MANUAL DO ALUNO

DISCIPLINA SISTEMAS DIGITAIS E ARQUITETURA DE COMPUTADORES

Módulo 6

República Democrática de Timor-Leste
Ministério da Educação

FICHA TÉCNICA

TÍTULO
MANUAL DO ALUNO - Disciplina de Sistemas Digitais e Arquitetura de Computadores
Módulo 6

AUTOR
BRUNO MORAIS

COLABORAÇÃO DAS EQUIPAS TÉCNICAS TIMORENSES DA DISCIPLINA
XXXXXXX

COLABORAÇÃO TÉCNICA NA REVISÃO

DESIGN E PAGINAÇÃO
UNDESIGN - JOAO PAULO VILHENA
EVOLUA.PT

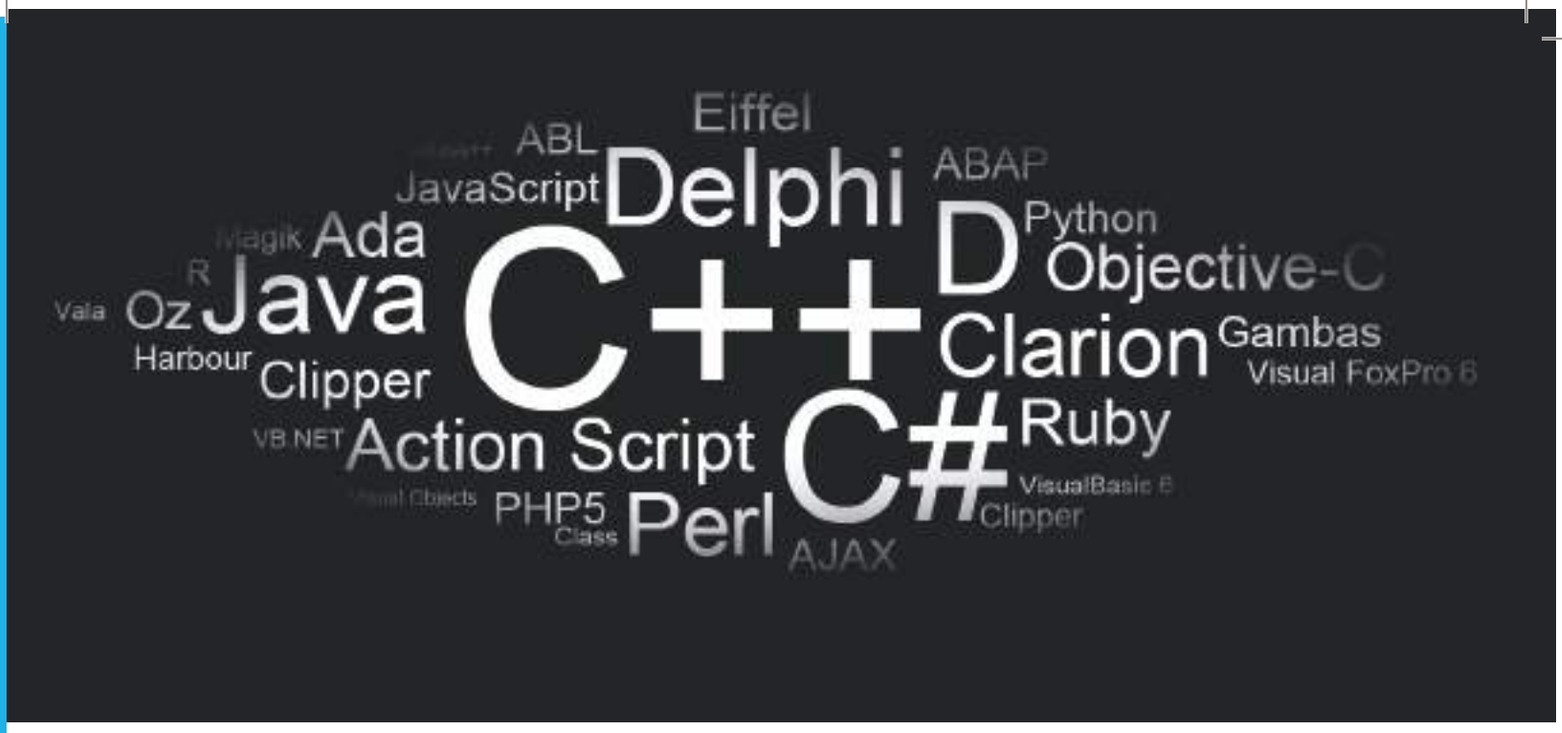
IMPRESSÃO E ACABAMENTO
XXXXXX

ISBN
XXX - XXX - X - XXXXX - X

TIRAGEM
XXXXXXX EXEMPLARES

COORDENAÇÃO GERAL DO PROJETO
MINISTÉRIO DA EDUCAÇÃO DE TIMOR-LESTE
2014

Índice



Fundamentos de Programação

Módulo 6

Apresentação

Neste módulo os alunos irão tomar contato com a criação de algoritmos, como sendo um conjunto ordenado de ações, sujeitas a critérios de decisão, condicionados a variáveis, através dos quais, se podem realizar e mecanizar determinados objetivos.

Para isso, serão abordadas diferentes fases, tais como:

- Análise do problema:Conhecer o problema: ouvir o problema, entendê-lo, perceber qual o objetivo.
    - Descrever o problema: subdividir o problema (esquematizar), detalhar.
- Resolução do problema: escrever passo a passo o raciocínio da solução do problema
- Teste da solução: verificar se não existe ambiguidade.
- Implementação: esta fase acontece apenas após o problema estar resolvido e consiste em implementar o algoritmo numa linguagem de programação.

Este é um módulo que se pretende integrador dos vários saberes, justificado pela interdependência dos conteúdos abordados, não sendo, por isso, aconselhável a sua modularização.

A ideia é ir introduzindo aos poucos os vários conceitos fazendo pequenos exercícios independentes.

À medida que os formandos vão assimilando os vários conceitos, começa-se a construir um projeto.

(leia-se programa) que vai ser complementado no decurso do módulo.

Objetivos de aprendizagem

- Iniciar o desenvolvimento de raciocínios algorítmicos e aquisição de métodos de desenvolvimento de programas.
- Abordar linguagens: sintaxe e semântica.
- Conhecer os conceitos de instrução, dados e programa.
- Utilizar uma linguagem natural e uma linguagem gráfica, no desenho e teste de Algoritmos.
- Utilizar um ambiente integrado de desenvolvimento de programas para edição, compilação e teste.
- Estudar tipos de dados simples.
- Conhecer e utilizar instruções de decisão.
- Conhecer e utilizar instruções de controlo de fluxo.
- Iniciar o desenvolvimento de raciocínios algorítmicos e aquisição de métodos de desenvolvimento de programas.
- Abordar linguagens: sintaxe e semântica.
- Conhecer os conceitos de instrução, dados e programa.
- Utilizar uma linguagem natural e uma linguagem gráfica, no desenho e teste de Algoritmos.
- Utilizar um ambiente integrado de desenvolvimento de programas para edição, compilação e teste.
- Estudar tipos de dados simples.
- Conhecer e utilizar instruções de decisão.
- Conhecer e utilizar instruções de controlo de fluxo.
- Saber fazer DEBUGGING e o visionamento passo-a-passo da execução de algoritmos.
- Conhecer e manipular estruturas de dados estáticas (vetores e matrizes).
- Saber decompor um programa em sub-programas (modularização).
- Utilizar parâmetros na construção de sub-programas.
- Conhecer os níveis de visibilidade (“Scope”) das variáveis de um programa.
- Estudar tipos de dados compostos.

- Saber analisar as necessidades de estruturas de informação utilizando as estruturas de dados apropriadas.
- Estudar as formas de armazenamento de informação em memória secundária como forma de manter.

Âmbito de conteúdos

- Definição de Linguagem.
- Conceitos de Sintaxe, Semântica, Gramática e Expressão.
- Exemplo com um subconjunto da Linguagem Natural.
- Erros de Sintaxe e de Semântica nas frases (expressões) de uma Linguagem (gramática).
- Definição de Algoritmo como processo descritivo de uma Linguagem.
- Exemplificação (meramente conceptual) de algoritmos simples.
- Desenvolvimento conceptual de tipos de informação e respetivo armazenamento (conceito intuitivo de variável num algoritmo).
- Valores Numéricos, Alfanuméricos e Lógicos.
- Desenvolvimento conceptual da possibilidade de o algoritmo dispor da capacidade de recolher informação do utilizador e de enviar informação para o utilizador.
- Exemplos em linguagem natural envolvendo mecanismos intuitivos de Decisão Binária e Decisão Múltipla.
- Exemplos em linguagem natural envolvendo mecanismos de repetição condicionada por uma expressão lógica.
- Desenvolvimento de algoritmos, fazendo uso de uma linguagem gráfica com o objetivo de analisar o seu fluxo de execução sequencial.
- Estudo e utilização de um ambiente integrado de desenvolvimento de programas para edição, compilação e teste de programas:
    - Estrutura de um programa.
    - Tipos de variáveis. Tipos simples.
    - Instruções: Afetação, Input e Output de informação.
    - Mecanismos de controlo de programa:
        - Seleção simples.
        - Seleção múltipla.
        - Repetição condicional.
        - Repetição incondicional.
    - Funções Simples.

- Implementação de Algoritmos de complexidade crescente.
- Utilização das ferramentas de Debugging disponíveis:
  - Observação do valor de variáveis.
  - Execução de algoritmos em modo "*STEP by STEP*."
  - Definição de "*Breakpoints*".
  - Execução de algoritmos por Troços.
- Estruturas de dados estáticas (unidimensionais):
  - Declaração e Manipulação.
- Estudo de algoritmos de manipulação de *Arrays*:
  - Algoritmos de iniciação.
  - Algoritmos de pesquisa sequencial.
  - Algoritmos de inserção e remoção de elementos: No Inicio (à Cabeça - FIFO); no Fim (à Cauda - LIFO).
  - Algoritmos de ordenação.
- Estruturas de dados estáticas (multidimensionais).
- Análise *top-down*, versus *bottom-up*:
  - Diferenças.
  - Declaração.
  - Utilização.
- Regras de "*Scope*" para a utilização de variáveis:
- Variáveis Globais e Variáveis Locais:
  - Período de existência das variáveis.
  - Regras de "*Scope*" para a utilização de variáveis.
- Passagem de parâmetros a sub-programas:
  - Passagem por Valor.
  - Passagem por Referência de Endereço.

Tipos de dados compostos:

- Sintaxe.
- Manipulação.
- Estruturas de dados compostos.
  - Desenho de aplicações que envolvam estruturas de dados de baixa complexidade.

- Ficheiros como variáveis suportadas em disco.
- Declaração de variáveis do tipo ficheiro da mesma forma que são declaradas variáveis em RAM.
- Associação do nome físico do ficheiro (ao nível do Sistema Operativo) ao nome lógico do ficheiro (ao nível do programa).
- Abertura e fecho de ficheiros.
- Acesso a ficheiros.
- Manipulação de ficheiros.

Exemplos de ficheiros de texto pré definidos: COM (Porta Série), LPT1 ou PRN (Porta Paralela. Exercícios envolvendo estes ficheiros (por exemplo, imprimir um ficheiro de texto).

Conceitos Básicos

Desde o início da existência do homem que este tem procurado criar maquinas que o auxiliem nos seus trabalhos, diminuindo esforços e economizando tempo. Dentre essas máquinas, o computador tem-se mostrado uma das mais versáteis, rápidas e seguras.
O computador é capaz de auxiliar em qualquer coisa que lhe seja solicitada, é consciente, trabalhador e possui muita energia, mas não tem iniciativa, nenhuma independência, não é criativo nem inteligente, por isso precisa receber instruções nos mínimos detalhes.
A finalidade de um computador é receber, manipular e armazenar dados. Se for apenas visto como uma caixa composta por circuitos eletrónicos, cabos e fontes de alimentação, certamente ele não tem utilidade alguma. O computador só consegue armazenar dados em discos, imprimir relatórios, gerar gráficos, realizar cálculos, entre outras funções, por meio de programas. Portanto, a sua finalidade principal é realizar a tarefa de *processamento de dados,* isto é, receber dados por um dispositivo de entrada (por exemplo, teclado, rato, scanner, entre outros), realizar operações com esses dados e gerar uma resposta que será expressa num dispositivo de saída (por exemplo, impressora, monitor, entre outros).

Portanto, um computador possui duas partes diferentes que trabalham juntas: o hardware, composto pelas partes físicas, e o software, composto pelos programas.
Quando queremos criar ou desenvolver um software para realizar determinado tipo de processamento de dados, devemos escrever um programa ou vários programas interligados. No entanto, para que o computador compreenda e execute esse programa, devemos escreve-lo usando uma linguagem que tanto o computador quanto o criador de software entendam. Essa linguagem e chamada de *linguagem de programação.*

As etapas para o desenvolvimento de um programa são:

- **Analise** - Nesta etapa estuda-se o enunciado do problema para definir os dados de entrada, o processamento e os dados de saída.
- **Algoritmo** - Ferramentas do tipo descrição narrativa, fluxograma ou português estruturado são utilizadas para descrever o problema com as suas soluções.
- **Codificação** - O algoritmo é transformado em códigos da linguagem de programação escolhida para se trabalhar.

Portanto, um programa é a codificação de um algoritmo numa linguagem de programação.

Conceito de Algoritmo

A seguir são apresentados alguns conceitos de algoritmos.

"Algoritmo é uma sequência de passos que visa atingir um objetivo bem definido." (FORBELLONE, 1999)

" Algoritmo é a descrição de uma sequência de passos que deve ser seguida para a realização de uma tarefa." (ASCENCIO, 1999)

" Algoritmo é uma sequência finita de instruções ou operações cuja execução, em tempo finito, resolve um problema computacional, qualquer que seja a sua instância." (SALVETTI, 1999)

" Algoritmo são regras formais para a obtenção de um resultado ou da solução de um problema, englobando fórmulas de expressões aritméticas.'' (MANZANO, 1997)

" Ação é um acontecimento que, a partir de um estado inicial, apos um período de tempo finito, produz um estado final previsível e bem definido. Portanto, um algoritmo é a descrição de um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações." (FARRER, 1999)

Analisando as definições anteriores, podemos perceber que executamos no dia-a-dia vários algoritmos, como se pode observar nos exemplos a seguir.

Algoritmo 1 - Somar três números

Passo 1 — Receber os três números.

Passo 2 — Somar os três números.

Passo 3 — Mostrar o resultado obtido.

Algoritmo 2 - Fazer um sanduiche

Passo 1 — Buscar o pão.

Passo 2 — Cortar o pão ao meio.

Passo 3 — Pegar na maionese.

Passo 4 — Passar a maionese no pão.

Passo 5 — Buscar e cortar alface e tomate.

Passo 6 — Colocar alface e tomate no pão.

Passo 7 — Buscar o hambúrguer.

Passo 8 — Fritar o hambúrguer.

Passo 9 — Colocar o hambúrguer no pão.

Algoritmo 3 - Trocar uma lâmpada

Passo 1 — Buscar uma lâmpada nova.

Passo 2 — Buscar uma escada.

Passo 3 — Posicionar a escada de baixo da lâmpada queimada,

Passo 4 — Subir a escada com a lâmpada nova na mão.

Passo 5 — Retirar a lâmpada queimada.

Passo 6 — Colocar a lâmpada nova.

Passo 7 — Descer da escada.

Passo 8 — Testar o interruptor.

Passo 9 — Guardar a escada.

Passo 10 — Colocar a lâmpada velha no lixo.

Algoritmo 4 - Ir para a escola

Passo 1 — Acordar cedo.

Passo 2 — Ir à casa de banho.

Passo 3 — Abrir o armário para escolher uma roupa.

Passo 4 — Se o tempo estiver quente, escolher uma t-shirt e umas calças; caso contrário, escolher um agasalho e uma calças.

Passo 5 — Vestir a roupa escolhida.

Passo 6 — Tomar pequeno almoço.

Passo 7 — Apanhar Microlet.

Passo 8 — Sair próximo à escola.

Algoritmo 5 – Levantar dinheiro no Multibanco

Passo 1 — Ir até uma caixa multibanco.

Passo 2 — Colocar o cartão.

Passo 3 — Marcar a senha

Passo 4 — Escolher a quantia desejada

Passo 5 — Se o saldo for maior ou igual à quantia desejada, levantar; caso contrário, mostrar mensagem de impossibilidade de levantamento.

Passo 6 — Retirar o cartão.

E se estivermos a pensar: " Mas eu realizo essas atividades de maneira diferente!". Este pensamento está correto, pois às vezes um problema pode ser resolvido de diversas maneiras, porem, obtendo a mesma resposta, ou seja, podem existir vários algoritmos para solucionar o mesmo problema.

Método para a Construção de algoritmos

Para a construção de qualquer tipo de algoritmo, é necessário seguir estes passos:

a. Compreender completamente o problema a ser resolvido, destacando os pontos mais importantes e os objetos que o compõem.
b. Definir os dados de entrada, ou seja, quais os dados que serão fornecidos e quais objetos fazem parte do cenário/problema.
c. Definir o processamento, ou seja, quais os cálculos a serem efetuados e quais as restrições para esses cálculos. O processamento é responsável pela transformação dos dados de entrada em dados de saída.
d. Alem disso, deve-se verificar quais os objetos que são responsáveis pelas atividades.
e. Definir os dados de saída, ou seja, quais os dados que se obtêm depois do processamento.
f. Construir o algoritmo utilizando um dos tipos descritos na próxima secção.
g. Testar o algoritmo realizando simulações.

Tipos de Algoritmos

Os três tipos mais utilizados de algoritmos são: *descrição narrativa, fluxograma* e *pseudocódigo*.

Descrição narrativa

A descrição narrativa consiste em analisar o enunciado do problema e escrever, utilizando uma linguagem natural (por exemplo, a língua portuguesa ou tetun), os passos a serem seguidos para a sua resolução.

Vantagem: não é necessário aprender nenhum conceito novo, pois uma língua natural, neste ponto, já é bem conhecida.

Desvantagem: a língua natural abre espaço para varias interpretações, o que posteriormente dificultara a transcrição desse algoritmo para o programa.

Fluxograma

O fluxograma consiste em analisar o enunciado do problema e escrever, utilizando símbolos gráficos predefinidos (Tabela 1.1), os passos a serem seguidos para a sua resolução.

Vantagem: o entendimento de elementos gráficos é mais simples que o entendimento de textos.

Desvantagem: é necessário aprender a simbologia dos fluxogramas e, alem disso, o algoritmo resultante não apresenta muitos detalhes, dificultando a sua transcrição para um programa.

Pseudocodigo

O pseudocómico consiste em analisar o enunciado do problema e escrever, por meio de regras predefinidas, os passos a serem seguidos para a sua resolução.

Vantagem: a passagem do algoritmo para qualquer linguagem de programação é quase imediata, bastando conhecer as palavras reservadas dessa linguagem que serão utilizadas.

Desvantagem: é necessário aprender as regras do pseudocodigo, que serão apresentadas nas próximas secções.

	Símbolo utilizado para indicar o início e o fim do algoritmo
	Permite indicar o sentido do fluxo de dados. Serve exclusivamente para conectar os símbolos ou blocos existentes.
	Símbolo utilizado para indicar cálculos e atribuições de valores.
	Símbolo utilizado para representar a entrada de dados
	Símbolo utilizado para representar a saída de dados
	Símbolo utilizado para indicar que deve ser tomada uma decisão, apontando a possibilidade de desvios

Tabela 1.1: Conjunto de símbolos utilizados no fluxograma

Exemplos de Algoritmos

Os exemplos a seguir mostram alguns algoritmos desenvolvidos com os três tipos citados anteriormente,

a. Faca um algoritmo para mostrar o resultado da multiplicação de dois números.

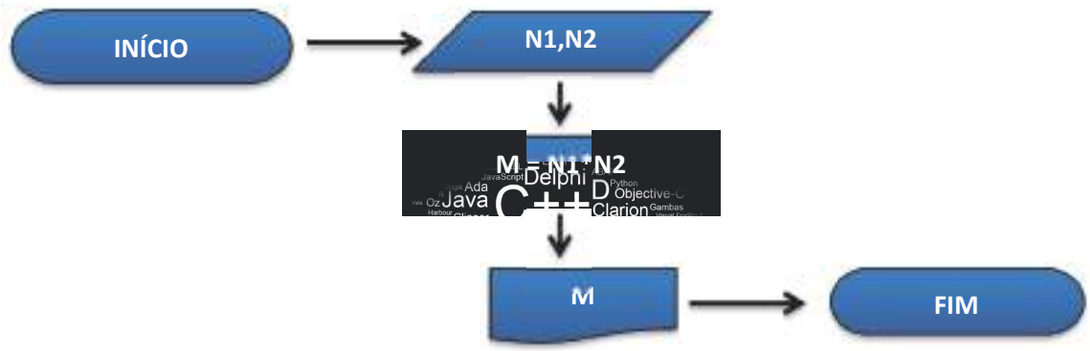
Algoritmo em descrição narrativa:

Passo 1 — Receber os dois números que serão multiplicados.

Passo 2 — Multiplicar os números.

Passo 3 — Mostrar o resultado obtido na multiplicação

Algoritmo em fluxograma:



Algoritmo em pseudocodigo:

ALGORITMO

DECLARE N1, N2, M NUMÉRICO

ESCREVA “Escreva dois números”

LEIA N1, N2

M ← N1 * N2

ESCREVA “Multiplicação = “, M

FIM_ALGORITMO.

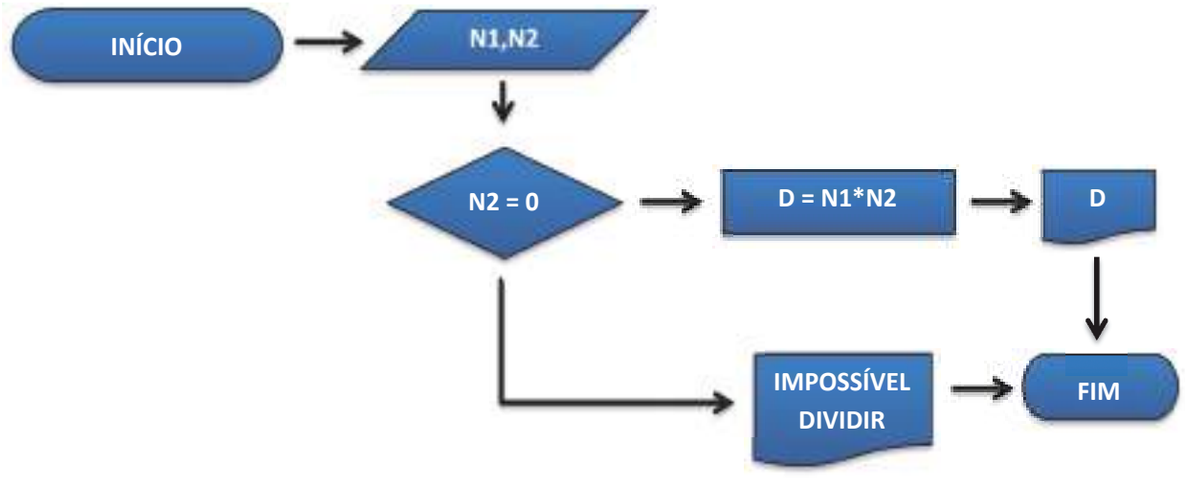
b. Faça um algoritmo para mostrar o resultado da divisão de dois números.

Algoritmo em descrição narrativa:

Passo 1 — Receber os dois números que serão divididos.

Passo 2 — Se o segundo número for igual a zero, não poderá ser feita a divisão, pois não existe divisão por zero; caso contrario, dividir os números e mostrar o resultado da divisão.

Algoritmo em fluxograma:



Algoritmo em pseudocodigo:

ALGORITMO

DECLARE N1, N2, D NUMÉRICO

ESCREVA “Escreva dois números”

LEIA N1, N2

SE N2 = 0

ENTÃO ESCREVA “Impossível dividir”

SENÃO INICIO

D ← N1/N2

ESCREVA “Divisão = “, D

FIM

FIM ALGORITMO.

c. Faça um algoritmo para calcular a média aritmética entre duas notas de um aluno e mostrar a sua situação, que pode ser aprovado ou reprovado.

Algoritmo em descrição narrativa:

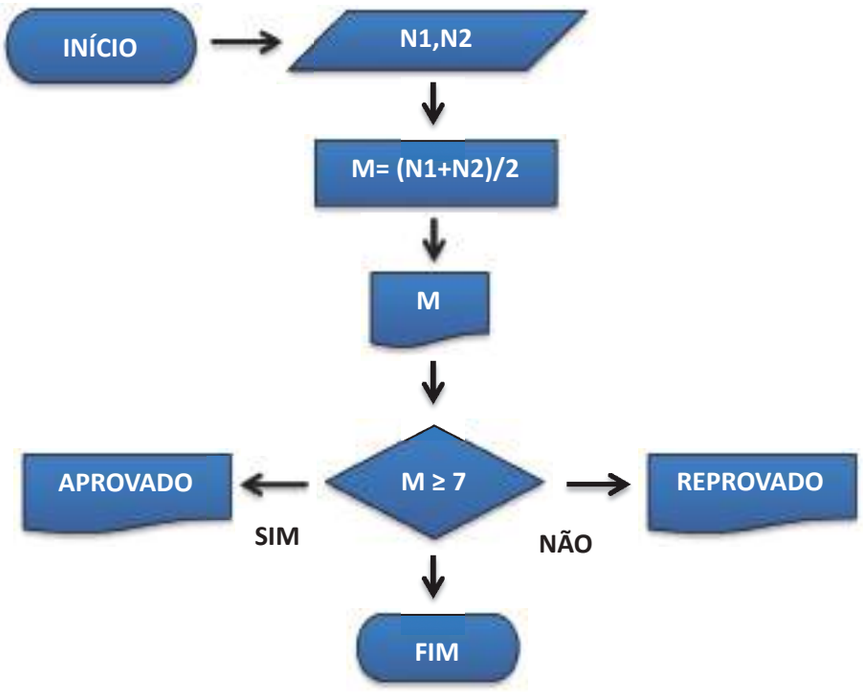
Passo 1 — Receber as duas notas.

Passo 2 — Calcular a média aritmética.

Passo 3 — Mostrar a média aritmética.

Passo 4 — Se a média aritmética for maior ou igual a 5, então a situação do aluno é aprovado; caso contrário, a situação é reprovado.

Algoritmo em fluxograma:



Algoritmo em pseudocodigo:

ALGORITMO

DECLARE N1, N2, M NUMÉRICO

ESCREVA “Escreva as duas notas”

LEIA N1, N2

M ← (N1 + N2)/2

ESCREVA “Media = “,M

SE M 5

ENTÃO ESCREVA “Aprovado”

SENÃO ESCREVA “Reprovado”

FIM_ALG0RITM0.

d. Faça um algoritmo para calcular o novo salário de um funcionário. Sabe-se que os funcionários que recebem atualmente salário de até $ 500 terão aumento de 20%; os demais terão aumento de 10%.

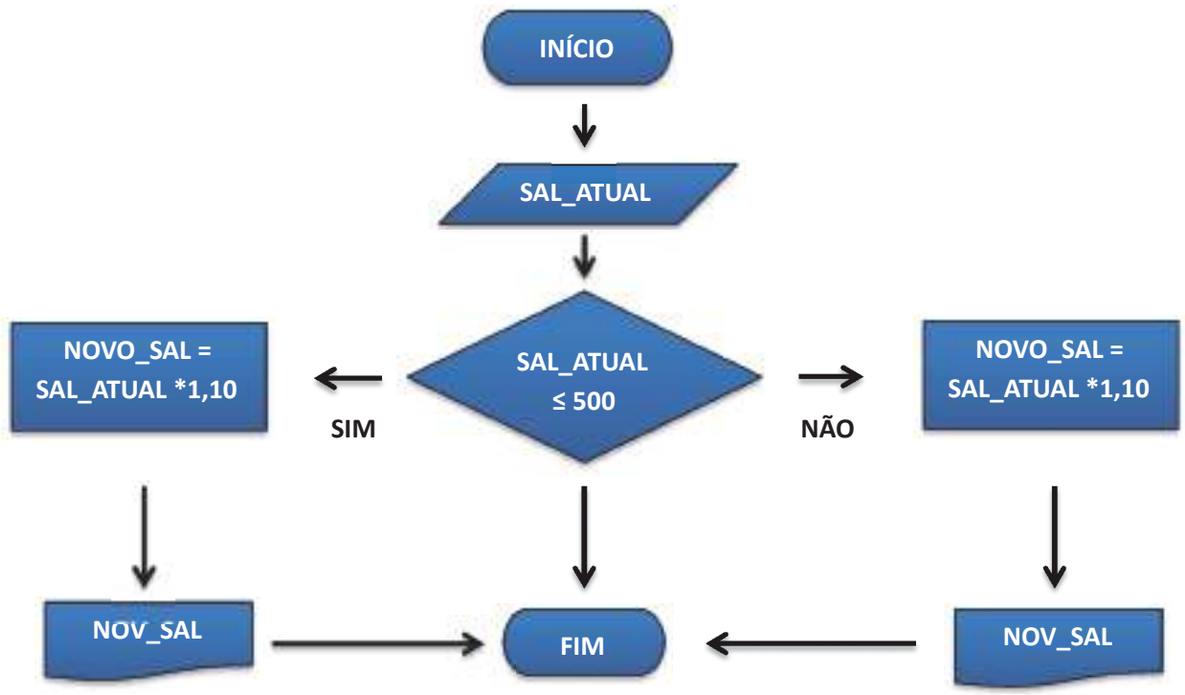
Algoritmo em descrição narrativa:

Passo 1 — Receber o salário atual do funcionário.

Passo 2 — Se o salário atual do funcionário for de até $ 500, calcular o novo salário com percentual de aumento de 20% ; caso contrário, calcular o novo salário com percentual de aumento de 10%.

Passo 3 — Mostrar o novo salário.

Algoritmo em fluxograma:



Algoritmo em pseudocodigo:

ALGORITMO

DECLARE SAL_ATUAL, NOVO_SAL NUMÉRICO

ESCREVA “Escreva o salário atual do funcionário”

LEIA SAL_ATUAL

SE SAL_ATUAL 500

ENTÃO NOVO_SAL ← SAL_ATUAL * 1,20

SENÃO N0VO_SAL ← SAL_ATUAL * 1,10

ESCREVA “Novo salário = “,NOVO_SAL

FIM ALGORITMO.

Conceito de variável

Duas pessoas estão a conversar e precisam de realizar uma conta.

A primeira pessoa diz: " Vamos somar dois números" e continua: "O primeiro número é 5".

A segunda pessoa guarda o primeiro número na cabeça, ou seja, na memória.

A primeira pessoa diz: "O segundo numero é 3" .

A segunda pessoa também guarda o segundo número na cabeça, sem esquecer o primeiro número, ou seja, cada numero foi armazenado em posições diferentes da memoria humana, sem sobreposição.

A primeira pessoa pergunta: "Qual é o resultado da soma?"

A segunda pessoa recupera os valores armazenados na memória, realiza a conta e responde que o resultado é 8.

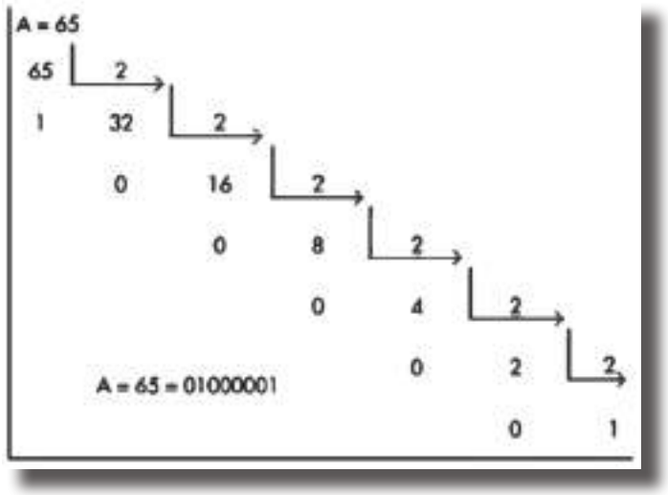
Um algoritmo é, posteriormente, um programa, recebem dados, que precisam ser armazenados no computador para serem utilizados no processamento. Esse armazenamento é feito na memória.

Todos os computadores trabalham com sistema numérico binário. Neste sistema, os dados são transformados em 0 e 1 ('zeros' e 'uns') para, então, serem armazenados na memória. Cada dígito binário (0 ou 1) ocupa porções de memória chamadas bytes (8 bits), e cada byte é identificado e acedido por meio de um endereço.

Todos os caracteres existentes possuem um correspondente numérico na tabela ASCII, que é transformado em caractere binário pelo método da divisão para, então, ser armazenado na memória.

Desta maneira, uma variável representa uma posição de memória. Possui nome e tipo, e o seu conteúdo pode variar ao longo do tempo, durante a execução de um programa. Embora uma variável possa assumir diferentes valores, ela só pode armazenar um valor a cada instante.

Exemplo de transformação em binário:



Todos os computadores possuem uma tabela de alocação que contem o nome da variável, o tipo (para saber quantos bytes ocupara) e o seu endereço inicial de armazenamento. Desta maneira, quando queremos procurar algum dado na memória, basta sabermos o nome da variável que o computador, por meio da tabela de alocação, procura automaticamente.

Tipos de dados

Os tipos de dados mais utilizados são: *numéricos*, *lógicos* e *literais* ou *caracteres*.

Numéricos

Os dados numéricos dividem-se em dois grupos: *inteiros* e *reais.*

Os números inteiros podem ser positivos ou negativos e *não* possuem parte fracionária.

Exemplos de dados numéricos inteiros:

- -23
- 98
- 0
- -357
- 237
- -2

Os números reais podem ser positivos ou negativos e possuem parte fracionária.

Exemplos de dados numéricos reais:

- 23,45
- 346,89
- -34,88
- 0 , 0
- -247,0

Lógicos

São também chamados dados booleanos (por causa da álgebra de Boole) e podem assumir os valores: *verdadeiro* ou *falso.*

OBSERVSÇÂO: Os números reais seguem a notação da língua inglesa, ou seja, a parte decimal é separada da parte inteira por um . (ponto) e não por uma , (virgula).

Caracteres

São dados formados por um único caractere ou por uma cadeia de caracteres. Estes caracteres podem ser as letras maiúsculas, as letras minúsculas, os números (não podem ser usados para cálculos) e os caracteres especiais (& , #, @, ?, +).

Exemplos de dados literais:

- ‘aluno’
- ‘1234’
- ‘@ internet’
- ‘0 .34 ‘
- ‘1 + 2’

Formação de identificadores

Os identificadores são os nomes das variáveis, dos programas, das constantes, das rotinas, das unidades etc.

As regras básicas para a formação dos identificadores são:

- Os caracteres que se podem utilizar são: os números, as letras maiúsculas, as letras minúsculas e o caractere sublinhado.
- O primeiro caractere deve ser sempre uma letra ou o caractere sublinhado.
- Não são permitidos espaços em branco e caracteres especiais (@, $, +, -, % , !).
- Não podemos usar as palavras reservadas nos identificadores, ou seja, palavras que pertençam a uma linguagem de programação.

Exemplos de identificadores

Exemplos de identificadores validos:

- A
- a
- nota
- NOTA
- X5
- A32
- NOTA1
- MATRICULA
- nota_1
- dia
- IDADE

Exemplos de identificadores inválidos:

- 5b - por começar com número;
- e 12 - por conter espaço em branco;
- x-y - por conter o caractere especial;
- prova 2n - por conter espaço em branco;
- nota (2) - por conter os caracteres especiais ();
- case - por ser palavra reservada;
- set - por ser palavra reservada.

Linguagem PASCAL

A linguagem PASCAL foi desenvolvida em 1968 por Niklaus Wirth, na Suíça, destinada principalmente à programação cientifica, mas a sua grande evolução permitiu que, nos dias de hoje, seja utilizada para qualquer fim.

Por se tratar de uma linguagem estruturada, isto é, uma linguagem que possui regras para a escrita dos seus programas, é muito utilizada por alunos que começam a aprender programação. A linguagem PASCAL possui um ambiente integrado de desenvolvimento chamado Pascal Zim com as seguintes características:

- Apresenta um editor que permite ao desenvolvedor do programa escrever, guardar e modificar o código dos seus programas.
- Possui um compilador que converte os códigos dos programas em instruções de máquina e permite-lhe compilar, ou seja, verificar a existência de erros de sintaxe nos programas sem retornar ao sistema operacional.
- Dispõe de um depurador que permite inspecionar um programa durante a sua execução, facilitando a localização de erros.
- Conta com um sistema de ajuda ativo que oferece diferentes níveis de informações.
- Possui ainda o ambiente de execução propriamente dito, que permite executar os programas sem sair do Pascal Zim (com ficheiros de extensão PAS) ou, se preferir, permite criar ficheiros a serem executados fora do ambiente do Pascal Zim (com ficheiros de extensão EXE).

Portanto, para fazer um programa utilizando a linguagem de programação PASCAL devemos:

- Analisar o enunciado do problema, algoritmo e codificação, utilizando o editor do ambiente de desenvolvimento
- Pascal Zim.
- Compilar, utilizando o compilador do ambiente de desenvolvimento Pascal Zim.
- Executar.

Paradigmas de Programação

Um paradigma de programação está intimamente relacionado à forma de pensar do programador e como ele procura a solução para os problemas. É o paradigma que permite ou proíbe a utilização de algumas técnicas de programação. Ele é capaz, ainda, de mostrar como o programador analisou e abstraiu o problema a resolver.

Existem vários paradigmas de programação: estruturado, orientado a objetos, logico, funcional, dentre outros.

Vamos analisar com mais detalhes os paradigmas estruturados e orientados a objetos.

Pelo *paradigma estruturado* (também conhecido como *imperativo*), qualquer problema pode ser resolvido utilizando três estruturas: sequencial, condicional e iterativa (repetição). Alem disso, procura encontrar uma forma de resolver um problema complexo em partes mais pequenas sendo assim mais simples do que, trabalhadas conjuntamente, permitam soluciona-lo.

A ideia é que, utilizando corretamente tais estruturas, o recurso da modularização e a parametrização, seja possível criar programas com menor repetição possível de linhas de comandos.

Já o *paradigma orientado a objetos* compreende o problema como uma coleção de objetos interagindo por meio de trocas de mensagem. Os objetos são estruturas de dados incluindo logica. Desta maneira, um conjunto de objetos com informações comuns e com o mesmo comportamento dá origem a uma classe.

Exemplificando, um programador que utiliza o paradigma estruturado analisa o problema tentando relacionar as ações que deverão ser executadas e como poderão ser divididas em módulos. Um programador que utilize o paradigma orientado a objetos analisaria o mesmo problema tentando identificar os objetos que compõem essa realidade e como interagem.

Como o paradigma está ligado à forma de pensar do programador, o simples fato de se utilizar, por exemplo, uma linguagem com suporte nativo à orientação a objetos não implica que a solução apresentada seja orientada a objetos, ou, então, muitas soluções não estruturadas são feitas utilizando linguagens com suporte à estruturação.

Estrutura Sequencial

Estrutura Sequencial em Algoritmos

ALGORITMO

DECLARE

bloco de comandos

FIM_ALGORITMO.

Declaração de Variáveis em Algoritmos

As *variáveis* são declaradas após a palavra declare e os tipos mais utilizados são: numérico (para variáveis que receberão números), caractere (para variáveis que receberão caracteres) e lógico (para variáveis que receberão apenas dois valores: verdadeiro ou falso).

Exemplo:

DECLARE

X NUMERICO

Y, Z LITERAL

TESTE LÓGICO

Comando de Atribuição em Algoritmos

O *comando de atribuição* é utilizado para conceder valores ou operações a variáveis, sendo representado pelo símbolo.

Exemplo:

x ← 4

x ← x + 2

y ← “aula”

teste ← falso

Comando de Entrada em Algoritmos

O *comando de entrada* é utilizado para receber dados escritos pelo utilizador, que serão armazenados em variáveis. Esse comando é representado pela palavra leia.

Exemplo:

LEIA X

Um valor escrito pelo utilizador será armazenado na variável x.

LEIA Y

Um ou vários caracteres escritos pelo utilizador serão armazenados na variável y.

Comando de Saída em Algoritmos

O *comando de saída* é utilizado para mostrar dados no ecrã ou na impressora. Este comando é representado pela palavra escreva, e os dados podem ser conteúdos de variáveis ou mensagens.

Exemplo:

ESCREVA X

Mostra o valor armazenado na variável x.

ESCREVA "Conteúdo de Y = ", Y

Mostra a mensagem "Conteúdo de y = " e em seguida o valor armazenado na variável y.

Estrutura Sequencial em Pascal

```
PROGRAM nome;
USES nomes_das_unidades;
VAR nome_das_variaveis : tipo;
BEGIN
    bloco de comandos;
END.
```

As unidades são bibliotecas utilizadas pela linguagem PASCAL para a correta execução do programa. A unidade CRT é obrigatória em todos os programas, pois faz a adequação do hardware com o programa.

Declaração de variáveis em PASCAL

As *variáveis* são declaradas após a palavra var e os tipos mais utilizados são: integer (para números inteiros), real (para números reais), char (para um caractere), string (para vários caracteres) e boolean (para verdadeiro ou falso).

Exemplo:

```
VAR X: INTEGER;
        Y, Z: REAL;
        NOME: STRING;
        SEXO: CHAR;
        TESTE: BOOLEAN;
```

Os identificadores são os nomes das variáveis, dos programas, das constantes, das rotinas e unidades, entre outras.

As regras básicas para a formação dos identificadores são:

- Podem ter qualquer tamanho. Entretanto, apenas os 63 primeiros caracteres são utilizados pelo compilador.
- Os caracteres que podem ser utilizados na formação dos identificadores são: os números, as letras maiúsculas, as letras minúsculas e o caractere sublinhado.
- O compilador não faz distinção entre letras maiúsculas e minúsculas, portanto, o identificador NUM é exatamente igual ao identificador num.
- O primeiro caractere deve ser sempre uma letra ou o caractere sublinhado.
- Não são permitidos espaços em branco e caracteres especiais (@, $, +, -, %, !).
- Não é permitido usar palavras reservadas.

PALAVRAS RESERVADAS são nomes utilizados pelo compilador para representar comandos, operadores e nomes de secções de programas. As palavras reservadas da linguagem PASCAL são:		
and	goto	program
asm	if	record
array	implementation	repeat
begin	in	set
case	inherited	shl
const	inline	shr
constructor	interface	string
destructor	label	then
div	library	to
do	mod	type
downto	nil	unit
else	not	until
end	object	uses
exporte	of	var
file	or	while
for	packed	with
function	procedure	xor

Os tipos de dados mais utilizados na linguagem PASCAL estão descritos na tabela a seguir:

TIPO	FAIXA DE VALORES	TAMANHO (aproximado)
shortint	-128 a 127	8 bits
integer	-32.768 a 32.767	16 bits
longint	-2.147.483.648 a 2.147.483.647	32 butes
byte	0 a 255	8 bites
word	0 a 65.535	16 bites
real	$2,9 \times 10^{-39}$ a $1,7 \times 10^{38}$ (11 a 12 dígitos com sinal)	6 bytes
single	$1,5 \times 10^{-45}$ a $3,4 \times 10^{38}$ (7 a 8 dígitos com sinal)	4 bytes
double	$5,0 \times 10^{-324}$ a $1,7 \times 10^{308}$ (15 a 16 dígitos com sinal)	8 bytes
extended	$5,0 \times 10^{-4932}$ a $1,7 \times 10^{4932}$ (19 a 20 dígitos com sinal)	10 bytes
comp	$-9,2 \times 10^{18}$ a $9,2 \times 10^{18}$ (19 a 20 dígitos com sinal)	8 bytes
boolean	true ou false	8 bites
wordbool	true ou false	16 bites
longbool	true ou false	32 bits
bytebool	true ou false	8 bits
char	1 caractere qualquer	1 byte
string	cadeia de caracteres (no max 255)	tantos bytes quantos forem os caracteres

Comando de atribuição em PASCAL

O *comando de atribuição* é utilizado para dar valores ou operações a variáveis, sendo representado por : = (o sinal de dois-pontos e de igualdade).

Exemplo:

```
x : = 4 ;
x : = x + 2 ;
y : = 2.5;
nome : = 'AULA';
sexo : = 'F ';
teste : = false;
```

Em PASCAL, os caracteres literais são representados entre apóstrofos; os números reais utilizam o ponto como separador decimal; cada comando é finalizado com o sinal de ponto-e-vírgula.

Comando de entrada em PASCAL

O *comando de entrada* é utilizado para receber dados escritos pelo utilizador. Estes dados são armazenados em variáveis. Este comando é representado pela palavra readln. A sua sintaxe está representada a seguir:

Sintaxe:

READLN (nome_da_variavel);

READLN (nome_da_variavell,nome_da_variavel2);

Exemplo:

READLN (X) ;

Um valor escrito pelo utilizador será armazenado na variável x.

READLN (NOME);

Um ou vários caracteres escritos pelo utilizador serão armazenados na variável NOME.

Comando de saída em PASCAL

O *comando de saída* é utilizado para mostrar dados no ecrã ou na impressora. Este comando é representado pelas palavras write ou writeln e os dados podem ser conteúdos de variáveis ou mensagens.

Sintaxe:

WRITE (nome_da_variável);

WRITELN (nome_da_variável);

WRITE (‘ mensagem’);

WRITELN (‘mensagem’);

WRITE (‘ mensagem’, nome_da_variável);

WRITELN (‘mensagem’, nome_da_variável);

Exemplo:

WRITELN (X);

WRITE (X);

Mostra o valor armazenado na variável x.

WRITELN (‘Conteúdo de Y = ‘ , Y) ;

WRITE (‘Conteúdo de Y = \Y);

Mostra a mensagem “Conteúdo de y = “ e de seguida o valor armazenado na variável y. A diferença entre estes comandos é que o comando writeln mostra o seu conteúdo e passa o cursor para a linha de baixo, enquanto o comando write mantem o cursor na mesma linha após mostrar a mensagem.

Comentários em PASCAL

Os comentários não são interpretados pelo compilador, servem apenas para esclarecer o programador. Constituem excelentes instrumentos de documentação e devem sempre estar entre { .....} ou entre (* ..... *).

Operadores e funções predefinidas em PASCAL

A linguagem PASCAL possui operadores e funções predefinidas destinados a cálculos matemáticos. Alguns são apresentados a seguir.

OPERADOR	EXEMPLO	COMENTÁRIO
: =	x : = y	O conteúdo da variável Y é atribuído à variável X. (A uma variável pode ser atribuído o conteúdo de outra, um valor constante ou, ainda, o resultado de uma função.)
+	x + y	Soma o conteúdo de X e de Y.
-	x – y	Subtrai o conteúdo de Y do conteúdo de X.
*	x * y	Multiplica o conteúdo de X pelo conteúdo de Y.
/	x / y	Obtém o quociente da divisão de X por Y.
div	x div y	Obtém o quociente inteiro da divisão de X por Y.
mod	x mod y	Obtém o resto da divisão de X por Y.

Observações:

a. Os operadores div e mod só podem ser aplicados com operandos do tipo inteiro.
b. O operador / conduz sempre a um resultado real.
c. Com os operadores * e /, se pelo menos um dos operandos for real, então o resultado será real.

