Funções na Linguagem C

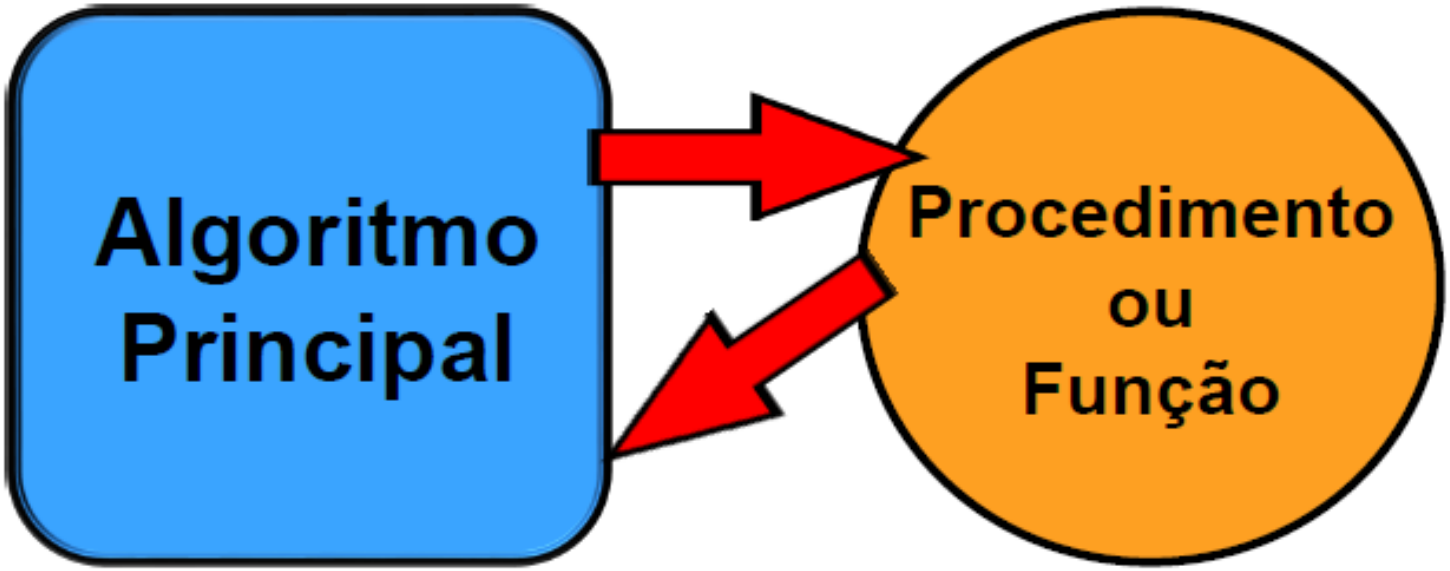
Profa. Dra. Giani Carla Ito

Disciplina: Linguagem de Programação Estruturada

Funcionamento

- Um algoritmo completo é dividido em algoritmo principal e a diversos subalgoritmos.
  - **Algoritmo principal:** é aquele por onde a execução do algoritmo sempre se inicia. Este pode invocar subalgoritmos;
  - **Subalgoritmos:** são executados quando "chamados" pelo algoritmo principal. Ao terminar sua execução devolvem o controle ao algoritmo principal;

Funcionamento



Funções:

- São trechos de código fora do programa principal
- Implementam um subalgoritmo do programa
- São utilizadas (chamadas) em diversos pontos do programa

Funções

Vantagens:

- Organiza o código:
  - Cada função é um algoritmo
  - Dividir um programa complicado em várias funções simples
- Evita repetição de código semelhante
  - Escrever o código apenas uma vez
  - Usar a mesma função várias vezes para variáveis diferentes
- Simplifica e agiliza a programação

Funções

Exemplo: comparar a média das duas melhores notas do aluno A e do aluno B

Algoritmo:

- Ler 3 notas dos alunos A e B
- Calcular a média de A
- Calcular a média de B
- Comparar as médias
- Imprimir resultados

Funções

Alguns problemas no exemplo ...

- Ler 3 notas dos alunos A e B
- Calcular a média de A
- Calcular a média de B
- Comparar as médias
- Imprimir resultados

Repetição do mesmo algoritmo

Solução: Uma função genérica para calcular a média

Sintaxe da Função

Tipo de retorno **nomefuncao** **(tipo1 parametro1, tipo2 parametro2){**

variáveis a serem usadas na função;

comandos;

return (***valor a ser retornado***);

}

Onde:

- **Tipo de retorno:** indica qual e o tipo do valor a ser retornado pela função;
- **Parâmetros** : variáveis que serão recebidas para serem processadas na função (locais);

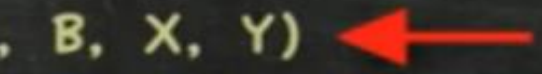
Escopo de Variáveis

- Pode ser de dois tipos: **Global e Local**;

- **Variáveis Globais:** Declaradas externamente a todas as funções e podem ser lidas e escritas por todas as funções do programa;

- **Variáveis Locais:** Variáveis declaradas internamente as funções e podem ser lidas e escritas somente enquanto a função estiver sendo executada.

```
Algoritmo "testeEscopo"
var
   N1, N2: Inteiro
Procedimento Rotina(A, B: Inteiro)
var
    X, Y: Inteiro
Inicio
    X <- A
    Y <- B
    Escreva(N1, N2, A, B, X, Y)
FimProcedimento
Inicio
    N1 <- 5
    N2 <- 3
    Rotina(N1, N2)
    Escreva(N1, N2, A, B, X, Y)
FimAlgoritmo
```



LOCAL

GLOBAL

Exemplo 1

- Escrever um programa que contenha uma função que recebe dois valores e retorna a soma dos dois. O programa principal deverá exibir o resultado.

