DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Algoritmia e Programação

AULAS PRÁTICAS

2 0 0 5 / 2 0 0 6

Ana Madureira, Ana Almeida, José Avelino Marinho, Ricardo Almeida, Paulo Baltarejo Sousa, Jorge Santos, António Vieira de Castro, Alexandre Gouveia

ÍNDICE

1. ESTRUTURA DE CONTROLO DE SEQUÊNCIA

Objectivos

Introduzir a noção de execução sequencial de instruções e de estruturas de dados. Abordar o conceito de operações, operandos e operadores.

Exercício 1

Escreva um algoritmo que permita converter um valor em horas para segundos e mostre o valor em segundos.

Exercício 2

Escreva um algoritmo que faça a divisão de dois números inteiros e mostre o resultado na forma *dividendo/divisor = quociente* *E*x: 5/2=2,5.

Exercício 3

Escreva um algoritmo que permita calcular a área de um triângulo dada a sua base e altura e visualize o seu resultado.

Exercício 4

Escreva um algoritmo que permita calcular uma coroa circular (π*(raio_maior^2-raiomenor^2)). Considere o valor de π = 3,1415

Exercício 5

Escreva um algoritmo que permita converter um valor em *bytes* para *bits* e *kilobytes*.

Exercício 6

Escreva um algoritmo que calcule o resto da divisão de dois números inteiros.

2. INSTRUÇÕES DE CONTROLO DE DECISÃO E REPETIÇÃO

Objectivos

Identificar situações em que é necessário usar uma instrução condicional ou de repetição. Capacidade de optar por uma ou outra face a um enunciado.

Exercício 7

Escreva um algoritmo que permita determinar se um determinado número é par ou impar.

Exercício 8

Escreva um algoritmo que permita classificar um triângulo dados os seus lados:

Exercício 9

Faça um algoritmo para multiplicar dois números sem recurso ao operador multiplicação (*).

Exercício 10

Escreva um algoritmo que permita localizar o quadrante dum ponto, dadas as suas coordenadas.

3. INSTRUÇÕES DE CONTROLO DE DECISÃO E REPETIÇÃO EM ESTRUTURAS ENCAIXADAS

Objectivos

Identificar situações em que é necessário usar uma instrução condicional e/ou de repetição em estruturas encaixadas.

Exercício 11

Faça um algoritmo que peça ao utilizador que introduza dois números a somar e o resultado. Utilizando a *prova dos nove* o programa deve dizer se o resultado está correcto ou não.

Exercício 12

Escreva um algoritmo que, dados 2 números, diga quantos números pares existem no intervalo constituído pelos dois números. Especifique as estruturas de dados necessárias à resolução do problema.

Exercício 13

Ler n números, determinar o maior, o menor e a média dos n números lidos. Elabore o algoritmo.

Exercício 14

Faça um algoritmo que permita simular o funcionamento de uma máquina de café presente no ISEP. O Custo de cada café é de 0.35€. A máquina aceita moedas de 0.05€, 0.1€, 0.2€, 0.5€, 1 e 2€. A máquina deve visualizar o troco de forma a devolver o menor número possível e quantas moedas de cada deve devolver

Nota: O problema deve ser resolvido sem recurso à operação divisão

Exercício 15

Dois números inteiros, X, Y, dizem-se amigos quando a soma dos divisores de X é igual a Y e a soma dos divisores de Y é igual a X. Escreva um algoritmo que determine se dois números são amigos.

Exercício 16

Construa um programa com o objectivo de determinar o valor, ao fim de *n* anos, de um depósito bancário da quantia *q*, sabendo que a taxa de juro inicial *j* (superior a 5.0%), decresce todos os anos de um valor 0.5% até

atingir o mínimo 5.0%. Suponha que os juros são sempre capitalizados. Indique para cada ano, o capital inicial, taxa de juro, juros e capital final.

4. FUNÇÕES

Objectivos

Utilização de funções de vários tipos, com e sem argumentos, e de variáveis locais e globais.

Capacidades da linguagem C++ utilizadas pela primeira vez

Funções de tipo void, inteiro e char
Variáveis globais e locais
Passagem de parâmetros por valor referência e apontador

Exercício 17

Recorrendo a funções elabore um programa que dados dois números inteiros a e b determine:
a) a sua soma;
b) a sua multiplicação;
c) indique qual o maior dos dois
Nota: o resultado de cada operação deve ser retornado pela respectiva função

Exercício 18

Elabore uma função que dado um ano verifique se ele é ou não bissexto.