OPERADOR	EXEMPLO	COMENTÁRIO
=	x = y	O conteúdo de X e igual ao conteúdo de Y.
<>	x <> y	O conteúdo de X e diferente do conteúdo de Y.
<=	x <= y	O conteúdo de X e menor ou igual ao conteúdo de Y.
>=	x >= y	O conteúdo de X e maior ou igual ao conteúdo de Y.
<	x < y	O conteúdo de X e menor que o conteúdo de Y.
>	x > y	O conteúdo de X e maior que o conteúdo de Y.

FUNÇÕES MATEMÁTICAS		
FUNÇÃO	**EXEMPLO**	**COMENTÁRIO**
abs	abs (x)	Obtém o valor absoluto de X.
exp	exp (x)	Obtém o logaritmo natural e elevado a potencia X.
log	log (x)	Obtém o logaritmo natural de X.
trunc	trunc (x)	Obtém a parte inteira do numero real armazenado em X.
frac	frac (x)	Obtém a parte fracionaria do numero real armazenado em X.
round	round (x)	Arredonda X.
sin	sin (x)	Calcula o seno de X (X deve estar representado em radianos).
cos	cos (x)	Calcula o cosseno de X (X deve estar representado em radianos).
pi	Pi	Retorna o valor de π
sqrt	sqrt (x)	Calcula a raiz quadrada de x
sqr	sqr (x)	Calcula X elevado ao quadrado
inc	inc (x,y)	Incrementa a variável X com o valor da variável Y.
dec	dec (x,y)	Decrementa a variável X com o valor da variável Y.

Observação 1:

Por não existir o operador de potenciação, temos:

A^B = EXP (B*LN(A))

Exemplo:

3^4 = exp (4*ln (3))

5^{10} = exp (10*ln (5))

Observação 2:

As funções SIN e COS esperam receber argumentos no formato de radianos; para receber argumentos em graus, siga o próximo exemplo. Na linguagem PASCAL não existe uma função para tangente;

Sendo assim, utilizamos seno/cosseno.

Exemplo com variável para o valor de π:

```
VALORPI := 3.1415;
READLN (X); { X EM GRAUS }
Y := SIN ((VALORPI * X) / 180);
```

Exemplo utilizando a função pi:

```
READLN (X); { X EM GRAUS }
Y := SIN ((PI * X) / 180);
```

As prioridades entre os operadores são:

Iº) ( )

2º) funções

3º) *, /, DIV, MOD

4º) + , -

Quando se tem uma expressão em que os operadores têm a mesma prioridade, a expressão e resolvida da esquerda para a direita.

Exemplos:

2 +3 - 4 = 5 - 4 = 1

2 * 4/2 = 8/2 = 4

Exercícios Resolvidos

Exercício 1

Faça um programa que receba quatro números inteiros, calcule e mostre a soma desses números.

Solução Algoritmo

```
ALGORITMO
        DECLARE n1, n2, n3, n4, soma NUMÉRICO
        LEIA n1, n2, n3, n4
        soma ← n1 + n2 + n3 + n4
        ESCREVA soma
FIM_ALGORITMO.
```

Solução 1 PASCAL

```
PROGRAM EX1;
        USES CRT;
        VAR n1, n2, n3, n4, soma: INTEGER;
BEGIN
        {Recebe os quatro números}
        READLN(n1, n2, n3, n4);
        {Soma os números escritos}
        soma := n1 + n2 + n3 + n4;
        {Mostra o resultado da soma}
        WRITELN(soma);
        {Para o programa à espera de um enter}
READLN;

END.
```

Solução 2 PASCAL

```
PROGRAM EX1;
        USES CRT;
        VAR n1, n2, n3, n4, soma: INTEGER;
```

```
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem antes da leitura dos números}
        WRITELN('Escreva quatro números');
        {Recebe os quatro neros}
        READLN(n1, n2, n3, n4);
        {Soma os números digitados}
        soma := n1 + n2 + n3 + n4;
        {Mostra mensagem e o resultado da soma}
        WRITELN('Resultado da soma = ',soma);
{Para o programa à espera de um enter}
READLN;

END.
```

Exercício 2

Faça um programa em PASCAL que receba três notas, calcule e mostre a média aritmética entre elas.

Solução 1 Algoritmo

```
ALGORITMO
        DECLARE nota1, nota2, nota3, media NUMÉRICO
        LEIA nota1, nota2, nota3
        media ← (nota1 + nota2 + nota3)/3
        ESCREVA media
FIM_ALGORITMO.
```

Solução 2 Algoritmo

```
ALGORITMO
        DECLARE nota1, nota2, nota3, soma, media NUMÉRICO
        LEIA nota1, nota2, nota3
```


```
        soma ← notai + nota2 + nota3
        media ← soma/3
        ESCREVA média
FIM_ALGORITMO.
```

Solução 1 PASCAL

```
PROGRAM EX2;
        USES CRT;
        VAR nota1, nota2, nota3, media: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Recebe as três notas}
        READLN(nota1, nota2, nota3);
        {Calcula a média aritmética}
        media := (nota1 + nota2 + nota3)/3;
        {Mostra a média formatada para duas casas decimais}
        WRITELN(media:4:2);
        {Para o programa à espera de um enter}
        READLN;

END.
```

Solução 2 PASCAL

```
PROGRAM EX2;
        USES CRT;
        VAR nota1, nota2, nota3, soma, media: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem antes da leitura das notas}
        WRITELN('Digite as três notas');
```

```
{Recebe as três notas}
READLN(nota1, nota2, nota3);
{Calcula a média}
soma := (nota1 + nota2 + nota3);
media := soma/3;
{Mostra a média formatada com duas casas decimais}
WRITELN('Média = ',media:4:2);
{Para o programa à espera de um enter}
READLN;

END.
```

Exercício 3

Faça um programa que receba três notas e seus respetivos pesos, calcule e mostre a média ponderada dessas notas.

Solução 1 Algoritmo

```
ALGORITMO
    DECLARE nota1, nota2, nota3, peso1, peso2, peso3, média NUMÉRICO
    LEIA nota1, nota2, nota3, peso1, peso2, peso3
    média ← (nota1 * peso1 + nota2 * peso2 + nota3 * peso3)/(peso1 + peso2 +
peso3)
    ESCREVA média
FIM_ALGORITMO.
```

Solução 2 Algoritmo

```
ALGORITMO
    DECLARE nota1, nota2, nota3, peso1, peso2, peso3 NUMÉRICO
        soma1, soma2, soma3, total, media NUMERICO
    LEIA notai1 nota2, nota3, peso1, peso2, peso3
    soma1 ← nota1 * peso1
    soma2 ← nota2 * peso2
```

```
        soma3 ← nota3 * peso3
        total ← peso1 + peso2 + peso3
        média (soma1 + soma2 + soma3)/total
        ESCREVA média
FIM_ALGORITMO.
```

Solução 1 PASCAL

```
PROGRAM EX3;
        USES CRT;
        VAR nota1, nota2, nota3, peso1, peso2, peso3, media: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Recebe as três notas e os três pesos}
        READLN(nota1, nota2, nota3, peso1, peso2, peso3);
        {Calcula a média ponderada}
        media := (nota1 * peso1 + nota2 * peso2 + nota3 * peso3)/(peso1 + peso2 +
peso3);
        {Mostra a média ponderada formatada com duas casas decimais}
        WRITELN(media:5:2);
        {Para o programa à espera de um enter}
        READLN;

END.
```

Solução 2 PASCAL

```
PROGRAM EX3;
        USES CRT;
        VAR     nota1, nota2, nota3, peso1, peso2, peso3: REAL;
                soma1, soma2, soma3, total, media: REAL;
BEGIN
        {Limpa o ecrã}
```

```
        CLRSCR;
        {Mostra mensagem antes da leitura das três notas}
        WRITELN(‘ Escreva as três notas’);
        {Recebe as três notas}
        READLN(nota1, nota2, nota3);
        {Mostra mensagem antes da leitura dos três pesos}
        WRITELN(‘ Escreva os três pesos’);
        {Recebe os três pesos}
        READLN(peso1, peso2, peso3);
        {Calcula a média ponderada}
        soma1:=nota1 * peso1;
        soma2:=nota2 * peso2;
        soma3:=nota3 * peso3;
        total:=peso1 + peso2 + peso3;
        media := (soma1 + soma2 + soma3)/total;
        {Mostra a média ponderada formatada com duas casas decimais}
        WRITELN(‘Média Ponderada = ‘,media:5:2);
        {Para o programa à espera de um enter}
        READLN;
END.
```

Exercício 4

Faça um programa que receba o salário de um funcionário, calcule e mostre o novo salário, sabendo-se que este sofreu um aumento de 25%.

Solução 1 Algoritmo

```
ALGORITMO
        DECLARE sal, novosal NUMÉRICO
        LEIA sal
        novosal  sal + sal * 25/100
        ESCREVA novosal
FIM_ALGORITMO.
```

Solução 2 Algoritmo

```
ALGORITMO
        DECLARE sal, aumento, novosal NUMÉRICO
        LEIA sal
        aumento ← sal * 25/100
        novosal ← sal + aumento
        ESCREVA novosal
FIM_ALGORITMO.
```

Solução 1 PASCAL

```
PROGRAM EX4;
        USES CRT;
        VAR sal, novosal: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem antes da leitura do sal rio}
        WRITELN(Escreva o salário do funcionário');
        {Recebe o salário}
        READLN(sal);
        {Calcula o novo sal rio}
        novosal := sal + sal * 25/100;
        {Mostra o novo sal rio calculado com duas casas decimais}
        WRITELN('Novo salário = ',novosal:5:2);
        {Para o programa à espera de um enter}
        READLN;
END.
```

Solução 2 PASCAL

```
PROGRAM EX4;
        USES CRT;
        VAR sal, aumento, novosal: REAL;
```

```
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem antes da leitura do salário}
        WRITELN('Escreva o salário');
        {Recebe o salário}
        READLN(sal);
        {Calcula o aumento}
        aumento := sal * 25/100;
        {Calcula o novo salário}
        novosal := sal + aumento;
        {Mostra o novo sal rio formatado com duas casas decimais}
        WRITELN('Novo salário = ', novosal:5:2);
        {Para o programa à espera de um enter}
        READLN;

END.
```

Exercício 5

Faça um programa que receba o salário de um funcionário e o percentual de aumento, calcule e mostre o valor do aumento e o novo salário.

Solução Algoritmo

```
ALGORITMO
        DECLARE sal, perc, aumento, novosal NUMÉRICO
        LEIA sal, perc
        aumento ← sal * perc/100
        ESCREVA aumento
        novosal ← sal + aumento
        ESCREVA novosal
FIM_ALGORITMO.
```


Solução PASCAL

```
PROGRAM EX5;
        USES CRT;
        VAR sal, perc, aumento, novosal: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem antes da leitura do salário}
        WRITELN('Digite o sal rio do funcion rio');
        {Recebe o salário}
        READLN(sal);
        {Mostra mensagem antes da leitura do percentual de aumento}
        WRITELN('Escreva o percentual de aumento');
        {Recebe o percentual de aumento}
        READLN(perc);
        {Calcula o valor do aumento}
        aumento := sal * perc/100;
        {Mostra o valor do aumento formatado com duas casas decimais}
        WRITELN('Aumento = ',aumento:5:2);
        {Calcula o novo salário}
        novosal := sal + aumento;
        {Mostra o valor do novo salário formatado com duas casas decimais}
        WRITELN('Novo salário = ',novosal:5:2);
        {Para o programa à espera de um enter}
        READLN;
END.
```

Exercícios Propostos

Exercício 1

Faça um programa que receba o salário base de um funcionário, calcule e mostre o salário a receber, sabendo-se que o funcionário tem gratificação de 5% sobre o salário base e paga imposto de 7% sobre este salário.

Exercício 2

Faça um programa que receba o salário base de um funcionário, calcule e mostre o seu salário a receber, sabendo-se que o funcionário tem gratificação de $50 e paga imposto de 10% sobre o salário base.

Exercício 3

Faça um programa que receba o valor de um depósito e o valor da taxa de juros, calcule e mostre o valor do rendimento e o valor total depois do rendimento.

Exercício 4

Faça um programa que calcule e mostre a área de um triângulo. Sabe-se que: Área = (base * altura) /2

Exercício 5

Faça um programa que calcule c mostre a área de um circulo. Sabe-se que: Área = π * R^2.

Exercício 6

Faça um programa que receba um número positivo e maior que zero, calcule e mostre:

a) o numero escrito ao quadrado;

b) o numero escrito ao cubo;

c) a raiz quadrada do numero escrito;

d) a raiz cubica do numero escrito.

Exercício 7

Faça um programa que receba dois números maiores que zero, calcule e mostre um elevado ao outro.

Exercício 8

Sabe-se que:

1 pé = 12 polegadas

1 jarda = 3 pés

1 milha = 1.760 jardas

Faça um programa que receba uma medida em pés, faça as conversões a seguir e mostre os resultados.

a. polegadas;
b. jardas;
c. milhas.

Exercício 9

Faça um programa que receba o ano de nascimento de uma pessoa e o ano atual, calcule e mostre:

a. a idade dessa pessoa.
b. quantos anos ela terá em 2050.

Exercício 10

O custo ao consumidor de um carro novo é a soma do preço de fábrica com o percentual de lucro do distribuidor e dos impostos aplicados ao preço de fábrica. Faça um programa que receba o preço de fábrica de um veículo, o percentual de lucro do distribuidor e o percentual de impostos, calcule e mostre:

a. o valor correspondente ao lucro do distribuidor;
b. o valor correspondente aos impostos;
c. o preço final do veículo.

Exercício 11

Faça um programa que receba o número de horas trabalhadas e o valor do salário mínimo, calcule e mostre o salário a receber seguindo estas regras:

a. a hora trabalhada vale a metade do salário mínimo.
b. o salário bruto equivale ao numero de horas trabalhadas multiplicado pelo valor da hora trabalhada.

c. o imposto equivale a 3% do salário bruto.

d. o salário a receber equivale ao salário bruto menos o imposto.

Exercício 12

Um trabalhador recebeu o seu salário e depositou-o na sua conta bancaria. Esse trabalhador emitiu dois cheques e agora deseja saber o seu saldo atual. Sabe-se que cada operação bancaria de retirada paga uma taxa de 0,38% e o saldo inicial da conta está a zero.

Exercício 13

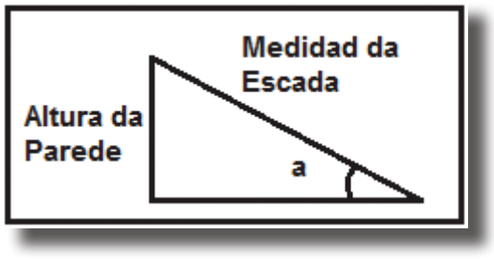
O Pedro comprou um saco de ração com peso em quilos. Ele tem dois gatos, para os quais fornece a quantidade de ração em gramas. A quantidade diária de ração fornecida para cada gato é sempre a mesma. Faça um programa que receba o peso do saco de ração e a quantidade de ração fornecida para cada gato, calcule e mostre quanto restara de ração no saco após cinco dias.

Exercício 14

Cada degrau de uma escada tem X de altura. Faça um programa que receba essa altura e a altura que o utilizador deseja alcançar ao subir a escada, calcule e mostre quantos degraus ele devera subir para atingir o seu objetivo, sem se preocupar com a altura do utilizador. Todas as medidas fornecidas devem estar em metros.

Exercício 15

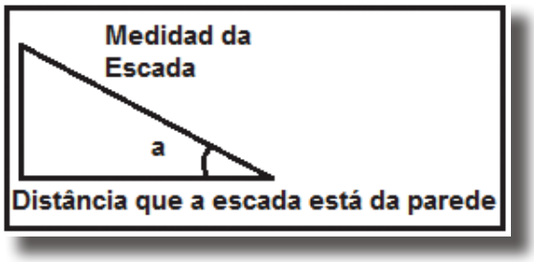
Faça um programa que receba a medida do angulo formado por uma escada apoiada no chão e encostada na parede e a altura da parede onde está a ponta da escada, calcule e mostre a medida desta escada.



Exercício 16

Uma pessoa deseja pregar um quadro numa parede. Faça um programa para calcular e mostrar a que distancia a escada deve estar da parede. A pessoa deve fornecer o

tamanho da escada e a altura a que deseja pregar o quadro. Lembre-se de que o tamanho da escada deve ser maior que a altura que se deseja alcançar.



X – Altura em que deseja pregar o quadro

Y – Distância em que deverá ficar a escada

Z – Tamanho da escada

Exercício 17

Sabe-se que o quilowatt de energia custa um quinto do salário mínimo. Faça um programa que receba o valor do salário mínimo e a quantidade de quilowatts consumida por uma residência, calcule e mostre:

a. o valor de cada quilowatt;
b. o valor a ser pago por essa residência;
c. o valor a ser pago com desconto de 15%.

Exercício 18

Faça um programa que receba um número real, calcule e mostre:

a. a parte inteira desse numero;
b. a parte fracionaria desse numero;
c. o arredondamento desse numero.
d.

Exercício 19

Faça um programa que receba uma hora formada por hora e minutos (um numero real), calcule e mostre a hora escrita apenas em minutos. Lembre-se de que:

e. para quatro e meia, deve-se digitar 4.30;
f. os minutos vão de 0 a 59.

Exercício 20

Faça um programa que receba o custo de um espetáculo teatral e o preço do convite desse espetáculo. Este programa deverá calcular e mostrar a quantidade de convites que devem ser vendidos para que pelo menos o custo do espetáculo seja alcançado.

Estrutura Condicional

Estrutura Condicional Em Algoritmos

Estrutura Condicional Simples

SE condição

ENTÃO comando

O comando só será executado se a condição for verdadeira. Uma condição é uma comparação que possui dois valores possíveis: verdadeiro ou falso.

SE condição

ENTÃO INICIO

comando1

comando2

comando3

FIM

Os comandos 1, 2 e 3 só serão executados se a condição for verdadeira. As palavras início e fim serão necessárias apenas quando dois ou mais comandos forem executados.

Estrutura Condicional Composta

SE condição

ENTÃO comando1

SENÃO comando2

Se a condição for verdadeira, será executado o comando1 caso contrario, será executado o comando2.

SE condição

ENTÃO INICIO

comando1

```
        comando2
        FIM
SENÃO INICIO
        comando3
        comando4
        FIM
```

Se a condição for verdadeira, o comando1 e o comando2 serão executados; caso contrário, o comando3 e o comando4 serão executados.

Estrutura Condicional Em Pascal

Estrutura Condicional Simples

```
IF condição
THEN comando;
```

O comando só será executado se a condição for verdadeira. Uma condição é uma comparação que possui dois valores possíveis: verdadeiro ou falso.

```
IF condição
THEN BEGIN
        comando1;
        comando2;
        comando3;
END;
```

Os comandos 1, 2 e 3 só serão executados se a condição for verdadeira.

Estrutura Condicional Composta

```
IF condição
THEN comando1
ELSE comando2;
```

Se a condição for verdadeira, será executado o comando1; caso contrario, será executado o comando2.

```
IF condição
THEN BEGIN
        comando1;
        comando2;
        END
ELSE BEGIN
        comando3;
        comando4;
        END;
```

Se a condição for verdadeira, o comando1 e o comando2 serão executados; se for falsa, o comando3 e o comando4 serão executados.

NOTA: Antes do comando ELSE não existe ponte e virgula.

Estrutura Case

Em alguns programas, existem situações mutuamente exclusivas, isto é, se uma situação for executada, as demais não serão. Quando for este o caso, um comando seletivo será o mais indicado, e esse comando, em PASCAL, tem a seguinte sintaxe:

```
CASE seletor OF
        lista de alvos1: comando1;
        lista de alvos2: comando2;
        alvo3: comando3;
        alvo4: BEGIN
                comando4;
                comando5;
                END;
        END;
```


Se o seletor atingir a lista de alvos 1, o comando1 será executado; se atingir a lista de alvos2, o comando2 sera executado; se atingir o alvo3, o comando3 será executado; se atingir o alvo4, então o comando4 e o comandos serão executados. Se nenhum alvo for atingido, nada será executado.

```
CASE seletor OF
        lista de alvos1:        BEGIN
                                comando1;
                                comando2;
                                END;
        lista de alvos2: cornando3;
        ELSE comando4;
        END;
```

Se o seletor atingir a lista de alvos1, o comando1 e o comando2 serão executados; se atingir a lista de alvos2, o comando3 será executado. Se nenhum alvo for atingido, será executado o comando4.

Os alvos podem ser valores únicos, listas de valores (separados por virgulas) ou faixas de valores, como no exemplo que se segue.

Exemplo:

```
program teste;
uses crt;
var i: integer;
begin
clrscr;
writeln('Escreva um numero');
readln(i) ;
case i of
        1: writeln ('Numero 1');
        2,5,6:writeln ('Numero 2 ou numero 5 ou numero 6');
        7..10:writeln{'Numero entre 7 e 10');
        else writeln{'outro número');
end;
```

```
readln;
end.
```

A restrição da estrutura case é que o seletor só pode ser uma variável do tipo char, integer ou boolean.

Operadores Lógicos

Os principais operadores lógicos são: and , or e not. que significam *e*, *ou*, *não* e são usados para conjunção, disjunção e negação, respetivamente.

TABELA E	TABELA OU	TABELA NÃO
V e V = V	V ou V = V	NÃO V = F
V e F = F	V ou F = V	NÃO F = V
F e V = F	F ou V = V	
F e F = F	F ou F = F	

NOTA: Na linguagem PASCAL, quando existe mais de uma condição, elas devem estar entre parenteses.

Exemplos:

```
IF x = 3
THEN WRITELN ('Numero igual a 3');
```

No exemplo, existe apenas uma condição, logo, os parenteses são opcionais.

```
IF (X > 5) AND (X < 10)
THEN WRITELN ('Numero entre 5 e 10');
```

No exemplo anterior, existe mais de uma condição, logo, os parenteses são obrigatórios, ou seja, cada condição deve estar entre parenteses.

```
IF ((X = 5) AND (Y = 2)) OR (Y = 3)
THEN WRITELN ('X é igual a 5 e Y é igual a 2, ou Y é igual a 3');
```

No exemplo acima, existe mais de uma condição e mais de um tipo de operador logico, logo, alem dos parenteses de cada condição, devem existir ainda parenteses que indiquem a prioridade de execução das condições.

Neste exemplo, as condições com o operador and, ou seja, ((x = 5) and (Y = 2)), serão testadas, e o resultado será testado com a condição or (y = 3).

Aqui, as condições com o operador or, ou seja, ((y = 2) or (y = 3)), serão testadas, e o resultado será testado com a condição and (x = 5).

Exercícios Resolvidos

Exercício 1

A nota final de um estudante é calculada a partir de três notas atribuídas, respetivamente, a um trabalho de laboratório, a uma avaliação semestral e a um exame final. A média das três notas mencionadas obedece aos pesos a seguir:

Nota	Peso
Trabalho de laboratório	**2**
Avaliação semestral	**3**
Exame Final	**5**

Faça um programa que receba as três notas, calcule e mostre a media ponderada e o conceito que segue a tabela:

MÉDIA PONDERADA	CONCEITO
8,0 ------- 10,0	**A**
7,0 ------- 8,0	**B**
6,0 ------- 7,0	**C**
5,0 ------- 6,0	**D**
0,0 ------- 5,0	**E**

Solução Algoritmo

```
ALGORITMO
        DECLARE nota_trab, aval_sem/ exame, media NUMÉRICO
ESCREVA “Escreva a nota do trabalho de laboratório: “
LEIA nota_trab
ESCREVA “Escreva a nota da avaliação semestral: “
LEIA aval_sem
ESCREVA “Escreva a nota do exame final: “
LEIA exame
media <— (nota_trab * 2 + aval_sem * 3 + exame * 5) / 10
```

```
ESCREVA “Media ponderada: “ , media
SE media >= 8 E media <= 10
        ENTÃO ESCREVA “Obteve conceito A”
SE media >= 7 E media < 8
        ENTÃO ESCREVA “Obteve conceito B”
SE media >= 6 E media < 7
        ENTÃO ESCREVA “Obteve conceito C”
SE media >= 5 E media < 6
        ENTÃO ESCREVA “Obteve conceito D”
SE media >= 0 E media < 5
        ENTÃO ESCREVA “Obteve conceito E”
FIM ALGORITMO
```

Solução 1 PASCAL

```
PROGRAM EX1;
        USES CRT;
        VAR nota_trab, aval_sem, exame, media: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem antes da leitura da nota do laboratório}
        WRITE(‘Escreva a nota do trabalho em laboratório: ‘);
        {Recebe a nota do laboratório}
        READLN(nota_trab);
        {Mostra mensagem antes da leitura da nota semestral}
        WRITE(‘Escreva a nota da avaliação semestral: ‘);
        {Recebe a nota semestral}
        READLN(aval_sem);
        {Mostra mensagem antes da leitura da nota do exame}
        WRITE(‘ Escreva a nota do exame final: ‘);
        {Recebe a nota do exame}
        READLN(exame);
```


```
        {Calcula a média ponderada}
        media := (nota_trab * 2 + aval_sem * 3 + exame * 5) / 10;
        {Mostra a m,dia calculada com formata‡ao}
        WRITELN('Média ponderada: ',media:5:2);
        {Mostra o conceito de acordo com a m,dia}
        IF (media >=8) AND (media <=10)
                THEN WRITELN('Obteve conceito A');
        IF (media >=7) AND (media < 8)
                THEN WRITELN('Obteve conceito B');
        IF (media >= 6) AND (media < 7)
                THEN WRITELN('Obteve conceito C');
        IF (media >= 5) AND (media < 6)
                THEN WRITELN('Obteve conceito D');
        IF (media >= 0) AND (media < 5)
                THEN WRITELN('Obteve conceito E');
        {Para o programa à espera de um ENTER}
        READLN;
END.
```

Solução 2 PASCAL

```
PROGRAM EX1;
        USES CRT;
        VAR nota_trab, aval_sem, exame, media: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem antes da leitura da nota do laboratório}
        WRITE('Escreva a nota do trabalho em laboratório: ');
        {Recebe a nota do laboratório}
        READLN(nota_trab);
        {Mostra mensagem antes da leitura da nota semestral}
        WRITE('Escreva a nota da avaliação semestral: ');
```

```
{Recebe a nota semestral}
READLN(aval_sem);
{Mostra mensagem antes da leitura da nota do exame}
WRITE(' Escreva a nota do exame final: ');
{Recebe a nota do exame}
READLN(exame);
{Calcula a média ponderada}
media := (nota_trab * 2 + aval_sem * 3 + exame * 5) / 10;
{Mostra a média calculada com formatação}
WRITELN('Média ponderada: ',media:5:2);
{Mostra o conceito de acordo com a m,dia}
IF media >=8
        THEN WRITELN('Obteve conceito A')
        ELSE IF media >=7
                THEN WRITELN('Obteve conceito B')
                ELSE IF media >= 6
                        THEN WRITELN('Obteve conceito C')
                        ELSE IF media >= 5
                                THEN WRITELN('Obteve conceito D')
                                ELSE WRITELN('Obteve conceito E');
{Para o programa à espera de um ENTER}
READLN;
END.
```

Exercício 2

Faça um programa que receba três notas de um aluno, calcule e mostre a média aritmética e a mensagem constante na tabela a seguir. Aos alunos que ficaram para exame, calcule e mostre a nota que deverão tirar para serem aprovados, considerando que a média exigida é 6,0.

MÉDIA ARITMÉTICA	CONCEITO
0,0 ------- 3,0	**Reprovado**
3,0 ------- 7,0	**Exame**
7,0 ------- 10,0	**Aprovado**

Solução Algoritmo

```
ALGORITMO
DECLARE nota1, nota2, nota3, media, nota_exame NUMÉRICO
ESCREVA “Escreva a primeira nota: “
LEIA nota1
ESCREVA “Escreva a segunda nota: “
LEIA nota2
ESCREVA “Escreva a terceira nota: “
LEIA nota3
media ←(nota1 + nota2 + nota3) / 3
ESCREVA “Media aritmética: “, media
SE media >= 0 E media < 3
        ENTÃO ESCREVA “Reprovado”
SE media >= 3 E media < 7
        ENTÃO INICIO
                ESCREVA “Exame”
                nota_exame <— 12 - media;
                ESCREVA “Deve tirar nota”, nota_exame, “para ser aprovado”
        FIM
SE media >= 7 E media <= 10
        ENTÃO ESCREVA “Aprovado”
FIM ALGORITMO
```

Solução 1 PASCAL

```
PROGRAM EX2;
        USES CRT;
        VAR nota1, nota2, nota3, media, nota_exame: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem antes da leitura da primeira nota}
        WRITE(‘Escreva a primeira nota: ‘);
```

```
        {Recebe a primeira nota}
        READLN(nota1);
        {Mostra mensagem antes da leitura da segunda nota}
        WRITE(' Escreva a segunda nota: ');
        {Recebe a segunda nota}
        READLN(nota2);
        {Mostra mensagem antes da leitura da terceira nota}
        WRITE(' Escreva a terceira nota: ');
        {Recebe a terceira nota}
        READLN(nota3);
        {Calcula a média aritmética}
        media := (nota1 + nota2 + nota3) / 3;
        {Mostra a média aritmética}
        WRITELN('Média aritmética: ',media:5:2);
        {Mostra a situação}
        IF (media >=0) AND (media <3)
                THEN WRITELN('Reprovado');
        IF (media >=3) AND (media < 7)
                THEN BEGIN
                        WRITELN('Exame ');
                        nota_exame := 12 - media;
                        WRITELN('Deve tirar nota ',nota_exame:4:2,' para ser
                        aprovado');
        END;
        IF (media >= 7) AND (media < 10)
                THEN WRITELN('Aprovado');
        {Para o programa à espera de um ENTER}
        READLN;
END.
```


Solução 2 PASCAL

```
PROGRAM EX2;
        USES CRT;
        VAR nota1, nota2, nota3, media, nota_exame: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem antes da leitura da primeira nota}
        WRITE('Escreva primeira nota: ');
        {Recebe a primeira nota}
        READLN(nota1);
        {Mostra mensagem antes da leitura da segunda nota}
        WRITE(' Escreva a segunda nota: ');
        {Recebe a segunda nota}
        READLN(nota2);
        {Mostra mensagem antes da leitura da terceira nota}
        WRITE(' Escreva a terceira nota: ');
        {Recebe a terceira nota}
        READLN(nota3);
        {Calcula a média aritmética}
        media := (nota1 + nota2 + nota3) / 3;
        {Mostra a média aritmética}
        WRITELN('Média aritmética: ',media:5:2);
        {Mostra a situação}
        IF (media >=0) AND (media <3)
                THEN WRITELN('Reprovado')
                ELSE IF media < 7
                        THEN BEGIN
                        WRITELN('Exame ');
                        nota_exame := 12 - media;
                        WRITELN('Deve tirar nota ',nota_exame:4:2,' para ser
                        aprovado');
```

```
        END
        ELSE WRITELN('Aprovado');
{Para o programa à espera de um ENTER}
READLN;
END.
```

Exercício 3

Faça um programa que receba dois números e mostre o maior.

Solução Algoritmo

```
ALGORITMO
DECLARE num1, num2 NUMÉRICO
ESCREVA "Escreva o primeiro numero: "
LEIA num1
ESCREVA "Escreva o segundo numero: "
LEIA num2
SE num1 > num2
        ENTÃO ESCREVA "O maior número é: ", num1
SE num2 > num1
        ENTÃO ESCREVA "O maior número é: ", num2
SE num1 = num2
        ENTÃO ESCREVA "Os números são iguais "
FIM ALGORITMO.
```

Solução 1 PASCAL

```
PROGRAM EX3;
        USES CRT;
        VAR  num1, num2: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem de leitura do primeiro número}
        WRITE('Escreva o primeiro número: ');
```


```
        {Recebe o primeiro número}
        READLN(num1);
        {Mostra mensagem de leitura do segundo número}
        WRITE(' Escreva o segundo número: ');
        {Recebe o segundo número}
        READLN(num2);
        {Verifica qual o maior número}
        IF num1 > num2
                THEN WRITELN('O maior número : ',num1:5:2);
        IF num2 > num1
                THEN WRITELN('O maior número : ',num2:5:2);
        IF num1 = num2
                THEN WRITELN('Os números são iguais');
        {Para o programa à espera de um ENTER}
        READLN;
END.
```

Solução 2 PASCAL

```
PROGRAM EX3;
        USES CRT;
        VAR  num1, num2: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem de leitura do primeiro número}
        WRITE('Escreva o primeiro número: ');
        {Recebe o primeiro número}
        READLN(num1);
        {Mostra mensagem de leitura do segundo número}
        WRITE('Escreva o segundo número: ');
        {Recebe o segundo número}
        READLN(num2);
```

```
        {Verifica qual o maior número}
        IF num1 > num2
                THEN WRITELN('O maior número : ',num1:5:2)
                ELSE IF num2 > num1
                        THEN WRITELN('O maior numero : ',num2:5:2)
                        ELSE WRITELN('Os números são iguais');
        {Para o programa à espera de um ENTER}
        READLN;
END
```

Exercício 4

Faça um programa que receba três números e mostre-os em ordem crescente. Suponha que o utilizador escreve três números diferentes.

Solução Algoritmo

```
ALGORITMO
DECLARE num1, num2, num3 NUMÉRICO
ESCREVA "Escreva o primeiro número: "
LEIA num1
ESCREVA "Escreva o segundo número: "
LEIA num2
ESCREVA "Escreva o terceiro numero: "
LEIA num3
SE num1 < num2 E num1 < num3
        ENTÃO SE num2 < num3
                ENTÃO ESCREVA
                        ENTÃO ESCREVA "A ordem crescente é:", num1, "-",num2, "-", num3
                        SENÃO ESCREVA "A ordem crescente é:", num1, "-",num3, "-", num2
        SE num2 <num1 E num2 < num3
                ENTÃO SE num 1 < num3
ENTÃO ESCREVA "A ordem crescente é:", num2, "-",num1, "-", num3
SENVO ESCREVA "A ordem crescente é:", num2, "-",num3, "-", num1
        SE num3 <num1 E num3 < num2
```

```
                ENTÃO SE num 1 < num2
ENTÃO ESCREVA “A ordem crescente é:”, num3, “-”,num1, “-“, num2
SENÃO ESCREVA “A ordem crescente é:”, num3, “-”,num2, “-“, num1
FIM_ALGORITMO
```

Solução 1 PASCAL

```
PROGRAM EX4;
        USES CRT;
        VAR num1, num2, num3: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem solicitando o primeiro número}
        WRITE(‘Escreva primeiro número: ‘);
        {Recebe o primeiro número}
        READLN(num1);
        {Mostra mensagem solicitando o segundo número}
        WRITE(‘Escreva o segundo número: ‘);
        {Recebe o segundo número}
        READLN(num2);
        {Mostra mensagem solicitando o terceiro número}
        WRITE(‘ Escreva o terceiro número: ‘);
        {recebe o terceiro número}
        READLN(num3);
        {Mostra os números em ordem crescente}
        IF (num1 < num2) AND (num1 < num3)
                THEN BEGIN
                        IF num2 < num3
                                THEN WRITELN(‘A ordem crescente ,: ‘,num1:5:2,’ -
                        ‘,num2:5:2,’ - ‘,num3:5:2);
                        IF num3 < num2
                                THEN WRITELN(‘A ordem crescente ,: ‘,num1:5:2,’ -
```

```
                ',num3:5:2,' - ',num2:5:2);
        END;
        IF (num2 < num1) AND (num2 < num3)
                THEN BEGIN
                        IF num1 < num3
                                THEN WRITELN('A ordem crescente ,: ',num2:5:2,' -
                        ',num1:5:2,' - ',num3:5:2);
                        IF num3 < num1
                                THEN WRITELN('A ordem crescente ,: ',num2:5:2,' -
                        ',num3:5:2,' - ',num1:5:2);
        END;
        IF (num3 < num1) AND (num3 < num2)
                THEN BEGIN
                        IF num1 < num2
                                THEN WRITELN('A ordem crescente ,: ',num3:5:2,' -
                        ',num1:5:2,' - ',num2:5:2);
                        IF num2 < num1
                                THEN WRITELN('A ordem crescente ,: ',num3:5:2,' -
                        ',num2:5:2,' - ',num1:5:2)
        END;
        {Para o programa à espera de um ENTER}
        READLN;
END.
```

Solução 2 PASCAL

```
PROGRAM EX4;
        USES CRT;
        VAR num1, num2, num3: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem solicitando o primeiro número}
```


```
WRITE('Escreva o primeiro número: ');
{Recebe o primeiro número}
READLN(num1);
{Mostra mensagem solicitando o segundo número}
WRITE(' Escreva o segundo número: ');
{Recebe o segundo número}
READLN(num2);
{Mostra mensagem solicitando o terceiro número}
WRITE(' Escreva o terceiro número: ');
{recebe o terceiro número}
READLN(num3);
{Mostra os números em ordem crescente}
IF (num1 < num2) AND (num1 < num3)
        THEN BEGIN
                IF num2 < num3
                THEN WRITELN('A ordem crescente ,: ',num1:5:2,' -
        ',num2:5:2,' - ',num3:5:2)
                        ELSE WRITELN('A ordem crescente ,: ',num1:5:2,' -
                ',num3:5:2,' - ',num2:5:2);
                END
ELSE IF (num2 < num1) AND (num2 < num3)
        THEN BEGIN
                IF num1 < num3
                        THEN WRITELN('A ordem crescente ,: ',num2:5:2,' -
                ',num1:5:2,' - ',num3:5:2)
                        ELSE WRITELN('A ordem crescente ,: ',num2:5:2,' -
                ',num3:5:2,' - ',num1:5:2);
                END
        ELSE BEGIN
                IF num1 < num2
                        THEN WRITELN('A ordem crescente ,: ',num3:5:2,' -
                ',num1:5:2,' - ',num2:5:2)
```

```
                    ELSE WRITELN('A ordem crescente ,: ',num3:5:2,' -
            ',num2:5:2,' - ',num1:5:2);
            END;
    {Para o programa à espera de um ENTER}
    READLN;
END.
```

Exercício 5

Faça um programa que receba três números obrigatoriamente em ordem crescente e um quarto número que não siga essa regra. Mostre, em seguida, os quatro números em ordem decrescente. Suponha que o utilizador escreve quatro números diferentes.

Solução Algoritmo

```
ALGORITMO
DECLARE num1, num2, num3, num4 NUMERICO
ESCREVA "Escreva três números em ordem crescente:
LEIA num1
LEIA num2
LEIA num3
ESCREVA "Escreva um número (fora de ordem): "
LEIA num4
SE num4 > num3
ENTÃO ESCREVA "A ORDEM DECRESCENTE É: ",num4,"-",num3,"-",num2,"-",num1
SE num4 > num2 E num4 < num3
ENTÃO ESCREVA "A ORDEM DECRESCENTE É: ",num3,"-",num4,"-",num2,"-",num1
SE num4 > num1 E num4 < num2
ENTÃO ESCREVA "A ORDEM DECRESCENTE É: ",num3,"-",num2,"-",num4,"-",num1
SE num4 < num1
ENTÃO ESCREVA "A ORDEM DECRESCENTE É: ",num3,"-",num2,"-",num1,"-",num4
FIM_ALGORITMO
```

Solução 1 PASCAL

```
PROGRAM EX5;
```

```
        USES CRT;
        VAR num1, num2, num3, num4: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem solicitando os três números}
        WRITELN('Escreva três números em ordem crescente: ');
        {Recebe o primeiro número}
        READLN(num1);
        {Recebe o segundo número}
        READLN(num2);
        {Recebe o terceiro número}
        READLN(num3);
        {Mostra mensagem solicitando o quarto número}
        WRITELN('Escreva um número (fora de ordem): ');
        {Recebe o quarto número}
        READLN(num4);
        {Mostra os números em ordem}
        IF num4 > num3
                THEN WRITELN('A ordem decrescente ,: ', num4:5:2,' - ',num3:5:2,' -
        ',num2:5:2, ' - ',num1:5:2);
        IF (num4 > num2) AND (num4<num3)
                THEN WRITELN('A ordem decrescente ,: ', num3:5:2,' - ',num4:5:2,' -
        ',num2:5:2, ' - ',num1:5:2);
        IF (num4 > num1) AND (num4 < num2)
                THEN WRITELN('A ordem decrescente ,: ', num3:5:2,' - ',num2:5:2,' -
                ',num4:5:2, ' - ',num1:5:2);
        IF num4 < num1
                THEN WRITELN('A ordem decrescente ,: ', num3:5:2,' - ',num2:5:2,' -
                ',num1:5:2, ' -',num4:5:2);
                {Para o programa à espera de um ENTER}
        READLN;
```

```
END.
```

Solução 2 PASCAL

```
PROGRAM EX5;
        USES CRT;
        VAR num1, num2, num3, num4: REAL;
BEGIN
        {Limpa o ecrã}
        CLRSCR;
        {Mostra mensagem solicitando os três números}
        WRITELN('Escreva três números em ordem crescente: ');
        {Recebe o primeiro número}
        READLN(num1);
        {Recebe o segundo número}
        READLN(num2);
        {Recebe o terceiro número}
        READLN(num3);
        {Mostra mensagem solicitando o quarto número}
        WRITELN('Digite um número (fora de ordem): ');
        {Recebe o quarto número}
        READLN(num4);
        {Mostra os números em ordem}
        IF num4 > num3
                THEN WRITELN('A ordem decrescente ,: ', num4:5:2,' - ',num3:5:2,' -
        ',num2:5:2, ' - ',num1:5:2)
                ELSE IF (num4 > num2) AND (num4<num3)
                        THEN WRITELN('A ordem decrescente ,: ', num3:5:2,' -
                ',num4:5:2,' - ',num2:5:2, ' - ',num1:5:2)
                        ELSE IF (num4 > num1) AND (num4 < num2)
                                THEN WRITELN('A ordem decrescente ,: ', num3:5:2,' -
                                ',num2:5:2,' - ',num4:5:2, ' - ',num1:5:2)
                                ELSE WRITELN('A ordem decrescente ,: ', num3:5:2,' -
```


```
                ',num2:5:2,' - ',num1:5:2, ' -',num4:5:2);
    {Para o programa à espera de um ENTER}
    READLN;
END.
```

Exercícios Propostos

Exercício 1

Faça um programa que receba um numero inteiro e verifique se e par ou impar.

Exercício 2

Faça um programa que receba quatro valores: I, A, B e C. Desses valores, 1 é inteiro e positivo, A, B e C são reais. Escreva os números A, B e C obedecendo à tabela a seguir. Suponha que o valor escrito para I seja sempre um valor valido, ou seja, 1, 2 ou 3, e que os números escritos sejam diferentes uns dos outros.

VALOR DE I	FORMA DE ESCREVER
1	**A, B e C em ordem crescente**
2	**A, B e C em ordem decrescente**
3	**O maior fica entre os outros dois números**

Exercício 3

Faça um programa que mostre o menu de opções a seguir, receba a opção do utilizador e os dados necessários para executar cada operação.

Menu de opções:

1. Somar dois números.
2. Raiz quadrada de um número.

Escreva a opção desejada.

Exercício 4

Faça um programa que mostre a data e a hora do sistema nos seguintes formatos: DD/MM/AAAA – mês por extenso e hora: minuto.

Exercício 5

Faça um programa que determine a data cronologicamente maior entre duas datas fornecidas pelo utilizador. Cada data deve ser composta por três valores inteiros, em que o primeiro representa o dia, o segundo, o mês e o terceiro, o ano.

Exercício 6

Faça um programa que receba a hora do início de um jogo e a hora final (cada hora e composta por duas variáveis inteiras: hora e minuto). Calcule e mostre a duração do jogo (horas e minutos), sabendo-se que o tempo máximo de duração do jogo é de 24 horas e que ele pode iniciar-se num dia e terminar no dia seguinte.

Exercício 7

Faça um programa que receba o código correspondente ao cargo de um funcionário e o seu salário atual e mostre o cargo, o valor do aumento e o seu novo salário. Os cargos estão na tabela abaixo.

CÓDIGO	CARGO	PERCENTUAL
1	**Escriturário**	**50%**
2	**Secretário**	**35%**
3	**Caixa**	**20%**
4	**Gerente**	**10%**
5	**Diretor**	**Não tem aumento**

Exercício 8

Faça um programa que apresente o menu a seguir, permita ao utilizador escolher a opção desejada, receba os dados necessários para executar a operação e mostre o resultado. Verifique a possibilidade de opção invalida e não se preocupe com restrições, como salário negativo.

Menu de opções:

1. Imposto
2. Novo salário
3. Classificação

Escreva a opção desejada.

Na opção 1: receber o salário de um funcionário, calcular e mostrar o valor do imposto usando as regras a seguir:

SALÁRIO	PERCENTUAL DE IMPOSTO
Menor que US$ 500,00	**5%**
De US$ 500,00 a US$ 850,00	**10%**
Acima de US$ 850,00	**15%**

Na opção 2: receber o salário de um funcionário, calcular e mostrar o valor do novo salário, usando as regras a seguir:

SALÁRIO	AUMENTO
Maior que US$ 1500,00	**US$ 25,00**
De US$ 750,00 (Inclusive) a US$ 1500,00 (Inclusive)	**US$ 50,00**
De US$ 450,00 (Inclusive) a US$ 750,00	**US$ 75,00**
Menor que US$ 450,00	**US$ 100,00**

Na opção 3: receber o salário de um funcionário e mostrar a sua classificação usando a tabela a seguir:

SALÁRIO	CLASSIFICAÇÃO
Até US$ 700,00 (Inclusive)	**Mal remunerado**
Maiores que US$ 700,00	**Bem remunerado**

Exercício 9

SALÁRIO	BONIFICAÇÃO
Até US$ 500,00	5% Salário
Entre US$ 500,00 a US$ 1200,00	12% do salário
Acima de US$ 1200,00	Sem bonificação

Faça um programa que receba o salário de um funcionário, calcule e mostre o novo salário, acrescido da bonificação e de auxílio escola.

SALÁRIO	AUXÍLIO ESCOLA
Até US$ 600,00	US$ 150,00
Mais que US$ 600,00	US$ 100,00

Exercício 10

Faça um programa que receba o valor do salário mínimo, o número de horas trabalhadas, o número de dependentes do funcionário e a quantidade de horas extras trabalhadas. Calcule e mostre o salário a receber do funcionário de acordo com as regras a seguir:

- O valor da hora trabalhada é igual a 1/5 do salário mínimo.
- O salário do mês é igual ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada.
- Para cada dependente, acrescentar $ 32,00.
- Para cada hora extra trabalhada, calcular o valor da hora trabalhada acrescida de 50%.
- O salário bruto é igual ao salário do mês mais o valor dos dependentes, mais o valor das horas extras.
- Calcular o valor do imposto de renda retido na fonte de acordo com a tabela a seguir:

IRRF	SALÁRIO BRUTO
Isento	**Inferior a US$ 200,00**
10%	**De US$ 200,00 até US$ 500,00**
20%	**Superior a US$ 500,00**

- O salário líquido é igual ao salário bruto menos 1RRF.

SALÁRIO LÍQUIDO	GRATIFICAÇÃO
Até US$ 350,00	**US$ 100,00**
Superior a US$ 350,00	**US$ 50,00**

- A gratificação de acordo com a tabela a seguir:
- O salário a receber do funcionário é igual ao salário líquido mais a gratificação.

