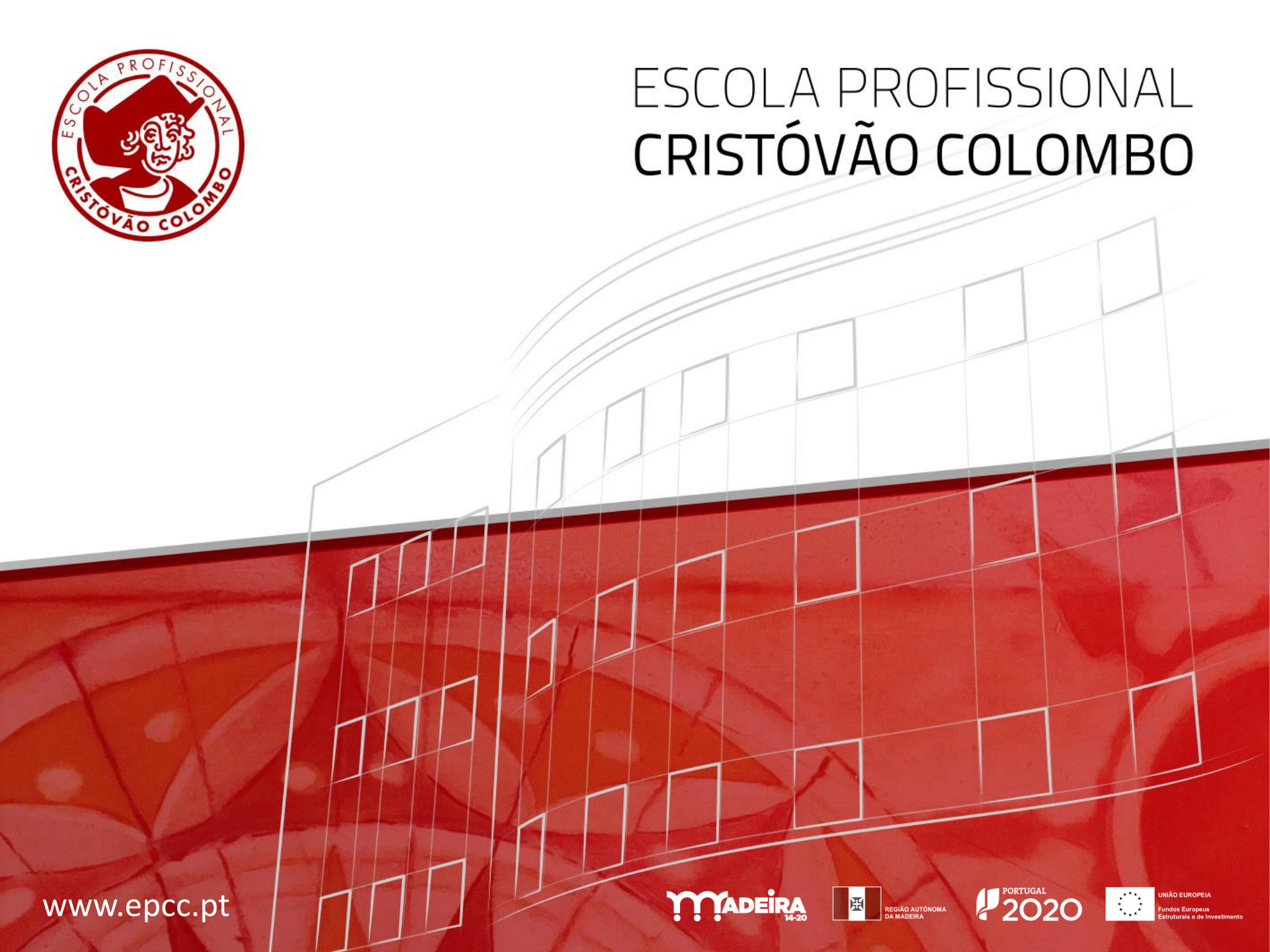
ESCOLA PROFISSIONAL
CRISTÓVÃO COLOMBO



www.epcc.pt

MADEIRA 14-20 · REGIÃO AUTÓNOMA DA MADEIRA · PORTUGAL 2020 · UNIÃO EUROPEIA · Fundos Europeus Estruturais e de Investimento

Sistemas Operativos – Processamento Computacional

Funções

Autor da apresentação

09/09/2014

Funções

Conjunto de comandos agrupados num **bloco**, que recebe um **nome** é através deste que pode ser **evocado**.