Exemplo 1

```
#include<stdio.h>
int soma (int x, int y);
void main(){
    int a,b,s;
    printf("Digite dois valores :");
    scanf("%d%d",&a,&b);
    s= soma(a,b);
    printf("\nO valor da soma e: %d.",s);
}
int soma (int x, int y){
    int r;
    r=x+y;
    return(r);
}
```

O valor da soma e: 40

Exemplos

- Faça um algoritmo que recebe uma idade e envia para uma função que retorna se um individuo atingiu a maioridade ou não. O resultado deve ser impresso no programa principal

- Faça um algoritmo que leia duas notas de um aluno e envia para uma função que retorna o valor da média aritmética. Imprimir na tela o resultado.

Passagem de parâmetros

- Consiste na transferência de informações para as funções;
- A passagem de parâmetros pode ser de duas formas: **Por cópia** ou **Por referência;**

- **Por cópia:** Quanto é feita uma "cópia" do valor das variáveis utilizadas para chamar a função nos parâmetros. Alterações feitas localmente às funções não alteram o valor das variáveis no programa principal.

Exemplo – Passagem de parâmetro por cópia

```c
#include<stdio.h>
int soma (int x, int y);
void main(){
    int a,b;
    printf("Digite dois valores :");
    scanf("%d%d",&a,&b);
    printf("\nAntes da funcao %d %d",a, b);
    printf("\n***O valor da soma e: %d.***",soma(a,b));
    printf("\nDepois da funcao %d %d",a, b);
}
int soma (int x, int y){
    int r;
    r=x+y;
    x=20;
    y=25;
    printf("\nDentro da funcao %d %d",x, y);
    return(r);
}
```

Passagem de Parâmetros

- **Por referência:** Se dá a partir do uso de ponteiros.
- Um **ponteiro** é uma variável utilizada para receber um endereço de memória, ou seja, aponta para um local da memória.
- Assim, as variáveis de parâmetro recebem o endereço de memória das variáveis do programa principal.
- Desta forma, as alterações realizadas nas funções são executadas diretamente nas variáveis originais, ou seja, no endereço de memória.

Ponteiros

Endereço na memória	
x	1000
y	1001
z	1002

Variável na memória
1002

Ponteiros

Os operadores de ponteiro são:

& - o endereço de: que devolve um endereço da memória do seu operando ou seja a variável que o segue ;
* - no endereço : complemento de &, que devolve o valor da variável localizada no endereço que o segue.

Exemplo: cont tem endereço 2000 e valor 100 ...

m=&cont *(m recebeu o endereço de cont=2000)*;
q=*m *(que recebeu o valor de que está no endereço m= 100)*
Pergunta : Quem é o ponteiro?

Passagem de Parâmetro por referência

- **Ponteiros** podem ser de qualquer tipo primitivo;
- Devem ser criados a partir do uso do símbolo *;

- **Sintaxe:**

  tipo *nomeponteiro;

- **Exemplo:**

```
int *p_num;
char *p_nome;
...
```

Exemplo 2

```c
#include<stdio.h>
int soma (int *x, int *y);
void main(){
    int a,b;
    printf("Digite dois valores :");
    scanf("%d%d",&a,&b);
    printf("\nAntes da funcao %d %d",a, b);
    printf("\n***O valor da soma e: %d.***",soma(&a,&b));
    printf("\nDepois da funcao %d %d",a, b);
}
int soma (int *x, int *y){
    int r;
    r=*x+*y;
    *x=20;
    *y=25;
    printf("\nDentro da funcao %d %d",*x, *y);
    return(r);
}
```

Passando vetores/Matrizes

Passando vetores para funções: Na programação não podemos passar uma matriz inteira para uma função, mas sim um ponteiro de uma matriz para uma função especificando o nome do vetor sem um índice.

Exemplo:

...

Int i[10];

func1(i);

....

Se uma função recebe um vetor, para declarar o parâmetro formal podemos escolher quaisquer das três formas abaixo:

Void func1(int*x)// como ponteiro;

Void func1(int x[10])// vetor dimensionado;

Void func1(int x[])//vetor não dimensionado;

Passando Structs para uma função

```c
#include <stdio.h>

typedef struct carro
{
        char modelo[30];
        float potenciaMotor;
        int anoFabricacao,
            numPortas;
} CARRO;

void Exibe(CARRO car)
{
    printf("Modelo: %s\n", car.modelo);
    printf("Motor: %.1f\n", car.potenciaMotor);
    printf("Ano: %d\n", car.anoFabricacao);
    printf("%d portas\n", car.numPortas);
}
void mostra (int vet[]){
int i;
for(i=0;i<10;i++) printf("%d",vet[i]);
}
int main(void)
{
    int i[10]={1,2,3,4,5,6,7,8,9,0};

    CARRO fusca = {"Fuscao preto", 1.5, 74, 2};
    Exibe(fusca);
    mostra(i);

    return 0;
}
```