Exercício 11

Um supermercado deseja reajustar os preços dos seus produtos usando o seguinte critério: o produto poderá ter o seu preço aumentado ou diminuído. Para o preço ser alterado, o produto deve preencher pelo menos um dos requisitos a seguir:

VENDA MÉDIA MENSAL	PREÇO ATUAL	% DE AUMENTO	% DE DIMINUIÇÃO
< 500	**< US$ 30,00**	**10**	-
>= 500 E < 1200	**>= US$ 30,00 e < US$ 80,00**	**15**	-
>= 1200	**>= US$ 80,00**	-	**20**

Faca um programa que receba o preço atual e a venda média mensal do produto, calcule e mostre o novo preço.

Exercício 12

Faça um programa para resolver equações do 2º grau.

$ax^2 + bx + c = 0$

A variável *a* deve ser diferente de zero.

$\Delta = b^2 - 4 * a * c$

$\Delta < 0$ —> não existe raiz real

$\Delta = 0$ —> existe uma raiz real

x = (- b) / (2 * a)

$\Delta > 0$ —> existem duas raízes reais

x1 = (- b +) / (2 * a)

x2 = (- b -) / (2 * a)

Exercício 13

Dados três valores X, Y e Z, verifique se eles podem ser os comprimentos dos lados de um triângulo e, se forem, verifique se é um triângulo equilátero, isósceles ou escaleno. Se eles não formarem um triângulo, escreva uma mensagem. Considere que:

- O comprimento de cada lado de um triângulo é menor do que a soma dos outros dois lados.
- Chama-se equilátero o triângulo que tem três lados iguais.
- Denomina-se isósceles o triângulo que tem o comprimento de dois lados iguais.
- Recebe o nome de escaleno o triângulo que tem os três lados diferentes.

Exercício 14

Faça um programa que receba a altura e o peso de uma pessoa. De acordo com a tabela a seguir, verifique e mostre a classificação dessa pessoa.

ALTURA	PESO ATÉ 60	ENTRE 60 E 90 (INCLUSIVE)	ACIMA DE 90
Menores que 1,20	**A**	**D**	**G**
De 1,20 a 1,70	**B**	**E**	**H**
Maiores que 1,70	**C**	**F**	**I**

Exercício 25

Faça um programa que receba:

- O código de um produto comprado, supondo que a digitação do código do produto seja sempre valida, ou seja, um numero inteiro entre 1 e 10.
- O peso do produto em quilos.
- O código do pais de origem, supondo que a digitação do código seja sempre valida, ou seja, um numero inteiro entre 1 e 3.

TABELAS:

CÓDIGO DO PAÍS DE ORIGEM	IMPOSTO
1	**0%**
2	**15%**
3	**25%**

CÓDIGO DO PRODUTO	PREÇO POR GRAMA
1 a 4	**10**
5 a 7	**25**
8 a 10	**35**

Calcule e mostre:

- O peso do produto convertido em gramas.
- O preço total do produto comprado.
- O valor do imposto, sabendo-se que ele é cobrado sobre o preço total do produto comprado e depende do pais de origem.
- O valor total, preço total do produto mais imposto

Exercício 16

Faça um programa que receba:

- O código do estado de origem da carga de um camião, supondo que a digitação do código do estado seja sempre valida, ou seja, um numero inteiro entre 1 e 5.
- O peso da carga do camião em toneladas.
- O código da carga, supondo que a digitação do código seja sempre valida, ou seja, um numero inteiro entre 10 e 40.

Tabelas:

CÓDIGO DO ESTADO	IMPOSTO
1	**35%**
2	**25%**
3	**15%**
4	**5%**
5	**Isento**

CÓDIGO DA CARGA	PREÇO POR QUILO
10 a 20	**100**
21 a 30	**250**
31 a 40	**340**

Calcule e mostre:

- O peso da carga do camião convertido em quilos.
- O preço da carga do camião.
- O valor do imposto, sabendo que o imposto é cobrado sobre o preço da carga do caminhão e depende do estado de origem.
- O valor total transportado pelo camião, preço da carga mais imposto.

Exercício 17

Faça um programa que receba o salário base e o tempo de serviço de um funcionário.

Calcule e mostre:

- O imposto, apresentado na tabela a seguir.

SALÁRIO BASE	% SOBRE O SALÁRIO BASE
< US$ 200,00	**Isento**
Entre US$ 200,00 (Inclusive) e US$ 400,00 (Inclusive)	**3%**
Entre US$ 450,00 e US$ 700,00	**8%**
>= US$ 700,00	**12%**

- A gratificação, que se encontra na tabela abaixo.

SALÁRIO BASE	TEMPO DE SERVIÇO	GRATIFICAÇÃO
Superior a US$ 500,00	Até 3 anos	20
	Mais de 3 anos	30
Até US$ 500,00	Até 3 anos	23
	Entre 3 e 6 anos	35
	De 6 anos para cima	33

- O salário liquida, ou seja, salário base menos imposto mais gratificação.
- A categoria, que esta na tabela a seguir.

SALÁRIO LIQUIDO	CLASSIFICAÇÃO
Até US$ 350,00	**A**
Entre US$ 350,00 e US$ 600,00	**B**
De US$ 600,00 para cima	**C**

Exercício 18

Faça um programa que receba o valor do salário mínimo, o turno de trabalho (M - matutino, V - vespertino ou N - noturno), a categoria (O - operário, G - gerente) e o número de horas trabalhadas no mês de um funcionário. Suponha a digitação apenas de dados validos e, quando houver digitação de letras, utilize maiúsculas. Calcule e mostre:

- O coeficiente do salário, de acordo com a tabela a seguir.

TEMPO DE TRABALHO	VALOR DO COEFICIENTE
M – Matutino	**10% do salário mínimo**
V - Vesperino	**15% do salário mínimo**
N - Noturno	**12% do salário mínimo**

- O valor do salário bruto, ou seja, o número de horas trabalhadas multiplicado pelo valor do coeficiente do salário.

- O imposto, de acordo com a tabela a seguir.

CATEGORIA	SALÁRIO BRUTO	IMPOSTO SOBRE O SALÁRIO BRUTO
O – Operário	**>= US$ 300,00**	**5%**
	< US$ 300,00	**3%**
G - Gerente	**>= US$ 400,00**	**6%**
	< US$ 400,00	**4%**

Se o funcionário preencher todos os requisitos a seguir, a sua gratificação será de R$ 50,00; caso contrario, será de R$ 30,00. Os requisitos são:

Turno: Noturno

Número de horas trabalhadas: Superior a 80 horas

- A gratificação, de acordo com as regras que se seguem.

Se o funcionário preencher algum dos requisitos abaixo, o seu subsídio de alimentação será de um terço do seu salário bruto; caso contrario, será de metade do seu salário bruto. Os requisitos são:

Categoria: Operario

Coeficiente do salário: < = 25

- O subsídio de alimentação, de acordo com as seguintes regras.

O salário líquido, ou seja, salário bruto menos imposto mais gratificação mais subsídio de alimentação.

A classificação, de acordo com a tabela a seguir.

SALÁRIO LÍQUIDO	MENSAGEM
Menor que US$ 350,00	**Mal remunerado**
Entre US$ 350,00 e US$ 600,00	**Normal**
Maior que US$ 600,00	**Bem remunerado**

Exercício 19

Faça um programa que receba o preço, o tipo (A - alimentação, L - limpeza e V - vestuário) e a refrigeração (S - produto que precisa de refrigeração e N - produto que não precisa de refrigeração) de um produto. Suponha que haverá apenas a digitação de dados validos e, quando houver digitação de letras, utilize maiúsculas. Calcule e mostre:

- O valor adicional, de acordo com a tabela a seguir.

REFRIGERAÇÃO	TIPO	PREÇO	VALOR ADICIONAL
N	A	< US$ 15,00	US$ 2,00
		>= US$ 15,00	US$ 5,00
	L	< US$ 10,00	US$ 1,50
		>= US$ 10,00	US$ 2,50
	V	< US$ 30,00	US$ 3,00
		>= US$ 30,00	US$ 2,50
S	A		US$ 8,00
	L		US$ 0,00
	V		US$ 0,00

- O valor do imposto, de acordo com a regra a seguir.

PREÇO	PERCENTUAL SOBRE O PREÇO
< US$ 25,00	5%
>= US$ 25,00	8%

- O preço de custo, ou seja, preço mais imposto.

- O desconto, de acordo com a regra a seguir.

O produto que não preencher nenhum dos requisitos abaixo terá desconto de 3%, caso contrario, 0 (zero).

Os requisitos são:

Tipo: A

Refrigeração: S

- O novo preço, ou seja, preço de custo mais adicional menos desconto.

- A classificação, de acordo com a regra a seguir.

NOVO PREÇO	CLASSIFICAÇÃ0
< = US$ 50,00	Barato
Entre US$ 50,00 e US$ 100,00	Normal
>= US$ 100,00	Caro

Exercício 20

Faça um programa que receba a medida de um angulo em graus. Calcule e mostre o quadrante em que se encontra esse angulo. Considere os quadrantes da trigonometria e, para ângulos maiores que 360° ou menores que -360°, reduzi-los, mostrando também o número de voltas e o sentido da volta (horário ou anti-horário).

Estrutura de Repetição

Estrutura de repetição em algoritmos

Uma estrutura de repetição é utilizada quando uma "fração" do algoritmo ou até mesmo o algoritmo inteiro precisa de ser repetido. O numero de repetições pode ser fixo ou estar atrelado a uma condição. Assim, existem estruturas para tais situações, descritas a seguir.

Estrutura de repetição para número definido de repetições (estrutura PARA)

Essa estrutura de repetição é utilizada quando se sabe o numero de vezes que uma "fração" do algoritmo deve ser repetido. O formato geral dessa estrutura é:

PARA I ← valor inicial ATÉ valor final FAÇA [PASSO n]
INICIO
Comando1
comando2
…
comando_m
FIM

O comando1, o comando2 e o comando_m serão executados utilizando-se a variável i como controlo, e o seu conteúdo vai variar do valor inicial até o valor final. A informação do passo está entre parenteses retos porque é opcional. O passo indica como será a variação da variável de controlo. Por exemplo, quando for indicado PASSO 2, a variável de controlo será aumentada em 2 unidades a cada iteração até atingir o valor final.
Quando a informação do passo for suprimida, isso significa que o incremento ou o decremento da variável de controlo será de 1 unidade.
Quando houver apenas um comando a ser repetido, os marcadores de bloco INICIO e FIM poderão ser suprimidos.

Exemplos:

```
PARA I ← 1 ATE 10 FAÇA
ESCREVA I
```

O comando ESCREVA I será executado dez vezes, ou seja, para I a variar de 1 a 10. Assim, os valores de I serão: 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10.

```
PARA J ← 1 ATE 9 FAÇA PASSO 2
ESCREVA J
```

O comando ESCREVA J será executado cinco vezes, ou seja, para J a variar de 1 a 10, de 2 em 2. Assim, os valores de J serão: 1, 3, 5, 7 e 9.

```
PARA I ← 10 ATE 5 FAÇA
ESCREVA I
```

O comando ESCREVA I será executado seis vezes, ou seja, para I a variar de 10 a 5. Assim, os valores de I serão: 10, 9, 8, 7, 6 e 5.

```
PARA J ← 15 ATE 1 FAÇA PASSO -2
ESCREVA J
```

O comando ESCREVA J será executado oito vezes, ou seja, para J a variar de 15 a 1, de 2 em 2. Assim, os valores de J serão: 15, 13, 11, 9, 7, 5, 3 e 1.

Estrutura de repetição para número indefinido de repetições e teste no início (ESTRUTURA ENQUANTO)

Esta estrutura de repetição é utilizada quando não se sabe o número de vezes que uma fração do algoritmo deve ser repetido, embora também possa ser utilizada quando se conhece esse número.

Esta estrutura baseia-se na análise de uma condição. A repetição será feita enquanto a condição se mostrar verdadeira.

Existem situações em que o teste condicional da estrutura de repetição, que fica no início, resulta num valor falso logo na primeira comparação. Nesses casos, os comandos

de dentro da estrutura de repetição não serão executados.

ENQUANTO condição FAÇA

comando1

Enquanto a condição for verdadeira, o comando1 será executado.

ENQUANTO condição FAÇA

INICIO

comando1

comando2

comando3

FIM

Enquanto a condição for verdadeira, o comando1, o comando2 e o comando3 serão executados.

Exemplos:

x ← 1

Y ← 5

ENQUANTO X < Y FAÇA

INICIO

X ← X + 2

Y ← Y + 1

FIM

Simulação:

X	Y	
1	**5**	**Valores iniciais**
3	**6**	**Valores obtidos dentro da estrutura de repetição**
5	**7**	
7	**8**	
9	**9**	

Na fração do algoritmo anterior, portanto, os comandos que estão dentro da estrutura de repetição serão repetidos quatro vezes.

```
x ← 1
y ← 1
ENQUANTO X <= 5 FAÇA
INICIO
y ← y * x
x ← x + 1
FIM
```

Simulação:

X	Y	
1	**1**	**Valores iniciais**
1	**2**	**Valores obtidos dentro da estrutura de repetição**
2	**3**	
6	**4**	
24	**5**	
120	**6**	

Na fração do algoritmo anterior, portanto, os comandos que se localizam na estrutura de repetição serão repetidos cinco vezes. Nesse exemplo, a estrutura ENQUANTO é utilizada para repetir a fração do algoritmo um número definido de vezes.

Estrutura de repetição para número indefinido de repetições e teste no final (ESTRUTURA REPITA)

Essa estrutura de repetição e utilizada quando não se sabe o numero de vezes que uma fração do algoritmo deve ser repetido, embora também possa ser utilizada quando se conhece esse número.

Essa estrutura baseia-se na análise de uma condição. A repetição será feita até a condição se tornar verdadeira.

A diferença entre a estrutura ENQUANTO e a estrutura REPITA é que nesta última os comandos serão repetidos pelo menos uma vez, já que a condição de paragem se encontra no final.

REPITA

comandos

ATE condição

Repita os comandos até a condição se tornar verdadeira.

Exemplos:

x ← 1

Y ← 5

REPITA

x ← x + 2

y ← y + 1

ATE x >= y

Simulação:

X	Y	
1	**5**	**Valores iniciais**
3	**6**	**Valores obtidos dentro da estrutura de repetição**
5	**7**	
7	**8**	
9	**9**	

No excerto do algoritmo anterior, portanto, os comandos de dentro da estrutura de repetição serão repetidos quatro vezes.

x ← 1

Y ← 1

REPITA

y ← y * x

x ← x + 1

ATE x = 6

Simulação:

X	Y	
1	**1**	**Valores iniciais**
1	**2**	**Valores obtidos dentro da estrutura de repetição**
2	**3**	
6	**4**	
24	**5**	
120	**6**	

Na fração do algoritmo anterior, portanto, os comandos que se localizam dentro da estrutura de repetição serão repetidos cinco vezes. Nesse exemplo, a estrutura REPITA é utilizada para repetir a fração do algoritmo um número definido de vezes.

Estrutura de repetição em PASCAL

Estrutura de repetição "for"

Essa estrutura de repetição é utilizada quando se sabe o número de vezes que uma fração do programa deve ser repetida.

```
FOR I := valor inicial TO valor final DO
comando;
```

O comando será executado utilizando a variável I como controlo, o seu conteúdo vai variar do valor inicial até ao valor final, de 1 em 1, incrementando automaticamente.

```
FOR J := valor inicial TO valor final DO
BEGIN
comando1;
comando2;
END;
```

O comando1 e o comando2 serão executados utilizando a variável J como controlo, e o seu conteúdo vai variar do valor inicial até ao valor final, de 1 em 1, incrementando automaticamente.

```
FOR K := valor inicial DOWNTO valor final DO
comando;
```

O comando será executado utilizando a variável k como controlo, e o seu conteúdo vai variar do valor inicial até ao valor final, de 1 em 1, decrementando automaticamente.

```
FOR H := valor inicial DOWNTO valor final DO
BEGIN
comando1;
comando2;
comando3;
END;
```

O comando1, o comando2 e o comando3 serão executados utilizando a variável H como controlo, e o seu conteúdo vai variar do valor inicial até ao valor final, de 1 em 1, decrementando automaticamente.

Observação: Na linguagem PASCAL, a estrutura de repetição FOR funciona obrigatoriamente e automaticamente de 1 em 1, incrementando ou decrementando.

Exemplos:

```
FOR i := 1 TO 5 DO
WRITELN(i);
```

Na fração de programa acima, o comando WRITELN(i), será executado cinco vezes, ou seja, para i a valer 1, 2, 3, 4 e 5.

```
FOR i := 10 DOWNTO 1 DO
WRITELN(i);
```

No excerto de programa acima, o comando WRITELN(i), será executado dez vezes, ou seja, para i a valer 10, 9, 8, 7, 6, 5, 4, 3, 2 e 1.

Estrutura de repetição WHILE

A estrutura de repetição WHILE é utilizada quando o número de repetições necessárias não é fixo, apesar de também poder ser utilizada quando se conhece a quantidade de repetições.

Nesta estrutura, os comandos serão repetidos enquanto a condição for verdadeira e o teste condicional ocorre no início. Isto significa que existe a possibilidade da repetição não ser executada quando a condição assumir um valor falso logo na primeira verificação.

```
WHILE condição DO
comando;
```

Enquanto a condição for verdadeira, o comando será executado.

```
WHILE condição DO
BEGIN
comando1;
comando2;
END;
```

Enquanto a condição for verdadeira, o comando1 e o comando2 serão executados.

Exemplos:

```
X = 0;
WHILE X o 5 DO
BEGIN
WRITELN('Valor de X = ' ,X) ;
X = X + 1;
END;
WRITELN('Valor de X depois que sair da estrutura = 1 ,X);
```

Na fração do programa acima, os comandos WRITELN ('valor de x = ' ,x) e x = x + i; serão executados cinco vezes. O teste condicional avaliará x a valer 0, I, 2, 3,4 e 5.

Simulação:

TELA	X	
	0	**VALOR INICIAL**
Valor de X = 0	**1**	**Valores obtidos dentro da estrutura de repetição**
Valor de X = 1	**2**	**Valores obtidos dentro da estrutura de repetição**
Valor de X = 2	**3**	
Valor de X = 3	**4**	
Valor de X = 4	**5**	**Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição**
Valor de X depois que sair da estrutura = 5		

```
        X = 1;
Y = 10;
WHILE Y > X DO
BEGIN
WRITELN (' Valor de Y = ',Y );
Y = Y - 2;
END;
WRITELN ('Valor de Y depois que sair da estrutura = ' ,Y);
```

Na fração de programa acima, os comandos WRITELN ('Valor de y = ' y) ; e y = y - 2 ; serão executados cinco vezes. O teste condicional avaliara y a valer 10, 8, 6,4, 2 e 0.

Simulação:

TELA	X	Y	
	1	**10**	**VALORES INICIAIS**
Valor de Y = 10	**1**	**8**	**Valores obtidos dentro da estrutura de repetição**
Valor de Y = 8	**1**	**6**	
Valor de Y = 6	**1**	**4**	
Valor de Y = 4	**1**	**2**	
Valor de Y = 2	**1**	**0**	**Valor obtido dentro da estrutura de repetição, que torna a condição falsa e interrompe a repetição**
alor de Y depois que sair da estrutura = 0			

```
X = 1;
Y - 1;
WHILE X < Y DO
BEGIN
WRITELN('Valor de X = ',X);
X = X + 1;
END;
```

Na fração de programa acima, os comandos WRITELN ('valor de x = ' ,x); e x = x + 1; não serão executados, pois com os valores iniciais de x e y a condição é falsa, logo, não ocorrera a entrada na estrutura de repetição para execução de seus comandos.

Estrutura de repetição REPEAT

A estrutura de repetição REPEAT é utilizada quando o número de repetições necessárias não é fixo, apesar de também poder ser utilizada quando se conhece o número de repetições.

Nessa estrutura, os comandos serão repetidos até à condição se tornar verdadeira e o teste condicional ocorre no final, o que significa que a repetição será executada no mínimo uma vez.

```
REPEAT
comandos;
UNTIL condição;
```

Os comandos serão repetidos até que a condição se tome verdadeira.

Exemplos:

```
X = 0;
REPEAT
WRITELN ('Valor de X = 1,X);
X = X + 1;
UNTIL X = 5;
WRITELN ('Valor de X depois de sair da estrutura = ',X);
```

Na fração de programa acima, os comandos WRITELN (‘valor de x = ‘ , x); e x = x + 1; serão executados cinco vezes. O teste condicional avaliará x a valer 1, 2, 3, 4 e 5.

Simulação:

TELA	X	
	0	**VALOR INICIAL**
Valor de X = 0	**1**	**Valores obtidos dentro da estrutura de repetição**
Valor de X = 1	**2**	
Valor de X = 2	**3**	
Valor de X = 3	**4**	
Valor de X = 4	**5**	**Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição**
Valor de X depois que sair da estrutura = 5		

Na fração do algoritmo anterior, portanto, os comandos que se localizam na estrutura de repetição serão repetidos cinco vezes. Nesse exemplo, a estrutura REPITA está a ser utilizada para repetir a fração do algoritmo um número definido de vezes.

```
X = 1;
Y = 10;
REPEAT
WRITELN (‘Valor de Y = ‘Y);
Y = Y - 2;
UNTIL Y <= X;
WRITELN(‘Valor de Y depois que sair da estrutura =’ ,Y);
```

Na fração de programa acima, os comandos WRITELN (‘Valor de y =’ y) ; e y = y - 2 ; serão executados cinco vezes. O teste condicional avaliara y a valer 8, 6,4, 2 e 0.

Simulação:

TELA	X	Y	
	1	**10**	**VALORES INICIAIS**
Valor de Y = 10	**1**	**8**	**Valores obtidos dentro da estrutura de repetição**
Valor de Y = 8	**1**	**6**	
Valor de Y = 6	**1**	**4**	**Valores obtidos dentro da estrutura de repetição**
Valor de Y = 4	**1**	**2**	
Valor de Y = 2	**1**	**0**	**Valor obtido dentro da estrutura de repetição, que torna a condição verdadeira e interrompe a repetição**
Valor de Y depois que sair da estrutura = 0			

Exercícios Resolvidos

Exercício 1

Um funcionário de uma empresa recebe aumento salarial anualmente. Sabe-se que:

a. Esse funcionário foi contratado em 2005, com salário inicial de $ 1.000,00.
b. Em 2006, ele recebeu aumento de 1,5% sobre seu salário inicial.
c. A partir de 2007 (inclusive), os aumentos salariais sempre corresponderam ao dobro do percentual do ano anterior.

Faça um programa que determine o salário atual desse funcionário.

Solução Algoritmo

```
ALGORITMO
    DECLARE     i, ano_atual, salario NUMERICO
                novo_salario/ percentual NUMERICO
    LEIA ano_atual
    salario ← 1000
    percentual ← 1,5/100
    novo_salario ← salario + percentual * salario
```

```
        PARA i ← 2007 ATÉ ano_atual FAÇA
        INICIO
                percentual ← 2 * percentual
                novo_salario ← novo_salario + percentual * novo_salario
        FIM
        ESCREVA novo_salario
FIM ALGORITMO.
```

Solução 1 PASCAL

```
PROGRAM EX1;
        USES CRT;
        VAR     i, ano_atual: INTEGER;
                salario, novo_salario, percentual: REAL;
BEGIN
        CLRSCR;
        WRITELN('Escreva o ano atual');
        READLN(ano_atual);
        salario := 1000;
        percentual := 1.5/100;
        novo_salario := salario + percentual * salario;
        FOR i := 2007 TO ano_atual DO
                BEGIN
                        percentual := 2 * percentual;
                        novo_salario := novo_salario + percentual * novo_salario;
                END;
        WRITELN('Novo sal rio = ',novo_salario:5:2);
        READLN;
END.
```


Solução 2 PASCAL

```
PROGRAM EX1;
        USES CRT;
        VAR     i, ano_atual: INTEGER;
                salario, novo_salario, percentual: REAL;
BEGIN
        CLRSCR;
        WRITELN('Escreva o ano atual');
        READLN(ano_atual);
        salario := 1000;
        percentual := 1.5/100;
        novo_salario := salario + percentual * salario;
        i := 2007;
        WHILE i <= ano_atual DO
                BEGIN
                        percentual := 2 * percentual;
                        novo_salario := novo_salario + percentual * novo_salario;
                        i := i + 1;
                END;
        WRITELN('Novo sal rio = ',novo_salario:5:2);
        READLN;
END.
```

Exercício 2

Faça um programa que leia um valor N inteiro e positivo, calcule e mostre o valor de E, conforme a fórmula a seguir:

$$E = 1 + 1/1! + 1/2! + 1/3! + \ldots + 1/N!$$

Solução Algoritmo

```
ALGORITMO
        DECLARE n, e, i, j, fat NUMERICO
        LEIA n
```

```
        e ← 1
        PARA i ← 1 ATÉ n FAÇA
        INICIO
                fat ← 1
                PARA j ← 1 ATÉ i FAÇA
                        INICIO
                                fat ← fat * j
                        FIM
                e ← e + 1/fat
        FIM
ESCREVA e
FIM_ALG0RITMO.
```

Solução 1 PASCAL

```
PROGRAM EX2;
        USES CRT;
        VAR n, i, j: INTEGER;
        e, fat: real;
BEGIN
        CLRSCR;
        WRITELN('Escreva o valor de N');
        READLN(n);
        e := 1;
        FOR i := 1 TO n DO
                BEGIN
                        fat := 1;
                        FOR j := 1 TO i DO
                                BEGIN
                                        fat := fat * j;
                                END;
                        e := e + 1/fat;
                END;
```


```
        WRITELN('Valor de E = ',e:5:2);
        READLN;
END.
```

Solução 2 PASCAL

```
PROGRAM EX2;
        USES CRT;
        VAR n, i, j: INTEGER;
        e, fat:REAL;
BEGIN
        CLRSCR;
        WRITELN('Escreva o valor de N');
        READLN(n);
        e := 1;
        i := 1;
        REPEAT
                j := 1;
                fat := 1;
                        REPEAT
                                fat := fat * j;
                                j := j + 1;
                        UNTIL j > i;
                                 i := i + 1;
                                e := e + 1/fat;
        UNTIL i > n;
        WRITELN('Valor de E = ',e:5:2);
        READLN;
END.
```

Exercício 3

Faça um programa que leia um número N e que indique quantos valores inteiros e positivos devem ser lidos a seguir. Para cada número lido, mostre uma tabela contendo o valor lido e o fatorial desse valor.

Solução Algoritmo

```
ALGORITMO
        DECLARE n, num, i, j, fat NUMERICO
        LEIA n
        PARA i ← 1 ATÉ n FAÇA
                INICIO
                        LEIA num
                        fat ← 1
                        PARA j ← 1 ATÉ num FAÇA
                                INICIO
                                        Fat ← fat * j
                                FIM
                        ESCREVA fat
                FIM
FIM_ALG0RITM0.
```

Solução 1 PASCAL

```
ROGRAM EX3;
        USES CRT;
        VAR n, num, i, j:INTEGER;
        fat: REAL;
BEGIN
        CLRSCR;
        WRITELN('Escreva a quantidade de números que serão lidos');
        READLN(n);
        FOR i := 1 TO n DO
                BEGIN
```


```
                    WRITELN;
                    WRITELN('Digite o ', i,'º número');
                    READLN(num);
                    fat := 1;
                        FOR j := 1 TO num DO
                            BEGIN
                                fat := fat * j;
                            END;
                        WRITELN('Fatorial de ',num,' = ',fat:5:2);
                END;
        READLN;
END.
```

Solução 2 PASCAL

```
ROGRAM EX3;
        USES CRT;
        VAR n, num, i, j:INTEGER;
        fat: REAL;
BEGIN
        CLRSCR;
        WRITELN('Escreva a quantidade de números que serão lidos');
        READLN(n);
        i := 1;
        WHILE i <= n DO
                BEGIN
                    WRITELN;
                    WRITELN('Escreva o ', i,'º número');
                    READLN(num);
                    fat := 1;
                    j := 1;
                    WHILE j <= num DO
                        BEGIN
```

```
                    fat := fat * j;
                    j := j +1;
                END;
            WRITELN('Fatorial de ',num,' = ',fat:5:2);
        i := i + 1;
        END;
    READLN;
END.
```

Exercício 4

Foi feita uma estatística em cinco cidades para coletar dados sobre acidentes de trânsito. Foram obtidos os seguintes dados:

a. Código da cidade;
b. Número de veículos de passeio (em 2007);
c. Número de acidentes de transito com vitimas (em 2007).

Deseja-se saber:

a. qual o maior e o menor índice de acidentes de transito e a que cidades pertencem;
b. qual a media de veículos nas cinco cidades juntas;
c. qual a media de acidentes de transito nas cidades com menos de 2.000 veículos de passeio.

Solução Algoritmo

```
ALGORITMO
    DECLARE     cont, cod, num_vei, num_acid NUMERICO
                maior, cid_maior, menor, cid_menor NUMERICO
                media_vei, soma_vei, media_acid NUMERICO
                soma_acid, cont_acid NUMERICO
    soma_vei ← 0
    soma_acid ← 0
    cont_acid ← 0
```

```
PARA cont ← 1 ATÉ 5 FAÇA
        INICIO
                LEIA cod, num_vei, num_acid
                SE cont = 1
                ENTÃO INICIO
                        maior ← num_acid
                        cid_maior ← cod
                        menor ←num_acid
                        cid_menor ← cod
                FIM
                SENÃO INICIO
                        SE num_acid > maior
                        ENTÃO INICIO
                                maior ← num_acid
                                cid_maior ← cod
                        FIM
                                SE num_acid < menor
                                ENTÃO INICIO
                                        menor ← num_acid
                                        cid_menor ← cod
                                FIM
                FIM
                soma_vei ← soma_vei + num_vei
        SE num_vei < 2000
        ENTÃO INICIO
                soma_acid soma_acid + num_acid
                cont_acid ← cont_acid + 1
        FIM
        FIM
ESCREVA maior, cid_maior
ESCREVA menor, cid_menor
media_vei ← soma_vei/5
```

```
        ESCREVA media_vei
        SE cont_acid = 0
                ENTÃO ESCREVA "Não foi escrita nenhuma cidade com menos de
                                        2000 veiculos"
                SENAO INICIO
                        media_acid soma_acid/cont_acid
                        ESCREVA media_acid
                FIM
FIM ALGORITMO.
```

Solução 1 PASCAL

```
PROGRAM EX4;
        USES CRT;
        VAR     cont, cod, num_vei, num_acid, soma_vei: INTEGER;
                maior, cid_maior, menor, cid_menor, soma_acid, cont_acid: INTEGER;
                media_vei, media_acid: REAL;
BEGIN
        CLRSCR;
        soma_vei := 0;
        soma_acid := 0;
        cont_acid := 0;
        FOR cont := 1 TO 5 DO
                BEGIN
                        WRITELN('Escreva o código da ',cont,'¦ cidade');
                        READLN(cod);
                                WRITELN('Digite o número de veículos de passeio da
                                ',cont,'¦ cidade');
                                READLN(num_vei);
                                WRITELN('Digite o número de acidentes de transito da
                                ',cont,'¦ cidade');
                                READLN(num_acid);
                        IF cont = 1
```


```
                THEN BEGIN
                        maior := num_acid;
                        cid_maior := cod;
                        menor := num_acid;
                        cid_menor := cod;
                END
        ELSE BEGIN
                IF num_acid > maior
                        THEN BEGIN
                                maior := num_acid;
                                cid_maior := cod;
                        END;
                        IF num_acid < menor
                                THEN BEGIN
                                        menor := num_acid;
                                        cid_menor := cod;
                                END;
        END;
        soma_vei := soma_vei + num_vei;
        IF num_vei < 2000
                THEN BEGIN
                        soma_acid := soma_acid + num_acid;
                        cont_acid := cont_acid + 1;
                END;
        END;
    WRITELN('Maior número de acidentes = ', maior, ' na cidade de código =
',cid_maior);
    WRITELN('Menor número de acidentes = ', menor, ' na cidade de código =
',cid_menor);
    media_vei := soma_vei/5;
    WRITELN('Média de acidentes nas 5 cidades = ',media_vei:5:2);
    IF cont_acid = 0
```

```
            THEN WRITELN('Não foi escrita nenhuma cidade com menos de
            2000 veículos')
                    ELSE BEGIN
                            media_acid := soma_acid/cont_acid;
                            WRITELN('M,dia de acidentes nas cidades com menos
                            de 2000 veículos = ',media_acid:5:2);
                    END;
        READLN
END.
```

Solução 2 PASCAL

```
PROGRAM EX4;
        USES CRT;
        VAR     cont, cod, num_vei, num_acid, soma_vei: INTEGER;
                maior, cid_maior, menor, cid_menor, soma_acid, cont_acid: INTEGER;
                media_vei, media_acid: REAL;
BEGIN
        CLRSCR;
        soma_vei := 0;
        soma_acid := 0;
        cont_acid := 0;
        cont := 1;
        REPEAT
                WRITELN('Escreva o código da ',cont,'ª cidade');
                READLN(cod);
                        WRITELN('Escreva número de veículos de passeio da ',cont,'ª
                        cidade');
                        READLN(num_vei);
                         WRITELN('Escreva número de acidentes de trânsito da
                ',cont,'ª cidade');
                         READLN(num_acid);
                IF cont = 1
```


```
                THEN BEGIN
                        maior := num_acid;
                        cid_maior := cod;
                        menor := num_acid;
                        cid_menor := cod;
                END
                ELSE BEGIN
                        IF num_acid > maior
                                THEN BEGIN
                                        maior := num_acid;
                                        cid_maior := cod;
                                END;
                        IF num_acid < menor
                                THEN BEGIN
                                        menor := num_acid;
                                        cid_menor := cod;
                                END;
                END;
                soma_vei := soma_vei + num_vei;
                IF num_vei < 2000
                        THEN BEGIN
                                soma_acid := soma_acid + num_acid;
                                cont_acid := cont_acid + 1;
                        END;
                        cont := cont + 1;
        UNTIL cont = 6;
        WRITELN('Maior número de acidentes = ', maior, ' na cidade de código =
',cid_maior);
        WRITELN('Menor número de acidents = ', menor, ' na cidade de código =
',cid_menor);
        media_vei := soma_vei/5;
        WRITELN('M,dia de acidentes nas 5 cidades = ',media_vei:5:2);
```

```
        IF cont_acid = 0
                THEN WRITELN('Não foi escrita nenhuma cidade com menos de
                2000 veículos')
                ELSE BEGIN
                        media_acid := soma_acid/cont_acid;
                                WRITELN('M,dia de acidentes nas cidades com menos
                                de 2000 veículos = ',media_acid:5:2);
                END;
        READLN
END.
```

Exercício 5

Faça um programa que leia o numero de termos e um valor positivo para X, calcule e mostre o valor da serie a seguir:

$$S = \frac{-x^2}{1!} \frac{-x^3}{2!} \frac{-x^4}{3!} \frac{-x^5}{4!} \frac{-x^6}{3!} \frac{-x^7}{2!} \frac{-x^8}{1!} \frac{-x^9}{2!} \frac{-x^{10}}{3!} \frac{-x^{11}}{4!} - \dots$$

Solução Algoritmo

```
ALGORITMO
        DECLARE         fim, i, j, x, expoente, num_termos NUMERICO
                        den, denominador, fat, s NUMERICO
        LEIA num_termos, x
        s ← 0
        denominador <— 1
        PARA i ← 1 TO num_termos FAÇA
        INICIO
                fim ← denominador
                fat ← 1
                PARA j 1 ATÉ fim FAÇA
                        INICIO
                                fat ←  fat * j
```

```
                FIM
            expoente ← i + 1
            SE RESTO (expoente/ 2) = 0
            ENTÃO s← s - x expoente/fat
            SENÃO s ← s + x expoente /fat
            SE denominador = 4
            ENTÃO den ← 1
            SE denominador = 1
            ENTÃO den 1
            SE den = 1
            ENTÃO denominador ← denominador + 1
            SENAO denominador ← denominador - 1
        FIM
        ESCREVA s
FIM ALGORITMO.
```

Solução 1 PASCAL

```
PROGRAM EX5;
        USES CRT;
        VAR     fim,i,j,x,expoente,num_termos,den,denominador: INTEGER;
                fat,s: REAL;
BEGIN
        CLRSCR;
        WRITELN('Escreva o número de termos da sequência ');
        READLN(num_termos);
        WRITELN('Digite o valor de X');
        READLN(x);
        s := 0;
        denominador := 1;
        FOR i := 1 TO num_termos DO
                BEGIN
                        fim := denominador;
```

```
                fat := 1;
                FOR j := 1 TO fim DO
                        BEGIN
                                fat := fat * j;
                        END;
                expoente := i + 1;
                IF expoente MOD 2 = 0
                        THEN   s := s - (EXP((expoente) * LN(x))/fat)
                        ELSE   s := s + (EXP((expoente) * LN(x))/fat);
                IF denominador = 4
                        THEN  den := -1;
                IF denominador = 1
                        THEN den := 1;
                IF den = 1
                        THEN denominador := denominador + 1
                        ELSE denominador := denominador - 1;
        END;
    WRITELN('Valor de S = ',s:5:2);
    READLN;
END.
```

Solução 2 PASCAL

```
PROGRAM EX5;
    USES CRT;
    VAR     i, x, j, num_termos, fim, denominador: INTEGER;
            s: REAL;
BEGIN
    CLRSCR;
    WRITELN('Escreva o número de termos da sequência ');
    READLN(num_termos);
    WRITELN('Escreva o valor de X');
    READLN(x);
```


```
        s := 0;
        denominador := 1;
        i := 1;
                WHILE  i <= num_termos DO
                        BEGIN
                                IF denominador <> 1
                                        THEN BEGIN
                                                fim := denominador;
                                                denominador := 1;
                                                FOR j := 1 TO fim DO
                                                        BEGIN
                                                                denominador :=
                                                                denominador * j;
                                                        END;
                                        END;
                                IF i MOD 2 = 0
                                        THEN   s := s + (EXP((i+1) *
                                LN(x))/denominador)
                                        ELSE   s := s - (EXP((i+1) *
                                        LN(x))/denominador);
                                IF i = 4
                                        THEN  denominador := 1
                                        ELSE  denominador := denominador + 1;
                                        i := i + 1;
                        END;
        WRITELN('Valor de S = ',s:5:2);
        READLN;
END.
```

Exercícios Propostos

Exercício 1

Uma empresa possui dez funcionários com as seguintes características: código, número de horas trabalhadas no mês, turno de trabalho (M - matutino, V - vespertino ou N - noturno), categoria (O - operário ou G - gerente), valor da hora trabalhada. Sabendo-se que essa empresa deseja informatizar a sua folha de pagamento, faça um programa que:

a. Leia as informações dos funcionários, exceto o valor da hora trabalhada, não permitindo que sejam informados turnos nem categorias inexistentes. Trabalhe sempre com a escrita de letras maiúsculas.

b. Calcule o valor da hora trabalhada, conforme a tabela a seguir. Adote o valor de $ 450,00 para o salário mínimo.

CATEGORIA	TURNO	VALOR DA HORA TRABALHADA
G	**N**	**18% do salário mínimo**
G	**M ou V**	**15% do salário mínimo**
O	**N**	**13% do salário mínimo**
O	**M ou V**	**10% do salário mínimo**

c. Calcule o salário inicial dos funcionários com base no valor da hora trabalhada e no numero de horas trabalhadas.

d. Calcule o valor do subsidio de alimentação (aux) recebido por funcionário de acordo com seu salário inicial, conforme a tabela a seguir.

SALÁRIO INICIAL	AUXÍLIO – AUMENTO
Até US$ 300,00	**20% do salário inicial**
Entre US 300,00 e US$ 600,00	**15% do salário inicial**
Acima de US$ 600,00	**5% do salário inicial**

e. Mostre o código, número de horas trabalhadas, valor da hora trabalhada, salário inicial, subsidio de alimentação e salário final (salário inicial + subsídio de alimentação).

Exercício 2

Faça um programa que monte os oito primeiros termos da sequência de Fibonacci.

0 – 1 – 1 – 2 – 3 – 5 – 8 – 13 – 21 – 34 – 55 ...

Exercício 3

Faça um programa que leia o número de termos, determine e mostre os valores de acordo com a serie a seguir:

Série = 2, 7, 3, 4, 21, 12, 8, 63, 48, 16, 189, 192, 32, 567, 768, ...

Exercício 4

Faça um programa que receba duas notas de seis alunos, calcule e mostre:

- a média aritmética das duas notas de cada aluno;
- a mensagem que esta na tabela a seguir:

MÉDIA ARITMÉTICA	MENSAGEM
Até 3	**Reprovado**
Entre 3 e 7	**Exame**
De 7 para cima	**Aprovado**

- o total de alunos aprovados;
- o total de alunos de exame;
- o total de alunos reprovados;
- a média da classe.

Exercício 5

Num campeonato de futebol existem cinco equipas e cada um possui onze jogadores. Faça um programa que receba a idade, o peso e a altura de cada um dos jogadores, calcule e mostre:

- a quantidade de jogadores com idade inferior a 18 anos;
- a média das idades dos jogadores de cada time;
- a média das alturas de todos os jogadores do campeonato;

- a percentagem de jogadores com mais de 80 quilos entre todos os jogadores do campeonato.

Exercício 6

Faça um programa que receba um número inteiro maior que 1, verifique se o número fornecido é primo ou não e mostre mensagem de número primo ou de numero não primo. Um número é primo quando é divisível apenas por 1 e por ele mesmo.

Exercício 7

Numa fábrica trabalham homens e mulheres divididos em três classes:

- trabalhadores que fazem ate 30 peças por mês - classe 1;
- trabalhadores que fazem de 31 a 21 peças por mês - classe 2;
- trabalhadores que fazem mais de 21 peças por mês - classe 3.

A classe 1 recebe salário mínimo. A classe 2 recebe salário mínimo mais 3% deste salário por peça, acima das 30 pecas iniciais. A classe 3 recebe salário mínimo mais 5% deste salário por peça, acima das 30 peças iniciais.

Faca um programa que receba o número do operário, o número de peças fabricadas no mês, o sexo do operário, e que também calcule e mostre:

- o número do operário e o seu salário;
- o total da folha de pagamento da fábrica;
- o numero total de peças fabricadas no mês;
- a media de peças fabricadas pelos homens;
- a media de peças fabricadas pelas mulheres;
- o numero do operário ou operária de maior salário.

A fábrica possui 15 operários.

Exercício 8

Foi feita uma pesquisa para determinar o índice de mortalidade infantil em certo período.

Faça um programa que:

- leia o número de crianças nascidas no período;
- identifique o sexo (M ou F) e o tempo de vida de cada criança nascida.

O programa deve calcular e mostrar:

- a percentagem de crianças do sexo feminino mortas no período;
- a percentagem de crianças do sexo masculino mortas no período;
- a percentagem de crianças que viveram 24 meses ou menos no período.

Exercício 9

Faça um programa que receba o valor de uma divida e mostre uma tabela com os seguintes dados:

- Valor da divida, valor dos juros, quantidade de parcelas e valor da parcela.

Os juros e a quantidade de parcelas seguem a tabela:

QUANTIDADE DE PARCELAS	% DE JUROS SOBRE O VALOR INICIAL DA DÍVIDA
1	**0**
3	**10**
6	**15**
9	**20**
12	**25**

Exemplo de saída do programa:

VALOR DA DÍVIDA	VALOR DOS JUROS	QUANTIDADE DE PARCELAS	VALOR DA PARCELA
US$ 1000,00	0	1	US$ 1000,00
US$ 1100,00	100	3	US$ 366,67
US$ 1150,00	150	6	US$ 191,67

Exercício 10

Faça um programa que receba o preço unitário, a refrigeração (S para os produtos que precisão de refrigeração e N para os que não precisam) e a categoria (A - alimentação, L - limpeza e V - vestuário) de doze produtos, e que calcule e mostre:

- O custo de reposição de stock, calculado de acordo com a tabela a seguir.

PREÇO UNITÁRIO	REFRIGERAÇÃO	CATEGORIA	CUSTO DO STOCK
Até 20		**A**	**US$ 2,00**
		L	**US$ 3,00**
		V	**US$ 4,00**
Entre 20 e 50 (Inclusive)	**S**		**US$ 6,00**
	N		**US$ 0,00**
Maior que 50	**S**	**A**	**US$ 5,00**
		L	**US$ 2,00**
		V	**US$ 4,00**
	N	**A ou V**	**US$ 0,00**
		L	**US$ 1,00**

- O imposto calculado de acordo com as regras a seguir:

Se o produto não preencher nenhum dos requisitos abaixo, o seu imposto será de 2% sobre o preço unitário, caso contrário, será de 4%.

Os requisitos são: categoria - A e refrigeração - S.

- O preço final, ou seja, preço unitário mais custo de reposição de stock mais o imposto.
- A classificação calculada usando a tabela a seguir.

PREÇO FINAL	CLASSIFICAÇÃO
Até US$ 20,00	**Barato**
Entre US$ 20,00 e US$ 100,00	**Normal**
Acima de US$ 100,00	**Caro**

- A média dos valores adicionais, ou seja, a média dos custos de reposição de stock e dos impostos dos doze produtos.
- O maior preço final.
- O menor preço final.
- O total dos impostos.
- A quantidade de produtos com classificação barato.
- A quantidade de produtos com classificação caro.
- A quantidade de produtos com classificação normal.

Exercício 11

Faça um programa para calcular a área de um triângulo, que não permita a entrada de dados inválidos, ou seja, medidas menores ou iguais a 0.

Exercício 12

Faça um programa que receba o salário de um funcionário chamado Carlos. Sabe-se que outro funcionário, João, tem salário equivalente a um terço do salário de Carlos. Carlos aplicará o seu salário integralmente na caderneta de poupança, que está a render 2% ao mês, e o João aplicará seu salário integralmente no fundo de renda fixa, que está a render 5% ao mês. O programa deverá calcular e mostrar a quantidade de meses necessários para que o valor pertencente ao João iguale ou ultrapasse o valor pertencente a Carlos.

Exercício 13

Faça um programa que leia um conjunto não determinado de valores, um de cada vez, e escreva uma tabela com cabeçalho, que deve ser repetido a cada vinte linhas. A tabela deverá conter o valor lido, o seu quadrado, o seu cubo e a sua raiz quadrada. Finalize a entrada de dados com um valor negativo ou zero.

Exercício 14

Faça um programa que leia um número não determinado de pares de valores [m,n], todos inteiros e positivos, um par de cada vez, e que calcule e mostre a soma de todos os números inteiros entre m e n (inclusive).

A digitação de pares terminará quando m for maior ou igual a n.

Exercício 15

Faça um programa para ler o código, o sexo (M - masculino, F - feminino) e o número de horas/aulas dadas mensalmente pelos professores de uma universidade, sabendo-se que cada hora/aula vale $ 30,00. Emita uma listagem que contenha o código, o salário bruto e o salário líquido (tendo em consideração os descontos explicados a seguir) de todos os professores. Mostre também a média dos salários líquidos dos professores do sexo masculino e a média dos salários líquidos dos professores do sexo feminino.

Considere:

- desconto para homens, 10% e, para mulheres, 5%;
- as informações terminarão quando for lido o código = 99999.