Exercício 19

Escreva um programa que permita calcular o valor da seguinte expressão, em que **n** e **r** são lidos do teclado, mas utilizando uma função para calcular o factorial.

$$nCr = \frac{factorial(n)}{factorial(r) * factorial(n-r)}$$

A função para o cálculo do factorial de um número deve ter as características seguintes:

a) A função não tem parâmetros e não devolve nenhum valor.
b) A função tem um parâmetro, o valor de n, e não devolve nenhum valor.
c) A função não devolve nenhum valor e tem dois parâmetros: o valor de n e o resultado.
d) A função devolve o resultado do cálculo e tem um parâmetro: o valor de n.

Exercício 20

Elabore um programa que que permita inserir uma sequência de caracteres até que o utilizador introduza o '*'. Por cada carcater intorduzido o programa deve apresentar o respectivo codigo ascii. Para tal elabore uma função que dado um carácter apresente o código ASCII correspondente

Exercício 21

Escreva um programa que converta um valor binário para decimal, utilizando uma função para o efeito.

Exercício 22

Escreva uma função que recebe dois valores inteiros(por apontador ou por referência) e troque os valores.

Exercício 23

Escreva uma função que dados três números, x, y e z(por apontador ou por referência), os ordene para que x <y<z.

5. VECTORES

Objectivos

Identificar situações em que é vantajosa a utilização de arrays unidimensionais.
Declaração de vectores.
Ler, armazenar e referenciar dados num vector.
Identificar limitações e erros frequentes no dimensionamento de arrays.

Capacidades da linguagem C++ utilizadas pela primeira vez

#define

Exercício 24

Faça um programa que defina um vector de *N* inteiros e implemente as seguintes funções:

- Uma função que faça a leitura de 10 valores (inteiros), guardando-os em memória RAM;
- Uma função que retorne o índice da primeira ocorrência de um dado valor no vector (-1 se não existir);
- Uma função que devolva a maior diferença entre dois elementos consecutivos;
- Uma função que devolva um vector apenas com os elementos que são maiores que os seus vizinhos (direita e esquerda), bem como o número de elementos do novo vector;

  Exemplo:

vector de entrada:	5	4	7	8	2	9	6

resultado:	5	8	9

- Uma função que rode os elementos do vector para a direita, uma posição.

  Exemplo:

vector de entrada:	5	4	7	8	2	9	6

resultado:	6	5	4	7	8	2	9

Exercício 25

Em qualquer experiência existe um certo erro associado aos valores obtidos. Uma técnica conhecida como *suavização* pode ser utilizada para reduzir o efeito desse erro na análise dos resultados. Escreva então um programa que defina um vector de *N* reais e implemente uma função que produza uma suavização sobre o vector, *V*.

Cada elemento v_i é substituído pela média dos valores de v_{i-1}, v_i e v_{i+1} (excepto o primeiro e o último). O primeiro elemento do vector é suavizado com base na média entre os dois primeiros valores e o último elemento é suavizado com base na média entre os dois últimos.

Exercício 26

Escreva um programa que defina 2 vectores de *N* inteiros e implemente as seguintes funções:

- Uma função que faça a leitura de 2 vectores de inteiros com 10 elementos cada, guardando-os em memória RAM;
- Uma função que devolva um vector com a reunião dos dois vectores, sem valores repetidos, bem como o número de elementos do novo vector;
- Uma função que devolva um vector com a intersecção dos dois vectores, bem como o número de elementos do novo vector.

Exercício 27

Considere um corredor com mil portas, numeradas de 1 a 1000, que se encontram todas fechadas. Por esse corredor passarão mil pessoas, que modificarão o estado das portas cujo número seja múltiplo do seu número de passagem: a pessoa com o número 3 modificará o estado (fechará se estiverem abertas ou abrirá se estiverem fechadas) das portas nº 3, 6, 9, 12, … e a pessoa com o número 7 fará o mesmo às portas 7, 14, 21, etc.

Construa um programa que permita saber quantas são e quais são as portas abertas após a passagem da milésima pessoa.