Porque usar:

- Reaproveitamento de código
- Evitar trecho de código que seja repetido várias vezes
- Facilitar a leitura do programa-fonte
- Separar o programa em partes(blocos)

Funções - Sintaxe

```
tipo_da_função nomedaFunção(parametros){
    //corpo da função
}
```

parametros - opcional a lista de argumentos

Funções sem parâmetros e sem retorno - *Exemplo*

```c
#include <stdio.h>
// Definição da função "EsperaEspaço"
void esperaEspaco(){
    int tecla;
    printf("Pressione ENTER\n");
    do{
        tecla = getchar();
        // Se nao for barra de espaço
        if (tecla !=32) {
            printf("Digite Barra e espaço pra sair\n");
        }
    } while(tecla != 32); // 32 e' o codigo ASCII da barra de espaço
}

int main(){
    esperaEspaco();      // Chamada da função definida antes
}
```

Funções com parâmetros

Definir os parâmetros de uma função devemos explicitá-los como se estivéssemos a declarar uma variável, entre os parênteses do cabeçalho da função. Caso precise declarar mais de um parâmetro, basta separá-los por vírgulas.

Funções com parâmetros - Exemplo

```c
#include <stdio.h>
#include <locale.h>

void soma(float x, float y){
    float result;
    result = x + y;
    printf("A soma é :%.0f",result);
}

int main(){
    setlocale(LC_ALL,"Portuguese");
    soma(2,8);
}
```

Funções com parâmetros – Exemplo 2

```c
#include <stdio.h>
#include <locale.h>

void soma(float x, int y){
    float result;
    result = x + y;
    //Arredonda para a 0 casas decimais
    //printf("A soma de %0.2f + %d é = %.0f",x,y,result);
    printf("A soma de %0.2f + %d é = %.2f",x,y,result);
}

int main(){
    setlocale(LC_ALL,"Portuguese");

    int a = 10;
    float b = 25.6;
    soma(b,a);
}
```

Funções onde criar?

Toda função deve ser *declarada* antes de ser usada.
A linguagem C permite que se **declare** uma função, antes de defini-la. Esta declaração é feita através do **protótipo** da função. O protótipo da função, nada mais é do que o trecho de código que especifica o nome e os parâmetros da função.

Funções prototipadas

Uma função é prototipada antes de ser usada e assim pode ser chamada antes de ser definida.

```c
#include <stdio.h>
#include <locale.h>

void soma(float x, int y); // Protótipo da função
int main(){
    soma(14.5,5); // Chamada da função antes de ser definida.
}

void soma(float x, int y){
    float result;
    result = x + y;
    printf("A soma de %0.2f + %d = %.2f",x,y,result);
}
```

Funções - Verificação de parâmetros

```c
#include <stdio.h>
#include <locale.h>
void soma(float x, int y); // Protótipo da função

int main(){
    float n = 12.8;
    /* A função espera um float, int
       no entanto passo um int,float
       Será feito cast de int para float no primeiro parametro
       No segundo parametro de float para int.
       soma((float)14,(int)n);
       vão entrar os valores:
       soma(14.0,12);
    */
    soma(14,n); // Chamada da função antes de ser definida.
}

void soma(float x, int y){
    printf("A soma de %0.2f + %d = %.2f",x,y,x + y);
}
```

Funções Parâmetros por referência

A passagem de parâmetros até ao momento era chamada de **passagem por valor**. Desta forma, a chamada da função passa o **valor** do parâmetro para a função. Não havendo alterações do parâmetro dentro da função, **não afectará** a variável usada na chamada da função.