Exercício 16

Faça um programa que receba vários números, calcule e mostre:

- a soma dos números escritos;
- a quantidade de números escritos;
- a media dos números escritos;
- o maior número escrito;
- o menor número escrito;
- a media dos números pares;
- a percentagem dos números impares entre todos os números escritos.

Finalize a entrada de dados com a escrita do número 30.000.

Exercício 17

Uma empresa decidiu fazer um levantamento em relação aos candidatos que se apresentarem para preenchimento de vagas nos seus quadros de funcionários. Supondo que você seja o programador dessa empresa, faça um programa que leia, para cada candidato, a idade, o sexo (M ou F) e a experiencia no serviço (S ou N).

Para encerrar a entrada de dados, escreva zero para a idade.

O programa também deve calcular e mostrar:

- o numero de candidatos do sexo feminino;
- o numero de candidatos do sexo masculino;
- a idade média dos homens que já têm experiencia no serviço;
- a percentagem dos homens com mais de 45 anos entre o total dos homens;
- o número de mulheres com idade inferior a 21 anos e com experiencia no serviço;
- a menor idade entre as mulheres que já tem experiencia no serviço.

Exercício 18

Faça um programa que receba o valor do salário mínimo, uma lista que contenha a quantidade de quilowatts gasta por consumidor e o tipo de consumidor (1 - residencial, 2 - comercial ou 3 - industrial) e que calcule e mostre:

- o valor de cada quilowatt, sabendo que o quilowatt custa um oitavo do salário mínimo;
- o valor a ser pago por cada consumidor (conta final mais acréscimo). O acréscimo encontra-se na tabela a seguir:

TIPO	% DE ACRÉSCIMO SOBRE O VALOR GASTO
1	**5**
2	**10**
3	**15**

- a faturação geral da empresa;
- a quantidade de consumidores que pagam entre $ 500,00 e $ 1.000,00.

Termine a entrada de dados com quantidade de quilowatts igual a zero.

Exercício 19

Faça um programa que represente o menu de opções a seguir, permita ao utilizador escolher a opção desejada, receba os dados necessários para executar a operação e mostre o resultado. Verifique a possibilidade de opção inválida e não se preocupe com restrições do tipo salário inválido.

Menu de opções:

1. Imposto
2. Novo salário
3. Classificação
4. Finalizar o programa

Escreva a opção desejada.

Na opção 1: receber o salário de um funcionário, calcular e mostrar o valor do imposto usando as regras a seguir.

SALÁRIO	% DE IMPOSTO
Menor que US$ 500,00	**5**
De US$ 500,00 a US$ 850,00	**10**
Acima de US$ 850,00	**15**

Na opção 2: receber o salário de um funcionário, calcular e mostrar o valor do novo salário usando as regras a seguir.

SALÁRIOS	AUMENTO
Maiores de US$ 1500,00	**US$ 25,00**
De US$ 750,00 (Inclusive) a US$ 1500,00 (Inclusive)	**US$ 50,00**
De US$ 450, 00 (Inclusive) a US$ 750,00	**US$ 75,00**
Menores que US$ 450,00	**US$ 100,00**

Na opção 3: receber o salário de um funcionário e mostrar a sua classificação usando esta tabela:

SALÁRIOS	CLASSIFICAÇÃO
Até US$ 700,00	**Mal remunerado**
Maiores que US$ 700,00	**Bem remunerado**

Exercício 20

Faça um programa que receba os dados a seguir de vários produtos: preço unitário, país de origem (1- Estados Unidos, 2 - México e 3 - outros), meio de transporte (T - terrestre, F - fluvial e A - aéreo), carga perigosa (S - sim, N - não), finalize a entrada de dados com um preço inválido, ou seja, menor ou igual a zero e que calcule e mostre:

- O valor do imposto, usando a tabela a seguir.

PREÇO UNITÁRIO	PERCENTUAL DE IMPOSTO SOBRE O PREÇO UNITÁRIO
Até US$ 100,00	**5%**
Maiores que US$ 100,00	**10%**

- O valor do transporte usando a tabela a seguir.

CARGA PERIGOSA	PAÍS DE ORIGEM	VALOR DO TRANSPORTE
S	**1**	**US$ 50,00**
	2	**US$ 21,00**
	3	**US$ 24,00**
N	**1**	**US$ 12,00**
	2	**US$ 21,00**
	3	**US$ 60,00**

- O valor do seguro, usando a regra a seguir.

Os produtos que vem do México e os produtos que utilizam transporte aéreo pagam metade do valor do seu preço unitário como seguro.

- O preço final, ou seja, preço unitário mais imposto mais valor do transporte mais valor do seguro.
- O total dos impostos.

Vetores

Vetor em Algoritmos

Definição de Vetor

Vetor também e conhecido como variável composta homogénea unidimensional. Isto quer dizer que se trata de um conjunto de variáveis do mesmo tipo, que possuem o mesmo identificador (nome) e são alocadas sequencialmente na memória. Como as variáveis tem o mesmo nome, o que as distingue é um índice que referencia a sua localização dentro da estrutura.

Declaração de Vetor

DECLARE nome[tamanho] tipo

Onde: nome é o nome da variável do tipo vetor; tamanho e a quantidade de variáveis que vão compor o vetor;
Tipo é o tipo básico dos dados que serão armazenados no vetor.

Exemplo de Vetor

DECLARE x[5] NUMÉRICO

X[1]

Atribuindo valores ao Vetor

As atribuições em vetor exigem que seja informada em qual das suas posições o valor ficará armazenado.

X[1] ← 45

No exemplo, o numero 45 será armazenado na posição de índice 1 do vetor.

X [4] ← 0

No exemplo, o numero 0 será armazenado na posição de índice 4 do vetor.

Preenchimento de um Vetor

Preencher um vetor significa atribuir valores a todas as suas posições. Assim, deve-se implementar um mecanismo que controle o valor do índice.

```
PARA i ← 1 ATE 5 FAÇA
INICIO
        ESCREVA "Escreva o ", i, "° número"
        LEIA X[i]
FIM
```

Neste exemplo, a estrutura de repetição PARA foi utilizada para garantir que a variável i assuma todos os valores possíveis para o índice do vetor. Assim, para cada execução da repetição, será utilizada uma posição diferente do vetor.

Simulação:

		MEMÓRIA	ECRA
I = 1	**X**	**95** | | | | **1 2 3 4 5**	**Escreva o primeiro número** **95**
I = 2	**X**	**95** | **13** | | | **1 2 3 4 5**	**Escreva o segundo número** **13**
I = 3	**X**	**95** | **13** | **-25** | | **1 2 3 4 5**	**Escreva o terceiro número** **-25**
I = 4	**X**	**95** | **13** | **-25** | **47** | **1 2 3 4 5**	**Escreva o quarto número** **47**
I = 5	**X**	**95** | **13** | **-25** | **47** | **0** **1 2 3 4 5**	**Escreva o quinto número** **0**

Apresentação dos elementos do Vetor

Mostrar os valores contidos num vetor também implica a utilização do índice.

```
PARA i ← 1 ATE 5 FAÇA
INICIO
        ESCREVA "Este é o ", i, "° numero do vetor"
        ESCREVA X[i]
FIM
```

Nesse exemplo, a estrutura de repetição PARA foi utilizada para garantir que a variável i assuma todos os valores possíveis para o índice do vetor. Assim, para cada execução da repetição, será utilizada uma posição diferente e, dessa forma, todos os valores do vetor serão apresentados.

Vetor em PASCAL

Definição de Vetor

As variáveis compostas homogéneas unidimensionais (vetores) são conhecidas na linguagem Pascal como ARRAY. Todas as posições do ARRAY possuem o mesmo identificador (mesmo nome) e são alocadas sequencialmente na memória.

Declaração de Vetor

VAR nome da variável: ARRAY [índice inicial ... índice final] OF tipo dos dados do vetor; onde:

Nome da variável é o nome da variável do tipo vetor;

Índice inicial é o número correspondente ao índice da primeira posição do vetor;

Índice final é o número correspondente ao índice da última posição do vetor;

Tipo dos dados do vetor é o tipo básico dos dados que serão armazenados no vetor.

Observação: O valor do índice inicial deve, obrigatoriamente, ser maior ou igual ao valor do índice final. As posições serão numeradas com valores inteiros dentro desse intervalo.

Exemplo 1:

VAR vetor1: ARRAY [1..10] OF INTEGER

Neste caso, o índice poderá assumir valores inteiros que vão de 1 ate 10.

Exemplo 2:

VAR vetor1: ARRAY [5..9] OF REAL

Neste caso, o índice poderá assumir valores inteiros que vão de 5 ate 9.

Exemplo de Vetor

VAR X : ARRAY [1..10] OF REAL;

X	10,5	20	13,1	14,65	87	1,2	35,6	78,2	15	65,9
	1	2	3	4	5	6	7	8	9	10

VAR VET: ARRAY [5..9] OF CHAR;

X	E	*	m	J	K
	5	6	7	8	9

Atribuindo valores ao Vetor

As atribuições em vetores exigem que seja informada em qual das suas posições o valor ficara armazenado.

X [4] := 5; atribui o valor 5 à posição do vetor cujo índice é 4

VET [3]: =’ F ‘; atribui a letra F à posição do vetor cujo índice é 3

Preenchimentos de um Vetor

Preencher um vetor significa atribuir valores a todas as suas posições. Assim, deve-se implementar um mecanismo que controle o valor do índice.

```
FOR i := 1 TO 7 DO
BEGIN
        READLN (X [i]);
END;
```

Neste exemplo, a estrutura de repetição FOR foi utilizada para garantir que a variável i assuma todos os valores possíveis para o índice do vetor (de 1 a 7). Assim, para cada execução da repetição, será utilizada uma posição diferente do vetor.

Apresentação dos elementos do vetor

Mostrar os valores contidos num vetor também exige a utilização do índice.

```
FOR i := 1 TO 10 DO
BEGIN
        WRITELN (X[i] ) ;
END;
```

Neste exemplo, a estrutura de repetição FOR foi utilizada para garantir que a variável i assuma todos os valores possíveis para o índice do vetor (de 1 a 10). Assim, para cada execução da repetição, será utilizada uma posição diferente do vetor e, dessa forma, todos os valores do vetor serão apresentados.

Exercícios Resolvidos

Exercício 1

Faça um programa que preencha um vetor com nove números inteiros, calcule e mostre os números primos e as suas respetivas posições.

Solução Algoritmo

```
ALGORITMO
        DECLARE     num[9] NUMERICO
                    i, j, cont NUMERICO
        PARA i ← 1 ATE 9 FAÇA
        INICIO
                LEIA num[i]
        FIM
        PARA i ← 1 ATE 9 FAÇA
        INICIO
                cont ← 0
                PARA j ← 1 ATE num[i] FAÇA
                        INICIO
                                SE RESTO(num[i] / j) = 0
                                ENTÃO cont ← cont + 1
                        FIM
                SE cont <= 2
                        ENTÃO INICIO
                                ESCREVA num[i]
                                ESCREVA i
                        FIM
        FIM
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX1;
        USES CRT;
        VAR     num: ARRAY[1..9] OF INTEGER;
                i, j, cont: INTEGER;
BEGIN
        CLRSCR;
        FOR i:=1 TO 9 DO
        BEGIN
                WRITELN('Escreva o ', i, 'º elemento do vetor ');
                READLN(num[i]);
        END;
        FOR i:=1 TO 9 DO
        BEGIN
                cont := 0;
                FOR j:=1 TO num[i] DO
                        BEGIN
                                IF  num[i] MOD j = 0
                                THEN cont := cont + 1;
                        END;
                IF cont = 2
                        THEN WRITELN ('O número', num[i],', primo e ocupa a
                        posição', i);
        END;
        READLN;
END.
```


Exercício 2

Faça um programa que receba a quantidade de pecas vendidas por vendedor e armazene essas quantidades num vetor. Receba também o preço da peça vendida de cada vendedor e armazene esses preços num outro vetor. Existem apenas dez vendedores e cada vendedor pode vender apenas um tipo de peça, isto é, para cada vendedor existe apenas um preço. Calcule e mostre a quantidade total de pecas vendidas por todos os vendedores e para cada vendedor calcule e mostre o valor total da venda, isto é, a quantidade de pecas * o preço da peca.

Solução Algoritmo

```
ALGORITMO
        DECLARE     qtd[10], preco[10] NUMERICO
                    i, tot_geral, tot_vend NUMERICO
        tot_geral ← 0
        PARA i ← 1 ATE 10 FAÇA
        INICIO
                LEIA qtd[i]
                LEIA preco[i]
        FIM
        PARA i ← 1 ATE 10 FAÇA
        INICIO
                tot_vend qtd[i] * preco [i]
                ESCREVA tot_vend
        FIM
        PARA i ← 1 ATE 10 FAÇA
        INICIO
                tot_geral tot_geral + qtd[i]
        FIM
        ESCREVA tot_geral
FIM_ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX2;
        USES CRT;
        VAR     qtd, preco: ARRAY[1..10] OF REAL;
                tot_geral, tot_vend: REAL;
                i: INTEGER;
BEGIN
        CLRSCR;
        tot_geral := 0;
        FOR i:=1 TO 10 DO
                BEGIN
                        WRITELN('Escreva a ', i, ' ¦ quantidade ');
                        READLN(qtd[i]);
                        WRITELN('Escreva o ', i, 'º preço ');
                        READLN(preco[i]);
                END;
        FOR i:=1 TO 10 DO
                BEGIN
                        tot_vend := qtd[i] * preco[i];
                        WRITELN('Total vendido pelo ', i, 'º vendedor ', tot_vend:5:2);
                END;
        FOR i:=1 TO 10 DO
                BEGIN
                        tot_geral := tot_geral + qtd[i];
                END;
        WRITELN('Total geral vendido ', tot_geral:5:2);
        READLN;
END.
```

Exercício 3

Faça um programa que preencha dois vetores de dez elementos numéricos cada um, e mostre o vetor resultante da intercalação deles.

Vetor 1

3	5	4	2	2	5	3	2	5	9
1	2	3	4	5	6	7	8	9	10

Vetor 2

7	15	20	0	18	4	55	23	8	6
1	2	3	4	5	6	7	8	9	10

Vetor resultante da Intercalação

3	7	5	15	4	20	2	0	2	18	5	4	3	55	2	23	5	8	9	6
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Solução Algoritmo

```
ALGORITMO
        DECLARE     vetl[10] , vet2[10], vet3[20] NUMERICO
                    i/ j NUMERICO
        j ← 1
        PARA i ← 1 ATE 10 FACA
        INICIO
                LEIA vet1[i]
                vet3 [ j] ← vet1 [i]
                j ← j + 1
                LEIA vet2[i]
                vet3 [ j]← vet2 [i]
                j ← j + 1
        FIM
        PARA i ← 1 ATE 20 FAÇA
        INICIO
                ESCREVA vet3[i]
        FIM
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX3;
        USES CRT;
        VAR     vet1, vet2: ARRAY[1..10] OF INTEGER;
                vet3:ARRAY[1..20] OF INTEGER;
                i, j: INTEGER;
BEGIN
        CLRSCR;
        j := 1;
        FOR I:=1 TO 10 DO
        BEGIN
                WRITELN('Escreva o ', i , 'º número do vetor 1 ');
                READLN(vet1[i]);
                vet3[j] := vet1[i];
                j := j + 1;
                WRITELN('Escreva o ', i, 'º número do vetor 2 ');
                READLN(vet2[i]);
                vet3[j] := vet2[i];
                j := j + 1;
        END;
        WRITELN('O vetor intercalado , ');
        FOR i:=1 TO 20 DO
        WRITE('  ',vet3[i]);
        READLN;
END.
```

Exercício 4

Faça um programa que preencha um vetor com oito números inteiros, calcule e mostre dois vetores resultantes. O primeiro vetor resultante deve conter os números positivos; o segundo deve conter os números negativos.

Cada vetor resultante vai ter, no máximo, oito posições, que poderão não ser completamente utilizadas.

Solução Algoritmo

```
ALGORITMO
        DECLARE     num[8], pos[8], neg[8] NUMERICO
                    cont, cont_n, cont_p, i NUMERICO
        cont_n ← 1
        cont_p ← 1
        PARA i ← 1 ATE 8 FAÇA
        INICIO
                LEIA num[i]
                SE num[i] > = 0
                ENTÃO INICIO
                        pos[cont_p] ← num[i]
                        cont_p ← cont_p + 1
                FIM
                SENÃO INICIO
                        neg[cont_n] ← num[i]
                        cont_n ← cont_n + 1
                FIM
        FIM
        SE cont_n = 1
                ENTÃO ESCREVA “Vetor de negativos vazio”
                SENÃO INICIO
                        PARA i ← 1 ATE cont_n - 1 FAÇA
                                INICIO
                                        ESCREVA neg[i]
                                FIM
                FIM
        SE cont_p = 1
                ENTÃO ESCREVA “Vetor de positivos vazio”
                SENÃO INICIO
                        PARA i ← 1 ATE cont_p - 1 FACA
                        INICIO
```

```
                    ESCREVA pos[i]
                FIM
            FIM
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX4;
    USES CRT;
    VAR   num, pos, neg: ARRAY[1..8] OF INTEGER;
          i, cont, cont_n, cont_p: INTEGER;
BEGIN
    CLRSCR;
    cont_n := 1;
    cont_p := 1;
    FOR i:=1 TO 8 DO
        BEGIN
            WRITE('Escreva o ', i, 'º elemento ');
            READLN(num[i]);
            IF num[i] >=0
            THEN BEGIN
                pos[cont_p] := num[i];
                cont_p := cont_p + 1;
            END
            ELSE BEGIN
                neg[cont_n] := num[i];
                cont_n := cont_n + 1;
            END;
        END;
    IF cont_n = 1
        THEN WRITELN('Vetor de negativos vazio')
        ELSE BEGIN
            WRITELN('Números negativos');
```


```
                FOR i:=1 TO (cont_n-1) DO
                WRITELN(neg[i]);
            END;
        IF cont_p = 1
            THEN WRITELN('Vetor de positivos vazio')
            ELSE BEGIN
                WRITELN('Números positivos');
                FOR  i:=1 TO (cont_p-1) DO
                WRITELN(pos[i]);
            END;
        READLN;
END.
```

Exercício 5

Faça um programa que preencha dois vetores, X e Y, com dez números inteiros cada. Calcule e mostre os seguintes vetores resultantes:

- A união de X com Y

  (todos os elementos de X e de Y sem repetições).

X	3	8	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10

Y	2	1	5	12	3	0	1	4	5	6
	1	2	3	4	5	6	7	8	9	10

UNIÃO	3	8	4	2	1	6	7	11	9	5	12	0
	1	2	3	4	5	6	7	8	9	10	11	12

- A diferença entre X e Y

(todos os elementos de X que não existam em Y, sem repetições).

X	3	8	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10

Y	2	1	5	12	3	0	1	4	5	6
	1	2	3	4	5	6	7	8	9	10

DIFERENÇA	8	7	11	9
	1	2	3	4

- A soma entre X e Y

(soma de cada elemento de X com o elemento de mesma posição em Y).

x	3	8	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10

Y	2	1	5	12	3	0	1	4	5	6
	1	2	3	4	5	6	7	8	9	10

SOMA	5	9	9	14	4	6	9	11	16	15
	1	2	3	4	5	6	7	8	9	10

- O produto entre X e Y

(multiplicação de cada elemento de X com o elemento de mesma posição em Y).

X	3	8	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10
Y	2	1	5	12	3	0	1	4	5	6
	1	2	3	4	5	6	7	8	9	10

PRODUTO	6	8	20	24	3	0	8	28	55	54
	1	2	3	4	5	6	7	8	9	10

- A interseção entre X e Y

  (apenas os elementos que aparecem nos dois vetores, sem repetições).

x	3	8	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10

Y	2	1	5	12	3	0	1	4	5	6
	1	2	3	4	5	6	7	8	9	10

Interseção	3	4	2	1	6
	1	2	3	4	5

Solução Algoritmo

```
ALGORITMO
    DECLARE     X [10], Y [10], U[20], D[10], S [10]# P [10]# IT[10] NUMERICO
                i, j, k, cont_u, cont_d, cont_i NUMERICO
    PARA i ← 1 ATE 10 FAÇA
    INICIO
        LEIA X [i]
        LEIA Y [i]
    FIM
    cont_u ← 1
    cont_d ← 1
    cont_i ← 1
    PARA i ← 1 ATE 10 FAÇA
    INICIO
        j← 1
        ENQUANTO (j < cont_u E  X[i]   U[j]) FAÇA
        INICIO
            j← j + 1
        FIM
        SE j >= cont_u
```

```
        ENTAO INICIO
                U[cont_u] ← X[i]
                cont_u ← cont_u + 1
        FIM
FIM
PARA i ← 1 ATE 10 FAÇA
        INICIO
                J← 1
                ENQUANTO (j < cont_u E Y[i]    U [j]) FAÇA
                INICIO
                        j ← j + 1
                FIM
                SE j >= cont_u
                        ENTAO INICIO
                                U[cont_u] ← Y [ i]
                                cont_u ← cont_u + 1
                        FIM
        FIM
PARA i ← 1 ATE cont_u - 1 FAÇA
        INICIO
                ESCREVA U[i]
        FIM
PARA i ← 1 ATE 10 FAÇA
        INICIO
                j ← 1
                ENQUANTO (X[i] * Y[j] E j <= 10) FAÇA
                INICIO
                        j ← j + 1
                FIM
                SE j > 10
                        ENTAO INICIO
                                k ← 1
```

```
                                ENQUANTO (k < cont_d E X[i]    D[k]) FAÇA
                                        INICIO
                                                k ← k + 1
                                        FIM
                        SE k >= cont_d
                                ENTAO INICIO
                                        D[cond_d] ← X [i]
                                        cont_d ← cont_d + 1
                                FIM
                FIM
FIM
PARA i ← 1 ATE cont_d - 1 FAÇA
INICIO
        ESCREVA (D[i])
FIM
PARA i ← 1 ATE 10 FAÇA
INICIO
        S [i] ← X [i] + Y [i]
        P[i] ← X[i] * Y [ i]
FIM
PARA i ← 1 ATE 10 FAÇA
INICIO
        ESCREVA S[i]
FIM
PARA i ← 1 ATE 10 FAÇA
INICIO
        ESCREVA P[i]
FIM
PARA i← 1 ATE 10 FAÇA
INICIO
        j ← 1
        ENQUANTO (j <= 10 E X[i] * Y [ j]) FACA
```

```
                INICIO
                        j ← j + 1
                FIM
        SE j <= 10
                ENTAO INICIO
                        k ← 1
                        ENQUANTO(k < cont_i E IT[k]   X[i])FAÇA
                        INICIO
                                k ← k + 1
                        FIM
        SE k >= cont_i
                ENTAO INICIO
                        IT[cont_i] ← X[i]
                        cont_i ← cont_i + 1
                FIM
        FIM
FIM
PARA i ← 1 ATE cont_i FAÇA
INICIO
        ESCREVA IT[i]
FIM
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX5;
        USES CRT;
        VAR     X: ARRAY[1..10] OF INTEGER;
                Y: ARRAY[1..10] OF INTEGER;
                U: ARRAY[1..20] OF INTEGER;
                D: ARRAY[1..10] OF INTEGER;
                S: ARRAY[1..10] OF INTEGER;
                P: ARRAY[1..10] OF INTEGER;
```

```
        IT: ARRAY[1..10] OF INTEGER;
        i, j, k, cont_u, cont_d, cont_i: INTEGER;
BEGIN
    CLRSCR;
    FOR i:=1 TO 10 DO
    BEGIN
        WRITELN('Escreva o ', i, 'º elemento do vetor X: ');
        READLN(X[i]);
        WRITELN('Escreva o ', i, 'º elemento do vetor Y: ');
        READLN(Y[i]);
    END;
    cont_u := 1;
    cont_d :=1;
    cont_i := 1;
    FOR i:=1 TO 10 DO
    BEGIN
        j := 1;
        WHILE (j < cont_u) AND (X[i] <> U[j]) DO
        BEGIN
            j:= j+1;
        END;
        IF (j >= cont_u) THEN
        BEGIN
            U[cont_u] := X[i];
            cont_u := cont_u + 1;
        END;
    END;
    FOR i:=1 TO 10 DO
    BEGIN
        j := 1;
        WHILE (j < cont_u) AND (Y[i] <> U[j]) DO
        BEGIN
```

```
                j := j + 1;
        END;
        IF (j >= cont_u) THEN
        BEGIN
                U[cont_u] := Y[i];
                cont_u := cont_u + 1;
        END;
END;
WRITELN('Vetor União ');
FOR i:=1 TO cont_u - 1 DO
WRITELN(U[i]);
READLN;
FOR i:=1 TO 10 DO
BEGIN
        j := 1;
        WHILE (j <= 10) AND (X[i] <> Y[j]) DO
        BEGIN
                j:= j+1;
        END;
        IF (j > 10) THEN
        BEGIN
                k :=1;
                WHILE (k < cont_d) AND (X[i] <> D[k]) DO
                BEGIN
                        k := k + 1;
                END;
                IF (k >= cont_d) THEN
                BEGIN
                        D[cont_d] := X[i];
                        cont_d := cont_d + 1;
                END;
        END;
```


```
END;
WRITELN('Vetor Diferença ');
FOR i:=1 TO cont_d - 1 DO
WRITELN(D[i]);
READLN;
FOR i:=1 TO 10 DO
BEGIN
        S[i] := X[i] + Y[i];
        P[i] := X[i] * Y[i];
END;

 WRITELN('Vetor Soma ');
FOR i:=1 TO 10 DO
WRITELN(S[i]);
READLN;
WRITELN('Vetor Produto ');
FOR i:=1 TO 10 DO
WRITELN(P[i]);
READLN;
FOR i:=1 TO 10 DO
BEGIN
        j := 1;
        WHILE (j <= 10) AND (X[i] <> Y[j]) DO
        BEGIN
                j := j + 1;
        END;
        IF (j <= 10) THEN
        BEGIN
                k := 1;
                WHILE (k < cont_i) AND (IT[k] <> X[i]) DO
                        BEGIN
                                k := k + 1;
```

```
                    END;
            IF (k >= cont_i) THEN
            BEGIN
                    IT[cont_i] := X[i];
                    cont_i := cont_i + 1;
            END;
        END;
    END;

        WRITELN('Vetor Interse‡ao ');
        FOR i:=1 TO cont_i - 1 DO
        WRITELN(IT[i]);
        READLN;
END.
```


Exercícios Propostos

Exercício 1

Faça um programa que preencha um vetor com dez números inteiros, calcule e mostre o vetor resultante de uma ordenação decrescente.

X	3	5	4	2	1	6	8	7	11	9
	1	2	3	4	5	6	7	8	9	10

Ordenado	11	9	8	7	6	5	4	3	2	1
	1	2	3	4	5	6	7	8	9	10

Exercício 2

Faça um programa que, no momento de preencher um vetor com oito números inteiros, já os armazene de forma crescente.

Exercício 3

Faça um programa que preencha dois vetores com cinco elementos numéricos cada e depois ordenados de maneira crescente. Devera ser criado um terceiro vetor com dez posições, composto pela junção dos elementos dos vetores anteriores, também ordenado de maneira crescente.

X	6	8	1	10	3
	1	2	3	4	5

X ordenado	1	3	6	8	10
	1	2	3	4	5

Y	20	0	7	2	5
	1	2	3	4	5

Y ordenado	0	2	5	7	20
	1	2	3	4	5

RESULTADO	0	1	2	3	5	6	7	8	10	20
	1	2	3	4	5	6	7	8	9	10

Exercício 4

Faça um programa que efetue reserva de passagens aéreas de uma companhia. O programa deverá ler informações sobre os voos (numero, origem e destino) e o número de lugares disponíveis para doze aviões (um vetor para cada um desses dados). Depois da leitura, o programa devera apresentar um menu com as seguintes opções:

- consultar
- efetuar reserva
- sair

Quando a opção escolhida for *Consultar*, deverá ser disponibilizado mais um menu com as seguintes opções:

- por numero do voo
- por origem
- por destino

Quando a opção escolhida for *Efetuar reserva*, devera ser perguntado o número do voo em que a pessoa deseja viajar. O programa devera dar as seguintes respostas:

- *reserva confirmada* - caso exista o voo e lugar disponível, dando baixa nos lugares disponíveis;
- *voo lotado* - caso não exista lugar disponível nesse voo;
- *voo inexistente* - caso o código do voo não exista.

A opção *Sair* é a única que permite encerrar a execução do programa. Sendo assim, apos cada operação de consulta ou reserva, o programa volta ao menu principal.

Exercício 5

Faça um programa para corrigir provas de escolha múltipla. Cada prova tem oito questões e cada questão vale um ponto. O primeiro conjunto de dados a ser lido é o nível da prova. Os outros dados são os números dos alunos e as respostas que deram as questões. Existem dez alunos matriculados. Calcule e mostre:

- o numero e a nota de cada aluno;
- a percentagem de aprovação, sabendo-se que a nota mínima é 6.

Exercício 6

Faça um programa que receba a temperatura média de cada mês do ano, armazenando num vetor. Calcule e mostre a maior e a menor temperatura do ano e em que mês ocorreu (mostrar o mês por extenso: 1 - janeiro, 2 - fevereiro...). Desconsidere empates.

Exercício 7

Faça um programa que preencha um vetor com os modelos de cinco carros (exemplos de modelos: BMW, MERCEDES, AUDI, etc.). Carregue outro vetor com o consumo desses carros, isto é, quantos quilómetros faz cada um deles com um litro de combustível, calcule e mostre:

- o modelo de carro mais económico;
- quantos litros de combustível cada um dos carros registados consome para percorrer uma distancia de 1.000 quilómetros.

Exercício 8

Faça um programa que preencha um vetor com dez números inteiros, calcule e mostre os números superiores a cinquenta e as suas respetivas posições. O programa devera mostrar mensagem se não existir nenhum número nessa condição.

Exercício 9

Faça um programa que preencha três vetores com cinco posições cada. O primeiro vetor recebera os nomes de cinco funcionários; o segundo e o terceiro vetor receberão, respetivamente, o salario e o tempo de serviço de cada um. Mostre um primeiro relatório apenas com os nomes dos funcionários que não terão aumento; mostre um segundo relatório apenas com os nomes e os novos salários dos funcionários que terão aumento. Sabe-se que os funcionários que terão direito ao aumento são aqueles que possuem tempo de serviço superior a cinco anos ou salario inferior a $ 400,00. Sabe-se ainda que, se o funcionário satisfaz as duas condições anteriores, tempo de serviço e salario, o aumento será de 35%; para o funcionário que satisfaz apenas a condição tempo de serviço, o aumento será de 25%; para aquele que satisfaz apenas a condição salario, o aumento será de 15%.

Exercício 10

Faça um programa que preencha um primeiro vetor com dez números inteiros e um segundo vetor com cinco números inteiros. O programa deverá mostrar uma lista dos números do primeiro vetor com os seus respetivos divisores armazenados no segundo vetor, bem como as suas posições.

Exemplo de saída do programa:

NUM	**5**	**12**	**4**	**7**	**10**	**3**	**2**	**6**	**23**	**16**
	1	2	3	4	5	6	7	8	9	10

DIVIS	**3**	**11**	**5**	**8**	**2**
	1	2	3	4	5

Numero 5

Divisível por 5 na posição 3

Numero 12

Divisível por 3 na posição 1

Divisível por 2 na posição 5

Numero 4

Divisível por 2 na posição 5

Numero 7

Não possui divisores no segundo vetor

Numero 10

Divisível por 5 na posição 3

Divisível por 2 na posição 5

Para saber se um numero é divisível por outro, deve-se testar o resto.

Exemplo: RESTO (5/5) = 0

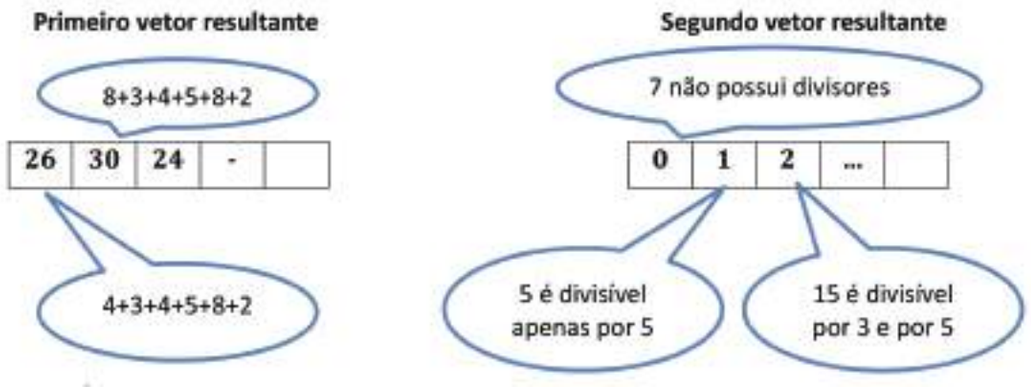
Exercício 11

Faça um programa que preencha um vetor com dez números inteiros e um segundo vetor com cinco números inteiros, calcule e mostre dois vetores resultantes. O primeiro vetor resultante será composto pelos números pares gerados pelo elemento do primeiro vetor somado a todos os elementos do segundo vetor; o segundo será composto pelos

números ímpares gerados pelo elemento do primeiro vetor somado a todos os elementos do segundo vetor.

Primeiro vetor	**5**	**12**	**4**	**7**	**10**	**3**	**2**	**6**	**23**	**16**
	1	2	3	4	5	6	7	8	9	10

Segundo vetor	**3**	**11**	**5**	**8**	**2**
	1	2	3	4	5



Exercício 12

Faça um programa que receba seis números inteiros e mostre:

- os números pares escritos;
- a soma dos números pares escritos;
- os números impares escritos;
- a quantidade de números impares escritos.

Vetor	**2**	**4**	**5**	**6**	**3**	**7**
	1	2	3	4	5	6

Relatório

Os números pares são:

Número 2 na posição 1

Número 4 na posição 2

Número 6 na posição 4

Soma dos pares = 12

Os números ímpares são:

Número 5 na posição 3

Número 3 na posição 5

Número 7 na posição 6

Quantidade de impares = 3

Exercício 13

Faça um programa que receba o número sorteado por um dado em vinte jogadas, mostre os números sorteados e a frequência com que apareceram.

Exercício 14

Faça um programa que preencha dois vetores, A e B, com vinte caracteres cada. A seguir, troque o 1º elemento de A com o 20º de B, o 2º de A com o 19º de B, e assim por diante, ate trocar o 20º de A com o 1º de B. Mostre os vetores antes e depois da troca.

Vetor 1 – Antes da troca

A	G	Y	W	S	V	S	8	6	J	G	A	W	2	M	C	H	Q	6	L
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Vetor 2 – Antes da troca

S	D	4	S	H	G	R	U	8	9	K	S	A	1	2	V	4	D	S	M
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Vetor 1 – Depois da troca

M	5	D	4	V	2	1	A	S	K	9	8	U	R	G	H	5	4	D	S
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Vetor 2 – Depois da troca

L	6	Q	H	C	M	2	W	A	G	J	6	8	S	V	S	W	Y	G	A
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Exercício 15

Faça um programa que leia um vetor com cinco posições para números reais e, depois, um código inteiro. Se o código for zero, finalize o programa; se for 1, mostre o vetor na ordem direta; se for 2, mostre o vetor na ordem inversa.

Exercício 16

Faça um programa que leia um conjunto de quinze valores e armazene-os Num vetor. A seguir, separe-os em dois outros vetores (P e I) com cinco posições cada. O vetor P armazena números pares e o vetor I, números ímpares. Como o tamanho dos vetores pode não ser suficiente para armazenar todos os números, deve-se sempre verificar se já estão cheios. Caso P ou I estejam cheios, deve-se mostra-los e recomeçar o preenchimento da primeira posição. Terminado o processamento, mostre o conteúdo restante dentro dos vetores P e I.

Exercício 17

Faça um programa que simule um controle bancário. Para tanto, devem ser lidos os códigos de dez contas e os seus respetivos saldos. Os códigos devem ser armazenados num vetor de números inteiros (não pode haver mais de uma conta com o mesmo código) e os saldos devem ser armazenados num vetor de números reais. O saldo devera ser cadastrado na mesma posição do código. Por exemplo, se a conta 504 foi armazenada na quinta posição do vetor de códigos, o seu saldo devera ficar na quinta posição do vetor de saldos. Depois de fazer a leitura dos valores, devera aparecer o seguinte menu no ecrã:

1. Efetuar depósito
2. Efetuar saque
3. Consultar o ativo bancário (ou seja, o somatório dos saldos de todos os clientes)
4. Finalizar o programa

- para efetuar deposito, deve-se solicitar o código da conta e o valor a ser depositado. Se a conta não estiver registada, deverá aparecer a mensagem *Conta não encontrada* e voltar ao menu. Se a conta existir, atualizar o seu saldo;
- para efetuar levantamentos, deve-se solicitar o código da conta e o valor a ser levantado. Se a conta não estiver registada, devera aparecer a mensagem *Conta*

não encontrada e voltar ao menu. Se a conta existir, verificar se o seu saldo é suficiente para cobrir o levantamento. (Estamos a supor que a conta não pode ficar com o saldo negativo.) Se o saldo for suficiente, realizar o levantamento e voltar ao menu. Caso contrário, mostrar a mensagem *Saldo insuficiente* e voltar ao menu;

- para consultar o ativo bancário, deve-se somar o saldo de todas as contas do banco. Depois de mostrar esse valor, voltar ao menu;
- o programa só termina quando for escrita a opção 4 - *Finalizar o programa.*

Exercício 18

Uma empresa possui Microlets com 48 lugares (24 nas janelas e 24 no corredor). Faça um programa que utilize dois vetores para controlar as poltronas ocupadas no corredor e na janela. Considere que 0 representa uma poltrona desocupada e 1, poltrona ocupada.

Janela	**0**	**1**	**0**	**0**	**...**	**1**	**0**	**0**
	1	2	3	4	...	22	23	24

Corredor	**0**	**0**	**0**	**1**	**...**	**1**	**0**	**0**
	1	2	3	4	...	22	23	24

Inicialmente, todas as cadeiras estão livres. Depois disso, o programa devera apresentar as seguintes opções:

- Vender passagem.
- Mostrar mapa de ocupação da microlet.
- Encerrar.

Quando a opção escolhida for Vender Passagem, devera ser perguntado se o utilizador deseja janela ou corredor e o número da cadeira. O programa devera, então, dar uma das seguintes mensagens:

- Venda efetuada - se a cadeira solicitada estiver livre, marcando-a como ocupada.
- Cadeira ocupada - se a cadeira solicitada não estiver disponível para venda.
- Microlet lotada - quando todas as cadeiras estiverem ocupadas.

Quando a opção escolhida for Mostrar Mapa de Ocupação da Microlet, devera ser mostrada uma listagem conforme a seguir:

JANELA	CORREDOR
1- Ocupada	1- Ocupada
2- Ocupada	2- Livre
3- Livre	3- Livre
4- Livre	4- Ocupada
5- Ocupada	5- Livre

Quando for escolhida a opção Encerrar, a execução do programa devera ser finalizada.

Exercício 19

Faça um programa que leia um vetor A de dez posições contendo números inteiros. Determine e mostre, a seguir, quais elementos de A estão repetidos e quantas vezes cada um se repete.

Exemplo

Vetor A	**5**	**4**	**3**	**18**	**5**	**3**	**4**	**18**	**4**	**18**
	1	2	3	4	5	6	7	8	9	10

Caso sejam escritos valores como os apresentados no vetor A, o programa devera mostrar no final as seguintes informações:

- O número 5 aparece duas vezes,
- O número 4 aparece três vezes,
- O número 3 aparece duas vezes,
- O número 18 aparece três vezes.

Exercício 20

Faça um programa que gere os dez primeiros números primos acima de 100 e os armazene num vetor. Escreva no final o vetor resultante.

Matriz

Matriz em Algoritmos

Definição de Matriz

Uma matriz é uma variável composta homogénea multidimensional. É formada por uma sequencia de variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome), e alocadas sequencialmente na memoria.

Uma vez que as variáveis têm o mesmo nome, o que as distingue são índices que referenciam a sua localização dentro da estrutura. Uma variável do tipo matriz precisa de um índice para cada uma das suas dimensões.

Declaração de Matriz

Um algoritmo pode declarar uma matriz, conforme descrito a seguir.

DECLARE nome [dimensaol, dimensao2, dimensao3, ..., dimensaoN] tipo

onde: nome é o nome da variável do tipo matriz;

dimensão1 é a quantidade de elementos da 1ª dimensão (muitas vezes chamada de linha);

dimensão2 é a quantidade de elementos da 2ª dimensão (muitas vezes chamada de coluna);

dimensão3 é a quantidade de elementos da 3ª dimensão (muitas vezes chamada de profundidade);

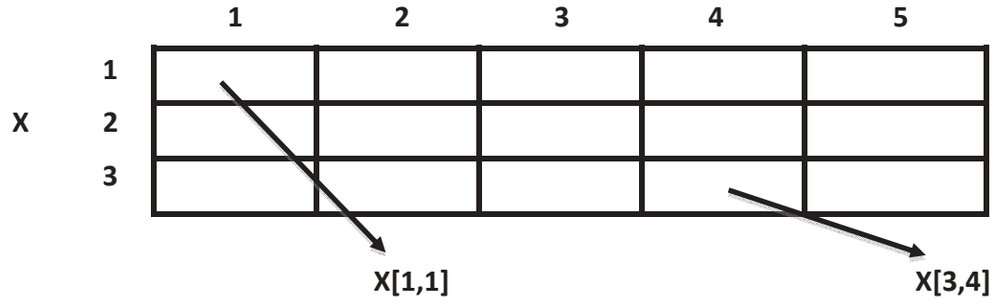
dimensãoN é a quantidade de elementos da n-exima dimensão;

tipo é o tipo de dados dos elementos da matriz.

Exemplo de Matriz

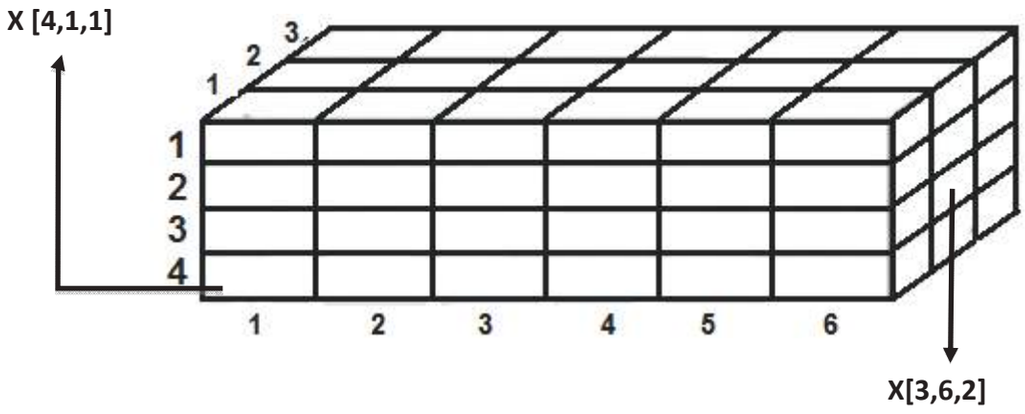
O exemplo a seguir define uma matriz bidimensional, onde o tamanho da Iª dimensão (linha) é 3 e o da 2ª dimensão (coluna) é 5.

DECLARE X [3,5] NUMERICO



O exemplo que se segue define uma matriz tridimensional, onde o tamanho da Iª dimensão (linha) é 4, o tamanho da 2ª dimensão (coluna) é 6 e o tamanho da 3ª dimensão (profundidade) é 3.

DECLARE X [4/6/3] NUMERICO



Atribuindo valores a uma Matriz

Cada elemento de uma matriz pode armazenar um valor. Para fazer este armazenamento, é necessário executar uma atribuição, a informar o número das dimensões.

X [2, 4] ← 45 X[3,1] ← 13

X [4 , 2,1] ← 0 X[3,5,3] ← -4

Preenchimento de uma Matriz

Para preencher uma matriz, é necessário identificar todas as suas posições. Isto exige a utilização de um índice para cada dimensão da matriz.

No exemplo a seguir, uma matriz bidimensional com três linhas e cinco colunas é mostrada. Observe que a variável i varia dentro do intervalo de 1 a 3, ou seja, exatamente nas linhas. Para cada valor de i, a variável j varia de 1 a 5, ou seja, as cinco colunas que cada linha possui.

```
PARA i ← 1 ATE 3 FAÇA
INICIO
        PARA j ← 1 ATE 5 FAÇA
        INICIO
                ESCREVA "Escreva o numero da linha ",i, " e coluna: ", j
                LEIA X [i,j]
        FIM
FIM
```

Simulação

MEMÓRIA		ECRA
i	**J**	
1	**1**	**Escreva o número da linha 1 e coluna 1 :12**
	2	**Escreva o número da linha 1 e coluna 2 :9**
	3	**Escreva o número da linha 1 e coluna 3 :3**
	4	**Escreva o número da linha 1 e coluna 4 :7**
	5	**Escreva o número da linha 1 e coluna 5 :-23**
2	**1**	**Escreva o número da linha 2 e coluna 1 :15**
	2	**Escreva o número da linha 2 e coluna 2 :4**
	3	**Escreva o número da linha 2 e coluna 3 :2**
	4	**Escreva o número da linha 2 e coluna 4 :34**
	5	**Escreva o número da linha 2 e coluna 5 :-4**

3	**1**	**Escreva o número da linha 3 e coluna 1 :3**
	2	**Escreva o número da linha 3 e coluna 2 :45**
	3	**Escreva o número da linha 3 e coluna 3 :3**
	4	**Escreva o número da linha 3 e coluna 4 :0**
	5	**Escreva o número da linha 3 e coluna 5 :-3**

Assim, podemos imaginar os elementos dispostos numa estrutura bidimensional, como uma tabela.