6. ESTRUTURAS E VECTORES DE ESTRUTURAS

Exercício 28

Sabendo que um ponto é constituído por três coordenadas e uma recta pode ser definida por dois pontos.

a. Defina as estruturas ponto e recta;
b. Desenvolva uma função que permita definir um ponto. A função tem o seguinte protótipo: *void inserirPonto(Ponto &p);*
c. Desenvolva uma função que mostre o valor das coordenadas de um ponto. A função tem o seguinte protótipo: *void inserirPonto(Ponto p);*
d. Desenvolva uma função que permita alterar valor das coordenadas de um ponto. A função tem o seguinte protótipo: *void alterarPonto(Ponto *p);*
e. Desenvolva uma função que permita definir uma recta. A função tem o seguinte protótipo: *void inserirRecta(Recta *r);*
f. Desenvolva a função que mostre o valor das coordenadas dos pontos que a constituem. A função tem o seguinte protótipo: *void inserirRecta(Recta &r);*
g. Desenvolva a função que calcule o comprimento de uma recta. A função tem o seguinte protótipo: *double comprimentoRecta(Recta r);*

Exercício 29

Uma empresa de construção civil pretende uma aplicação informática de gestão de recursos humanos. A empresa não prevê ultrapassar os 100 funcionários. Os dados dos funcionários são os seguintes: o número, o nome, a categoria, o vencimento e a data de entrada dos funcionários da empresa

a. Defina as estruturas de dados.
b. Escreva uma função para ler os dados de um funcionário.
c. Escreva uma função para listar os dados de um funcionário.
d. Escreva uma função para ler os dados dos n funcionários da empresa.
e. Escreva uma função para listar os dados dos n funcionários da empresa.

Exercício 30

Pretende-se um programa para a gestão de clientes de uma discoteca. A cada cliente é dado, à entrada, um cartão com um número. Os cartões estão previamente numerados e existe uma lista de produtos (código do produto, descrição e preço) também previamente definida. De cada vez que o cliente consome algum produto é registado no cartão o código do produto assim como a quantidade. É necessário verificar se o código do produto introduzido é valido. Se o cliente exceder os 10 consumos terá que liquidar a conta e

pedir novo cartão. Quando um cliente sai, o programa deverá calcular o preço a pagar pelo cliente e registar esse cartão como pago (deverá apresentar a relação dos consumos com totais parciais e totais).

a. Defina as estruturas necessárias para a resolução do problema.
b. Crie uma função que numere os cartões a partir de um número dado pelo utilizador, supondo que serão necessários, no máximo 600.
c. Crie uma função que inicialize a lista de produtos, supondo que existem 10 produtos.
d. Crie uma função para registar a entrada dos clientes.
e. Crie uma função para inserir consumos.
f. Crie uma função para calcular a despesa de um cliente.
g. Crie uma função para indicar o número de clientes na discoteca.
h. Crie uma função que permita listar os produtos.

Nota: se entender necessário pode criar funções auxiliares, além das pedidas.

Exercício 31

Dado número elevado de alunos inscritos na disciplina de Algoritmia e Programação (APRO) foi pedido ao programador Bill Gaitas que desenvolvesse uma aplicação para gerir a disciplina. Por motivos ainda desconhecidos o programador em causa foi raptado. Partindo do que já está desenvolvido termine a aplicação (Figura 1).

Pressupostos:

- Os alunos são identificados pelo número.
- Um aluno só pode estar inscrito a uma turma.
- Não podem existir turmas com o mesmo número.
- A avaliação consiste em duas componentes, componente Frequência e a Prova escrita. As notas da Frequência e da Prova escrita tem um peso de 50% cada na nota final. A nota da frequência é determinada em função das classificações dos 4 Mini-testes (cada com um peso de 10%) e de um trabalho (peso de 60%). Além disso e pela ordem que se apresenta, os alunos que:
  i. Faltem a mais de 30% das aulas efectivamente dadas são classificados com “NF” e reprovam.
  ii. Os alunos com nota inferior a 8 valores na nota da Frequência ou da Prova Escrita são classificados com “NC” e reprovam.
  iii. Os alunos com nota superior ou igual a 9.5 são classificados como aprovados outros como reprovados.
- O valor de qualquer classificação está compreendido entre 0 e 20 valores.

Desenvolva:

- Relativamente as turmas:
  - Uma função que permita inserir uma turma. Preferencialmente inserção ordenada.
  - Uma função que liste todas as turmas. Listar só a informação referente à turma (número, professor, aulas dadas e numero de alunos).