Funções Parâmetros por referência - Ex

```c
#include<stdio.h>
#include<stdlib.h>

/* Não existe alteração de valor */
void limpa(float a){
    a=0;
}
int main(){
    float f;
    f = 123.6;
    limpa(f);
    printf("%f",f);
}
```

Funções Parâmetros por referência

Aspectos a reter na passagem por referencia:

- Na chamada da função o operador & deve preceder a variável;
- No parâmetro da função deve-se indicar o ponteiro(*).
- No corpo da função usa-se o operador de referencia(*)

Funções Parâmetros por referência - Ex

```c
#include<stdio.h>
#include<stdlib.h>

//Definir que o parâmetro é a referencia
//variavel que foi passada
void limpa(float *a){
    //operador de referência para aceder à variavel
    *a=0;
}
int main(){
    float f;
    f = 123.6;
    // Passar o endereço da variavel f para a função limpa
    limpa(&f);
    printf("%f",f);
}
```

Funções com Parâmetros array

A passagem de vectores por parâmetro é **sempre por referência.**
Isto significa que não se deve, na chamada da função, passar o endereço do vector. Isto ocorre porque, por convenção, o nome do vector já representa o endereço inicial deste vector na memória.

Funções com Parâmetros array - Exemplo

```c
#include<stdio.h>
#include<stdlib.h>

void limpa(float v[], int qt){
    int i;
    for(i=0 ; i < qt ; i++)
        v[i]=1.0;
}
int main(){
    int i;
    float vetor[10];
    limpa(vetor,10);
    for(i=0 ; i < 10 ; i++)
        printf("%f\n",vetor[i]);
}
```

Funções com Parâmetros array - Exemplo

```c
#include<stdio.h>
#include<stdlib.h>

void limpa(float *v, int qt){
    int i;
    for(i=0 ; i < qt ; i++)
        v[i]=1.0;
}
int main(){
    int i;
    float vetor[10];
    limpa(vetor,10);
    for(i=0 ; i < 10 ; i++)
        printf("%f\n",vetor[i]);
}
```

Declarar o parâmetro como um ponteiro para o tipo de dado que forma o vector

Funções com retorno

Por definição, toda função em C retorna algo, algum valor, variável. Pode retornar um inteiro, um float, um caractere, um vector, struct ou outro tipo criado por nós.
O *void* indica que tais funções não retornavam nada.

Funções com retorno – Exemplo Somar

```c
#include<stdio.h>
#include<stdlib.h>

int soma(int a, int b){
    return a+b;
}
int main(){
    int n1,n2,r;
    printf("Insira dois valores inteiros :\n",r);
    scanf("%d %d", &n1,&n2);

    r = soma(n1,n2);
    printf("Resultado : %d\n",r);
}
```

Funções com retorno – Exemplo Multiplicar

```c
#include<stdio.h>
#include<stdlib.h>

int mult(float a, float b){
    return (a*b);
}
int main(){
    float n1,n2,r;
    printf("Insira dois valores a Multiplicar :\n");
    scanf("%f %f", &n1,&n2);
    fflush(stdin);
    r = mult(n1,n2);
    printf("Resultado : %f\n",r);
}
```

Função – getchar()

A função getchar() lê um carácter e retorna um inteiro que é: o código do carácter, ou o valor -1 que corresponde a *fim de ficheiro.*

```c
#include<stdio.h>
#include<stdlib.h>

main(){
int c;
c=getchar();     // Lê o primeiro caracter
printf("O caracter foi:%d \n",c); // Imprime o inteiro
}
```

Função – getchar() – Exemplo 2

```
/* Contar a ocorrência de leituras das:
letra a ou A
letra z ou Z
letra B   */

main(){
int c;
int naA = 0, nzZ=0, nB=0, nOutros =0;

while((c = getchar()) != '.'){
    switch(c){
        case 'a':
        case 'A': ++naA;break;
        case 'z':
        case 'Z': ++nzZ;break;
        case 'B': ++nB;break;
        default: ++nOutros;
    }
}
/* Introduza exemplo : sadsadmszzb.
Inserir um carater por linha contabilizará o \n(ao pressionao o enter)
*/
printf("\n A\t Z\t B\t O\n",c);
printf("\n %d\t %d\t %d \t%d\n",naA,nzZ,nB,nOutros); // Imprime o inteiro
}
```