		1	**2**	**3**	**4**	**5**
	1	12	9	3	7	-23
X	2	15	4	2	34	-4
	3	3	45	3	0	-3

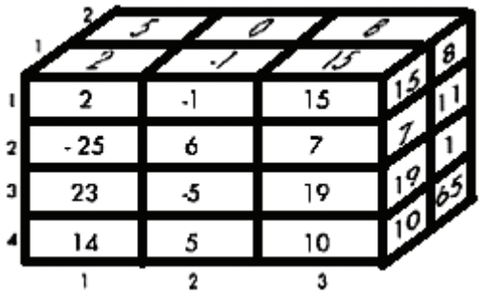
Já no exemplo que se segue, uma matriz tridimensional com quatro linhas, três colunas e profundidade dois é preenchida. Observe que a variável i oscila dentro do intervalo de 1 a 4, ou seja, exatamente nas linhas. Para cada valor de i, a variável j movimenta-se de 1 a 3, ou seja, as três colunas que cada linha possui, e, por fim, a variável k se alterna entre 1 e 2, que é a profundidade.

```
PARA i ← 1 ATE 4 FAÇA
INICIO
        PARA j ← 1 ATE 3 FAÇA
        INICIO
                PARA k ← 1 ATE 2 FAÇA
                INICIO
                        ESCREVA "Escreva o numero da linha ",i, " coluna ", j,
                        " e profundidade: ", k
                        LEIA X [i, j, k]
                FIM
        FIM
FIM
```

Simulação

MEMÓRIA			ECRA
i	**J**	**K**	
1	**1**	**1**	**Escreva o número da linha 1 e coluna 1 : 2**
		2	**Escreva o número da linha 1 e coluna 2 : 5**
	2	**1**	**Escreva o número da linha 2 e coluna 1 : -1**
		2	**Escreva o número da linha 2 e coluna 2 : 0**
	3	**1**	**Escreva o número da linha 3 e coluna 1 : -15**
		2	**Escreva o número da linha 3 e coluna 2 : 8**
2	**1**	**1**	**Escreva o número da linha 1 e coluna 1 : -25**
		2	**Escreva o número da linha 1 e coluna 2 : 3**
	2	**1**	**Escreva o número da linha 2 e coluna 1 : 6**
		2	**Escreva o número da linha 2 e coluna 2 : 9**
	3	**1**	**Escreva o número da linha 3 e coluna 1 : 7**
		2	**Escreva o número da linha 3 e coluna 2 : 11**
3	**1**	**1**	**Escreva o número da linha 1 e coluna 1 : 23**
		2	**Escreva o número da linha 1 e coluna 2 : -2**
	2	**1**	**Escreva o número da linha 2 e coluna 1 : -5**
		2	**Escreva o número da linha 2 e coluna 2 : 46**
	3	**1**	**Escreva o número da linha 3 e coluna 1 : -19**
		2	**Escreva o número da linha 3 e coluna 2 : 1**
4	**1**	**1**	**Escreva o número da linha 1 e coluna 1 : 14**
		2	**Escreva o número da linha 1 e coluna 2 : 27**
	2	**1**	**Escreva o número da linha 2 e coluna 1 : 5**
		2	**Escreva o número da linha 2 e coluna 2 : 4**
	3	**1**	**Escreva o número da linha 3 e coluna 1 : 10**
		2	**Escreva o número da linha 3 e coluna 2 : 65**

Assim, podemos imaginar os elementos dispostos numa estrutura tridimensional, como um cubo.



Apresentação dos elementos de uma Matriz

Para apresentar os elementos de uma matriz, é necessário identificar as suas posições. Isto exige a utilização de um índice para cada dimensão da matriz.

No exemplo a seguir, uma matriz bidimensional com três linhas e cinco colunas é mostrada. Observe que a variável i assume valores sequenciais no intervalo de 1 a 3, ou seja, exatamente nas linhas da matriz. Para cada valor assumido por i, a variável j assume valores sequenciais de 1 a 5, ou seja, as cinco colunas que cada linha possui.

```
PARA i ← 1 ATE 3 FAÇA
INICIO
        PARA j ← 1 ATE 5 FAÇA
        INICIO
                ESCREVA X [i,j]
        FIM
FIM
```

Matriz em PASCAL

Definição de Matriz

As variáveis compostas homogéneas multidimensionais (matrizes) são conhecidas na linguagem PASCAL como ARRAY. Uma estrutura do tipo ARRAY é uma sequencia de variáveis com o mesmo identificador (mesmo nome) e alocadas sequencialmente na memória. Todas as variáveis que compõem uma ARRAY devem ser do mesmo tipo.

Uma vez que as variáveis recebem o mesmo nome, o que as distingue são os índices que referenciam a sua posição em cada dimensão da estrutura. Assim, se a matriz for bidimensional precisaria de dois índices, se for tridimensional precisará de três índices, e assim por diante.

Declaração de Matriz

VAR nome da variável: ARRAY [inicio1. .fim1, inicio2 . .fim2, inicioN. .fimN] OF tipo dos dados;

onde:

nome da variável é o nome da variável do tipo matriz;

inicio1 é o índice inicial da primeira dimensão da matriz;

fim1 é o índice final da primeira dimensão da matriz;

inicio2 é o índice inicial da segunda dimensão da matriz;

fim2 é o índice final da segunda dimensão da matriz;

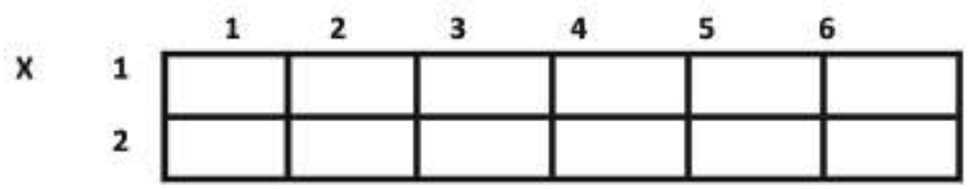
inicioN é o índice inicial da n-exima dimensão da matriz;

fimN é o índice final da n-exima dimensão da matriz;

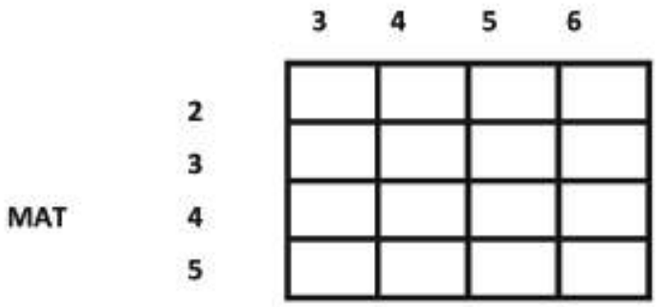
tipo dos dados é o tipo básico de dados que serão armazenados na matriz.

Exemplo de Matriz

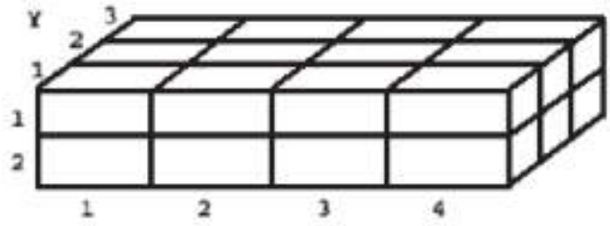
VAR X: ARRAY [1..2,1..6] OF REAL;



VAR MAT: ARRAY [2..5,3..6] OF CHAR;



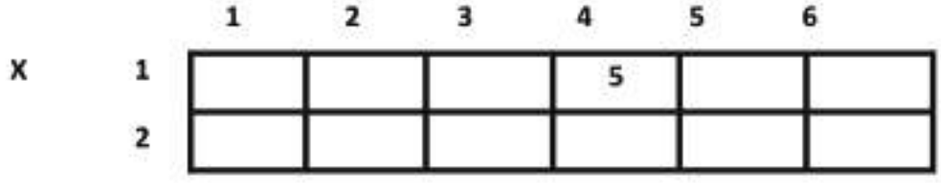
VAR y: ARRAY [1. .2,1. .4,1. .3] OF REAL;



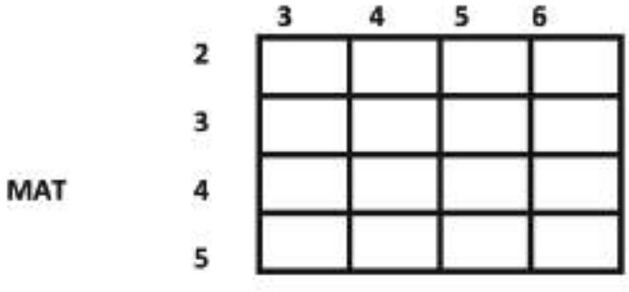
Atribuição de valores a uma matriz

Atribuir valor à matriz significa armazenar uma informação num dos seus elementos, identificado de forma única por meio dos seus índices.

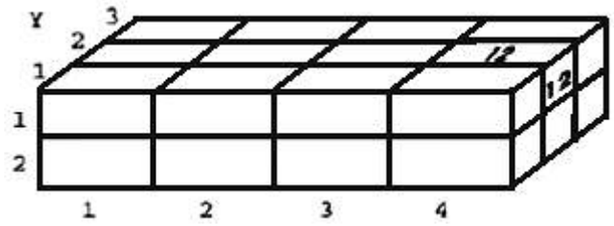
x [1 , 4] := 5 => Atribui o valor 5 à posição identificada pelos índices 1 (linha) e 4 (coluna).



MAT [4,5] := 'D' => Atribui a letra D à posição identificada pelos índices 4 (linha) e 5 (coluna).



Y [1, 4, 2] := 12 => Atribui o valor 12 à posição identificada pelos índices 1 (linha), 4 (coluna) e 2 (profundidade).



Preenchimento de uma Matriz

Preencher uma matriz significa percorrer todos os seus elementos, atribuindo-lhes um valor. Este valor pode ser recebido do utilizador, pelo teclado, ou pode ser criado pelo programa.

No exemplo a seguir, percorrem-se os elementos de uma matriz bidimensional, atribuindo-lhes valores escritos pelo utilizador e capturados por meio do comando READLN .

```
FOR i:= 1 TO 7 DO
BEGIN
        FOR j := 1 TO 3 DO
        BEGIN
                READLN(MAT[i,j]);
        END;
END;
```

Apresentação dos elementos de uma Matriz

Pode-se também percorrer todos os elementos da matriz, acedendo ao seu conteúdo. Para mostrar os valores armazenados numa matriz, supondo que ela tenha sido declarada como var x : ARRAY [1 . . 10, 1.. 6] OF REAL , podem-se executar os comandos a seguir.

```
FOR i :=        1 TO 10 DO
BEGIN
        FOR j := 1 TO 6 DO
```

```
        BEGIN
            WRITELN( X[i, j ] ) ;
        END;
END;
```

Exercícios Resolvidos

Exercício 1

Faca um programa que preencha uma matriz M (2 x 2), calcule e mostre a matriz R, resultante da multiplicação dos elementos de M pelo seu maior elemento.

Solução Algoritmo

```
ALGORITMO
    DECLARE mat[2,2], resultado[2, 2] , i, j, maior NUMERICO
    PARA i ← 1 ATE 2 FAÇA
    INICIO
        PARA j ← 1 ATE 2 FAÇA
        INICIO
            LEIA mat [i,j]
        FIM
    FIM
    maior ← mat[1,1]
    PARA i ← 1 ATE 2 FAÇA
    INICIO
        PARA j ← 1 ATE 2 FAÇA
        INICIO
            SE mat[i,j] > maior
            ENTAO maior ← mat[i,j]
        FIM
    FIM
    PARA i ← 1 ATE 2 FAÇA
    INICIO
```

```
            PARA j ← 1 ATE 2 FAÇA
            INICIO
                  resultado [i,j] ← maior * mat[i,j]
            FIM
      FIM
      PARA i ← 1 ATE 2 FAÇA
      INICIO
            PARA j ← 1 ATE 2 FAÇA
            INICIO
                  ESCREVA resultado[i,j]
            FIM
      FIM
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX1;
      USES CRT;
      VAR   mat, resultado: ARRAY[1..2, 1..2] OF INTEGER;
            i, j, maior: INTEGER;
BEGIN
      CLRSCR;
      FOR i:=1 TO 2 DO
      BEGIN
            FOR j:=1 TO 2 DO
            BEGIN
                  WRITE ('Escreva o elemento da linha ', i, ' e coluna ', j, ' ');
                  READLN(mat[i,j]);
            END;
      END;
      maior := mat[1][1];
      FOR i :=1 TO 2 DO
      BEGIN
```


```
            FOR j:=1 TO 2 DO
            BEGIN
                  IF mat[i][j] > maior THEN maior := mat[i,j];
            END;
      END;
      FOR i:=1 TO 2 DO
      BEGIN
            FOR j:=1 TO 2 DO
            BEGIN
                  resultado[i][j] := maior * mat[i][j];
            END;
      END;
      WRITELN;
      WRITELN('A imprimir a matriz resultante ');
      FOR i:=1 TO 2 DO
      BEGIN
            FOR j:=1 TO 2 DO
            BEGIN
                  WRITE(resultado[i][j], ' ');
            END;
      END;
      READLN;
END.
```

Exercício 2

Faça um programa que preencha uma matriz 10 * 3 com as notas de dez alunos em três provas. O programa devera mostrar um relatório com o número dos alunos (numero da linha) e a prova em que cada aluno obteve menor nota. No final do relatório, devera mostrar quantos alunos tiveram menor nota em cada uma das provas: na prova 1, na prova 2 e na prova 3.

Solução Algoritmo

```
ALGORITMO
        DECLARE     notas[10,3],q1, q2, q3, menor, p_menor, i, j NUMERICO
        PARA i ← 1 ATE 10 FAÇA
        INICIO
                PARA j ← 1 ATE 3 FAÇA
                INICIO
                        LEIA notas[i,j]
                FIM
        FIM
        q1 ← 0
        q2 ← 0
        q3 ← 0
        PARA i ← 1 ATE 10 FAÇA
        INICIO
                ESCREVA i
                menor ← notas [i,j]
                p__menor ← 1
                PARA j ← 1 ATE 3 FAÇA
                INICIO
                        SE notas[i, j] < menor
                        ENTAO INICIO
                                menor ← notas[i, j]
                                p_menor ← j
                        FIM
                FIM
                ESCREVA p_menor
                SE p_menor = 1
                        ENTAO q1 ← q1 + 1
                SE p_menor = 2
                        ENTAO q2 ← q2 + 1
                SE p_menor = 3
```


```
                    ENTAO q3 ← q3 + 1
        FIM
        ESCREVA q1, q2, q3
FIM_ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX2;
        USES CRT;
        VAR     notas:  ARRAY[1..10, 1..3] OF REAL;
                menor: REAL;
                q1, q2, q3, prova_menor, i, j: INTEGER;
BEGIN
        CLRSCR;
        FOR i:=1 TO 10 DO
        BEGIN
                FOR j:=1 TO 3 DO
                BEGIN
                        WRITE('Escreva a ', j, '¦ nota do aluno ', i, ' ');
                        READLN(notas[i,j]);
                END;
        END;

        q1 := 0;
        q2 := 0;
        q3 := 0;
        FOR i:=1 TO 10 DO
        BEGIN
                WRITELN('Aluno número ', i);
                menor := notas[i][1];
                prova_menor := 1;
                FOR j:=1 TO 3 DO
                BEGIN
```

```
                IF notas[i][j] < menor THEN
                BEGIN
                        menor := notas[i][j];
                        prova_menor := j;
                END;
        END;
        WRITELN('A menor nota do aluno ', i, ' foi na ', prova_menor, 'º prova ');
        IF prova_menor = 1 THEN q1 := q1 + 1;
        IF prova_menor = 2 THEN q2 := q2 + 1;
        IF prova_menor = 3 THEN q3 := q3 + 1;
 END;
WRITELN('Quantidade de alunos com menor nota na prova 1 = ', q1);
WRITELN('Quantidade de alunos com menor nota na prova 2 = ', q2);
WRITELN('Quantidade de alunos com menor nota na prova 3 = ', q3);
READLN;
END.
```

Exercício 3

Faça um programa que preencha:

- Um vetor com oito posições, que contenha nomes de lojas;
- Outro vetor com quatro posições, que contenha nomes de produtos;
- Uma matriz com os preços de todos os produtos em cada loja.

O programa devera mostrar todas as relações (nome do produto - nome da loja) em que o preço não ultrapasse $ 120,00.

Solução Algoritmo

```
ALGORITMO
        DECLARE         lojas[8],produtos[4] LITERAL
                        precos[4,8], i, j NUMERICO
        PARA j ← 1 ATE 8 FAÇA
        INICIO
```

```
            LEIA lojas [j]
        FIM
        PARA i ← 1 ATE 4 FAÇA
        INICIO
            LEIA produtos[i]
        FIM
        PARA i ← 1 ATE 4 FAÇA
        INICIO
            PARA j ← 1 ATE 8 FAÇA
            INICIO
                LEIA precos [i, j]
            FIM
        FIM
        PARA i ← 1 ATE 4 FAÇA
        INICIO
            PARA j ← 1 ATE 8 FAÇA
            INICIO
                SE precos[i, j] < 120
                ENTAO ESCREVA produtos [i] , lojas[j]
            FIM
        FIM
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX3;
    USES CRT;
    VAR   lojas: ARRAY[1..8] OF STRING;
          produtos: ARRAY[1..4] OF STRING;
          aux: STRING;
          pre: ARRAY[1..4, 1..8] OF REAL;
          i, j: INTEGER;
BEGIN
```

```
        CLRSCR;
        FOR j:=1 TO 8 DO
        BEGIN
                WRITE('Escreva o nome da ', j, '¦ loja ');
                READLN(lojas[j]);
        END;
        FOR i:=1  TO 4 DO
        BEGIN
                WRITE('Escreva o nome do ', i, 'º produto ');
                READLN(produtos[i]);
        END;
        FOR i:=1 TO 4 DO
        BEGIN
                FOR j:=1 TO 8 DO
                BEGIN
                        WRITE('Escreva o preço do produto ', produtos[i], ' na loja ',
                        lojas[j], ' ');
                        READLN(pre[i][j]);
                END;
        END;
        WRITELN('Listagem dos produtos e respetivas lojas cujos preços não
ultrapassam $ 120,00');
        FOR i:=1 TO 4 DO
        BEGIN
                FOR j:=1 TO 8 DO
                BEGIN
                        IF pre[i][j] < 120
                        THEN WRITELN(produtos[i], '-', lojas[j]);
                END;
        END;
        READLN;
END.
```


Exercício 4

Crie um programa que preencha uma matriz 10 x 20 com números inteiros e some cada uma das linhas, armazenando o resultado das somas num vetor. A seguir, o programa devera multiplicar cada elemento da matriz pela soma da linha correspondente e mostrar a matriz resultante.

Solução Algoritmo

```
ALGORITMO
        DECLARE     mat[10,20], soma[10], i, j NUMERICO
        PARA i ← 1 ATE 10 FAÇA
        INICIO
                PARA j ← 1 ATE 20 FAÇA
                INICIO
                        LEIA mat[i, j]
                FIM
        FIM
        PARA i ← 1 ATE 10 FAÇA
        INICIO
                soma [i] ← 0
                PARA j ← 1 ATE 20 FAÇA
                INICIO
                        soma[i] ← soma[i] + mat[i, j]
                FIM
        FIM
        PARA i ← 1 ATE 10 FAÇA
        INICIO
                PARA j ← 1 ATE 20 FAÇA
                INICIO
                        mat[i, j] ← mat[i, j] * soma[i]
                FIM
        FIM
        PARA i ←- 1 ATE 10 FAÇA
```

```
        INICIO
                PARA j ← 1 ATE 20 FAÇA
                INICIO
                        ESCREVA mat[i, j]
                FIM
        FIM
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX4;
        USES CRT;
        VAR     mat: ARRAY[1..10,1..20] OF REAL;
                soma: ARRAY[1..10] OF REAL;
                i, j: INTEGER;
BEGIN
        CLRSCR;
        FOR i:=1 TO 10 DO
        BEGIN
                FOR j:=1 TO 20 DO
                BEGIN
                        WRITE('Escreva o elemento da linha ', i, ' com a coluna ', j, ' da
                        matriz ');
                        READLN(mat[i][j]);
                END;
        END;
        FOR i:=1 TO 10 DO
        BEGIN
                soma[i] := 0;
                FOR j:=1 TO 20 DO
                BEGIN
                        soma[i] := soma[i] + mat[i][j];
                END;
```


```
        END;
        FOR i:=1 TO 10 DO
        FOR j:=1 TO 20 DO
        mat[i][j] := mat[i][j] * soma[i];
        WRITELN('Imprimindo a matriz resultante');
        FOR i:=1 TO 10 DO
        BEGIN
                WRITELN('Linha ', i);
                FOR j:=1 TO 20 DO
                WRITE(mat[i][j]:7:2, ' ');
        END;
        READLN;
END.
```

Exercício 5

Na teoria dos sistemas, define-se o elemento MINMAX de uma matriz como o maior elemento da linha em que se encontra o menor elemento da matriz. Elabore um programa que carregue uma matriz 4 x 7 com números reais, calcule e mostre o seu MINMAX e a sua posição (linha e coluna).

Solução Algoritmo

```
ALGORITMO
        DECLARE mat[4,7], menor, maior, i, j, l_menor, col NUMERICO
        PARA i ← 1 ATE 4 FAÇA
        INICIO
                PARA j ← 1 ATE 7 FAÇA
                INICIO
                        LEIA mat[i, j]
                FIM
        FIM
        menor ← mat[1, 1]
        1_menor ← 1
```

```
        PARA i ← 1 ATE 4 FAÇA
        INICIO
                PARA j ← 1 ATE 7 FAÇA
                INICIO
                        SE mat[i, j] < menor
                        ENTÃO INICIO
                                menor ← mat [i, j]
                                1_menor ← i
                        FIM
                FIM
        FIM
        maior mat[l_menor, 1]
        col ← 1
        PARA j ← 1 ATE 7 FAÇA
        INICIO
                SE mat [1_menor, j] > maior
                ENTAO INICIO
                        maior mat[1_menor, j]
                        col ← j
                FIM
        FIM
        ESCREVA maior, 1_menor, col
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX5;
        USES CRT;
        VAR     mat: ARRAY[1..4,1..7] OF INTEGER;
                menor, maior, i, j, linha_menor, coluna: INTEGER;
BEGIN
        CLRSCR;
        FOR i:=1 TO 4 DO
```


```
BEGIN
        FOR j:=1 TO 7 DO
        BEGIN
                WRITE('Escreva o elemento ', i, '-', j, ' ');
                READLN(mat[i,j]);
        END;
END;
menor := mat[1][1];
linha_menor := 1;
FOR i:=1 TO 4 DO
BEGIN
        FOR j:=1  TO 7 DO
        BEGIN
                IF mat[i][j] < menor
                THEN BEGIN
                        menor := mat[i][j];
                        linha_menor := i;
                END;
        END;
END;
maior := mat[linha_menor][1];
coluna := 1;
FOR j:=1  TO 7 DO
BEGIN
        IF mat[linha_menor][j] > maior
        THEN BEGIN
                maior := mat[linha_menor][j];
                coluna := j;
        END;
END;
WRITELN('O elemento minmax , = ', maior, ' e está  na linha ‹, linha_menor, ‹ e na
coluna ', coluna);
```

```
        READLN;
END.
```

Exercícios Propostos

Exercício 1

Crie um programa que preencha uma primeira matriz de ordem 4 x 5 e uma segunda matriz 5 x 2. O programa devera, também, calcular e mostrar a matriz resultante do produto matricial das duas matrizes anteriores, armazenando-o numa terceira matriz de ordem 4x2.

Exercício 2

Um elemento Aij de uma matriz é dito ponto de sela da matriz A se, e somente se, Aij for ao mesmo tempo o menor elemento da linha i e o maior elemento da coluna j. Faça um programa que carregue uma matriz de ordem 5x7, verifique se a matriz possui ponto de sela e, se possuir, mostre o seu valor e a sua localização.

Exercício 3

Elabore um programa que preencha uma matriz 6x4 com números inteiros, calcule e mostre quantos elementos dessa matriz são maiores que 30 e, em seguida, construa uma segunda matriz com os elementos diferentes de 30. No lugar do número 30 da segunda matriz, coloque o número zero.

Exercício 4

Crie um programa que preencha uma matriz 15x5 com números inteiros, calcule e mostre quais os elementos da matriz que se repetem e quantas vezes cada um se repete.

Exercício 5

Elabore um programa que preencha uma matriz 10x10 com números inteiros, execute as trocas especificadas a seguir e mostre a matriz resultante:

- A linha 2 com a linha 8;
- A coluna 4 com a coluna 10;

- A diagonal principal com a diagonal secundaria;
- A linha 5 com a coluna 10.

Exercício 6

Crie um programa que preencha uma matriz 8x8 com números inteiros e mostre uma mensagem a dizer se a matriz escrita é simétrica. Uma matriz só pode ser considerada simétrica se A [i, j] = A [j, i].

Exercício 7

Elabore um programa que preencha uma matriz 4 x 4 com números inteiros e verifique se essa matriz forma o chamado quadrado mágico. Um quadrado mágico é formado quando a soma dos elementos de cada linha é igual a soma dos elementos de cada coluna desta linha, e igual à soma dos elementos da diagonal principal e, também, é igual à soma dos elementos da diagonal secundaria.

Exercício 8

Faça um programa que preencha:

- Um vetor com os nomes de cinco produtos;
- Uma matriz 5 x 4 com os preços dos cinco produtos em quatro lojas diferentes;
- Outro vetor com o custo do transporte dos cinco produtos.

O programa devera preencher uma segunda matriz 5x4 com os valores dos impostos de cada produto, de acordo com a tabela a seguir.

PREÇO	% DE IMPOSTO
Até US$ 50,00	**5**
Entre US$ 50,01 a US$ 100 (Inclusive)	**10**
Acima de US$ 100,00	**20**

O programa deverá mostrar ainda um relatório com o nome do produto, o número da loja onde o produto é encontrado, o valor do imposto a pagar, o custo de transporte, o preço e o preço final (preço acrescido do valor do imposto e do custo do transporte).

Exercício 9

Faça um programa que receba:

- Um vetor com o nome de cinco cidades diferentes;
- Uma matriz 5x5 com a distância entre as cidades, sendo que na diagonal principal deve ser colocada automaticamente distancia zero, ou seja, não deve ser permitida a escrita;
- O consumo de combustível de um veículo, ou seja, quantos quilómetros este veículo percorre com um litro de combustível.

O programa devera calcular e mostrar:

- Os percursos que não ultrapassam 250 quilómetros (os percursos são compostos pelos nomes das cidades de origem e pelos nomes das cidades de destino);
- Todos os percursos (nome da cidade de origem e nome da cidade de destino), juntamente com a quantidade de combustível necessária para o veículo os percorrer;

Exercício 10

Crie um programa que leia um vetor V que contenha 18 elementos. A seguir, o programa devera distribuir esses elementos numa matriz 3 x 6 e, no final, mostrar a matriz criada. Veja a seguir um exemplo do que o seu programa devera fazer.

V	3	25	1	58	97	43	65	32	27	19	10	6	88	13	34	57	89	87

M					
3	**25**	**1**	**58**	**97**	**43**
65	**32**	**27**	**19**	**10**	**6**
88	**13**	**34**	**57**	**89**	**87**

Sub-Rotinas

Sub -Rotinas (programação modularizada)

As sub-rotinas, também são chamadas de subprogramas, são blocos de instruções que realizam tarefas específicas.

O código de uma sub-rotina é carregado uma vez e pode ser executado quantas vezes forem necessárias. Como o problema pode ser subdividido em pequenas tarefas, os programas tendem a ficar menores e mais organizados.

Os programas em geral são executados linearmente, uma linha apos a outra, até ao fim. Entretanto, quando são utilizadas sub-rotinas, é possível a realização de desvios na execução dos programas. Estes desvios são efetuados quando uma função é chamada pelo programa principal. Observe o exemplo a seguir (a numeração das linhas à esquerda está a ser utilizada apenas para facilitar a explicação).

```
1       ALGORITMO
2               DECLARE sal NUMERICO
3               LEIA sal
4               aum ← calculo (sal)
5               novo_sal sal + aum
6               ESCREVA “Novo salario é”, novo_sal
7 FIM_ALGORITMO
8 SUB-ROTINA calculo (sal NUMERICO)
9               DECLARE perc, valor NUMERICO
10      LEIA perc
11      valor ← sal * perc / 100
12      RETORNE valor
13 FIM_SUB_ROTINA calculo
```

O algoritmo apresentado tem como objetivo receber o valor do salario de um funcionário e calcular o novo salario. Para resolver este problema, utilizou-se o programa principal (representado pelo bloco de instruções entre as linhas 1 e 7) e uma sub-rotina (representada pelo bloco de instruções entre as linhas 8 e 13).

O programa principal é executado linearmente até à linha 4. Neste ponto, existe uma chamada à sub-rotina cálculo (que recebe como parâmetro o valor do salario inicial) e o programa principal fica temporariamente suspenso. A ordem de execução das instruções é, então, desviada para a linha 8, onde começa a sub-rotina calculo.

A execução só volta ao programa principal quando o comando RETORNE é executado (linha 12). Esse comando é responsável também por devolver ao programa principal um valor calculado dentro da sub-rotina (neste exemplo, foi devolvido o conteúdo da variável valor). A execução do programa principal e retomada exatamente no ponto em que foi interrompida; dessa maneira, o valor devolvido pela sub-rotina é atribuído à variável *aum* (linha 4). A partir dai, o programa volta a ser executado linearmente até ao fim (linha 7).

O exemplo apresentado mostra a utilização de uma sub-rotina que recebe um parâmetro (o valor atual do salario) é que, no final, retorna um valor (aumento que será dado ao salario) para quem a chamou. Porem, as sub-rotinas podem não receber parâmetros nem retornar nenhum valor.

Outro ponto que precisa de ser destacado é que, dentro das sub-rotinas, pode ocorrer declaração de variáveis, chamadas variáveis locais. Elas recebem este nome porque podem ser utilizadas apenas dentro da sub-rotina. Quando a execução desta chega ao fim, essas variáveis são destruídas e os seus conteúdos são perdidos.

Variáveis declaradas fora de qualquer sub-rotina são chamadas de globais. Elas recebem este nome porque qualquer ponto do programa, incluindo as sub-rotinas, pode utiliza-las. São destruídas quando a execução do programa chega ao fim.

Observação: Não se aconselha a utilização excessiva de variáveis globais, por tornar difícil a manutenção e a procura de erros nos programas.

Sub - Rotinas em PASCAL

A linguagem Pascal possibilita a modularização por meio de *procedures* (procedimentos) e *functions* (funções).

Procedures sem passagem de parâmetros

As *procedures* (procedimentos) são rotinas chamadas pelo programa principal para executar alguma operação específica, mas não retornam valor para quem as chamou. Possuem a seguinte sintaxe:

```
PROCEDURE nome da procedure;
declaração de variáveis locais;
BEGIN
        comandos;
END.
```

Quando o programa encontra uma linha que contenha o nome da *procedure,* o fluxo da execução é desviado para as linhas contidas dentro dela. Esta execução só retomara ao fluxo normal quando a execução da *procedure* chegar ao fim.

A seguir é apresentado um exemplo de *procedure* sem parâmetros (a numeração das linhas não faz parte do programa).

```
1 PROGRAM EXEMPLO;
2       USES CRT;
3       VAR I, P, NUM, CONT : INTEGER;
4       PROCEDURE PAR;
5       {Conta quantos n°s pares existem entre 1 e o número escrito no
        programa principal}
6       BEGIN
7               CONT:=2;
8               P : = 0 ;
9               WHILE CONT <= NUM DO
10              BEGIN
11                      P :=P+1;
12                      CONT:=CONT+2;
13              END;
14      END;
15      PROCEDURE IMPAR;
```

```
16      {Conta quantos n°s impares existem entre 1 e o número escrito no
        programa principal}
17      BEGIN
18              CONT:=1;
19              I:= 0;
20              REPEAT
21                      I:=I+1;
22                      CONT:=CONT+2;
23              UNTIL CONT > NUM;
24      END;
25      BEGIN
26              {inicio do programa principal}
27              WRITELN('ESCREVA O NUMERO DO INTERVALO ');
28              READLN (NUM);
29              PAR;
30              IMPAR;
31              WRITELN('QUANTIDADE DE PARES = ',P);
32              WRITELN('QUANTIDADE DE IMPARES = ',D;
33              READLN;
34      END.
```

O programa começa a sua execução no BEGIN principal, representado no exemplo anterior pela linha 25. Posteriormente, executa as linhas 26,27 e 28. Na linha 29 existe a chamada a uma *procedure.* O programa principal é desviado para a *procedure* denominada PAR. Assim, o fluxo de execução do programa vai para a linha 4, executando toda a *procedure*, ou seja, da linha 4 até à 14. Em seguida, ele retorna a linha 30, exatamente abaixo da linha onde ocorreu o primeiro desvio. Na linha 30 existe a chamada a outra *procedure.* O programa principal é desviado para a *procedure* denominada IMPAR. Assim, vai para a linha 15, executando toda a *procedure*, ou seja, da linha 15 até à 24. De seguida, retorna à linha 31, exatamente abaixo da linha onde ocorreu o segundo desvio. Finalmente, as linhas 31, 32, 33 e 34 são executadas e o programa chega ao fim.

Procedures com passagem de parametros

Pode-se utilizar *procedure* com passagem de parâmetros, ou seja, no momento em que a execução da *procedure é* solicitada, alguns valores são-lhe fornecidos. Observe a sintaxe:

```
PROCEDURE nome da procedure(x,y:tipo dos dados);
declaração de variáveis locais;
BEGIN
        comandos;
END.
```

A seguir é mostrado um exemplo de *procedure* com passagem de parâmetros (a numeração das linhas não faz parte do programa).

```
1 PROGRAM EXEMPLO;
2       USES CRT;
3       VAR I, C0L1, C0L2, LIN1, LIN2, X : INTEGER;
4       PROCEDURE DESENHA(Cl,C2,L1,L2 : INTEGER);
5       {Desenha um quadrado com *}
6       {Onde os limites do quadrado estão nas variáveis Cl, C2, LI e L2)
7       BEGIN
8               FOR I := Cl TO C2 DO
9               BEGIN
10                      GOTOXY(I,L1);
11                      WRITE('*');
12                      GOTOXY(I,L2);
13                       WRITE('*');
14              END;
15              FOR I:= L1 TO L2 DO
16              BEGIN
17                      GOTOXY(Cl,I);
18                      WRITELN('*');
19                      GOTOXY(C2,I);
20                      WRITELN('*');
```

```
21          END;
22    END;
23    BEGIN
24          {Programa principal - desenha 5 quadrados com *}
25          X:= 0;
26          REPEAT
27                CLRSCR;
28                X := X+1;
29                WRITELN('ESCREVA OS VALORES DAS BORDAS1);
30                READLN(C0L1,C0L2,LIN1,LIN2);
31                CLRSCR;
32                DESENHA(COL1,COL2,LIN1,LIN2);
33                READLN;
34                UNTIL X = 5;
35          END.
```

O programa anterior começa a sua execução no BEGIN principal, na linha 23. Posteriormente, executa as linhas 24 até 31. Na linha 32 existe a chamada a uma *procedure.* O programa principal é desviado para a *procedure* chamada DESENHA, levando os valores das variáveis col1, col2, lin1 e lin2, que são copiados, respetivamente, para as variáveis C1, C2, L1 e L2, descritas no cabeçalho da *procedure.* Quando a execução da *procedure* for concluída, o fluxo retomará ao programa principal, exatamente na linha 33, imediatamente abaixo do ponto de chamada da *procedure* DESENHA.

Function sem passagem de parâmetros

Uma *function* (função) tem o mesmo objetivo que uma *procedure*, ou seja, desviar a execução do programa principal para realizar uma tarefa específica, com uma única diferença: uma *function* retorna sempre um valor. A sintaxe de uma *function* é:

```
FUNCTION nome da function : tipo de dado do valor retornado;
declaração de variáveis locais;
BEGIN
```

```
        comandos;
END;
```

É importante ressaltar que, para que ocorra o retorno de algum valor para quem chamou a *function,* deve-se atribuir tal valor a uma variável cujo nome seja igual ao nome dado à *function.*

A chamada à *function* acontece atribuindo o seu nome a uma variável, que receberá o retorno produzido. A seguir, é apresentado um exemplo (a numeração das linhas não faz parte do programa).

```
1 PROGRAM EXEMPLO;
2       USES CRT;
3       VAR     CALC, X : REAL;v
4               FUNCTION RAIZ : REAL;
5               {Função que calcula a raiz quadrada de um numero}
6       BEGIN
7               RAIZ:= SQRT(X);
8       END;
9       BEGIN
10              {Programa principal}
11              CLRSCR;
12              WRITELNI'ESCREVA UM VALOR PARA CALCULAR A RAIZ');
13              READLN (X);
14              IF X < 0
15              THEN WRITELN ('NAO EXISTE RAIZ QUADRADA')
16              ELSE BEGIN
17                      CALC:= RAIZ;
18                      WRITELN ( 'RAIZ DE ' , X : 6 :2, ' = ',CALC:6:2);
19              END;
20              READLN;
21      END.
```

No exemplo anterior, o programa começa a sua execução no BEGIN principal, representado pela linha 9. Posteriormente, as linhas 10 até à 17 são executadas. Na linha 17 existe a chamada a *function* RAIZ . O fluxo de execução é então desviado para a linha 4 e toda a *function* é executada. Observe que na linha 7 é feita uma atribuição à variável RAIZ. Em seguida, o programa retorna à linha 17, que é a linha onde ocorreu o desvio.
Neste momento, o valor atribuído à RAIZ (na linha 7) é copiado para a variável CALC. Depois disto, as linhas 18, 19, 20 e 21 são executadas, finalizando o programa.

Function com passagem de parâmetros

Uma *function* pode receber parâmetros no momento em que é chamada. Os valores informados são copiados, sequencialmente, em variáveis descritas no seu cabeçalho. A sintaxe correta é a seguinte:

```
FUNCTION nome da function (x,y: tipo de dados) : tipo de dado do valor retornado;
declaração de variáveis locais;
BEGIN
        comandos;
END;
```

A chamada a uma *function* acontece atribuindo o seu nome a uma variável, que recebera o retorno no fim da sua execução. No momento da chamada, são informados também os parâmetros que deverão ser levados para a *function.* A seguir é apresentado um exemplo de *function* com passagem de parâmetros (a numeração das linhas não faz parte do programa).

```
1 PROGRAM EXEMPLO;
2       VAR     CALC,: REAL;
3               I: INTEGER;
4       FUNCTION RAIZ (NUM:REAL) : REAL;
5       {Função que calcula a raiz quadrada de um numero}
6       BEGIN
7               RAIZ := SQRT(NUM);
```

```
8       END;
9       BEGIN
10              {Programa principal}
11              CLRSCR;
12              WRITELN('ESCREVA UM VALOR ');
13              READLN(I);
14              IF I <= 0
15              THEN WRITELN('VALOR INVALIDO');
16              ELSE BEGIN
17                      CALC:= RAIZ(I);
18                      WRITELN ('RAIZ DE ',1,' = ', CALC:8:3);
19              END;
20              READLN;
21      END.
```

Neste exemplo, o programa começa a sua execução no BEGIN principal, representado pela linha 9. Posteriormente, executa as linhas 10 à 19. Na linha 17 existe a chamada a uma *function.* O programa principal é desviado para a, *function* denominada RAIZ, levando o valor da variável i, que é copiado para a variável NUM, descrita no cabeçalho da *function.* Observe que na linha 7 é feita uma atribuição à variável RAIZ . Em seguida, o programa retorna a linha 17, que é a linha onde ocorreu o desvio. Neste momento, o valor atribuído á RAIZ (na linha 7) é copiado para a variável CALC. Depois disto, as linhas 18, 19, 20 e 21 são executadas, finalizando o programa.

Exercícios Resolvidos

Exercício 1

Faça um programa que contenha uma sub-rotina que retorne 1 se o número escrito for positivo ou 0 se for negativo.

Solução Algoritmo

```
ALGORITMO
        DECLARE num, x NUMERICO
        LEIA num
        x ← verifica (num)
        SE x = 0
        ENTAO ESCREVA “Número positivo”
        SENAO ESCREVA “Numero negativo”
FIM_ALGORITMO
SUB-ROTINA verifica (num NUMERICO)
        DECLARE res NUMERICO
        SE num >= 0
        ENTAO res 1
        SENAO res <- 0
        RETORNE res
FIM SUB ROTINA verifica
```

Solução PASCAL

```
PROGRAM EX1;
        USES CRT;
        VAR     num: INTEGER;
                x: BOOLEAN;

        FUNCTION verifica(num:REAL): BOOLEAN;
        BEGIN
                IF num >= 0
                THEN verifica := true
                ELSE verifica := false;
        END;
        BEGIN
                CLRSCR;
                WRITE(‘Escreva um número: ‘);
```


```
        READLN(num);
        x := verifica(num);
        WRITELN;
        IF x=true
        THEN WRITELN('Número positivo')
        ELSE WRITELN('Número negativo');
        READLN;
END.
```

Exercício 2

Faça uma função que receba dois números positivos por parâmetro e retorne a soma dos N números inteiros existentes entre eles.

Solução Algoritmo

```
ALGORITMO
        DECLARE num1, num2, s NUMERICO
        LEIA num1, num2
        s ← somar{num1, num2)
        ESCREVA "soma = ", s
FIM_ALGORITMO
SUB-ROTINA somar(num1, num2 NUMERICO)
        DECLARE i, s NUMERICO
        S ← 0
        PARA i ← num1 + 1 ATE num2-l FAÇA
        INICIO
                s ← s + i
        FIM
        RETORNE s
FIM SUB ROTINA somar
```

Solução PASCAL

```
PROGRAM EX2;
        USES CRT;
        VAR num1, num2, s: INTEGER;
        FUNCTION somar(num1, num2:INTEGER): INTEGER;
        VAR s, i: INTEGER;
        BEGIN
                s := 0;
                FOR i:=num1+1 TO num2-1 DO
                BEGIN
                        s := s + i;
                END;
                somar := s;
        END;
        BEGIN
                CLRSCR;
                WRITE('Escreva o 1º número: ');
                READLN(num1);
                WRITE('Digite o 2º número: ');
                READLN(num2);
                s := somar(num1, num2);
                WRITELN('Soma = ', s);
                READLN;
        END.
```


Exercício 3

Crie uma função que receba três números inteiros *a, b* e *c*, sendo *a* maior que 1. A função devera somar todos os inteiros entre *b* e *c* que sejam divisíveis por *a* (inclusive *b* e c) e retomar o resultado para a função principal.

Solução Algoritmo

```
ALGORITMO
        DECLARE a, b, c, result NUMERICO
        REPITA
                LEIA a
                ATE a>l
                LEIA b,c
                result ← divisores (a, b, c)
        ESCREVA "Soma dos inteiros = ", result
FIM ALGORITMO
SUB-ROTINA divisores(a, b, c NUMERICO)
        DECLARE i, s, r NUMERICO
        s ← 0
        PARA i ← b ATE c FAÇA
        INICIO
                r ← RESTO (i / a)
                SE r = 0
                ENTAO s ← s + i
        FIM
        RETORNE s
FIM SUB ROTINA divisores
```

Solução PASCAL

```
PROGRAM EX3;
    USES CRT;
    VAR a, b, c, result: INTEGER;
    FUNCTION divisores(a, b, c: INTEGER): INTEGER;
    VAR i, s, r: INTEGER;
    BEGIN
        s := 0;
        FOR i:=b TO c DO
        BEGIN
            r := i MOD a;
            IF r = 0
            THEN s := s + i;
        END;
        divisores := s;
    END;
    BEGIN
        CLRSCR;
        REPEAT
        WRITE('Escreva o valor de a: ');
        READLN(a);
        UNTIL a > 1;
        WRITE('Escreva o valor de b: ');
        READLN(b);
        WRITE('Digite o valor de c: ');
        READLN(c);
        result := divisores(a, b, c);
        WRITELN('Soma dos inteiros = ', result);
        READLN;
    END.
```


Exercício 4

Faça uma função que receba um único valor representando segundos. Essa função devera converte-lo para horas, minutos e segundos. Todas as variáveis devem ser passadas como parâmetro, não havendo variáveis globais.

Solução Algoritmo

```
ALGORITMO
        DECLARE seg NUMERICO
        LEIA seg
        transformacao(seg);
FIM_ALGORITMO
SUB-ROTINA transformacao(segundos NUMERICO)
        DECLARE h, m, s, r NUMERICO
        h <- segundos / 3600
        r ← RESTO(segundos / 3600)
        m ← r / 60
        s = RESTO(r / 60)
        ESCREVA h, m, s
FIM SUB ROTINA transformacao
```

Solução PASCAL

```
PROGRAM EX4;
        USES CRT;
        VAR seg: INTEGER;
        PROCEDURE transformacao(segundos:INTEGER);
        VAR h, r, m, s: INTEGER;
        BEGIN
                h := segundos DIV 3600;
                r := segundos MOD 3600;
                m := r DIV 60;
                s := r MOD 60;
                WRITELN(segundos, ‘ segundo(s) equivale(m) a ‘, h, ‘ hora(s), ‘, m, ‘
```

```
        minuto(s) e ', s, ' segundo(s)');
        READLN;
    END;
    BEGIN
        CLRSCR;
        WRITE('Escreva o valor em segundos: ');
        READLN(seg);
        transformacao(seg);
    END.
```

Exercício 5

Crie um programa que receba os valores antigos e atual de um produto. Chame uma sub-rotina que determine a percentagem de acréscimo entre esses valores. O resultado devera ser mostrado no programa principal.

Solução Algoritmo

```
ALGORITMO
    DECLARE precoantigo, precoatual, acrescimo NUMERICO
    LEIA precoantigo
    LEIA precoatual
    acrescimo ← calculo_reajuste(precoantigo, precoatual)
    ESCREVA acrescimo
FIM_ALGORITMO
SUB-ROTINA calculo_reajuste(PA, PN NUMERICO)
    DECLARE result NUMERICO
    result ← (100 * PN - 100 * PA) / PA
    RETORNE result
FIM_SUB_ROTINA calculo_reajuste
```

Solução PASCAL

```
PROGRAM EX5;
        USES CRT;
        VAR precoantigo, precoatual, acrescimo: REAL;

        FUNCTION calculo_reajuste(PA, PN:REAL): REAL;
        BEGIN
                calculo_reajuste := (100 * PN - 100 * PA) / PA;
        END;
        BEGIN
                CLRSCR;
                WRITE('Escreva o preço antigo: ');
                READLN(precoantigo);
                WRITE('Escreva o preço final: ');
                READLN(precoatual);
                acrescimo := calculo_reajuste(precoantigo, precoatual);
                WRITELN('O reajuste foi de ', acrescimo:5:2, '%');
                READLN;
        END.
```

1								
2	4							
3	6	9						
4	8	12	16					
5	10	15	20	25				
6	12	18	24	30	36			
7	14	21	28	35	42	49		
8	16	24	32	40	48	56	64	
9	18	27	36	45	54	63	72	81

Exercício 2

Elabore um programa que contenha uma sub-rotina que receba as três notas de um aluno como parâmetros e uma letra. Se a letra for A, a sub-rotina devera calcular a média aritmética das notas do aluno; se for P, devera calcular a média ponderada, com pesos 5,3 e 2. A média calculada devera ser devolvida ao programa principal para, então, ser mostrada.

Exercício 3

Crie uma sub-rotina que receba como parâmetro a hora de inicio e a hora de termino de um jogo, ambas subdivididas em dois valores distintos: horas e minutos. A sub-rotina devera retornar a duração expressa em minutos, considerando que o tempo máximo de duração de um jogo é de 24 horas e que ele pode começar num dia e terminar no outro.

Exercício 4

Faça uma sub-rotina que leia cinco valores inteiros, determine e mostre o maior e o menor deles.

Exercício 5

Crie unia sub-rotina que receba como parâmetro um valor inteiro e positivo N e retorne o valor de S, obtido pelo seguinte calculo:

$$S = 1 + 1/1! + 1/2! + 1/3! + \ldots + 1/N!$$

Exercício 6

Foi realizada uma pesquisa sobre algumas características fisicas de cinco habitantes de uma região. Foram coletados os seguintes dados de cada habitante: sexo, cor dos olhos (A - azuis ou C - castanhos), cor dos cabelos (L - loiros, P - pretos ou C - castanhos) e idade.