- Uma função que liste uma dada turma. Informação da turma assim como dos alunos.
- Uma função que permita inserir as aulas efectivamente dadas aquela turma.

- Relativamente aos alunos.
  - Uma função que permita inserir um aluno. Preferencialmente inserção ordenada.
  - Uma função que mostre os dados de um aluno (número, nome, faltas, mini-testes, trabalho e exame)
  - Uma função que permita inserir as faltas e as classificações.
  - Uma função que elimine um aluno.
  - Uma função que calcule a classificação final de um aluno.

```
#include <iostream>
#include <iomanip>
#include <ctype.h>
#include <time.h>

using namespace std;

#define NOME_TAM            40
#define NUM_TURMA_TAM   4
#define NUM_ALUNOS          20
#define NUM_MINI_TESTES  4
#define NUM_TURMAS          10
#define IGNORE                  10
struct FREQ{
        int miniteste[NUM_MINI_TESTES];//posicao 0 para o 1º MT, posicao 1 para o
2ºMT ...
        int trabalho;
};
struct ALUNO{
        long numero;
        char nome[NOME_TAM];
        FREQ frequencia;
        int exame;
        int faltas;
};
struct TURMA{
        char numero[NUM_TURMA_TAM];
        char professor[NOME_TAM];
        int num_alunos;
        ALUNO alunos[NUM_ALUNOS];
        int aulas_dadas;
};
int menuAlunos()
{
        int opcao;
        do {
                cout<<"\n\nALUNOS"<<endl;
                cout<<" 1 - Inserir Aluno\n";
                cout<<" 2 - Listar Aluno\n";
                cout<<" 3 - Inserir Faltas-Classificacoes\n";
                cout<<" 4 - Eliminar Aluno\n";
                cout<<" 5 - Calcular Classificacao\n";
                cout<<" 0 - Voltar \n";
                cout<<" Opcao --> ";
```

```
                cin>>opcao;
        }while(opcao<0 || opcao>5);
        cin.ignore(IGNORE,'\n');
        return opcao;
}
int menuTurmas()
{
        int opcao;
        do {
                cout<<"\n\nTURMAS"<<endl;
                cout<<" 1 - Inserir Turma\n";
                cout<<" 2 - Listar Turmas(Todas)\n";
                cout<<" 3 - Listar Turma(Uma)\n";
                cout<<" 4 - Inserir Aulas Dadas\n";
                cout<<" 0 - Voltar \n";
                cout<<" Opcao --> ";
                cin>>opcao;
        }while(opcao<0 || opcao>4);
        cin.ignore(IGNORE,'\n');
        return opcao;
}
int menu()
{
        int opcao;
        do {
                cout<<"\n\nALGORITMIA E PROGRAMACAO"<<endl;
                cout<<" 1 - Turmas \n";
                cout<<" 2 - Alunos \n";
                cout<<" 0 - Sair \n";
                cout<<" Opcao --> ";
                cin>>opcao;
        }while(opcao<0 || opcao>2);
        cin.ignore(IGNORE,'\n');
        return opcao;
}
void manutencaoAlunos(TURMA t[], int &n)
{
        int opcao;
        do {
                opcao=menuAlunos();
                switch(opcao) {
                        case 1:
                        break;
                        case 2:
                        break;
                        case 3:
                        break;
                        case 4:
                        break;
                        case 5:
                        break;
                }
        }while(opcao);
}
void manutencaoTurma(TURMA t[], int &n)
{
        int opcao;
        do {
                opcao=menuTurmas();
                switch(opcao) {
```

```
                        case 1:
                        break;
                        case 2:
                        break;
                        case 3:
                        break;
                        case 4:
                        break;
                }
        }while(opcao);
}
void main()
{
        TURMA turmas[NUM_TURMAS];
        int num_turmas=0;
        int opcao;
        do {
                opcao=menu();
                switch(opcao) {
                        case 1: manutencaoTurma(turmas, num_turmas);
                        break;
                        case 2: manutencaoAlunos(turmas, num_turmas);
                        break;
                }
        }while(opcao);
}
```

Figura 1 – Parte desenvolvida

7. MATRIZES

Objectivos

Identificar situações em que é vantajosa a utilização arrays bidimensionais.
Declaração de matrizes.
Ler, armazenar e referenciar dados numa matriz.