- Faça uma função que leia esses dados, armazenando-os em vetores.
- Faça uma função que determine e devolva ao programa principal a media de idade das pessoas com olhos castanhos e cabelos pretos.
- Faça uma função que determine e devolva ao programa principal a maior idade entre os habitantes.
- Faça uma função que determine e devolva ao programa principal a quantidade de indivíduos do sexo feminino com idade entre 18 e 35 anos (inclusive) e que tenham olhos azuis e cabelos louros.

Exercício 7

Elabore uma sub-rotina que retorne ao programa principal um vetor com os três primeiros números perfeitos. Sabe-se que um número é perfeito quando é igual à soma de seus divisores (exceto por ele mesmo).

Exemplo: os divisores de 6 são 1, 2 e 3, e 1 + 2 + 3=6, logo 6 é perfeito.

Exercício 8

Faça uma sub-rotina que receba um vetor A de dez elementos inteiros como parâmetro. No final desta função, devera ter sido gerado um vetor B que contenha o fatorial de cada elemento de A. O vetor B devera ser mostrado no programa principal.

Exercício 9

Crie uma sub-rotina que receba como parâmetro dois vetores de dez elementos inteiros positivos e mostre o vetor união dos dois primeiros.

Exercício 10

Faça uma sub-rotina que receba como parâmetro um vetor A com cinco números reais e retorne esses números ordenados de forma crescente.

Manipulação de cadeia de Caracteres

Manipulação de cadeias de caracteres em PASCAL

As cadeias de caracteres em PASCAL são representadas pelo tipo de dado conhecido como STRING. Uma STRING em PASCAL é uma sequência de símbolos delimitada por apóstrofos. Quando um apóstrofo fizer parte da STRING, devera aparecer duplicado.
Quando uma variável é declarada do tipo STRING, é possível definir o seu tamanho máximo. Quando a definição do tamanho for omissa, a STRING assumirá o tamanho máximo permitido de 255 caracteres.

Exemplos de variáveis do tipo STRING:

VAR NOME: STRING [20];	A variável NOME poderá armazenar até 20 caracteres.
VAR SEXO: STRING [3];	A variável SEXO poderá armazenar ate três caracteres.
VAR CURSO: STRING;	A variável CURSO não tem tamanho predefinido,
logo	poderá armazenar ate 255 caracteres.
NOME: = ‘Maria’ ;	O conteúdo da variável NOME é Maria.
CURSO:= ‘D’’ÁGUA’;	O conteúdo da variável CURSO é D’ÁGUA. Observe
a	duplicação do apóstrofo.

Inicialização de cadeias de caracteres

As variáveis que armazenam cadeias de caracteres podem ser inicializadas automaticamente pelo programa ou podem receber um valor através do teclado. A seguir exemplificamos alguns casos.

a. Inicialização através da atribuição

```
VAR cadeia: STRING;
cadeia := ‘Programa’;
```

A variável cadeia recebe um valor constante, a palavra Programa.

b. Inicialização através do teclado

```
VAR cadeia: STRING;
READLN(cadeia);
```

O comando READLN armazena dentro da variável cadeia todos os símbolos escritos ate à ocorrência do ENTER.

Cópias de cadeias de caracteres

cadeia2 := COPY (cadeia1, posição, número);

A função COPY copia da cadeia1, a partir da posição dada, o número de caracteres estipulados. Os caracteres copiados serão armazenados na cadeia2.

cadeia1:= cadeia2;

O conteúdo da variável cadeia2, que é uma variável do tipo STRING, será copiado para a variável cadeia1, que também é do tipo STRING.

Exemplos deste tipo de função COPY podem ser vistos mais à frente nos exercícios resolvidos 1 e 2.

Ligação de cadeias de caracteres

cadeia3 := CONCAT(cadeia1,cadeia2) ; ou cadeia3 := cadeia1+cadeia2;

A função CONCAT liga/encadeia a cadeia de caracteres cadeia1 e a cadeia de caracteres cadeia2, ou seja, junta os conteúdos das cadeias de caracteres cadeia1 e cadeia2. A cadeia de caracteres resultante desta ligação é armazenada na cadeia3. A kunção pode ser feita também pelo sinal de +.

Exemplos deste tipo de junção de cadeias podem ser encontrados no exercício resolvido 3.

Comparação de cadeias de caracteres

A comparação entre cadeias de caracteres em PASCAL é feita pelo sinal de igual, recorde que letras maiúsculas são diferentes de letras minúsculas.
Exemplos de comparação de cadeias de caracteres podem ser encontrados no exercício resolvido 5.

Descobrir o número de caracteres de uma cadeia

```
tamanho := LENGTH(cadeia);
```

A função LENGTH retorna, para a variável tamanho, o número de caracteres da cadeia.
Um exemplo da função LENGTH pode ser encontrado no exercício resolvido 6.

Verificação da posição de uma cadeia de caracteres dentro de outra cadeia de caracteres

```
posi := POS (cadeia1,cadeia2);
```

A função POS retorna, para a variável posi, a posição inicial em que a cadeia1 aparece dentro da cadeia2.

Um exemplo da função POS pode ser encontrado no exercício resolvido 7.

Apagar caracteres de uma cadeia de caracteres

```
DELETE(cadeia, posição, numero);
```

A função delete apaga da variável cadeia o número de caracteres estipulados, a partir da posição informada.

Um exemplo da função DELETE pode ser encontrado no exercício resolvido 8.

Inserir caracteres numa cadeia de caracteres

INSERT(cadeia1, cadeia2, posicao);

A função INSERT insere a cadeia1 na cadeia2, a partir da posição dada.

Um exemplo da função INSERT pode ser encontrado no exercício resolvido 9.

Alteração dos caracteres de uma cadeia de caracteres

Não há uma função específica para alterar os caracteres de uma cadeia. Assim, a alteração requerera que os caracteres sejam acedidos individualmente para receberem os novos valores. O acesso é obtido através de um índice cujo valor começa em 1 é incrementado até ao final da cadeia. Logo, se desejar alterar o primeiro caractere da cadeia, basta escrever CADEIA[i] := 1 a 1; e será feita a alteração do primeiro caractere para A.

Exemplos da alteração de caracteres de uma cadeia pode ser encontrado no exercício resolvido 10 e 11.

Descobrir um caractere a partir do seu valor ASCII

carac := CHR(número);

A função CHR retoma para a variável carac, o caractere ASCII que é representado pelo número.

Um exemplo da função CHR pode ser encontrado no exercício resolvido 12.

Descobrir o valor ASCII de um caractere

valor := ORD(caractere);

A função ORD retoma, para a variável valor, o valor numérico que representa o caractere na tabela ASCII .

Um exemplo da função ORD pode ser encontrado no exercício resolvido 13.

Descobrir o caractere sucessor

carac := SUCC(caractere);

A função SUCC retorna, para a variável carac, o caractere sucessor do caractere na tabela ASCII .

Um exemplo da função SUCC pode ser encontrado exercício resolvido 14.

Descobrindo o caractere antecessor ou predecessor

carac := PRED(caractere);

A função PRED retorna, para a variável carac, o caractere antecessor ou predecessor do caractere na tabela ASCII.

Um exemplo da função PRED pode ser encontrado no exercício resolvido 15.

Converter caracteres para maiúsculo

UPCASE(caractere[posição]);

A função UPCASE converte o caractere da posição especificada para maiúsculo.

STRUPPER(cadeia de caracteres);

A função STRUPPER converte a cadeia de caracteres para maiúsculo. Esta função exige a utilização da biblioteca STRINGS .

Exemplos das funções UPCASE e STRUPPER podem ser encontrados nos exercícios resolvido 16 e 17.

OBSERVAÇÃO: Em ambas as conversões, as letras acentuadas e o ç não são convertidos.

Convertendo caracteres para minúsculo

Não existe uma função correspondente à função UPCASE para converter caracteres de maiúsculo para minúsculo. Assim, a conversão é obtida somente por meio de cadeias de caracteres.

STRLOWER(cadeia de caracteres);

A função STRLOWER converte a cadeia de caracteres para minúsculo. Esta função exige a utilização da biblioteca STRINGS .

OBSERVAÇÃO: Em ambas as conversões, as letras acentuadas e o ç não são convertidos.

Conversão de um valor numérico em caracteres

STR(valor numérico, cadeia de caracteres);

A função STR converte o valor numérico para a cadeia de caracteres.
Um exemplo da função STR pode ser encontrado no exercício resolvido 18.

Conversão de caracteres em valor numérico

VAL(cadeia de caracteres, valor numérico, erro);

A função VAL converte o valor numérico para a cadeia de caracteres. Se algum erro ocorrer durante a conversão, a variável erro recebera um valor diferente de zero.
Um exemplo da função VAL pode ser encontrado no exercício resolvido 19.

Exercícios Resolvidos

Exercício 1

Faça um programa que receba uma frase, calcule e mostre a quantidade de vogais da frase escrita. O programa devera contar vogais maiúsculas e minúsculas.

Solução

- Escrever uma frase.
- Ver o tamanho da frase.
- Percorrer a frase, vendo caractere a caractere.
- Comparar cada caractere com as vogais (maiúsculas e minúsculas).
- Quando encontrar uma vogal, acrescentar um na quantidade.

Solução PASCAL

```
PROGRAM EX1;
    USES CRT;
    VAR FRASE: STRING;
    QTDE, TAM, I: INTEGER;
    BEGIN
        CLRSCR;
        WRITELN('ESCREVA UMA FRASE');
        READLN(FRASE);
        TAM := LENGTH(FRASE);
        QTDE := 0;
        FOR I:=1 TO TAM DO
        BEGIN
            IF (FRASE[I] = 'A') OR (FRASE[I] = 'E')
            OR (FRASE[I] = 'I') OR (FRASE[I] = 'O')
            OR (FRASE[I] = 'U') OR (FRASE[I] = 'a')
            OR (FRASE[I] = 'e') OR (FRASE[I] = 'i')
            OR (FRASE[I] = 'o') OR (FRASE[I] = 'u')
            THEN QTDE := QTDE + 1;
        END;
        WRITELN('QUANTIDADE DE VOGAIS = ',QTDE);
        READLN;
    END.
```

Exercício 2

Faça um programa que receba uma frase, calcule e mostre a quantidade de consoantes da frase escrita. O programa devera contar consoantes maiúsculas e minúsculas.

Solução

- Escrever uma frase.
- Ver o tamanho da frase.
- Percorrer a frase, vendo caractere a caractere.
- Comparar cada caractere com as consoantes (maiúsculas e minúsculas).
- Quando encontrar uma consoante, acrescentar um na quantidade.

Solução PASCAL

```
PROGRAM EX2;
    USES CRT;
    VAR   frase: STRING;
          tam,i,qtde,num: INTEGER;
    BEGIN
          qtde:=0;
          WRITELN('Escreva uma frase');
          READLN(frase);
          tam:=LENGTH(frase);
          FOR i:=1 TO tam DO
          BEGIN
                num:=ORD(frase[i]);
                IF ((num >= 65) AND (num <= 90)) OR ((num >= 97) AND
          (num <= 122))
                THEN BEGIN
                      {, uma letra, maiúscula ou minúscula}
                      IF (num <> 65) AND (num <> 69)
                      AND (num <> 73) AND (num <> 79)
                      AND (num <> 85) AND (num <> 97)
                      AND (num <> 101) AND (num <> 105)
```

```
                    AND (num <> 111) AND (num <> 117)
                    THEN qtde := qtde + 1;
                END;
        END;
        WRITELN('Quantidade de consoantes = ',qtde);
        READLN;
END.
```

Exercício 3

Faça um programa que receba uma frase, calcule e mostre a quantidade de palavras da frase escrita.

Solução

- Escrever uma frase.
- Ver o tamanho da frase.
- Percorrer a frase, vendo caractere por caractere.
- Comparar cada caractere com espaço em branco.
- Quando encontrar um espaço, acrescentar um na quantidade.
- Como apos a ultima palavra não tem espaço, acrescentar um na quantidade.

Solução PASCAL

```
PROGRAM EX3;
    USES CRT;
    VAR   frase: STRING;
          tam,i,qtde: INTEGER;
    BEGIN
          qtde:=0;
          WRITELN('Escreva uma frase');
          READLN(frase);
          tam:=LENGTH(frase);
          FOR i:=1 TO tam DO
          BEGIN
```


```
            if frase[i] = ' '
            THEN qtde := qtde + 1;
        END;
        WRITELN('Quantidade de palavras = ',qtde+1);
        READLN;
END.
```

Exercício 4

Faça um programa que receba uma frase e mostre as letras que se repetem, juntamente com o número de repetições.

Exemplo: A PROVA FOI ADIADA

- A letra A apareceu 5 vezes.
- A letra O apareceu 2 vezes.
- A letra I apareceu 2 vezes.
- A letra D apareceu 2 vezes.

Solução

- Escrever uma frase.
- Ver o tamanho da frase escrita.
- Percorrer a frase, vendo caractere por caractere.
- Verificar se é a primeira vez que esse caractere aparece na frase.
- Caso seja a primeira vez, atribuir um ao contador de aparições.
- Caso contrário, incrementar o contador de aparições numa unidade.
- Mostrar todas as letras que apareceram mais de uma vez (que se repetiram), juntamente com o total de repetições.

Solução PASCAL

```
PROGRAM EX4;
    USES CRT;
    VAR   frase,letra: STRING;
          letras:ARRAY[1..26] OF STRING;
          rep: ARRAY[1..26] OF INTEGER;
```

```
        qtde, tam1, i, j, achou: INTEGER;
BEGIN
        CLRSCR;
        FOR  i := 1 TO 26 DO
        rep[i] := 0;
        FOR  i := 1 TO 26 DO
        letras[i] := '';
        WRITELN('Escreva uma frase ');
        READLN(frase);
        tam1 := LENGTH(frase);
        qtde:=0;
        achou:=0;
        FOR  i := 1 TO tam1 DO
        BEGIN
                letra := COPY(frase,i,1);
                IF letra <> ' '
                THEN BEGIN
                        IF qtde <> 0
                        THEN BEGIN
                                FOR j := 1 TO qtde DO
                                BEGIN
                                        IF letras[j] = letra
                                        THEN achou:= 1;
                                END;
                        END;
                        IF achou = 0
                        THEN BEGIN
                                achou:=1;
                                FOR j := i+1 TO tam1 DO
                                BEGIN
                                        IF letra = COPY(frase,j,1)
                                        then achou:=achou + 1;
```


```
                    END;
                    IF achou > 1
                    THEN BEGIN
                        qtde:=qtde+1;
                        letras[qtde] := letra;
                        rep[qtde]:= achou;
                    END;
                END;
            END;
            achou:=0;
        END;
        FOR i := 1 TO QTDE DO
        BEGIN
            WRITELN('A letra ', LETRAS[i], ' apareceu ', rep[i],' vezes');
        END;
        IF qtde = 0
        THEN WRITELN('Nenhuma letra foi repetida');
        READLN;
END.
```

Exercício 5

Faca um programa para criptografar uma frase dada pelo utilizador (a criptografia troca as vogais da frase por *).

Exemplo:

Frase: EU ESTOU NA ESCOLA

Saída: ** * ST** N* *SC*L*

Solução

- Escrever uma frase.
- Ver o tamanho da frase.
- Percorrer a frase, comparando cada caractere com as vogais.
- Quando encontrar uma vogal, substitui-la por um asterisco.

Solução PASCAL

```
PROGRAM EX5;
        USES CRT,STRINGS;
        VAR     letra : CHAR;
                frase: array[0..200] OF CHAR;
                tam, i : INTEGER;
        BEGIN
                CLRSCR;
                WRITELN('Digite uma frase');
                READLN(frase);
                STRUPPER(frase);
                tam := LENGTH(frase);
                FOR i := 0 TO tam DO
                BEGIN
                        letra:=frase[i];
                        IF (letra = 'A') OR (letra = 'E')
                        OR (letra = 'I') OR (letra = 'O') OR (letra = 'U')
                        THEN frase[i] := '*';
                END;
                WRITELN(frase);
                READLN;
        END.
```

Exercício 6

Faça um programa que receba duas frases e crie uma terceira que represente a combinação das palavras das duas frases recebidas.

Exemplo:

Frase 1: Hoje está um belo dia

Frase 2: Talvez chova amanhã

Saída: Hoje talvez esta chova um amanha belo dia

Solução

- Escrever duas frases.
- Ver o tamanho das frases escritas.
- Percorrer a primeira frase, vendo caractere a caractere, até chegar a um espaço em branco ou ao fim da frase.
- Quando encontrar o espaço em branco ou o fim da primeira frase, foi obtida uma palavra completa. Junta-se à nova frase.
- Percorrer a segunda frase, vendo caractere a caractere, ate chegar a um espaco em branco ou ao fim da frase.
- Quando encontrar o espaço em branco ou o fim da segunda frase, foi obtida uma palavra completa. Junta-se- à nova frase.
- Repetir os procedimentos anteriores, percorrendo as duas frases escritas até ao final.

Solução PASCAL

```
ROGRAM EX6;
        USES CRT;
        VAR     frase1, frase2, nova_frase, letra: STRING;
                tam1, tam2, i, j : INTEGER;
        BEGIN
                CLRSCR;
                WRITELN('Escreva a primeira frase ');
                READLN(frase1);
                WRITELN('Escreva a segunda frase ');
                READLN(frase2);
                tam1 := LENGTH(frase1);
                tam2 := LENGTH(frase2);
                i := 1;
                j := 1;
                WHILE ((i<=tam1) OR (j<=tam2)) DO
                BEGIN
                        IF (i<=tam1)
```

```
            THEN BEGIN
                letra := COPY(frase1,i,1);
                WHILE (letra <> ' ') AND (i<=tam1) DO
                BEGIN
                    nova_frase := nova_frase + letra;
                    i := i + 1;
                    letra := copy(frase1,i,1);
                END;
                nova_frase:=nova_frase + ' ';
                i := i + 1;
            END;
            IF (j<=tam2)
            THEN BEGIN
                letra := COPY(frase2,j, 1);
                WHILE (letra <> ' ') AND (j<=tam2)  DO
                BEGIN
                    nova_frase := nova_frase + letra;
                     j := j + 1;
                    letra := COPY(frase2, j, 1);
                END;
                nova_frase :=nova_frase + ' ';
                j := j + 1;
            END;
        END;
        WRITELN(nova_frase);
        READLN;
END.
```


Exercício 7

Faça um programa que receba uma frase e coloque as palavras da frase em ordem alfabética.

Exemplo:

Entrada: A informática está em constante evolução

Saída: A constante em está evolução informática

Solução

- Escreva uma frase.
- Ver o tamanho da frase escrita.
- Percorrer a frase, vendo caractere a caractere.
- Cada vez que encontrar um espaço em branco ou o fim da frase, foi obtida uma palavra completa.
- Se a nova frase (onde as palavras ficarão em ordem alfabética) estiver vazia, copiar nela a palavra obtida.
- Caso contrário, percorrer a nova frase até encontrar a posição adequada para colocar a palavra extraída da frase escrita.
- Caso a palavra extraída da frase escrita seja maior do que todas as palavras que já estão armazenadas na nova frase, juntar essa palavra ao final da nova frase.

Solução PASCAL

```
PROGRAM EX7;
        USES CRT;
        VAR     frase, frase_nova, palavra1, palavra2: STRING;
                achou, comeco, tam1, tam2, i, j: INTEGER;
        BEGIN
                CLRSCR;
                WRITELN('Escreva uma frase ');
                READLN(frase);
                frase := frase + ' ';
                frase_nova := '';
```

```
palavra1:='';
palavra2:='';
tam1:=length(frase);
FOR i :=1 TO tam1 DO
BEGIN
        palavra2:='';
        IF (COPY(frase,i,1) <> ' ')
        THEN BEGIN
                palavra1:= palavra1 + COPY(frase,i, 1);
        END
        ELSE BEGIN
                tam2:= LENGTH(frase_nova);
                IF (tam2=0)
                THEN begin
                        frase_nova := palavra1 + ' ';
                END
                ELSE BEGIN
                        achou:=0;
                        j := 1;
                        comeco := 1;
                        WHILE  (j<=tam2) AND (achou=0) DO
                        BEGIN
                                IF (COPY(frase_nova,j,1) <> ' ')
                                THEN BEGIN
                                        palavra2 := palavra2 +
                                        COPY(frase_nova, j, 1);
                                        j := j + 1;
                                END
                                ELSE BEGIN
                                        comeco := j - length(palavra2);
                                        IF palavra1 < palavra2
                                        THEN BEGIN
```


```
                                        achou:=1;
                                        palavra1:=palavra1+' ';
                                    insert (palavra1,frase_nova,comeco);

                                        END
                                        ELSE palavra2 := '';
                                        j := j + 1;
                                    END;
                                END;
                                if achou = 0
                                then frase_nova:=frase_nova + palavra1+' ';
                            END;
                            palavra1:='';
                        END;
                END;
                WRITELN(frase_nova);
                READLN;
END.
```

Exercício 8

Faça um programa que receba uma frase, calcule e mostre a quantidade de vezes que a palavra AULA aparece na frase escrita.

Solução

- Escrever uma frase.
- Ver o tamanho da frase escrita.
- Percorrer a frase, vendo caractere a caractere.
- Cada vez que encontrar um espaço em branco ou o fim da frase, foi obtida uma palavra completa.
- Se a palavra completa for igual à cadeia AULA, acrescentar um no contador.

Solução PASCAL

```
PROGRAM EX8;
        USES CRT;
        VAR     frase,palavra: STRING;
                tam, i, qtde: INTEGER;
        BEGIN
                CLRSCR;
                WRITELN('Escreva uma frase');
                READLN(frase);
                tam := LENGTH(frase);
                FOR i:=1 TO tam DO
                frase[i]:=UPCASE(frase[i]);
                FOR i:=1 TO tam DO
                BEGIN
                        palavra := COPY(frase,i,4);
                        if (palavra = 'AULA')
                        THEN BEGIN
                                IF (i+3 = tam) OR (COPY(frase,i+4,1) = ' ')
                                THEN qtde := qtde + 1;
                        END;
                END;
                WRITELN('Quantidade de palavra AULA na frase = ',qtde);
                READLN;
        END.
```

Exercício 9

Faça um programa que receba uma frase e uma palavra, calcule e mostre a quantidade de vezes que a palavra escrita aparece na frase.

Exemplo:

Frase: EU ESTOU NA ESCOLA E A ESCOLA É BOA.

Palavra: ESCOLA

Resposta: A palavra ESCOLA apareceu 2 vezes na frase.

Solução

- Escrever uma frase.
- Ver o tamanho da frase escrita.
- Percorrer a frase, vendo caractere a caractere.
- Cada vez que encontrar um espaço em branco ou o fim da frase, foi obtida uma palavra completa.
- Se a palavra completa for igual à palavra escrita, acrescentar um no contador

Solução PASCAL

```
PROGRAM EX9;
    USES CRT;
    VAR   frase, palavra, procura: STRING;
          tam, tam2, i, qtde: INTEGER;
    BEGIN
          CLRSCR;
          WRITELN('Escreva uma frase');
          READLN(frase);
          WRITELN('Escreva a palavra');
          READLN(palavra);
          tam := LENGTH(frase);
          tam2 := LENGTH(palavra);
          FOR i:=1 TO tam DO
          frase[i]:=UPCASE(frase[i]);
          FOR i:=1 TO tam2 DO
          palavra[i]:=UPCASE(palavra[i]);
          FOR i:=1 TO tam DO
          BEGIN
                procura := COPY(frase,i,tam2);
                if (palavra = procura)
                THEN BEGIN
                      IF (i+tam2-1 = tam) OR (COPY(frase,i+tam2,1) = ' ')
                      THEN qtde := qtde + 1;
```

```
            END;
        END;
        WRITELN('Quantidade de palavra ',palavra,' na frase = ',qtde);
        READLN;
END.
```

Exercício 10

Faça um programa que receba uma frase e troque a palavra ALUNO por ESTUDANTE e a palavra ESCOLA por UNIVERSIDADE.

Exemplo: EU SOU ALUNO DA ESCOLA

Saída: EU SOU ESTUDANTE DA UNIVERSIDADE

Solução

- Escrever uma frase.
- Ver o tamanho da frase escrita.
- Percorrer a frase, vendo caractere a caractere.
- Cada vez que encontrar um espaço em branco ou o fim da frase, foi obtida uma palavra completa.
- Se a palavra completa for igual à cadeia ALUNO, esta deverá ser removida da frase escrita e no seu lugar deverá ser inserida a cadeia ESTUDANTE.
- Percorrer a frase, vendo caractere a caractere.
- Cada vez que encontrar um espaço em branco ou o fim da frase, foi obtida uma palavra completa.
- Se a palavra completa for igual à cadeia ESCOLA, esta devera ser removida da frase escrita e no seu lugar deverá ser inserida a cadeia UNIVERSIDADE.

Solução PASCAL

```
PROGRAM EX10;
    USES CRT;
    VAR   frase, nova: STRING;
          palavra: STRING;
          tam, i: INTEGER;
```


```
BEGIN
        CLRSCR;
        WRITELN('Escreva uma frase');
        READLN(frase);
        tam := LENGTH(frase);
        FOR i:= 1 TO tam DO
        BEGIN
                frase[i]:=UPCASE(frase[i]);
        END;
        i:=1;
        WHILE i<=tam-4 DO
        BEGIN
                palavra := COPY(frase,i,5);
                if palavra = 'ALUNO'
                THEN BEGIN
                        DELETE(frase,i,5);
                        INSERT('ESTUDANTE',frase,i);
                END;
                tam := LENGTH(frase);
                i:= i + 1;
        END;
        i:=1;
        WHILE i<=tam-5 DO
        BEGIN
                palavra := COPY(frase,i,6);
                if palavra = 'ESCOLA'
                THEN BEGIN
                        DELETE(frase,i,6);
                        INSERT('UNIVERSIDADE',frase,i);
                END;
                tam := LENGTH(frase);
                i:= i + 1;
```

```
        END;
        WRITELN('Nova frase = ',frase);
        READLN;
END.
```

Exercício 11

Faça um programa que receba uma frase e, a cada ocorrência da palavra TECLADO, insira o texto OU RATO.

Exemplo:

Frase: PODE-SE UTILZAR O TECLADO PARA ENTRADA DE DADOS.

Resposta: PODE-SE UTILZAR O TECLADO OU RATO PARA ENTRADA DE DADOS.

Solução

- Escrever uma frase.
- Ver o tamanho da frase escrita.
- Percorrer a frase, vendo caractere a caractere.
- Cada vez que encontrar um espaço em branco ou o fim da frase, foi obtida uma palavra completa.
- Se a palavra completa for igual à cadeia TECLADO, devera ser inserida (junção) após a sua ocorrência à cadeia OU RATO.
-

Solução PASCAL

```
PROGRAM EX11;
    USES CRT;
    VAR     frase, nova: STRING;
            palavra: STRING;
            tam, i: INTEGER;
    BEGIN
            CLRSCR;
            WRITELN('Escreva uma frase');
            READLN(frase);
            tam := LENGTH(frase);
```


```
FOR i:= 1 TO tam DO
BEGIN
        frase[i]:=UPCASE(frase[i]);
END;
i:=1;
WHILE i<=tam-4 DO
BEGIN
        palavra := COPY(frase,i,7);
        if palavra = 'TECLADO'
        THEN BEGIN
                INSERT(' OU RATO',frase,i+7);
        END;
        tam := LENGTH(frase);
        i:= i + 1;
END;
WRITELN('Nova frase = ',frase);
READLN;
END.
```

Exercício 12

Faça um programa para criptografar uma frase dada pelo utilizador, ou seja, a criptografia deverá inverter a frase.

Exemplo:

Frase: EU ESTOU NA ESCOLA

Saída: ALOCSE AN UOTSE UE

Solução

- Escrever uma frase.
- Ver o tamanho da frase escrita.
- Percorrer a frase do último caractere ao primeiro, copiando-o para a nova cadeia.

Solução PASCAL

```
PROGRAM EX12;
        USES CRT;
        VAR     frase, nova: STRING;
                tam, i: INTEGER;
        BEGIN
                CLRSCR;
                WRITELN('Escreva uma frase');
                READLN(frase);
                tam := LENGTH(frase);
                nova:='';
                FOR i:= tam DOWNTO 1 DO
                BEGIN
                        nova:=nova+copy(frase,i,1);
                END;
                WRITELN('Nova frase = ',nova);
                READLN;
        END.
```

Exercício 13

Faça um programa para criptografar uma frase dada pelo utilizador, ou seja, a criptografia devera inverter cada palavra da frase.

Exemplo:

Frase: EU ESTOU NA ESCOLA

Saída: UE UOTSE AN ALOCSE

Solução

- Escrever uma frase.
- Ver o tamanho da frase Escrita.
- Percorrer a frase, vendo caractere a caractere.
- Cada vez que encontrar um espaço em branco ou o fim da frase, foi obtida uma palavra completa.

- Ver o tamanho da palavra completa.
- Percorrer a palavra completa do ultimo caractere ao primeiro, copiando-o para a nova palavra.
- Juntar a nova palavra completa na nova frase.

Solução PASCAL

```
PROGRAM EX13;
    USES CRT;
    VAR   palavra, frase, nova, letra: STRING;
          tam, tam2, j, i: INTEGER;
    BEGIN
          CLRSCR;
          WRITELN('Escreva uma frase');
          READLN(frase);
          tam := LENGTH(frase);
          nova:='';
          palavra:='';
          i:=1;
          WHILE i<=tam DO
          BEGIN
                letra:=COPY(frase,i,1);
                if (letra <> ' ')
                THEN BEGIN
                      palavra:=palavra+letra;
                END;
                IF (letra = ' ') OR (i = tam)
                THEN BEGIN
                      tam2:=LENGTH(palavra);
                      FOR j:=tam2 DOWNTO 1 DO
                      BEGIN
                            letra:=COPY(palavra,j,1);
                            nova:= nova+letra;
```

```
                    END;
                    IF i <> tam
                    THEN nova:=nova+' ';
                    palavra:='';
                END;
                i:=i+1;
            END;
            WRITELN('Nova frase = ',nova);
            READLN;
    END.
```

Exercício 14

Faça um programa que receba uma frase com letras minúsculas e converta a primeira letra de cada palavra da frase para maiúscula.

Exemplo:

Entrada: fazer exercícios faz bem.

Saída: Fazer Exercícios Faz Bem.

Solução

- Escrever uma frase.
- Ver o tamanho da frase escrita.
- Converter o primeiro caractere da frase para maiúsculo.
- Ver a frase, vendo caractere a caractere.
- Cada vez que encontrar um espaço em branco, converter o próximo caractere para maiúsculo.

Solução PASCAL

```
PROGRAM EX14;
    USES CRT,STRINGS;
    VAR   frase, nova, letra: STRING;
          tam,i,qtde: INTEGER;
```


```
BEGIN
        qtde:=0;
        WRITELN('Escreva uma frase');
        READLN(frase);
        tam:=LENGTH(frase);
        i:=1;
        WHILE i<=tam DO
        BEGIN
                letra:=COPY(frase,i,1);
                if letra = ' '
                THEN BEGIN
                        nova:= nova+' '+UPCASE(frase[i+1]);
                        i := i + 2;
                END
                ELSE BEGIN
                        IF i = 1
                        THEN letra:=UPCASE(frase[1]);
                        nova:=nova+letra;
                        i:=i+1;
                END;
        END;
        WRITELN('Nova frase = ',nova);
        READLN;
END.
```

Exercício 15

Faça um programa que receba um nome e crie como saída o nome escrito bem como o seu login. Lembre-se de respeitar as letras minúsculas e maiúsculas, já que o login será sempre com letras minúsculas. A regra para criação do login é: a primeira letra do nome e, caso exista apenas um sobrenome, deve-se acrescenta-lo; caso existam dois sobrenomes, deve-se criar a primeira letra do nome, mais a primeira letra do primeiro sobrenome, seguido do último sobrenome: caso existam três ou mais sobrenomes, deve-

se proceder como na situação anterior, considerando o nome, o primeiro sobrenome o ultimo sobrenome.

Exemplos:

Nome: Pedro Hansdorf

Login: phansdorf

Nome: Robson Soares Silva

Login: rssilva

Nome: Jaqueline Oliveira Fernandes Espanhola

Login: joespanhola

Solução

- Escrever um nome.
- Converter todas as letras do nome escrito para minúsculas.
- Inicializar o login (nova cadeia) com a primeira letra do primeiro nome.
- Verificar se o nome escrito tem apenas um espaço em branco. Caso tenha, juntar ao login os caracteres da posição posterior ao espaço até ao fim do nome escrito.
- Se o nome tiver mais de um espaço em branco, procurar o primeiro espaço e juntar ao login o caractere da posição posterior à posição do espaço; procurar o ultimo espaço e juntar ao login os caracteres da posição posterior a esse ultimo espaço até ao fim do nome escrito.

Solução PASCAL

```
PROGRAM EX15;
        USES    CRT,STRINGS;
        VAR     NOME:ARRAY[0..80] OF CHAR;
                LOGIN:STRING;
                ACHOU, I, TAM, POSI, QTDE: INTEGER;
        BEGIN
                CLRSCR;
                WRITELN('Escreva o nome');
```

```
READLN(NOME);
TAM:=LENGTH(NOME);
STRLOWER(NOME);
LOGIN:='';
LOGIN:=COPY(NOME,0,1);
QTDE:=0;
FOR I:=0 TO TAM DO
BEGIN
        IF NOME[I] = ' '
        THEN BEGIN
                QTDE:=QTDE+1;
                POSI:=I;
        END;
END;
IF QTDE = 1
THEN LOGIN:=LOGIN+COPY(NOME,POSI+2,TAM-POSI)
ELSE BEGIN
        ACHOU:=0;
        I:=0;
        WHILE (ACHOU = 0) DO
        BEGIN
                IF NOME[I] = ' '
                THEN BEGIN
                        ACHOU:=1;
                        POSI:=I;
                END;
                I:=I+1;
        END;
        LOGIN:=LOGIN+COPY(NOME,POSI+2,1);
        ACHOU:=0;
        I:=TAM;
        WHILE (ACHOU = 0) DO
```

```
                BEGIN
                        IF NOME[I] = ' '
                        THEN BEGIN
                                ACHOU:=1;
                                POSI:=I;
                        END;
                        I:=I-1;
                END;
                LOGIN:=LOGIN+COPY(NOME,POSI+2,TAM-POSI);
        END;
        WRITELN('Nome = ',nome);
        WRITELN('Login = ',login);
        READLN;
END.
```

Exercício 16

Faça um programa que receba uma palavra e verifique se constitui um palíndromo, ou seja, se a palavra escrita do fim para o início fica igual à palavra escrita do início para o fim.

Exemplos:

ANA

MIRIM

Solução

- Escrever uma palavra.
- Criar uma nova palavra, que será a palavra escrita invertida.
- Comparar a palavra escrita com a palavra invertida.
- Se as palavras forem iguais, trata-se de palíndromo; caso contrario, não se trata de palíndromo.

Solução PASCAL

```
PROGRAM EX16;
        USES CRT;
        VAR     PALAVRA1:STRING;
                TAM, I, J, ACHOU: INTEGER;
        BEGIN
                CLRSCR;
                WRITELN('ESCREVA A PALAVRA');
                READLN(PALAVRA1);
                TAM:=LENGTH(PALAVRA1);
                J:=TAM;
                ACHOU:=0;
                FOR I:=1 TO TAM DO
                BEGIN
                        IF (PALAVRA1[I] <> PALAVRA1[J])
                        THEN ACHOU:=1;
                        J:=J-1;
                END;
                IF ACHOU = 0
                THEN WRITELN(PALAVRA1,' □ PALINDROME')
                ELSE WRITELN(PALAVRA1,' NAO □ PALINDROME');
                READLN;
        END.
```

ver símbolos antes de PALINDROME'

Exercício 17

Faça um programa que receba uma frase e uma palavra. Caso a frase contenha a palavra ESCOLA, deverá substitui-la pela palavra escrita.

Exemplo:

Frase: EU MORO PERTO DE UMA ESCOLA. MAS ESTA ESCOLA NÃO É A MELHOR.

Palavra: PADARIA

Resposta: EU MORO PERTO DE UMA PADARIA. MAS ESTA PADARIA NÃO É A MELHOR.

Solução

- Escrever uma frase.
- Escrever a palavra que substituirá a palavra ESCOLA.
- Converter a palavra e a frase Escritas para letras maiúsculas.
- Ver o tamanho da frase.
- Percorrer a frase, vendo de seis em seis caracteres, tendo em vista que a palavra ESCOLA possui seis caracteres, e criar uma nova palavra.
- Comparar a nova palavra com a palavra ESCOLA.
- Quando encontrar a palavra ESCOLA, apaga-la e no seu lugar inserir a palavra escrita.
- Depois de substituir, atualizar o tamanho da frase a ser percorrida.

Solução PASCAL

```
PROGRAM EX17;
        USES CRT;
        VAR     frase, palavra_escrita, palavra_frase: STRING;
                tam, i: INTEGER;
        BEGIN
                CLRSCR;
                WRITELN('Escreva uma frase');
                READLN(frase);
                FOR i:=1 TO LENGTH(frase) DO
                frase[i]:=UPCASE(frase[i]);
                WRITELN('Escreva a palavra para substituição');
                READLN(palavra_escrita);
                FOR i:=1 to LENGTH(palavra_escrita) DO
                palavra_escrita[i]:=UPCASE(palavra_ escrita [i]);
                tam := LENGTH(frase);
                i:=1;
                WHILE i<= tam DO
                BEGIN
                        palavra_frase := COPY(frase,i,6);
```

```
                    if palavra_frase = 'ESCOLA'
                    THEN BEGIN
                            DELETE(frase,i,6);
                            INSERT(palavra_ escrita, frase,i);
                    END;
                    tam := LENGTH(frase);
                    i:= i + 1;
            END;
            WRITELN(frase);
    READLN;
END
```

Exercícios Propostos

Exercício 1

8. Faça um programa que se comporte como vírus, ou seja, que duplique cada uma das palavras escritas pelo utilizador.

Exemplo:

Frase: EU ESTOU NA ESCOLA

Saída: EU EU ESTOU ESTOU NA NA ESCOLA ESCOLA

Exercício 2

Faça um programa que receba uma frase. Caso na frase apareça o nome de um mês do ano por extenso, devera substitui-lo pelo seu número correspondente, como no exemplo.

Exemplo:

Frase: NO MÊS DE JANEIRO FAZ CALOR.

Nova frase: NO MES 01 FAZ CALOR.

Exercício 3

Faça um programa que receba o nome completo de uma pessoa e mostre os nomes intermediários entre o primeiro nome e o ultimo sobrenome abreviados.

Exemplo:

Nome: Maria Silva Costa

Saída: Maria S. Costa

Nome: João Carlos Gomes Marques

Saída: João C. G. Marques

Exercício 4

Faça um programa que receba o nome completo de uma pessoa e reescreva-o de acordo com o exemplo a seguir.

Exemplo:

Nome: Maria Silva Costa

Saída: Costa, M. S.

Nome: João Carlos Gomes Marques

Saída: Marques, J. C. G.

Exercício 5

Faça um programa que receba um nome e crie um login e uma senha. O login deverá ser composto pela primeira letra de cada nome em letras maiúsculas e as mesmas letras minúsculas; a senha será composta pela representação ASCII de cada letra do login.

Exemplo:

Nome: Ana Beatriz Costa

Login: ABCabc

Senha: 656667979899

Exercício 6

Faça um programa para criptografar uma frase em que cada caractere devera ser substituído pelo caractere que está três posições à sua frente na tabela ASCII. Os Três últimos caracteres da tabela ASCII deverão ser substituídos pelos três primeiros.

Exemplo:

BONECO ZABUMBA

ERQHFR CDEXPED

Exercício 7

Faça um programa que receba um verbo regular terminado em AR e mostre a sua conjugação no presente.

Exemplo:

Verbo: andar

Eu ando

Tu andas

Ele anda

Ela anda

Nos andamos

Vos andais

Eles andam

Elas andam

Exercício 8

Faça um programa que receba uma frase e conte quantos verbos existem nela, considerando que os verbos terminam em R.

Registos

Definição de Registos

Registos são estruturas de dados capazes de agregar várias informações. Desta maneira, os programadores podem criar novos tipos de dados, não se limitando apenas à utilização dos tipos de dados primitivos fornecidos pelas linguagens de programação.
Cada informação contida num registo é chamada de campo. Os campos podem ser de diferentes tipos primitivos ou, ainda, podem representar outros registos. Por isso, os registos são conhecidos como variáveis compostas heterogéneas.

Declaração de registos em algoritmos

A declaração de uma variável registo é o primeiro passo para sua utilização. Isso significa especificar o nome dos seus campos com os seus respetivos tipos.
Como acontece com qualquer outro tipo de dado, uma variável registo pode ser simples, um vetor ou uma matriz. A sintaxe correta para a declaração de uma variável registo é apresentada a seguir.

Declare nome_da_variavel_registo REGISTO (nome_campo1 TIP0_DO_CAMP01, nome_campo2 TIP0_D0_CAMP02, ..., nome_campoN TIPO_DO_CAMPON)

Exemplo 1

Declare conta REGISTO (num, saldo NUMERICO, nome LITERAL)

Neste exemplo, foi declarada uma variável chamada conta. Esta variável é um registo composto por três campos: num e saldo, capazes de armazenar valores numéricos, e nome, capaz de armazenar um valor literal.

Exemplo 2

Declare conta[3] REGISTO (num, saldo NUMERICO, nome LITERAL)

Neste exemplo, foi declarada uma variável chamada conta. Esta variável é um vetor de três posições (de 1 a 3). Em cada posição será armazenado um registo, composto por três campos: num e saldo, capazes de armazenar valores numéricos, e nome, capaz de armazenar um valor literal.

Acesso aos campos de um registo em algoritmos

Ao considerar que um registo contem várias informações, para aceder a um campo individualmente é necessário indicar o nome da variável e o nome do campo desejado, separados por um ponto. Observe o exemplo a seguir:

nome_da_variável_do_tipo_ registo.nome_do_campo

Exemplo 1

conta.num ← 12

O exemplo coloca o número 12 no campo *num* da variável registo denominada conta.

Exemplo 2

conta [2] .num ← 13

O exemplo armazena o número 13 no campo *num* da segunda posição da variável registo denominada conta.

Exemplo 3

ESCREVA conta[3, 2].num

O exemplo mostra o conteúdo do campo *num*, localizado na linha 3, coluna 2 da variável registo denominada conta.

Declaração de registos em PASCAL

Em PASCAL, a utilização de registos requer dois passos: a definição da estrutura do registo, em que se utilizam as palavras reservadas TYPE e RECORD, e a declaração da variável registo, que segue a sintaxe de declaração de qualquer variável.

```
TYPE nome_da_variavel_registo =    RECORD
                                   Campo1:tipo1;
```

```
                    campo2:tipo2;
                        • • •
                    campoN:tipoN;
                    END;
VAR nome_da_variavel: nome_da_variavel_registo;
```

Exemplo 1

```
TYPE REGISTO =      RECORD
                    num : INTEGER;
                    nome : STRING[35];
                    saldo : REAL;
                    END;
VAR conta: REGISTO;
```

No exemplo, um registo chamado REGISTO foi definido. Isto significa que o programa poderá utilizar um novo tipo de dado. Depois disso, a variável conta foi declarada como REGISTO. Portanto, *conta* terá espaço para armazenar *num*, *nome* e *saldo*.

Exemplo 2

```
TYPE EXEMPLO =      RECORD
                    num : INTEGER;
                    nome : STRING[35];
                    saldo : REAL;
                    END;
VAR conta: ARRAY[1..15] OF EXEMPLO;
```

No exemplo, um registo chamado EXEMPLO foi definido. Isto significa que o programa poderá utilizar um novo tipo de dado. Depois disso, a variável conta foi declarada como um vetor de 15 posições. Cada posição será um registo EXEMPLO. Portanto, cada posição do vetor conta terá espaço para armazenar *num, nome* e *saldo*.

Acesso aos campos de um registro em PASCAL

Considerando que uma variável do tipo registo contém várias informações, é necessária uma maneira de acede-las individualmente. Assim, devemos indicar o nome da variável e também o nome do campo desejado, separados por um ponto. Observe o exemplo a seguir.

```
nome_da_variavel_do_tipo_registro.nome_do_campo
```

Exemplo 1

```
conta.num:= 12;
```

O exemplo mostra como armazenar o número 12 no campo *num* da variável registo denominada *conta*.

Exemplo 2

```
conta[2].num:= 13;
```

O exemplo mostra como armazenar o número 13 no campo *num* da posição 2 da variável registo denominada *conta*. É importante observar que o acesso começa a fazer referência ao vetor, depois é definida uma das suas posições e, finalmente, o campo desejado.

Exemplo 3

```
writeln(conta [3, 2] .num);
```

O exemplo mostra o conteúdo do campo *num*, localizado na linha 3, coluna 2 da variável registo denominada *conta*. É importante observar que o acesso começa a fazer referência à matriz, depois são especificados os números da linha e da coluna e, finalmente, o campo desejado.

Exercícios Resolvidos

Exercício 1

Faça um programa que realize o registo de contas bancarias com as seguintes informações: número da conta, nome do cliente e saldo. O banco permitirá o registo de apenas quinze contas e não poderá haver mais que uma conta com o mesmo número. Crie o menu de opções a seguir.

Menu de opções:

1. Registar contas.

2. Visualizar todas as contas de determinado cliente.

3. Excluir a conta com menor saldo (supondo a não existência de saldos iguais).

4. Sair.

Solução Algoritmo

```
ALGORITMO
        DECLARE     conta [15] REGISTO (num, saldo NUMERICO, nome LITERAL)
                    i, op, posi, achou, num_conta,menor_saldo NUMERICO
                    nome_cliente LITERAL
        PARA i 1 ATE 15 FACA
        INICIO
                conta [i] .num ← 0
                conta [i] .nome ← “
                conta[i].saldo ← 0
        FIM
        posi ← 1
        REPITA
ESCREVA “Menu de Opções”
ESCREVA “1 – Registar contas”
ESCREVA “2 - Visualizar todas as contas de determinado cliente”
ESCREVA “3 - Excluir conta de menor saldo”
ESCREVA “4 - Sair”
ESCREVA “Escolha a sua opção”
LEIA op
SE op < 1 OU op > 4
ENTAO ESCREVA “Opção Invalida”
SE op = 1
                ENTAO INICIO
                        SE posi > 15
                        ENTAO ESCREVA “Todas as contas já foram registadas!”
```

```
        SENAO INICIO
                achou ← 0
                ESCREVA “Escreva o numero da conta a ser incluída”
                LEIA num_conta
                PARA i ← 1 ATE posi - 1 FAÇA
                INICIO
                        SE num_conta = conta [i] .num
                        ENTAO achou ← 1
                FIM
                SE achou = 1
                ENTAO ESCREVA “Já existe conta regstada com esse
                  numero”
                SENAO INICIO
                        conta[posi] ← num_conta
                        ESCREVA “Escreva o nome do cliente”
                        LEIA conta[posi].nome
                        ESCREVA “Escreva o saldo do cliente”
                        LEIA conta[posi].saldo
                        ESCREVA “Conta registada com sucesso”
                        posi ← posi + 1
                FIM
        FIM
FIM
SE Op = 2
ENTAO INICIO
        ESCREVA “Escreva o nome do cliente a ser consultado”
        LEIA nome_cliente
        achou ← 0
        PARA i ← 1 ATE posi - 1 FAÇA
INICIO
        SE conta[i].nome = nome_cliente
        ENTAO INICIO
```

```
                    ESCREVA conta[i] .num, conta [i] .saldo
                    achou ← 1
                FIM
            FIM
            SE achou = 0
            ENTAO ESCREVA “Não existe conta registada para este cliente”
FIM
SE op = 3
ENTAO INICIO
            SE posi = 1
            ENTAO “Nenhuma conta foi registada”
            SENAO INICIO
                menor_saldo ← conta(1).saldo
                achou ← 1
                i ← 2
                ENQUANTO i < posi FAÇA
                INICIO
                    SE conta [i].saldo < menor_saldo
                    ENTAO INICIO
                        menor_saldo ← conta[i].saldo
                        achou ← i
                    FIM
                    i ← i + 1
                FIM
                PARA i ← achou ATE posi - 1 FAÇA
                INICIO
                    conta[i-1] .num ← conta [i] .num
                    conta[i-1].nome ← conta[i].nome
                    conta [i-1] .saldo ← conta[i] .saldo
                FIM
                ESCREVA “Conta excluída com sucesso”
                posi ← posi - 1
```


```
            FIM
        FIM
        ATE op = 4
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX1;
    USES CRT;
    TYPE REGISTO =      RECORD
                        num : INTEGER;
                        nome : STRING[35];
                        saldo : REAL;
                        END;
    VAR conta:      ARRAY[1..15] OF REGISTO;
                    i, op, posi, achou, num_conta : INTEGER;
                    saldo_cliente, menor_saldo : REAL;
                    nome_cliente :STRING[35];
    BEGIN
        FOR i := 1 TO 15 DO
        BEGIN
            conta[i].num := 0;
            conta[i].nome:='';
            conta[i].saldo := 0;
        END;
        REPEAT
            CLRSCR;
    WRITELN('Menu de Opções');
    WRITELN('1 - Registar Contas');
    WRITELN('2 - Visualizar todas as contas de um determinado      cliente');
    WRITELN('3 - Excluir conta de menor saldo');
    WRITELN('4 - Sair');
    WRITELN('Escreva a sua opção');
```

```
READLN(op);
IF (op < 1) OR (op > 4)
THEN WRITELN('Opção Inválida');
IF op = 1
THEN BEGIN
                achou := 0;
                WRITELN('escreva o número da conta a ser incluída');
                READLN(num_conta);
                FOR i := 1 TO 15 DO
                BEGIN
                        IF num_conta = conta[i].num
                        THEN achou := 1;
                END;
                IF achou = 1
                THEN WRITELN('J  existe conta registada com esse
        número')
                ELSE BEGIN
                        posi := 0;
                        i := 1;
                        WHILE i <= 15 DO
                BEGIN
                        IF conta[i].num = 0
                        THEN BEGIN
                                posi := i;
                                i := 16;
                        END;
                        i := i + 1;
                END;
                IF posi = 0
                THEN WRITELN('Impossível registar novas contas')
                ELSE BEGIN
                        WRITELN('Escreva o nome do cliente');
```


```
                    READLN(nome_cliente);
                    WRITELN('Escreva o saldo do cliente');
                    READLN(saldo_cliente);
                    conta[posi].num := num_conta;
                    conta[posi].nome := nome_cliente;
                    conta[posi].saldo := saldo_cliente;
                    WRITELN('Conta registada com sucesso');
                END;
            END;
            READLN;
        END;
    IF op = 2
    THEN BEGIN
            WRITELN('Escreva o nome do cliente a ser consultado');
            READLN(nome_cliente);
            achou := 0;
            FOR i := 1 TO 15 DO
            BEGIN
                IF conta[i].nome = nome_cliente
                THEN BEGIN
                    WRITELN('Nº CONTA  SALDO');
                    WRITELN(conta[i].num, '        ',conta[i].saldo:5:2);
                    achou := 1;
                END;
            END;
            IF achou = 0
            THEN WRITELN('Não existe conta registada para este cliente');
            READLN;
    END;
    IF op = 3
    THEN BEGIN
            i := 1;
```

```
                achou := 0;
                WHILE i <= 15 DO
                BEGIN
                        IF conta[i].num <> 0
                        THEN BEGIN
                                menor_saldo := conta[i].saldo;
                                achou := 1;
                                posi := i;
                                i := 16;
                        END;
                        i := i + 1;
                END;
                IF achou = 0
                THEN WRITELN('Nenhuma conta foi registada)
                ELSE BEGIN
                        FOR i := 1 TO 15 DO
                        BEGIN
                                IF (conta[i].saldo< menor_saldo) AND
                                (conta[i].num <> 0)
                                THEN posi := i;
                        END;
                        WRITELN('A conta de número ', conta[posi].num,' foi
                        excluída com sucesso');
                        conta[i].num := 0;
                        conta[i].nome := '';
                        conta[i].saldo := 0;
                END;
                READLN;
        END;
        UNTIL op = 4;
END.
```


Exercício 2

Uma empresa prestadora de serviços armazena informações sobre os serviços prestados. Sabe-se que a empresa pode realizar no máximo três serviços diariamente. É de interesse da direção manter um histórico mensal (30 dias) sobre os serviços prestados.

A empresa realiza quatro tipos de serviços: 1) pintura; 2) jardinagem; 3) limpeza e 4) reparações.

Cada serviço realizado deve ser registado com as seguintes informações: numero do serviço, valor do serviço, código do serviço e código do cliente.

Registe os quatro tipos de serviços (código e descrição) que a empresa poderá realizar. Para isso, utilize um vetor de quatro posições.

O programa devera mostrar o seguinte menu de opções:

1. Registar os tipos de serviços.
2. Registar os serviços prestados.
3. Mostrar os serviços prestados em determinado dia.
4. Mostrar os serviços prestados dentro de um intervalo de valor.
5. Mostrar um relatório geral (separado por dia), que exiba, inclusive, a descrição do tipo do serviço.
6. Finalizar.

Para a opção 1: deve-se registar os tipos de serviços oferecidos pela empresa, com código e descrição.

Para a opção 2: deve considerar-se que deverão ser registados os serviços prestados ao longo do mês.

Por dia podem ser registados, no máximo, três serviços prestados.

Utilize uma matriz capaz de armazenar em cada posição todas as informações referentes a um serviço prestado. Cada linha representa um dia do mês. Dessa maneira, considere a matriz com dimensão 30 x 3.

Solicite o dia em que o serviço foi prestado e as demais informações.

Lembre-se de que a empresa só pode prestar os serviços que já tenham sido registados no vetor de tipo de serviços.

Caso o utilizador escreva um código de tipo de serviço invalido, o programa devera mostrar uma mensagem de erro.

Quando o utilizador tentar registar mais de três serviços prestados no mesmo dia, também deverá mostrar uma mensagem de erro.

Para a opção 3: o programa deverá receber o dia que se deseja consultar e mostrar os respetivos serviços prestados.

Para a opção 4: o programa devera receber o valor mínimo e o valor máximo e mostrar os serviços prestados que estiverem nesse intervalo.

Para a opção 5: o programa deverá mostrar todos os serviços prestados, conforme o exemplo a seguir.

DIA 01				
N.º de Serviço	**Valor do Serviço**	**Código do serviço**	**Descrição**	**Código do cliente**
100	**$ 200,00**	**1**	**Pintura**	**1**
150	**$ 100,00**	**3**	**Limpeza**	**5**
DIA 02				
N.º de Serviço	**Valor do Serviço**	**Código do serviço**	**Descrição**	**Código do cliente**
301	**$ 600,00**	**4**	**Reparações**	**3**
280	**$ 352,00**	**1**	**Pintura**	**2**

Solução Algoritmo

```
ALGORITMO
        DECLARE     tipos [4] REGISTO (cod NUMERICO, desc LITERAL)
                    serv[30,3] REGISTRO (num, valor, cod_serv, cod_cliente
                NUMERICO)
                    i, j, op, codigo_serv, achou, conta_tipo NUMERICO
                    dia, codigo_cliente, valor_serv, num_serv, valida NUMERICO
                    valor_inicial, valor_final, k NUMERICO
                    desc serv LITERAL
        conta_tipo ← 1
        PARA i ← 1 ATE 30 FAÇA
        INICIO
                PARA j 1 ATE 3 FAÇA
                INICIO
```

```
                serv[i, j].num ← 0
                serv[i, j] .valor ← 0
                serv[i, j].cod_serv ← 0
                serv[i, j].cod_cliente ← 0
        FIM
FIM
REPITA
        ESCREVA “Menu de Opções”
        ESCREVA “1 - Registar os tipos de serviços”
        ESCREVA “2 - Registar os serviços prestados”
        ESCREVA “3 - Mostrar os serviços prestados em determinado dia”
        ESCREVA “4 - Mostrar os serviços prestados dentro de um intervalo
         de valor”
        ESCREVA “5 - Mostrar um relatório geral, separado por dia”
        ESCREVA “6 - Finalizar”
        ESCREVA “Escolha a sua opção”
LEIA op
SE op < 1 OU op > 6
ENTAO ESCREVA “Opção Invalida”
SE op = 1
ENTAO INICIO
        SE conta_tipo > 4
        ENTAO ESCREVA “Registo de tipos de serviço lotado. “
        SENAO INICIO
                ESCREVA “Escreva o código do serviço a ser registado”
                LEIA codigo_serv
                achou ← 0
                PARA i ← 1 ATE conta_tipo - 1
                INICIO
                        SE tipos [i].cod = codigo_serv
                        ENTAO achou ← i
                FIM
```

```
                SE achou = 0
                ENTAO ESCREVA "Já existe tipo de serviço registado com esse
                                código"
                SENAO INICIO
                        ESCREVA "Escreva a descrição do tipo de serviço a ser
                        registado"
                        LEIA desc_serv
                        tipos[achou].cod ← codigo_serv
                        tipos[achou].desc ← desc_serv
                        ESCREVA "Tipo de serviço registado com sucesso"
                FIM
        FIM
FIM
SE op = 2
ENTAO INICIO
        ESCREVA "Escrever o dia em que deseja registar o serviço prestado"
        LEIA dia
        achou ← 0
        PARA j ← 1 ATE 3 FAÇA
        INICIO
                SE serv[dia, j] num = 0
                ENTAO achou ← j
        FIM
        SE achou = 0
        ENTAO ESCREVA "Todos os serviços prestados neste dia já foram
registados"
        SENAO INICIO
                ESCREVA "Escreva o código do serviço a ser registado"
                LEIA codigo_serv
                valida ← 0
                PARA i ← 1 ATE conta_tipo - 1 FAÇA
        INICIO
```


```
                SE tipos [i] .cod = codigo_serv
                ENTAO valida ← 1
            FIM
            SE valida = 0
            ENTAO ESCREVA "Código de serviço invalido"
            SENAO INICIO
                ESCREVA "Escreva o numero do serviço"
                LEIA num_serv
                ESCREVA "Escreva o valor do serviço"
                LEIA valor_serv
                ESCREVA "Escreva o código do cliente"
                LEIA codigo_cliente
                serv[dia, achou].num ← num_serv
                serv[dia, achou].valor ← valor_serv
                serv[dia, achou].cod_serv ← codigo_serv
                serv[dia, achou].cod_cliente ← codigo_cliente
                ESCREVA "Serviço prestado registado com sucesso"
            FIM
        FIM
        FIM
        SE op = 3
        ENTAO INICIO
            ESCREVA "Escreva o dia em que deseja consultar os serviços prestados"
            LEIA dia
            achou ← 0
            PARA j ← 1 ATE 3 FAÇA
            INICIO
                SE serv[dia, j].num  0
                ENTAO achou ← 1
            FIM
            SE achou = 0
            ENTAO ESCREVA "Nenhum serviço foi prestado neste dia"
```

```
        SENAO INICIO
                ESCREVA “Serviços prestados no dia”, dia
                PARA j ← 1 ATE 3 FAÇA
                INICIO
                        SE serv[dia, j].num   0
                        ENTAO INICIO
                                ESCREVA serv[dia, jj.num, serv[dia, j].valor
                                ESCREVA serv[dia, j].cod_serv
                                PARA i ← 1 ATE conta_tipo - 1 FACA
                                INICIO
                                        SE tipos [i] .cod = serv[dia, j].cod_serv
                                        ENTAO ESCREVA tipos[i].desc
                                FIM
                                ESCREVA serv[dia, j].cod_cliente
                        FIM
                FIM
        FIM
FIM
SE op = 4
ENTAO INICIO
        ESCREVA “Escreva o valor inicial”
        LEIA valor_inicial
        ESCREVA “Escreva o valor final”
        LEIA valor_final
        achou ← 0
        PARA i ← 1 ATE 30 FAÇA
        INICIO
                PARA j ← 1 ATE 3 FACA
                INICIO
                        SE serv[i, j].valor >= valor_inicial E serv[i, j].valor <=
                valor_final
                        ENTAO INICIO
```


```
                                    Achou ← 1
                                    ESCREVA serv[i, j].num, serv[i, j].valor
                                    ESCREVA serv[i, j].cod_serv
                                    PARA k ← 1 ATE conta_tipo - 1 FAÇA
                                    INICIO
                                        SE tipos[k].cod = serv[i, j].cod_serv
                                        ENTAO ESCREVA tipos[k].desc
                                    FIM
                                    ESCREVA serv[i, j].cod_cliente
                            FIM
                    FIM
            FIM
            SE achou = 0
            ENTAO ESCREVA “Nenhum serviço prestado está entre os valores
citados”
FIM
SE op = 5
ENTAO INICIO
            achou  ← 0
            PARA i ← 1 ATE 30 FAÇA
            INICIO
                    ESCREVA “Dia “,i
                    PARA j ← 1 ATE 3 FAÇA
                    INICIO
                            SE serv(i, j].num  0
                            ENTAO INICIO
                            achou ← 1
                            ESCREVA serv[i, j].num, serv[i, j].valor
                            ESCREVA serv[i, j].cod_serv
                            PARA k ← 1 ATE conta_tipo - 1 FAÇA
                            INICIO
                                    SE tipos[k].cod = serv[i, j].cod_serv
```

```
                                    ENTAO ESCREVA tipos[k].desc
                                FIM
                                ESCREVA serv[i, j].cod_cliente
                        FIM
                FIM
        FIM
        SE achou = 0
        ENTAO ESCREVA “Nenhum serviço prestado foi registado”
FIM
        ATE op = 6
FIM ALGORITMO.
```

Solução PASCAL

```
ROGRAM EX2;
USES CRT;
TYPE REGISTRO1 =    RECORD
                    cod : INTEGER;
                    desc : STRING[20];
                    END;
TYPE REGISTRO2 =    RECORD
                    num, cod_serv, cod_cliente : INTEGER;
                    valor : REAL;
                    END;
VAR tipos:          ARRAY[1..4] OF REGISTRO1;
                    serv: ARRAY[1..30,1..3] OF REGISTRO2;
                    i, j, op, codigo_serv, achou,dia, codigo_cliente, num_serv, valida,
k                   : INTEGER;
                    valor_inicial, valor_final, valor_serv : REAL;
                    desc_serv : STRING[20];
BEGIN
        FOR i := 1 TO 4 DO
        BEGIN
```


```
                tipos[i].cod := 0;
                tipos[i].desc := '';
        END;
        FOR i := 1 TO 30 DO
        BEGIN
                FOR j := 1 TO 3 DO
                BEGIN
                        serv[i, j].num := 0;
                        serv[i, j].valor := 0;
                        serv[i, j].cod_serv := 0;
                        serv[i, j].cod_cliente := 0;
                END;
        END;
        REPEAT
        CLRSCR;
        WRITELN('Menu de Opções');
WRITELN('1 - Registar tipos de serviços');
WRITELN('2 - Registar serviços prestados');
WRITELN('3 - Mostrar os serviços prestados num determinado dia');
WRITELN('4 - Mostrar os serviços prestados dentro de um intervalo de valor');
WRITELN('5 - Mostrar um relatório geral, separado por dia');
WRITELN('6 - Sair');
WRITELN('Escolha a sua opção');
READLN(op);
IF (op < 1) OR (op > 6)
        THEN WRITELN('Opção Inv lida');
IF op = 1
THEN BEGIN
                WRITELN('Escreva o código do serviço a ser registado');
                READLN( codigo_serv);
                achou := 0;
                FOR i := 1 TO 4 DO
```

```
        BEGIN
            IF tipos[i].cod = 0
            THEN achou := i;
        END;
        IF achou = 0
        THEN WRITELN('Registo de tipos de serviços lotado')
        ELSE BEGIN
            FOR i := 1 TO 4 DO
            BEGIN
                IF tipos[i].cod = codigo_serv
                THEN achou := 0;
            END;
            IF achou = 0
            THEN WRITELN('J  existe tipo de serviço registado com este
            código')
            ELSE BEGIN
                WRITELN('Escreva a descrição do tipo de serviço a ser
                registado');
                READLN( desc_serv);
                tipos[achou].cod := codigo_serv;
                tipos[achou].desc := desc_serv;
                WRITELN('Tipo de serviço registado com sucesso');
            END;
        END;
        READLN;
        END;
IF op = 2
THEN BEGIN
        WRITELN('Escreva o dia em que deseja registar o serviço prestado');
        READLN( dia);
        achou := 0;
        FOR j := 1 TO 3 DO
```


```
        BEGIN
            IF serv[dia, j].num = 0
            THEN  achou := j;
        END;
        IF achou = 0
        THEN WRITELN('Todos os serviços prestados neste dia j  foram
registados')
        ELSE BEGIN
            WRITELN('Escreva o código do serviço a ser registado');
            READLN( codigo_serv);
            valida := 0;
            FOR i := 1 TO 4 DO
            BEGIN
                IF tipos[i].cod = codigo_serv
                THEN  valida := 1;
            END;
            IF valida = 0
            THEN WRITELN('Código de serviço inválido')
            ELSE BEGIN
                WRITELN('Escreva o número do serviço');
                READLN( num_serv);
                WRITELN('Escreva o valor do serviço');
                READLN( valor_serv);
                WRITELN('Escreva o código do cliente');
                READLN( codigo_cliente);
                serv[dia, achou].num := num_serv;
                serv[dia, achou].valor := valor_serv;
                serv[dia, achou].cod_serv := codigo_serv;
                serv[dia, achou].cod_cliente := codigo_cliente;
                WRITELN('Serviço prestado registado com sucesso');
             END;
        END;
```

```
        READLN;
END;
IF op = 3
THEN BEGIN
        WRITELN('Escreva o dia em que deseja consultar os serviços
        prestados');
        READLN( dia);
        achou := 0;
        FOR j := 1 TO 3 DO
        BEGIN
                IF serv[dia, j].num <> 0
                THEN achou := 1;
        END;
        IF achou = 0
        THEN WRITELN('Nenhum serviço foi prestado neste dia')
        ELSE BEGIN
                WRITELN('Serviços prestados no dia ', dia);
                FOR j := 1 TO 3 DO
                BEGIN
                        IF serv[dia, j].num <> 0
                        THEN BEGIN
                                writeln;
                                WRITE('Nº do serviço: ',serv[dia, j].num, '    Valor:
                        ',serv[dia, j].valor:5:2);
                                WRITELN('  Código do serviço: ', serv[dia,
                        j].cod_serv);
                                FOR i := 1 TO 4 DO
                                BEGIN
                                        IF tipos[i].cod = serv[dia, j].cod_serv
                                        THEN WRITE('Descrição: ',tipos[i].desc);
                                END;
                                WRITELN('  Cliente: ',serv[dia, j].cod_cliente);
```


```
                    END;
                END;
            END;
            READLN;
END;
IF op = 4
THEN BEGIN
            WRITELN('Escreva o valor inicial');
            READLN( valor_inicial);
            WRITELN('Escreva o valor final');
            READLN( valor_final);
            achou := 0;
            FOR i := 1 TO 30 DO
            BEGIN
                FOR j := 1 TO 3 DO
                BEGIN
                    IF (serv[i, j].valor >= valor_inicial) AND (serv[i, j].valor <=
                valor_final)
                    THEN  BEGIN
                        achou := 1;
                        WRITELN;
                        WRITE('Nº do serviço: ',serv[i, j].num, '    Valor: '
                        ,serv[i, j].valor:5:2);
                        WRITELN('  Código do serviço: ', serv[i,
                        j].cod_serv);
                        FOR k := 1 TO 4 DO
                        BEGIN
                            IF tipos[k].cod = serv[i, j].cod_serv
                            THEN WRITE('Descrição: ',tipos[k].desc);
                        END;
                        WRITELN('Código cliente: ',serv[i, j].cod_cliente,'
                    Dia: ',i);
```

```
                    END;
                END;
            END;
            IF achou = 0
            THEN WRITELN('Nenhum serviço prestado entre os valores citados');
            READLN;
END;
IF op = 5
THEN  BEGIN
            achou := 0;
            FOR i := 1 TO 30 DO
            BEGIN
                FOR j := 1 TO 3 DO
                BEGIN
                    IF (serv[i, j].num <> 0)
                    THEN BEGIN
                        achou := 1;
                        WRITELN('Dia: ',i);
                        WRITE('Nº do serviço: ',serv[i, j].num, '    Valor:
                        ',serv[i, j].valor:5:2);
                        WRITELN(' Código do serviço: ', serv[i,
                        j].cod_serv);
                        FOR k := 1 TO 4 DO
                        BEGIN
                            IF tipos[k].cod = serv[i, j].cod_serv
                            THEN WRITE('Descrição: ',tipos[k].desc);
                        END;
                        WRITELN('Cliente:  ',serv[i, j].cod_cliente);
                    END;
                END;
            END;
            IF achou = 0
```


```
            THEN WRITELN('Nenhum serviço prestado foi registado');
            READLN;
        END;
        UNTIL op = 6;
END.
```

Exercício 3

Faça um programa que utilize os registos a seguir.

CLIENTES	DOCUMENTOS
cod_cli	**num_doc**
nome	**cod_cli**
telefone	**data_venc**
endereço	**data_pag**
	valor
	juros

Sabe-se que um documento só pode ser registado para um cliente que já exista. Considere que podem existir, no máximo, quinze clientes e trinta documentos. Crie um vetor para clientes e outro para documentos.

Crie um menu para a realização de cada uma das operações especificadas a seguir.

a. *Registar clientes* - não pode existir mais do que um cliente com o mesmo código.
b. b*Registar documentos* - ao registar um documento, se a data de pagamento for maior que a data de vencimento, calcular o campo 'juros' do registo documentos (5% sobre o valor original do documento).
c. *Excluir clientes* - um cliente só poderá ser excluído se não existir nenhum documento associado a ele.
d. *Excluir documentos individuais* – através do seu número. Caso o documento não exista, o programa devera mostrar a mensagem *Documento não encontrado.*
e. *Excluir documentos por cliente* - o programa deverá informar o código do cliente e excluir todos os seus documentos. Caso o cliente não exista, deverá mostrar a mensagem *Cliente não encontrado.*

f. *Excluir documentos por período* - o programa deverá informar a data inicial e a data final e excluir todos os documentos que possuírem data de vencimento nesse período.

g. *Alterar as informações sobre os clientes* - só não pode ser alterado o código do cliente.

h. *Mostrar o total de documentos de determinado cliente.*

i. Sair.

Quando forem excluídos clientes ou documentos, os vetores deverão ser reorganizados, ou seja, todas as posições não preenchidas dos vetores deverão ficar no final. Exemplo: se for necessário excluir o número 8 do vetor a seguir, tanto o 9 como o 1 deverão ser movidos uma casa para a esquerda e a ultima posição deverá ficar livre para uma nova inclusão.

Vetor inicial

12	5	8	9	1

Vetor modificado com uma posição livre no final

12	5	9	1

Solução Algoritmo

```
ALGORITMO
DECLARE     clientes[15] REGISTRO <cod_cli NUMERICO, nome, fone, ende LITERAL)
            docs [30] REGISTRO (num_doc, cod_cli, dv, mv, av, dp, mp, ap, valor, juros
            NUMERICO)
            posi, op, i, cliente_livre, doc_livre NUMERICO
            achou, achou2, codigo, numero, diav, mesv, anov NUMERICO
            diap, mesp, anop, valor, juros, total NUMERICO
            nome, fone, ende LITERAL
cliente livre ← 1
doc_livre ← 1
REPITA
ESCREVA “Menu de Opções”
```

```
ESCREVA “1 - Registar clientes”
ESCREVA “2 - Registar documentos”
ESCREVA “3 - Excluir clientes”
ESCREVA “4 - Excluir documentos individuais”
ESCREVA “5 - Excluir documentos por cliente”
ESCREVA “6 - Excluir documentos por período”
ESCREVA “7 - Alterar clientes”
ESCREVA “8 - Totalizar documentos”
ESCREVA “9 - Sair”
ESCREVA “Escolha opção”
LEIA op
SE op < 1 OU op > 9
ENTAO ESCREVA “Opção invalida”
SE op = 1
ENTAO INICIO
        SE cliente_livre > 15
        ENTAO ESCREVA “Registo de clientes lotado”
        SENAO INICIO
                ESCREVA “Escreva o código do cliente a ser registado”
                LEIA código
                achou ← 0
                PARA i ← 1 ATE cliente_livre - 1 FAÇA
                INICIO
                        SE clientes[i].cod_cli = codigo
                        ENTAO achou ← 1
                FIM
                SE achou = 1
                ENTAO ESCREVA “Já existe cliente registado com esse código”
                SENAO INICIO
                        ESCREVA “Escreva o nome do cliente”
                        LEIA nome
                        ESCREVA “Escreva o telefone do cliente”
```

```
                    LEIA fone
                    ESCREVA "Escreva endereço do cliente"
                    LEIA ende
                    clientes[cliente_livre].cod_cli ← codigo
                    clientes[cliente_livre].nome ← nome
                    clientes[cliente_livre].fone ← fone
                    clientes[cliente_livre].ende ← ende
                    ESCREVA "Cliente registado com sucesso"
                    cliente_livre ← cliente_livre + 1
                FIM
            FIM
FIM
SE op = 2
ENTAO INICIO
        SE doc_livre > 30
        ENTAO ESCREVA "Registo de documentos lotado"
        SENAO INICIO
                ESCREVA "Escreva o número do documento a ser registado"
                LEIA número
                achou ← 0
                PARA i ← 1 ATE doc_livre - 1 FAÇA
                INICIO
                        SE does[i].num_doc = numero
                        ENTAO achou ← 1
                FIM
                SE achou = 1
                ENTAO ESCREVA "Já existe um documento registado com esse código"
                SENAO INICIO
                        ESCREVA "Escreva o código do cliente dono do documento"
                        LEIA codigo
                        achou ← 0
                        PARA i ← 1 ATE cliente_livre - 1 FAÇA
```


```
INICIO
        SE clientes[i].cod_cli = codigo
        ENTAO achou ← 1
FIM
SE achou = 0
ENTAO ESCREVA “Não existe cliente registado com esse código”
SENAO INICIO
        ESCREVA “Escreva a data do vencimento do documento”
        LEIA diav, mesv,anov
        ESCREVA “Escreva a data do pagamento do documento”
        LEIA diap, mesp, anop
        ESCREVA “Escreva o valor do documento”
        LEIA valor
        SE anop > anov
        ENTAO juros ← valor * 5/100
        SENAO SE anop > anov
        ENTAO SE mesp > mesv
        ENTAO juros ← valor * 5/100
        SENAO SE mesp > mesv
        ENTAO SE diap > diav
        ENTAO juros ← valor * 5/100
        SENAO juros ← 0
        docs[doc_livre].num_doc ← numero
        docs[doc_livre].cod_cli ← codigo
        docs[doc_livre].dv ← diav
        docs[doc_livre].mv <- mesv
docs[doc_livre].av ← anov
docs [doc_livre] .dp ← diap
docs[doc_livre].mp ←mesp
docs[doc_livre].ap ← anop
docs[doc_livre].valor ← valor
docs[doc_livre].juros ← juros
```

```
                    ESCREVA “Documento registado com sucesso”
                    doc_livre doc_livre + 1
                FIM
            FIM
        FIM
FIM
SE op = 3
ENTAO INICIO
        ESCREVA “Digite o código do cliente a ser excluído”
        LEIA codigo
        achou ←0
        PARA i ← 1 ATE cliente_livre - 1 FAÇA
        INICIO
            SE clientes [i] .cod_cli = codigo
            ENTAO INICIO
                achou ← 1
                posi ← i
            FIM
        FIM
        SE achou = 0
        ENTAO ESCREVA “Não existe cliente registado com esse código”
        SENAO INICIO
            achou ← 0
            PARA i ← 1 ATE doc_livre - 1 FAÇA
            INICIO
                SE does[i].cod_cli = codigo
                ENTAO achou ← 1
            FIM
            SE achou = 1
            ENTAO ESCREVA “Este cliente não pode ser excluído, possui documento”
            SENAO INICIO
                PARA i ← posi ATE cliente_livre - 2 FAÇA
```


```
                    INICIO
                        clientes[i].cod_cli ← clientes[i+1].cod_cli
                        clientes[i].nome ← clientes[i+1].nome
                        clientes[i].fone ← clientes[i+1].fone
                        clientes[i].ende ← clientes[i+1].ende
                    FIM
                    cliente_livre ← cliente_livre - 1
                    ESCREVA “Cliente excluído com sucesso”
                FIM
            FIM
FIM
SE op = 4
ENTAO INICIO
        ESCREVA “Escreva o número do documento a ser excluído”
        LEIA numero
        achou ← 0
        PARA i ← 1 ATE doc_livre FAÇA
        INICIO
            SE does[i].num_doc = numero
            ENTAO INICIO
                achou ← 1
                posi ← i
            FIM
        FIM
        SE achou = 0
        ENTAO ESCREVA “Não existe documento registado com esse número”
        SENAO INICIO
            PARA i ← posi ATE doc_livre - 2 FAÇA
            INICIO
                docs[i].num_doc ← docs[i+1].num_doc
                docs[i] .cod_cli ← docs [i + 1] .cod_cli
                docs[i] .dv ← docs [i + 1] .dv
```

```
                    docs[i].mv ← docs[i+1].mv
                    docs[i].av ← docs[i+1].av
                    docs[i] .dp ← docs [i + 1] .dp
                    docs[i].mp ← docs[i+1].mp
                    docs[i] .ap ← docs [i + 1].ap
                    docs[i].valor ← docs[i+1].valor
                    docs[i].juros ← docs[i+1].juros
                FIM
                ESCREVA "Documento excluído com sucesso"
                doc_livre ← doc_livre - 1
        FIM
FIM
SE op = 5
ENTAO INICIO
        ESCREVA "Escreva o código do cliente do qual deseja excluir os documentos"
        LEIA codigo
        achou ← 0
        PARA i ← 1 ATE cliente_livre - 1 FAÇA
        INICIO
                SE clientes[i].cod_cli = codigo
                ENTAO achou ← 1
        FIM
        SE achou = 0
        ENTAO ESCREVA "Não existe cliente registado com esse código"
        SENAO INICIO
                SE doc_livre = 1
                ENTAO ESCREVA "Não existe nenhum documento registado"
                SENAO INICIO
                        k ←1
                        achou ← 0
                        ENQUANTO k < doc_livre FAÇA
                        INICIO
```


```
                                SE codigo = does[k].cod_cli
                                ENTAO INICIO
                                        PARA i ← k ATE doc_livre - 2 FAÇA
                                        INICIO
                                                achou ← 1
                                                docs[i].num_doc ← docs[i+l].num_doc
                                                docs[i].cod_cli ← docs[i+1].cod_cli
                                                docs[i].dv ← docs[i+1].dv
docsii] .mv ← docs [i + 1] .mv
docs[i].av ← docs[i+1].av
docs[i] .dp ← docs [i+1] .dp
docs[i].mp ← docs [i + 1].mp
docs[i] .ap ← docs [i+1] .ap
docs[i] .valor ← docs[i + 1] .valor
docs[i] .juros ← docs[i + 1] .juros
                                        FIM
                                        doc_livre ← doc_livre - 1
                                FIM
                                SENAO k ←k + 1
                        FIM
                        SE achou = 1
                        ENTAO ESCREVA “Documentos excluídos com sucesso”
                        SENAO ESCREVA “Não existe documento para este cliente”
                FIM
        FIM
FIM
SE op = 6
ENTAO INICIO
        ESCREVA “Escreva a data inicial dos documentos que serão excluídos”
        LEIA dia_inicial, mes_inicial, ano_inicial
        ESCREVA “Escreva a data final dos documentos que serão excluídos”
        LEIA dia_final, mes_final, ano_final
```

```
achou2 ← 0
ENQUANTO i < doc_livre FAÇA
INICIO
        achou ← 0
        SE docs[i] .av > ano_inicial E docs[i] .av < ano_iinal
        ENTAO SE docs[i] .mv > mes_inicial E docs[i] .mv < mes_final
        ENTAO SE docs[i] .dv > dia_inicial E docs[i] .dv < dia_final
        ENTAO INICIO
                posi ← i
                achou ← 1
        FIM
        SE achou = 1
        ENTAO INICIO
                achou2 ← 1
                PARA j ← posi ATE doc_livre - 2 FAÇA
                INICIO
                        docs [ j] .num_doc ← docs [ j+1] .num_doc
                        docs [ j ] .cod_cli ← does [j+1] .cod_cli
docs[j] .dv ← docs [j+1] .dv
docs ij] .mv ← docs [j+1] .mv
docs[j] .av ← docs [j+1] .av
docs[j] .dp ← docs [j+1] .dp
docs[j] .mp ← docs [j+1] .mp
docs[j] .ap ← docs[j+l] .ap
docs[j] .valor ← docs [j+1] .valor
docs [j ]. juros ← docs [j+1] . juros
                FIM
                doc_livre ← doc_livre – 1

        FIM
        SENAO i ← i + 1
        SE achou2 = 0
```

```
        ENTAO ESCREVA “Não existe documento registado neste período”
        SENAO ESCREVA “Documentos do período excluídos com sucesso”
FIM
SE op = 7
ENTAO INICIO
        ESCREVA “Escreva o código do cliente a ser alterado”
        LEIA codigo
        achou ← 0
        PARA i ← 1 ATE cliente_livre - 1 FAÇA
        INICIO
                SE clientes[i].cod_cli = codigo
                ENTAO INICIO
                        achou ← 1
                        posi ← i
                FIM
        FIM
        SE achou = 0
        ENTAO ESCREVA “Não existe cliente registado com esse código para ser
        alterado”
        SENAO INICIO
                ESCREVA “Escreva o novo nome do cliente”
                LEIA nome
                ESCREVA “Escreva o novo telefone do cliente”
                LEIA fone
                ESCREVA “Escreva o novo endereço do cliente”
                LEIA ende
                clientes[posi].nome ← nome
                clientes[posi].fone ← fone
                clientes[posi].ende ← ende
                ESCREVA “Cliente alterado com sucesso”
        FIM
FIM
```

```
        SE op = 8
        ENTAO INICIO
                ESCREVA "Escreva o código do cliente do qual deseja totalizar os
                documentos"
                LEIA codigo
                achou ← 0
                PARA i ← 1 ATE cliente_livre - 1 FAÇA
                INICIO
                        SE clientes[i].cod_cli = codigo
                        ENTAO achou ← 1
                FIM
                SE achou = 0
                ENTAO ESCREVA "Não existe cliente registado com esse código"
                SENAO INICIO
                        total ← 0
                        PARA i ← 1 ATE doc_livre - 1 FAÇA
                        INICIO
                                SE does[i].cod_cli = codigo
                                ENTAO INICIO
                                        total ← total + does [i] .valor
                                        total ← total + does [i] .juros
                                FIM
                        FIM
                        ESCREVA "Total dos documentos do cliente de código ", codigo,"
=                       ", total
                FIM
        FIM
        ATE op = 9
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX3;
USES CRT;
TYPE REGISTO1 =     RECORD
                    cod_cli : INTEGER;
                    nome, fone, ende : STRING;
                    END;
TYPE REGISTO2 =     RECORD
                    num_doc, cod_cli, dv, mv, av, dp, mp, ap : INTEGER;
                    valor, juros : REAL;
                    END;
                    VAR  clientes: ARRAY[1..15] OF REGISTO1;
                    docs: ARRAY[1..30] OF REGISTO2;
                    posi,op, i, cliente_livre, doc_livre, achou, codigo, numero, diav,
                    mesv, anov : INTEGER;
                    diap, mesp, anop, k,J : INTEGER;
                    valor, juros, total : REAL;
                    nome, fone, ende : STRING;
                    dia_inicial, mes_inicial, ano_inicial: INTEGER;
                    dia_final, mes_final, ano_final: INTEGER;
BEGIN
FOR i :=1 TO 15 DO
BEGIN
        clientes[i].cod_cli := 0;
clientes[i].nome := '';
clientes[i].fone := '';
clientes[i].ende := '';
END;
cliente_livre := 1;
FOR i := 1 TO 30 DO
BEGIN
docs[i].num_doc := 0;
```

```
docs[i].cod_cli := 0;
docs[i].dv := 0;
docs[i].mv := 0;
docs[i].av := 0;
docs[i].dp := 0;
docs[i].mp := 0;
docs[i].ap := 0;
docs[i].valor := 0;
docs[i].juros := 0;
END;
doc_livre := 1;
REPEAT
        CLRSCR;
        WRITELN('Menu de Op‡äes');
        WRITELN('1 - Registar clientes');
        WRITELN('2 - Registar documentos');
        WRITELN('3 - Excluir clientes');
        WRITELN('4 - Excluir documentos individuais');
        WRITELN('5 - Excluir documentos por cliente');
        WRITELN('6 - Excluir documentos por per¡odo');
        WRITELN('7 - Alterar clientes');
        WRITELN('8 - Totalizar documentos');
        WRITELN('9 - Sair');
        WRITELN('Escreva a sua opção');
        READLN(op);
        IF (op < 1) OR (op > 9)
        THEN WRITELN('Opção inválida');
        IF op = 1
        THEN BEGIN
                IF cliente_livre = 16
                THEN WRITELN('Registo de clientes lotado')
                ELSE BEGIN
```


```
                WRITELN('Escreva o código do cliente a ser registado');
                READLN(codigo);
                achou := 0;
                FOR i := 1 TO 15 DO
                BEGIN
                        IF clientes[i].cod_cli = codigo
                        THEN achou := 1;
                END;
                IF achou = 1
                THEN WRITELN('Já existe cliente registado com esse código')
                ELSE BEGIN
                        WRITELN('Escreva o nome do cliente');
                        READLN( nome);
                        WRITELN(' Escreva o telefone do cliente');
                        READLN( fone);
                        WRITELN(' Escreva o endere‡o do cliente');
                        READLN( ende);
                        clientes[cliente_livre].cod_cli := codigo;
                        clientes[cliente_livre].nome := nome;
                        clientes[cliente_livre].fone := fone;
                        clientes[cliente_livre].ende := ende;
                        WRITELN('Cliente registado com sucesso');
                        cliente_livre := cliente_livre + 1;
                END;
        END;
        READLN;
END;
IF op = 2
THEN BEGIN
        IF doc_livre = 31
        THEN WRITELN('Registo de documentos lotado')
        ELSE BEGIN
```

```
WRITELN(' Escreva o número do documento a ser registado');
READLN( numero);
achou := 0;
FOR i := 1 TO 30 DO
BEGIN
        IF docs[i].num_doc = numero
       THEN achou := 1;
END;
IF achou = 1
THEN WRITELN('Já  existe um documento registado com esse
código')
ELSE BEGIN
        WRITELN(' Escreva o código do cliente dono do
documento');
        READLN( codigo);
        achou := 0;
        FOR i := 1 TO 15 DO
        BEGIN
            IF clientes[i].cod_cli = codigo
            THEN achou := 1;
        END;
        IF achou = 0
        THEN WRITELN('Não existe cliente registado com esse
código')
        ELSE BEGIN
            WRITELN('Escreva a data do vencimento do
          documento');
            READLN( diav, mesv,anov);
            WRITELN('Escreva a data do pagamento do
          documento');
            READLN( diap, mesp,anop);
            WRITELN('Escreva o valor do documento');
```


```
                                        READLN( valor);
                                        IF anop > anov
                                        THEN juros := valor * 5/100
                                        ELSE IF mesp > mesv
                                                THEN juros := valor * 5/100
                                                ELSE IF diap > diav
                                                        THEN juros := valor * 5/100
                                                        ELSE juros := 0;
                                      docs[doc_livre].num_doc := numero;
                                      docs[doc_livre].cod_cli := codigo;
                                      docs[doc_livre].dv := diav;
                                      docs[doc_livre].mv := mesv;
                                      docs[doc_livre].av := anov;
                                      docs[doc_livre].dp := diap;
                                      docs[doc_livre].mp := mesp;
                                      docs[doc_livre].ap := anop;
                                      docs[doc_livre].valor := valor;
                                      docs[doc_livre].juros := juros;
                                      WRITELN('Documento registado com sucesso');
                                      doc_livre := doc_livre + 1;
                                END;
                        END;
                 END;
        READLN;
END;
IF op = 3
THEN BEGIN
        WRITELN('Escreva o código do cliente a ser excluído');
        READLN( codigo);
        achou := 0;
        FOR i := 1 TO 15 DO
        BEGIN
```

```
        IF clientes[i].cod_cli = codigo
        THEN BEGIN
                achou := 1;
                posi := i;
        END;
END;
IF achou = 0
THEN WRITELN('Não existe cliente registado com esse código')
ELSE BEGIN
        achou := 0;
        FOR i := 1 TO 30 DO
        BEGIN
                IF docs[i].cod_cli = codigo
                THEN achou := 1;
        END;
        IF achou = 1
        THEN WRITELN('Este cliente não pode ser excluído, existe
documento')
        ELSE BEGIN
                FOR i := posi TO (cliente_livre-2) DO
                BEGIN
                        clientes[i].cod_cli := clientes[i+1].cod_cli;
                        clientes[i].nome := clientes[i+1].nome;
                        clientes[i].fone := clientes[i+1].fone;
                        clientes[i].ende := clientes[i+1].ende;
                END;
                clientes[cliente_livre - 1].cod_cli := 0;
                clientes[cliente_livre - 1].nome := '';
                clientes[cliente_livre - 1].fone := '';
                clientes[cliente_livre - 1].ende := '';
                cliente_livre := cliente_livre -1;
                WRITELN('Cliente excluído com sucesso');
```


```
                END;
            END;
            READLN;
END;
IF op = 4
THEN BEGIN
            WRITELN('Escreva o número do documento a ser excluído');
            READLN( numero);
            achou := 0;
            FOR i := 1 TO 30 DO
            BEGIN
                IF docs[i].num_doc = numero
                THEN BEGIN
                        achou := 1;
                        posi := i;
                END;
            END;
            IF achou = 0
            THEN WRITELN('Não existe documento registado com esse número')
            ELSE BEGIN
                FOR i := posi TO (doc_livre - 2) DO
                BEGIN
                        docs[i].num_doc := docs[i+1].num_doc;
                        docs[i].cod_cli := docs[i+1].cod_cli;
                        docs[i].dv := docs[i+1].dv;
                        docs[i].mv := docs[i+1].mv;
                        docs[i].av := docs[i+1].av;
                        docs[i].dp := docs[i+1].dp;
                        docs[i].mp := docs[i+1].mp;
                        docs[i].ap := docs[i+1].ap;
                        docs[i].valor := docs[i+1].valor;
                        docs[i].juros := docs[i+1].juros;
```

```
                END;
                docs[doc_livre - 1].num_doc := 0;
                docs[doc_livre - 1].cod_cli := 0;
                docs[doc_livre - 1].dv := 0;
                docs[doc_livre - 1].mv := 0;
                docs[doc_livre - 1].av := 0;
                docs[doc_livre - 1].dp := 0;
                docs[doc_livre - 1].mp := 0;
                docs[doc_livre - 1].ap := 0;
                docs[doc_livre - 1].valor := 0;
                docs[doc_livre - 1].juros := 0;
                WRITELN('Documento excluído com sucesso');
                doc_livre := doc_livre - 1;
        END;
        READLN;
END;
IF op = 5
THEN BEGIN
        WRITELN('Escreva o código do cliente no qual deseja excluir os documentos');
        READLN( codigo);
        achou := 0;
        FOR i := 1 TO 15 DO
        BEGIN
                IF clientes[i].cod_cli = codigo
                THEN achou := 1;
        END;
        IF achou = 0
        THEN WRITELN('Não existe cliente resgitado com esse código')
        ELSE BEGIN
                IF doc_livre = 1
                THEN WRITELN('Não existe nenhum documento registado')
                ELSE BEGIN
```


```
k := 1;
achou := 0;
WHILE (k <= (doc_livre - 1)) DO
BEGIN
        IF codigo = docs[k].cod_cli
        THEN BEGIN
                FOR i := k TO (doc_livre - 2) DO
                BEGIN
                        achou := 1;
                        docs[i].num_doc := docs[i+1].num_doc;
                        docs[i].cod_cli := docs[i+1].cod_cli;
                        docs[i].dv := docs[i+1].dv;
                        docs[i].mv := docs[i+1].mv;
                        docs[i].av := docs[i+1].av;
                        docs[i].dp := docs[i+1].dp;
                        docs[i].mp := docs[i+1].mp;
                        docs[i].ap := docs[i+1].ap;
                        docs[i].valor := docs[i+1].valor;
                        docs[i].juros := docs[i+1].juros;
                END;
                docs[doc_livre - 1].num_doc := 0;
                docs[doc_livre - 1].cod_cli := 0;
                docs[doc_livre - 1].dv := 0;
                docs[doc_livre - 1].mv := 0;
                docs[doc_livre - 1].av := 0;
                docs[doc_livre - 1].dp := 0;
                docs[doc_livre - 1].mp := 0;
                docs[doc_livre - 1].ap := 0;
                docs[doc_livre - 1].valor := 0;
                docs[doc_livre - 1].juros := 0;
                doc_livre := doc_livre - 1;
                k := 1;
```

```
                    END
                    ELSE k := k + 1;
                END;
                IF achou = 1
                THEN WRITELN('Documentos excluídos com sucesso')
                ELSE WRITELN('Não existe documento registado para este
            cliente');
            END;
        END;
        READLN;
END;
IF op = 6
THEN BEGIN
        WRITELN('Escreva a data inicial dos documentos que serão excluídos');
        READLN(dia_inicial, mes_inicial, ano_inicial);
        WRITELN('Escreva a data final dos documentos que serão excluídos');
        READLN( dia_final, mes_final, ano_final);
        achou := 0;
        FOR i := 1 TO 30 DO
        BEGIN
            IF (docs[i].av >= ano_inicial) AND (docs[i].av <= ano_final)
            THEN IF (docs[i].mv >= mes_inicial) AND (docs[i].mv <= mes_final)
            THEN IF (docs[i].dv >= dia_inicial) AND (docs[i].dv <= dia_final)
            THEN BEGIN
                posi := i;
                achou := 1;
            END;
            IF achou = 1
            THEN BEGIN
                FOR j := posi TO (doc_livre - 2) DO
                BEGIN
                    docs[j].num_doc := docs[j+1].num_doc;
```


```
                    docs[j].cod_cli := docs[j+1].cod_cli;
                    docs[j].dv := docs[j+1].dv;
                    docs[j].mv := docs[j+1].mv;
                    docs[j].av := docs[j+1].av;
                    docs[j].dp := docs[j+1].dp;
                    docs[j].mp := docs[j+1].mp;
                    docs[j].ap := docs[j+1].ap;
                    docs[j].valor := docs[j+1].valor;
                    docs[j].juros := docs[j+1].juros;
                END;
                docs[doc_livre - 1].num_doc := 0;
                docs[doc_livre - 1].cod_cli := 0;
                docs[doc_livre - 1].dv := 0;
                docs[doc_livre - 1].mv := 0;
                docs[doc_livre - 1].av := 0;
                docs[doc_livre - 1].dp := 0;
                docs[doc_livre - 1].mp := 0;
                docs[doc_livre - 1].ap := 0;
                docs[doc_livre - 1].valor := 0;
                docs[doc_livre - 1].juros := 0;
                doc_livre := doc_livre - 1;
            END
        END;
        IF achou = 0
        THEN WRITELN('Não existe documento registado neste per¡odo')
        ELSE WRITELN('Documentos do período excluídos com sucesso');
        READLN;
END;
IF op = 7
THEN BEGIN
        WRITELN('Escreva o código do cliente a ser alterado');
        READLN( codigo);
```

```
        achou := 0;
        FOR i := 1 TO 15 DO
        BEGIN
                IF clientes[i].cod_cli = codigo
                THEN BEGIN
                        achou := 1;
                        posi:= i;
                END;
        END;
        IF achou = 0
        THEN WRITELN('Não existe cliente registado com esse código para ser
alterado')
        ELSE BEGIN
                WRITELN('Escreva o novo nome do cliente');
                READLN( nome);
                WRITELN('Escreva o novo telefone do cliente');
                READLN( fone);
                WRITELN('Escreva o novo endereco do cliente');
                READLN( ende);
                clientes[posi].nome := nome;
                clientes[posi].fone := fone;
                clientes[posi].ende := ende;
                WRITELN('Cliente alterado com sucesso');
        END;
        READLN;
END;
IF op = 8
THEN BEGIN
         WRITELN('Escreva o código do cliente do qual deseja totalizar os
documentos');
        READLN( codigo);
        achou := 0;
```


```
        FOR i := 1 TO 15 DO
        BEGIN
                IF clientes[i].cod_cli = codigo
                THEN achou := 1;
        END;
        IF achou = 0
        THEN WRITELN('Não existe cliente registado com esse c¢digo')
        ELSE BEGIN
                total := 0;
                FOR i := 1 TO 30 DO
                BEGIN
                        IF docs[i].cod_cli = codigo
                        THEN BEGIN
                                total := total + docs[i].valor;
                                total := total + docs[i].juros;
                        END;
                END;
                WRITELN('Total dos documentos do cliente de código ', codigo,' = ',
                total:5:2);
        END;
        READLN;
END;
UNTIL  OP = 9;
END.
```

Exercício 5

Uma empresa possui 18 funcionários, sobre os quais se tem estas informacoes: nome, numero de horas trabalhadas no mês, turno de trabalho (pode ser M - matutino, V - vespertino ou N - noturno), categoria (pode ser O - operario ou G - gerente) e valor da hora trabalhada. Sabendo-se que essa empresa deseja informatizar a sua folha de pagamento, faca um programa que leia o nome, o numero de horas trabalhadas no mês, o turno e a categoria dos funcionarios, nao permitindo que sejam informados turnos e

categorias inexistentes. O programa devera calcular o valor da hora trabalhada, conforme a tabela a seguir, adotando o valor de $ 380,00 para o salario minimo.