Exercício 32

Escreva um programa que defina uma matriz quadrada de dimensão máxima 4 e implemente as seguintes funções:

- Uma função que faça a leitura dos elementos da matriz, guardando os valores em memória RAM;
- Uma função que devolva a média da diagonal principal;
- Uma função que devolva um vector com os elementos cujo valor seja superior à média da diagonal principal;
- Uma função que devolva uma matriz com o número de ocorrências de cada elemento.

Exemplo:

matriz de entrada:

1	2	2
5	1	2
8	5	5

resultado:

1	2
2	3
5	3
8	1

Exercício 33

Escreva um programa que dada uma matriz quadrada de dimensão *n* determine se ela é ou não simétrica. Uma matriz *A* diz-se simétrica se $a_{ij} = a_{ji}$ com $1 \leq i, j \leq n$.

Exercício 34

Um elemento de uma matriz é considerado um máximo local se for superior a todos os seus vizinhos. Escreva um programa que dada uma matriz forneça todos os máximos locais e as respectivas posições, considerando que os elementos da periferia da matriz não podem ser máximos locais.

Exercício 35

Escreva um programa que permita:

- Ler um vector *Alunos* cujo elemento $alunos_{(i)}$ possui o número de matricula do aluno *i*, guardando os valores em memória RAM; considere que existem 10 alunos;
- Ler uma matriz *Notas* cujo elemento $notas_{(i,j)}$ contém a nota do estudante *i* no exercício *j*, guardando os valores em memória RAM; considere a existência de 4 exercícios;
- Ler um vector *Pesos* cujo elemento $pesos_{(j)}$ indica o peso do exercício *j*, guardando os valores em memória RAM;
- Produzir um novo vector, *Classif*, onde o elemento $classif_{(i)}$ possuirá a nota obtida pelo aluno *i* no exame. A nota do aluno no exame é calculada com base nas notas obtidas em cada exercício e respectivos pesos;
- Emitir a pauta final do exame ordenada, de forma descendente, por nota.

8. MANIPULAÇÃO DE STRINGS

Objectivos
Declaração de variáveis strings.
Inicializar e manipular strings.
Utilizar funções de manipulação de strings.

Exercício 36

Desenvolva as seguintes funções:

a) Uma função que recebe uma string e retorne o seu comprimento.
b) Uma função que copie o conteúdo de uma string para a outra. Portanto a função recebe duas strings, uma designada de origem e a outra de destino. O objectivo é copiar o conteúdo da de origem para a de destino. Note que as strings podem ter tamanhos diferentes. Daí que, a função recebe as duas string e um número com o comprimento da string de destino (comp_strDest). O objectivo da função é que copiar da string de origem para a string de destino no máximo comp_strDest - 1 caracteres. A função deve retornar o número de caracteres efectivamente copiados.
c) Uma função que receba duas strings e retorne: 1 se as strings forem iguais; e 0 se forem diferentes.
d) Uma função que receba duas strings e retorne o número de ocorrências de uma na outra. Por exemplo, a string "DEIDE DEDEI DEEI" tem duas ocorrências da string "DEI".
e) Uma função que recebe duas strings e verifica se uma é inversa de outra. Caso seja deve retornar 1 e 0 caso não seja. Note que uma string vazia não tem inversa.

Exercício 37

Manipulação de strings. Desenvolva as seguintes funções:

a. Uma função que receba um carácter e caso seja maiúsculo retorne o correspondente minúsculo.
b. Uma função que receba um carácter e caso seja minúsculo retorne o correspondente maiúsculo.
c. Uma função que elimine todos os espaços à esquerda do primeiro carácter (diferente de espaço).
d. Uma função que elimine todos os espaços á direita do último carácter (diferente de espaço).
e. Uma função que elimine todos os espaços múltiplos entre as palavras ou letras.
f. Faça uma função que receba uma string e retorne o número de palavras. Entendendo-se por palavra cadeias de caracteres terminadas pelo carácter espaço. As palavras têm que ter mais do que um carácter.
g. Faça uma função que receba uma string e a formate da seguinte forma:

i. As palavras começam sempre com letras maiúscula e o resto com letra minúscula.
ii. As palavras têm que ter mais do que um carácter.
iii. As letras (caracteres isolados em letra minúscula)