CATEGORIA	TURNO	VALOR DA HORA TRABALHADA
G	**N**	**18% do salário**
G	**M ou V**	**15% do salário**
O	**N**	**13% do salário**
O	**M ou V**	**10% do salário**

O programa devera calcular o salario inicial dos funcionarios, com base no valor da hora e no numero de horas trabalhadas. Todos recebem um auxilio-alimentacao, de acordo com o seu salario inicial, conforme a tabela a seguir.

SALÁRIO INICIAL	AUXÍLIO - ALIMENTAÇÃO
<= US$ 300,00	**20% do salário inicial**
> US$ 300,00 e < US$ 600,00	**15% do salário inicial**
>= US$ 600,00	**5% do salário inicial**

O programa devera mostrar o nome, o número de horas trabalhadas, o valor da hora trabalhada, o salario inicial, o auxilio-alimentacao e o salario final (salario inicial + auxilio-alimentacao) de todos os funcionarios.

Ele devera apresentar o seguinte menu de opções:

1. Registar funcionarios.
2. Mostrar folha de pagamento.
3. Sair.

Solução Algoritmo

```
ALGORITMO
        DECLARE     func[18] REGISTRO (num_horas_trab, valor_hora NUMERICO,
                    nome,  turno, cat LITERAL)
                    i, pos_livre, op, sal_minimo, sal_inicial, aux_alim, sal_final
                NUMERICO
PARA i ← 1 ATE 18 FAÇA
INICIO
```

```
        func[i].num_horas_trab ← 0
        func [i] .valor_hora ← 0
        func[i].nome ← “
        func[i].turno ← “
        func [i] .cat ← “
FIM
pos_livre ← 1
REPITA
ESCREVA “Menu de Opcoes”
ESCREVA “1 - registar funcionarios”
ESCREVA “2 - Mostrar folha de pagamento”
ESCREVA “3 - Sair”
ESCREVA “Escolha a opção desejada”
LEIA op
SE op < 1 OU op > 3
ENTAO ESCREVA “Opção Invalida”
SE op = 1
ENTAO INICIO
        SE pos_livre > 18
        ENTAO ESCREVA “Registo de funcionarios lotado”
        SENAO INICIO
                sal_minimo ← 380
                ESCREVA “Escreva o nome do funcionario que deseja incluir”
                LEIA fune[pos_livre].nome
                ESCREVA “Escreva o numero de horas trabalhadas”
                LEIA func[pos_livre].num_horas_trab
                ESCREVA “Escreva o turno de trabalho”
                REPITA
                        LEIA func[pos_livre].turno
                        ATE func[pos_livre].turno = “M” 0U func[pos_livre].turno = “V”
                OU func[pos_livre].turno = “N”)
                        ESCREVA “Escreva a categoria”
```

```
                REPITA
                        LEIA func[pos_livre].cat
                        ATE func[pos_livre].cat = “0” OU func[pos_livre].cat = “G”
                        SE func [pos_livre] .cat = “G”
                        ENTAO SE func[pos_livre].turno = “N”
                        ENTAO func[pos_livre].valor_hora ← sal_minimo * 18/100
                        SENAO func[pos_livre].valor_hora ← sal_minimo * 15/100
                        SE func[pos_livre].cat = “O”
                        ENTAO SE func[pos_livre].turno = “N”
                        ENTAO func[pos_livre].valor_hora ← sal_minimo * 13/100
                        SENAO func[pos_livre].valor_hora ← sal_minimo * 10/100
                        ESCREVA “Funcionario cadastrado com sucesso”
                        pos_livre ← pos_livre + 1
        FIM
FIM
SE op = 2
ENTAO INICIO
        ESCREVA “Folha de Pagamento”
        SE pos_livre = 1
        ENTAO ESCREVA “Nao existe funcionario Registado”
        SENAO INICIO
                PARA i ← 1 ATE (pos_livre - 1) FAÇA
                INICIO
                ESCREVA func[i].nome, func[i].num_horas_trab, func[i].valor_hora
                sal_inicial ← func[i].num_horas_trab *
                func[i].valor_hora
                ESCREVA sal_inicial
                SE sal_inicial < 300
                ENTAO aux_alim ← sal_inicial * 20/100
                SENAO SE sal_inicial < 600
                ENTAO aux_alim ← sal_inicial * 15/100
                SENAO aux_alim ← sal_inicial * 5/100
```


```
                ESCREVA aux_alim
                sal_iinal ← sal_inicial + aux_alim
                ESCREVA sal_final
                FIM
        FIM
FIM
ATE op = 3
FIM ALGORITMO.
```

Solução PASCAL

```
PROGRAM EX5;
        USES CRT;
        TYPE REGISTRO =     RECORD
                            num_horas_trab, valor_hora : REAL;
                            nome : STRING;
                            turno, cat : STRING;
                            END;
        VAR     func: ARRAY[1..18] OF REGISTRO;
                i, pos_livre, op : INTEGER;
                sal_minimo, sal_inicial, aux_alim, sal_final : REAL;
BEGIN
        FOR i := 1 TO 18 DO
        BEGIN
        func[i].num_horas_trab := 0;
        func[i].valor_hora := 0;
        func[i].nome := '';
        func[i].turno := '';
        func[i].cat := '';
END;
pos_livre := 1;
REPEAT
        CLRSCR;
```

```
        WRITELN('Menu de Opções');
        WRITELN('1 - Registar funcionários');
        WRITELN('2 - Mostrar folha de pagamento');
        WRITELN('3 - Sair');
        WRITELN('Escolha a opçãoo desejada');
        READLN(op);
        IF (op <> 1) AND (op <> 2) AND (op <> 3)
        THEN WRITELN('Op‡Æo Inv lida');
        IF op = 1
THEN BEGIN
        IF pos_livre = 19
        THEN WRITELN('Registo de funcionários lotado')
        ELSE BEGIN
                sal_minimo := 380;
                WRITELN('Escreva o nome do funcionário que deseja incluir');
                READLN(func[pos_livre].nome);
                WRITELN('Escreva o número de horas trabalhadas');
                READLN(func[pos_livre].num_horas_trab);
                WRITELN('Escreva o turno de trabalho');
                READLN(func[pos_livre].turno);
                WHILE (func[pos_livre].turno <> 'M')
               AND (func[pos_livre].turno  <> 'V')
                AND (func[pos_livre].turno  <> 'N') DO
                BEGIN
                        WRITELN('Turno inválido. Escreva novamente');
                        READLN(func[pos_livre].turno);
                END;
                WRITELN('Escreva a categoria');
                READLN(func[pos_livre].cat);
                WHILE (func[pos_livre].cat <> 'O') AND (func[pos_livre].cat <> 'G') DO
                BEGIN
                        WRITELN('Categoria inválida. Escreva novamente');
```


```
                READLN(func[pos_livre].cat);
            END;
            IF func[pos_livre].cat  = 'G'
            THEN IF func[pos_livre].turno =  'N'
            THEN func[pos_livre].valor_hora := sal_minimo * 18/100
            ELSE func[pos_livre].valor_hora := sal_minimo * 15/100;
            IF func[pos_livre].cat  = 'O'
            THEN IF func[pos_livre].turno = 'N'
            THEN func[pos_livre].valor_hora := sal_minimo * 13/100
            ELSE func[pos_livre].valor_hora := sal_minimo * 10/100;
            WRITELN('Funcionário registado com sucesso');
            pos_livre := pos_livre + 1;
        END;
END;
IF op = 2
THEN BEGIN
        WRITELN('Folha de Pagamento');
        IF pos_livre = 1
        THEN WRITELN('Não existe funcionário registado')
        ELSE BEGIN
            FOR  i := 1 TO (pos_livre - 1) DO
            BEGIN
                WRITELN('NOME = ',func[i].nome);
                WRITELN('NÚMERO DE HORAS TRABALHADAS
                 =',func[i].num_horas_trab:5:2);
                WRITELN('VALOR DA HORA TRABALHADA
            =',func[i].valor_hora:5:2);
                sal_inicial := func[i].num_horas_trab * func[i].valor_hora;
                WRITELN('SALÁRIO INICIAL = ',sal_inicial:5:2);
                IF sal_inicial <= 300
                THEN aux_alim := sal_inicial * 20/100
                ELSE IF sal_inicial < 600
```

```
                    THEN aux_alim := sal_inicial * 15/100
                    ELSE aux_alim := sal_inicial * 5/100;
                    WRITELN('AUXÍLIO ALIMENTAÇÃO = ',aux_alim:5:2);
                    sal_final := sal_inicial + aux_alim;
                    WRITELN('SALµRIO FINAL = ',sal_final:5:2);
                    WRITELN('Tecle enter');
                    READLN;
                END;
            END;
END;
READLN;
UNTIL op = 3;
END.
```

Exercícios Propostos

Exercício 1

Uma empresa contratou 15 funcionarios temporarios. De acordo com o valor das vendas mensais, os funcionarios ganham pontos que determinarão os seus salarios no final de cada mes. Sabe-se que eles trabalharão nos meses de novembro de 2014 a janeiro de 2014. Faca um programa que:

a. Registe os nomes dos funcionarios e as suas respectivas vendas mensais.
b. Calcule e mostre a pontuacao geral de cada funcionario nos três meses. Sabe-se que $ 100,00 equivalem a 1 ponto.
c. Calcule e mostre a pontuacao geral de todos os funcionarios em cada mês.
d. Determine e mostre a maior pontuacao atingida nos três meses, mostrando o nome do funcionario. Deverão ser desconsiderados empates.
e. Determine e mostre o valor total vendido.

Exercício 2

Crie um programa para ler o codigo, o sexo (M - masculino; F - feminino) e o número de horas/aula dadas pelos professores de uma escola durante um mês. Sabe-se que um professor ganha $ 40,50 hora aula e que a escola possui 10 professores. Apos a leitura, o programa deverá mostrar:

a. Uma listagem que contendo o codigo, o salário bruto, o desconto e o salário liquido de todos o professores.
b. A média aritmética dos salarios brutos dos professores do sexo masculino.
c. A média aritmética dos salarios brutos dos professores do sexo feminino.

Os descontos devem ser assim calculados:

Sexo	Até 70 horas/aula ao mês	Mais que 70 horas/aula ao mês
Maculino	10%	8%
Feminino	7%	5%

Exercício 3

Veja os campos de alguns registros:

Professor

(numero de registro, nome, cod_titulo, total h/a semanal)

Titulo

(cod_titulo, descricao, valor hora/aula)

Elabore um programa que:

Crie uma rotina para registar os titulos. Sabe-se que nesta escola existem cinco titulos.

Crie uma rotina para registar os professores. Sabe-se que nesta escola trabalham 14 professores, e cada um deve estar associado a um titulo previamente registado.

Crie uma rotina para mostrar a relação de professores, conforme o layout a seguir.

Nº DO REGISTO	NOME	TÍTULO (DESCRIÇÃO)	VALOR HORA/AULA	TOTAL H/A SEMANAL	TOTAL GERAL SEMANAL
111	João da Silva	Mestre	US$ 40,50	10	US$ 405,00
113	Maria Oliveira	Especialista	US$ 30,00	8	US$ 240,00

Exercício 4

Faca um programa que solucione o problema de preenchimento de vagas nos cursos de uma universidade. Cada aluno que entrou na universidade originou um registo com os seguintes campos: número de inscricao, idade, pontuação alcançada (de 0 a 5.000) e codigo do curso pretendido.

A universidade oferece seis cursos, com 40 vagas cada. O problema consiste em distribuir os candidatos entre os cursos, de acordo com a nota final e com a opcao apresentada pelo candidato. Em caso de empate, será atendido primeiro o candidato com maior idade.

Sabe-se que o final da leitura dos dados será determinado pelo campo de inscricao negativo.

Durante a leitura, os dados deverao ser repassados aos vetores (um vetor para cada curso). Como existe limite de vagas e nao há ordem na leitura, o programa devera criar vetores ordenados de maneira decrescente pela pontuação. Assim, aqueles que não conseguiram as 40 melhores pontuações serão descartados.

O programa devera mostrar a lista de alunos que foram aprovados em cada curso.

Exercício 5

A camara de uma cidade fez uma pesquisa entre os seus habitantes, recolhendo dados sobre o salario, idade, sexo e numero de filhos.

Crie um programa que leia os dados de um numero indeterminado de pessoas e, ao final, mostre:

a. a média de idade das mulheres com salario inferior a $ 300,00;
b. a media de salario da população;
c. a media do numero de filhos;
d. o maior salario;
e. a menor idade.

A leitura terminara quando for digitada idade igual a zero.

Ficheiros

Definição de ficheiros em Algoritmos

Estruturas de dados manipuladas fora do ambiente do programa são conhecidas como *ficheiros.* Considera-se como ambiente do programa a memória principal, onde nem sempre é possível ou conveniente manter certas estruturas de dados.

Um ficheiro, que é armazenado num dispositivo de memória secundaria, como discos, por exemplo, pode ser lido ou escrito por um programa e é formado por uma coleção de caracteres (ficheiro texto) ou bytes (ficheiro binário).

Um sistema de base de dados é composto por um ou vários ficheiros. Cada um destes ficheiros possui programas de manutenção, que são: *inclusão*, *exclusão logica* ou *exclusão física*, *alteração*, *consulta geral*, *consulta específica* e *relatórios.*

Existem dois tipos de exclusão de registos: a *exclusão física,* em que, apos a eliminação de um registo, os restantes registos são deslocados, e a *exclusão lógica*, em que os registos possuem um campo adicional, identificando se estão ativos ou inativos, isto é, se foram excluídos. Nos exemplos apresentados neste capítulo, convencionou-se que, se o registo possuir o campo ATIVO com valor zero, significa que foi excluído.

Trabalhar e declarar com ficheiros em PASCAL

O primeiro passo para trabalhar com ficheiros em Pascal é a criação de novos tipos de dados. O primeiro tipo representa um registo composto por vários campos, que informara a estrutura dos dados que serão arquivados. O segundo tipo representa um ficheiro de registos.

```
TYPE    nome_do_registo = RECORD
        nome_do_campo1: tipo_do_dado1;
        nome_do_campo2: tipo_do_dado2;
        nome_do_campoN: t ipo_do_dadoN;
        END;
nome_do_ficheiro = FILE OF nome_do_registo;
VAR     variavel_do_ficheiro : nome_do_ficheiro;
```

variavel_do_registo : nome_do_registo;

Exemplo 1

Neste exemplo (a numeração das linhas não faz parte do programa), temos a definição de dois tipos. O primeiro tipo, REGISTO, define as características de um registo, ou seja, informa que ele será composto pelos campos *nome*, *endereço* e *telefone*. Isto pode ser observado nas linhas de 1 a 5. O segundo tipo, *ficheiro*, servirá para definir variáveis capazes de referenciar ficheiros que armazenam dados no formato registo, como mostra a linha 6. Nas linhas 7 e 8 são declaradas variáveis dos tipos definidos anteriormente, ou seja, as informações geradas pelo programa serão armazenadas em RED. Posteriormente, a variável REG será gravada no ficheiro referenciado por AGENDA.

Exemplo do ficheiro AGENDA:

```
1       TYPE registro =     RECORD
2                           nome : string[30];
3                           endereco : string[20];
4                           telefone : string[10];
5                           END;
6       ficheiro = FILE OF registro;
7       VAR     AGENDA : arquivo;
8               REG : registro;
```

Exemplo 2

Neste exemplo (a numeração das linhas não faz parte do programa), temos a definição de dois tipos. O primeiro tipo, *carro*, define as características de um registo, ou seja, informa que ele será composto pelos campos *matricula*, *marca* e *ano*. Isto pode ser observado nas linhas de 1 a 5. O segundo tipo, frota, servira para definir variáveis capazes de referenciar ficheiros que armazenam dados no formato *carro*. Nas linhas 7 e 8 são declaradas variáveis dos tipos definidos anteriormente. Ou seja, as informações geradas pelo programa serão armazenadas em *carros*.

Posteriormente, a variável CARROS será gravada no ficheiro referenciado por DETRAN.

```
1       TYPE carro =    RECORD
2                       matricula : string[7];
```

```
3                   marca : string[20];
4                   ano : integer;
5                   END;
6       frota = FILE OF carro;
7       VAR     DETRAN : frota;
8               CARROS : carro;
```

Associando variáveis a ficheiros em PASCAL

O comando ASSIGN é utilizado para associar nomes de ficheiros físicos a variáveis de um programa. Desta forma, é possível estabelecer uma ligação entre a execução de um programa na memória principal e os dados armazenados na memória secundaria, por exemplo, o disco rígido.

O comando ASSINGN tem a sintaxe apresentada a seguir.

ASSIGN *{nome_da_variavel_do_tipo_ficheiro,* 'caminho do ficheiro no disco*:\ nome do ficheiro no disco');*

Exemplo 1

ASSIGN (AGENDA, 'AGENDA.DAT') ;

A variável agenda permitirá acesso ao ficheiro AGENDA.DAT. Como não foi especificado um caminho, o ficheiro AGENDA.DAT deverá estar gravado no mesmo local do ficheiro a ser executado.

Exemplo 2

ASSIGN {DETRAN,'C:\EXEMPLOS\CARROS.DAT') ;

A variável DETRAN permitirá o acesso ao ficheiro CARROS .DAT. O ficheiro CARROS.DAT deverá estar armazenado no caminho C:\EXEMPLOS.

Criação de um novo ficheiro em PASCAL

Na linguagem PASCAL, novos ficheiros podem ser criados utilizando o comando REWRITE. Se este comando for usado, a fazer referência a um ficheiro já existente, todos os seus dados serão destruídos, pois o comando REWRITE posiciona o seu ponteiro no primeiro

registo, apagando todo o seu conteúdo. Se o ficheiro referenciado não existir, o comando REWRITE forcara a sua criação.

A sintaxe correta para o comando REWRITE é apresentada a seguir.

REWRITE *(nome_da_variavel_do_tipo_ficheiro) ;*

Exemplo 1

REWRITE(AGENDA);

No exemplo, foi criado o ficheiro associado à variável AGENDA. Se ele já existir, os seus dados foram destruídos.

Exemplo 2

REWRITE(DETRAN);

No exemplo, foi criado o ficheiro associado à variável DETRAN. Se ele já existir, os seus dados foram destruídos

Abrir ficheiros já existentes

O comando RESET deverá ser utilizado cada vez que for necessário abrir ficheiros. Deve-se posicionar o ponteiro no primeiro registo, sem destruir os dados já existentes no ficheiro. A sintaxe desse comando é apresentada a seguir.

RESET (*nome_da_variavel_do_tipo_ficheiro)*;

Exemplo 1

RESET(AGENDA);

No exemplo, foi aberto o ficheiro associado à variável AGENDA, sem destruir os dados já existentes.

Exemplo 2

RESET(DETRAN);

No exemplo, foi aberto o ficheiro associado à variável DETRAN, sem destruir os dados já existentes.

Fechar um ficheiro

O comando CLOSE é utilizado para fechar ficheiros abertos pelo comando REWRITE ou pelo comando RESET. É importante salientar que todas as atualizações feitas num ficheiro só serão efetuadas quando ele for fechado. Alem disso, nenhuma mudança poderá ser feita num ficheiro fechado. Observe a sintaxe a seguir.

CLOSE *(nome_da_variavel_do_tipo_ficheiro)* ;

Exemplo 1

CLOSE(AGENDA);

No exemplo, foi fechado o ficheiro associado à variável AGENDA.

Exemplo 2

CLOSE(DETRAN);

No exemplo, foi fechado o ficheiro associado à variável DETRAN.

Ler dados de um ficheiro

Para ler os dados contidos num ficheiro, deve-se executar uma leitura no disco. Em PASCAL, isto pode ser feito através do comando READ, que possui a sintaxe a seguir.

READ(*nome_da_variavel_do_tipo_ficheiro, nome_da_variavel_do_tipo_registo)*;

Exemplo 1

READ(AGENDA, REG);

No exemplo, os dados lidos no ficheiro associado à variável AGENDA serão copiados para a variável de programa REG. A partir dai, o programa tratará esta variável como um registo qualquer.

Exemplo 2

READ(DETRAN, CARROS);

No exemplo, os dados lidos no ficheiro associado à variável DETRAN serão copiados para a variável de programa CARROS. A partir dai, o programa tratará esta variável como um

registo qualquer.

Gravar dados num ficheiro

A linguagem PASCAL permite a gravação de dados usando o comando WRITE, como mostra a sintaxe a seguir.

WRITE *(nome_da_variavel_do_tipo_ficheiro, nome_da_variavel_do_tipo_registo)*;

Exemplo 1

WRITE(AGENDA, REG);

No exemplo, os dados contidos na variável de programa REG serão copiados para o ficheiro associado à variável AGENDA.

Exemplo 2

WRITE(DETRAN, CARROS);

No exemplo, os dados contidos na variável de programa CARROS serão copiados para o ficheiro associado à variável DETRAN

Movimentar um ponteiro num ficheiro

O comando SEEK é utilizado para posicionar o ponteiro no registo desejado. O primeiro registo do ficheiro é sempre o de número zero. A sintaxe do comando SEEK é apresentada a seguir.

SEEK *{nome_da_variavel_do_tipo_ficheiro, numero_do_registo)* ;

Exemplo 1

SEEK(AGENDA, 2);

No exemplo, o ponteiro do ficheiro AGENDA está na segunda posição, ou seja, no inicio do terceiro registro gravado no ficheiro.

Exemplo 2

SEEK(DETRAN, 0);

No exemplo, o ponteiro do arquivo DETRAN está na posição zero, ou seja, no inicio do primeiro registo gravado no ficheiro.

Obter o número de registos de um ficheiro

O comando FILESIZE é utilizado para retornar o número de registos existentes num ficheiro. A sintaxe deste comando é mostrada a seguir.

FILESIZE {*nome_da_variavel_do_tipo_ficheiro*);

Exemplo 1

tamanho : = FILESIZE(AGENDA);

No exemplo, é retornada para a váriavel tamanho à quantidade de ficheiros gravados no ficheiro AGENDA.

Exemplo 2

tamanho := FILESIZE(DETRAN);

No exemplo, é retomado para a variável tamanho a quantidade de registos gravados no ficheiro DETRAN.

Obter a posição do ponteiro num ficheiro

O comando FILEPOS é utilizado para retornar o número do registo onde o ponteiro está localizado. A forma correta para utilizar este comando é:

posicao := FILEPOS*(nome_da_variavel_do_tipo_ficheiro)*;

Exemplo 1

posicao := FILEPOS(AGENDA);

No exemplo, é retornado para a variável posição o número do registo onde o ponteiro do ficheiro AGENDA está posicionado.

Exemplo 2

posicao := FILEPOS(DETRAN);

No exemplo, é retornado para a variável posição o número do registo onde o ponteiro do ficheiro DETRAN está posicionado.

Verificar o final do ficheiro

O comando NOT EOF é utilizado para verificar se o ponteiro chegou ao final do ficheiro. O retorno deste comando poderá ser verdadeiro, caso o final tenha sido encontrado, ou falso, se acontecer o contrário. A seguir e mostrada a sintaxe do comando NOR EOF.

```
WHILE NOT EOF {nome_da_variavel_do_tipo_ficheiro} DO
BEGIN
        comandos;
END;
```

Exemplo 1

```
WHILE NOT EOF(DETRAN) DO
BEGIN
        READ (DETRAN, REG) ;
END;
```

O exemplo acima mostra como percorrer todos os registos de um ficheiro Observe que, dentro da estrutura de repetição WHILE, foi colocado o comando READ. Isso quer dizer que, a cada leitura feita no ficheiro, o ponteiro movimenta-se para o registo seguinte. Desta forma, enquanto forem obtidas informações no ficheiro, a repetição continuara. Quando o ponteiro chegar ao final do ficheiro DETRAN, o WHILE será finalizado.

Exercícios Resolvidos

Exercício 1

Faça um programa para criar um ficheiro chamado ALUNOS:DAT, no qual cada registo será composto pelos seguintes campos: numero, nome, curso, nota1, nota2.

Solução Algoritmo

- Abrir ficheiro
- Fechar ficheiro

Solução 1 PASCAL

```
PROGRAM EX1;
        USES CRT;
        TYPE    aluno = RECORD
                numero : INTEGER;
                nome : STRING[20];
                curso: STRING[15];
                nota1, nota2 : REAL;
                END;
        classe = FILE OF aluno;
        VAR ALUNOS: classe;
        A: aluno;
BEGIN
        CLRSCR;
        ASSIGN(ALUNOS, 'C:\ALUNOS.DAT');
        REWRITE(ALUNOS);
        CLOSE(ALUNOS);
        WRITELN('Ficheiro ALUNOS criado com sucesso!!!');
        READLN;
END.
```

Exercício 2

Faça um programa para incluir alunos no ficheiro criado no Exercicio1, lembrando que não podem existir dois alunos com o mesmo numero

Solução Algoritmo

Os passos para a inclusão sequencial de registos num ficheiro são:

- Abrir o ficheiro que sofrerá inclusões.
- Escrever os dados que serão incluídos e fazer a sua validação.
- Se o ficheiro estiver vazio, gravar dados escritos no ficheiro.
- Se o ficheiro não estiver vazio, percorrer todo o ficheiro, do início ao fim, ou até encontrar o campo-chave igual ao que se deseja incluir.

- Se encontrar o campo-chave igual ao que se deseja incluir, mostrar mensagem.
- Se não encontrar o campo-chave igual ao que se deseja incluir, então gravar dados escritos no ficheiro.
- Fechar o ficheiro.

Os passos para a inclusão ordenada de registos num ficheiro são:

- Abrir o ficheiro que sofrerá inclusões.
- Escrever os dados que serão incluídos e fazer a sua validação.
- Se o ficheiro estiver vazio, gravar dados escritos no ficheiro.
- Se o ficheiro não estiver vazio, percorrer todo o ficheiro, do início ao fim, ou até encontrar o campo-chave igual ao que se deseja incluir.
- Se encontrar o campo-chave igual ao que se esta querendo incluir, mostrar mensagem.
- Se não encontrar o campo-chave igual ao que se deseja incluir, ocorrera o deslocamento de registos para gravar os dados escritos no ficheiro na posição ordenada.
- Fechar o ficheiro.

Solução PASCAL

```
PROGRAM EX2;
        USES CRT;
        TYPE    aluno = RECORD
                numero : INTEGER;
                nome : STRING[20];
                curso: STRING[15];
                nota1, nota2 : REAL;
                END;
        classe = FILE OF aluno;
        VAR     ALUNOS: classe;
                A: aluno;
                K, I : INTEGER;
                ACHOU : BOOLEAN;
```

```
            NUMERO: INTEGER;
            NOME: STRING[20];
            CURSO: STRING[15];
            NOTA1,NOTA2: REAL;
BEGIN
      CLRSCR;
      ASSIGN(ALUNOS, 'C:\ALUNOS.DAT');
      {$I-}
      RESET(ALUNOS);
      {$I+}
      IF IORESULT = 2
      THEN REWRITE(ALUNOS);
      K := FILESIZE(ALUNOS);
      WRITE('ESCREVA O NÚMERO DO ALUNO A SER INCLUÍDO ');
      READLN(NUMERO);
      IF K = 0
      THEN BEGIN
            WRITE('ESCREVA O NOME DO ALUNO A SER INCLUÍDO ');
            READLN(NOME);
            WRITE('ESCREVA O CURSO DO ALUNO A SER INCLUÍDO ');
            READLN(CURSO);
            REPEAT
            WRITE('ESCREVA A PRIMEIRA NOTA DO ALUNO A SER INCLUÍDO ');
            READLN(NOTA1);
            UNTIL (NOTA1 >= 0) AND (NOTA1 <= 10);
            REPEAT
            WRITE('ESCREVA A SEGUNDA NOTA DO ALUNO A SER INCLUÍDO ');
            READLN(NOTA2);
            UNTIL (NOTA2 >= 0) AND (NOTA2 <= 10);
            A.NUMERO := NUMERO;
            A.NOME := NOME;
            A.CURSO := CURSO;
```

```
        A.NOTA1:= NOTA1;
        A.NOTA2 := NOTA2;
        WRITE(ALUNOS, A);
        WRITELN('ALUNO INCLUÖDO');
END
ELSE BEGIN
        I := 0;
        ACHOU := FALSE;
        WHILE ( I <= (K-1)) DO
        BEGIN
                READ(ALUNOS, A);
                IF A.NUMERO = NUMERO
                THEN BEGIN
                        ACHOU := TRUE;
                        I := K+1;
                END
                ELSE BEGIN
                        I := I+1;
                        SEEK(ALUNOS, I);
                END;
        END;
        IF ACHOU = TRUE
        THEN WRITELN('ESTE ALUNO Jµ ESTµ CADASTRADO ')
        ELSE BEGIN
                WRITE('ESCREVA O NOME DO ALUNO A SER INCLUÍDO ');
                READLN(NOME);
                WRITE('ESCREVA O CURSO DO ALUNO A SER INCLUÍDO ');
                READLN(CURSO);
                REPEAT
                WRITE('ESCREVA A PRIMEIRA NOTA DO ALUNO A SER
        INCLUÍDO ');
                READLN(NOTA1);
```


```
                UNTIL (NOTA1 >= 0) AND (NOTA1 <= 10);
                REPEAT
                WRITE('ESCREVA A SEGUNDA NOTA DO ALUNO A SER
            INCLUÍDO ');
                READLN(NOTA2);
                UNTIL (NOTA2 >= 0) AND (NOTA2 <= 10);
                SEEK(ALUNOS, K);
                A.NUMERO := NUMERO;
                A.NOME := NOME;
                A.CURSO := CURSO;
                A.NOTA1:= NOTA1;
                A.NOTA2 := NOTA2;
                WRITE(ALUNOS, A);
                WRITELN('ALUNO INCLUÍDO');
            END;
        END;
        READLN;
        CLOSE(ALUNOS);
END.
```

Exercício 3

Faça um programa para alterar as notas dos alunos do ficheiro criado no Exercício 1.

Solução Algoritmo

- Abrir o ficheiro que sofrerá alterações.
- Escrever o campo-chave do registo que sofrera alterações.
- Se o ficheiro estiver vazio, mostrar mensagem.
- Se o ficheiro não estiver vazio, percorrer todo o ficheiro, do início ao fim, ou até encontrar o campo-chave que possui os dados que se deseja alterar.
- Se não encontrar o campo-chave igual ao que se deseja alterar os dados, mostrar mensagem.
- Se encontrar o campo-chave igual ao que se deseja alterar os dados, mostrar

os dados do registo que sofrerá alterações, escrever e validar os novos dados, posicionar no registo que sofrerá alterações e gravar.

- Fechar o arquivo.

Solução PASCAL

```
PROGRAM EX3;
USES CRT;
        TYPE aluno = RECORD
                        numero : INTEGER;
                        nome : STRING[20];
                        curso: STRING[15];
                        nota1, nota2 : REAL;
                        END;
        classe = FILE OF aluno;
        VAR     ALUNOS: classe;
                A: aluno;
                K, I, P, POSICAO: INTEGER;
                ACHOU : BOOLEAN;
                NUMERO: INTEGER;
                NOME: STRING[20];
                CURSO: STRING[15];
                NOTA1,NOTA2: REAL;
BEGIN
        CLRSCR;
        ASSIGN(ALUNOS, 'C:\ALUNOS.DAT');
        {$I-}
        RESET(ALUNOS);
        {$I+}
        IF IORESULT = 2
        THEN REWRITE(ALUNOS);
        K := FILESIZE(ALUNOS);
        I := 0;
```

```
ACHOU := FALSE;
WRITE('ESCREVA O NÚMERO DO ALUNO PARA ALTERAR AS NOTAS ');
READLN(NUMERO);
WHILE ( I <= K -1 ) DO
BEGIN
        SEEK(ALUNOS, I);
        READ(ALUNOS,A);
        IF A.NUMERO = NUMERO
        THEN BEGIN
                POSICAO := I;
                I := K + 1;
                ACHOU := TRUE;
        END
        ELSE I := I + 1;
END;
IF ACHOU = TRUE
THEN BEGIN
        WRITELN('NOTA 1 = ', A.NOTA1:5:2);
        REPEAT
        WRITE('ESCREVA A NOVA NOTA 1 ');
        READLN(NOTA1);
        UNTIL (NOTA1 >= 0) AND (NOTA1 <= 10);
        WRITELN('NOTA 2 = ', A.NOTA2:5:2);
        REPEAT
        WRITE('ESCREVA A NOVA NOTA 2 ');
        READLN(NOTA2);
        UNTIL (NOTA2 >= 0) AND (NOTA2 <= 10);
        A. NOTA1 := NOTA1;
        A.NOTA2 := NOTA2;
        SEEK(ALUNOS, POSICAO);
        WRITE(ALUNOS, A);
        WRITELN('ALTERACÇO EFETUADA ');
```

```
        END
        ELSE WRITELN(‘ ALUNO NÇO CADASTRADO ‘);
        READLN;
        CLOSE(ALUNOS);
END
```

Exercício 4

Faça um programa para alterar o curso dos alunos do ficheiro criado no Exercício 1.

Solução Algoritmo

- Abrir o ficheiro que sofrera alterações.
- Escrever o campo-chave do registo que sofrerá alterações.
- Se o ficheiro estiver vazio, mostrar mensagem.
- Se o ficheiro não estiver vazio, percorrer todo o ficheiro, do início ao fim, ou até encontrar o campo-chave igual ao que se deseja alterar.
- Se não encontrar o campo-chave contendo os dados que se deseja alterar, mostrar mensagem.
- Se encontrar o campo-chave contendo os dados que se deseja alterar, mostrar os dados do registo que sofrerá alterações, escrever e validar os novos dados, posicionar no registo que sofrerá alterações e gravar.
- Fechar o arquivo.

Solução PASCAL

```
PROGRAM EX4;
        USES CRT;
        TYPE aluno = RECORD
                        numero : INTEGER;
                        nome : STRING[20];
                        curso: STRING[15];
                        nota1, nota2 : REAL;
                        END;
        classe = FILE OF aluno;
```


```
        VAR     ALUNOS: classe;
        A: aluno;
        K, I, P, POSICAO: INTEGER;
        ACHOU : BOOLEAN;
        NUMERO: INTEGER;
        NOME: STRING[20];
        CURSO: STRING[15];
        NOTA1,NOTA2: REAL;
BEGIN
        CLRSCR;
        ASSIGN(ALUNOS, 'C:\ALUNOS.DAT');
        {$I-}
        RESET(ALUNOS);
        {$I+}
        IF IORESULT = 2
        THEN REWRITE(ALUNOS);
        K := FILESIZE(ALUNOS);
        I := 0;
        ACHOU := FALSE;
        WRITE('ESCREVA O NÚMERO DO ALUNO PARA ALTERAR O CURSO ');
        READLN(NUMERO);
        WHILE ( I <= K -1 ) DO
        BEGIN
                SEEK(ALUNOS, I);
                READ(ALUNOS,A);
                IF A.NUMERO = NUMERO
                THEN BEGIN
                        POSICAO := I;
                        I := K + 1;
                        ACHOU := TRUE;
                END
                ELSE I := I + 1;
```

```
        END;
        IF ACHOU = TRUE
        THEN BEGIN
                WRITELN('CURSO = ', A.CURSO);
                WRITE('ESCREVA O NOVO CURSO ');
                READLN(CURSO);
                A.CURSO := CURSO;
                SEEK(ALUNOS, POSICAO);
                WRITE(ALUNOS, A);
                WRITELN('ALTERAÇÃO EFETUADA ');
        END
        ELSE WRITELN('ALUNO NÇO CADASTRADO ');
        READLN;
        CLOSE(ALUNOS);
END.
```

Exercício 5

Faça um programa para excluir os alunos do arquivo criado no Exercício 1.

Solução Algoritmo

Conforme apresentado anteriormente, existem dois tipos de exclusão: física e logica.

Os passos para a exclusão física de registos num ficheiro são:

- Abrir o ficheiro que sofrerá exclusão.
- Escrever o campo-chave do registo que será excluído.
- Se o ficheiro estiver vazio, mostrar mensagem.
- Se o ficheiro não estiver vazio, percorrer todo o ficheiro, do começo ao fim, ou até encontrar o campo-chave que se deseja excluir.
- Se não encontrar o campo-chave que se quer excluir, selecionar mensagem.
- Se encontrar o campo-chave que se quer excluir, ocorrera o deslocamento de registos para sobrepor o registo que será excluído.
- Fechar o ficheiro.

Os passos para a exclusão logica de registos num ficheiro são:

- Abrir o ficheiro que sofrerá exclusão.
- Escrever o campo-chave do registo que será excluído.
- Se o ficheiro estiver vazio, selecionar mensagem.
- Se o ficheiro não estiver vazio, percorrer todo o ficheiro, do início ao fim, ou até encontrar o campo-chave que se quer excluir.
- Se não encontrar o campo-chave que se quer excluir, selecionar mensagem.
- Se encontrar o campo-chave que se objetiva excluir, o campo de marcação (campo ativo) do registo será marcado como excluído.
- Fechar o ficheiro.

Solução PASCAL

```
PROGRAM EX5;
        USES CRT;
        TYPE aluno = RECORD
                        numero : INTEGER;
                        nome : STRING[20];
                        curso: STRING[15];
                        nota1, nota2 : REAL;
                        END;
        classe = FILE OF aluno;
        VAR     ALUNOS, AUXILIAR: classe;
                A, AUX: aluno;
                K, I, P, POSICAO: INTEGER;
                ACHOU : BOOLEAN;
                NUMERO: INTEGER;
                NOME: STRING[20];
                CURSO: STRING[15];
                NOTA1,NOTA2: REAL;
BEGIN
        CLRSCR;
        ASSIGN(ALUNOS, 'C:\ALUNOS.DAT');
```

```
{$I-}
RESET(ALUNOS);
{$I+}
IF IORESULT = 2
THEN REWRITE(ALUNOS);
ASSIGN(AUXILIAR, 'C:\AUXILIO.DAT');
REWRITE(AUXILIAR);
K := FILESIZE(ALUNOS);
IF K = 0
THEN WRITELN('NÃO EXISTE ALUNO REGISTADO')
ELSE BEGIN
        WRITE('ESCREVA O NÚMERO DO ALUNO A SER EXCLUÍDO ');
        READLN(NUMERO);
        WHILE (ACHOU = FALSE) AND (NOT EOF(ALUNOS)) DO
        BEGIN
                READ(ALUNOS,A);
                IF A.NUMERO = NUMERO
                THEN BEGIN
                        ACHOU := TRUE;
                        POSICAO := FILEPOS(ALUNOS) -1;
                END;
        END;
        IF ACHOU = TRUE
        THEN BEGIN
                IF K = 1
                THEN BEGIN
                        REWRITE(ALUNOS);
                        WRITELN('ALUNO EXCLUÍDO COM SUCESSO');
                END
                ELSE BEGIN
                        ACHOU := FALSE;
                        K := 0;
```


```
SEEK(ALUNOS, K);
WHILE K < POSICAO DO
BEGIN
        SEEK(ALUNOS, K);
        READ(ALUNOS,A);
        AUX.NUMERO := A.NUMERO;
        AUX.NOME := A.NOME;
        AUX.CURSO := A.CURSO;
        AUX.NOTA1 := A.NOTA1;
        AUX.NOTA2 := A.NOTA2;
        WRITE(AUXILIAR, AUX);
        K := K +1;
END;
K := POSICAO + 1;
WHILE K <= FILESIZE(ALUNOS) - 1 DO
BEGIN
        SEEK(ALUNOS, K);
        READ(ALUNOS,A);
        AUX.NUMERO := A.NUMERO;
        AUX.NOME := A.NOME;
        AUX.CURSO := A.CURSO;
        AUX.NOTA1 := A.NOTA1;
        AUX.NOTA2 := A.NOTA2;
        WRITE(AUXILIAR, AUX);
        K := K +1;
END;
REWRITE(ALUNOS);
RESET(AUXILIAR);
WHILE NOT EOF(AUXILIAR) DO
BEGIN
        READ(AUXILIAR, AUX);
        A.NUMERO := AUX.NUMERO;
```

```
                    A.NOME := AUX.NOME;
                    A.CURSO :=  AUX.CURSO;
                    A.NOTA1:= AUX.NOTA1;
                    A.NOTA2:= AUX.NOTA2;
                    WRITE(ALUNOS, A);
                END;
                WRITELN('ALUNO EXCLUÍDO');
            END;
        END
        ELSE WRITELN('ESTE ALUNO NÇO ESTµ CADASTRADO');
    END;
    READLN;
    CLOSE(ALUNOS);
    CLOSE(AUXILIAR);
END.
```

Exercícios Propostos

Exercício 1

Faça um programa para consultar o número, o nome e a média de todos os alunos registados no ficheiro do Exercício 1.

Exercício 2

Faça um programa para consultar o número, o nome e a média de todos os alunos registados no ficheiro do Exercício 1 e que estejam aprovados, ou seja, com média maior ou igual a 7.

Exercício 3

Faça um programa para consultar o número, o nome e o curso de todos os alunos registados no ficheiro do Exercício 1 e que estejam reprovados, ou seja, com média inferior a 3.

Exercício 4

Faça um programa para consultar o nome de todos os alunos registados no ficheiro do Exercício 1 e que estejam de exame, ou seja, com média entre 3 (inclusive) e 7.

Exercício 5

Faça um programa para consultar o nome de todos os alunos de um curso.

Exercício 6

Faça um programa para criar um ficheiro chamado VENDAS.DAT, em que cada registo será composto pelos seguintes campos: codigo_vendedor, nome_vendedor, valor_venda e mês.

Exercício 7

Faça um programa para incluir um vendedor no ficheiro criado no Exercício 11, lembrando que não podem existir dois vendedores com o mesmo código e mesmo mês de vendas.

Exercício 8

Faça um programa para alterar o valor de uma venda no ficheiro criado no Exercício 11.

Exercício 9

Faça um programa para excluir um vendedor no ficheiro criado no Exercício 11.

Exercício 10

Faça um programa para consultar o valor da venda de um vendedor em determinado mês no ficheiro criado no Exercício 11.

Bibliografia

AZUL, Artur Augusto, *Técnicas e Linguagens de Programação.* Porto: Porto Editora, 1994.

BARATA, M.; FONSECA, J.; CARVALHO, M., *Princípios de Programação em Pascal*. Queluz: Edições EPGE, 1993.

BIASI, Ronaldo Sérgio, *Guia Rápido para Turbo C.* Rio de Janeiro: Editora Lutécia, 1990.

CARDOSO, Vasco, Fundamental do Turbo Pascal 6 e 7. Lisboa: FCA. sd.

CARRIÇO, José António; CARRIÇO, António João, *Programação em Visual Basic.Net.* Lisboa: CTI, 2002.

CUESTA, L.; PADILLA, A.; REMIRO, F., *Eletrónica Digital*. Amadora: McGrawHill, 1994.

DAMAS, Luís Manuel Dias, *Linguagem C.* Lisboa: FCA, 1999.

DICTOR, Evan, *Visual Basic Controls in a Nutshell.* Sebastopol, USA: O'Reilly. 1999.

FERREIRA, João, Técnicas Avançadas em Visual Basic 6. Lisboa: FCA, 2001.

GOTTFRIED, B., *Programação em Pascal*. Lisboa: McGraw-Hill, 1994.

GUERREIRO, Pedro, *Elementos de Programação com C.* Lisboa: FCA, 2001.

JENSEN, K.; WIRTH, N., *Pascal - User Manual and Report*. New York: Springer-Verlag, 1975.

LOMAX, Paul, *VB & VBA In a Nutshell.* Sebastopol, USA: O'Reilly, 1998.

MACDONALD, Matthew, Visual Basic 2005: A Developer's Notbook. Sebastopol, USA: O'Reilly, 2005.

NINA, Nuno, *Visual Basic 6,* 3ª ed.. Lisboa: FCA, 1999.

ROCHA, António Manuel, Adrego da, *Introdução à Programação Usando C.* Lisboa: FCA, 2006.

SCHILDT, Herbert, *C The Complete Reference.* 2ª ed. Berkeley, USA: McGrawHill, 1990.

SHAMMAS, Namir, *Programação em Turbo C++.* Lisboa: Editorial Presença, 1994.

SHARMA, Ashok, *Programmable Logic Handbook* .Berkeley, USA: MacGrawHill, 2003.

Notas