INSTITUTO FEDERAL
PARANÁ

LÓGICA DE PROGRAMAÇÃO

PROFª. M.Sc. JULIANA H Q BENACCHIO

Funções

INSTITUTO FEDERAL
PARANÁ

- Uma função é um bloco de código de programa que pode ser usado diversas vezes em sua execução
- O uso de funções permite que o programa fique mais legível e mais estruturado

Funções

INSTITUTO FEDERAL
PARANÁ

- Uma função pode retornar um valor e receber valores como parâmetros (lista de argumentos)

```
tipo_de_retorno nome_da_função (lista_de_argumentos)
{
    código_da_função
}
```

Tipo `void`

INSTITUTO FEDERAL
PARANÁ

- Uma função pode não ter parâmetros, neste caso a lista de argumentos é vazia. No entanto, os parênteses ainda são necessários.
- Tipo vazio → sem retorno

```
void mensagem(){
  printf("Ola! \n");
}
```

Funções simples

INSTITUTO FEDERAL
PARANÁ

```
void mensagem()    //sem retorno e sem parametro
{
  printf("Ola! \n");
}

int main()
{
  mensagem();  //chamada da funcao
  printf("Ola de novo!\n");
  return 0;
}
```

Regras de Escopo de Funções

INSTITUTO FEDERAL
PARANÁ

- São regras que governam se uma porção de código conhece ou tem acesso a outra porção de código ou dados.
- Em C, cada função é um bloco discreto de código, ou seja, um código de uma função é privativo àquela função.
- Variáveis que são definidas internamente a uma função são chamadas **variáveis locais**.

Regras de Escopo de Funções

INSTITUTO FEDERAL
PARANÁ

- O código e os dados que são definidos internamente a uma função não podem interagir com o código ou dados definidos em outra função porque as duas funções têm escopos diferentes.
- Uma variável local vem a existir quando ocorre a entrada da função e ela é destruída ao sair.
- Ou seja, variáveis locais não podem manter seus valores entre chamadas a funções.

Variáveis Locais

INSTITUTO FEDERAL
PARANÁ

```
void soma(){
   int a, b, res_soma; //variaveis locais
   printf("Digite dois numeros: ");
   scanf("%d %d", &a, &b);
   res_soma = a + b;
   printf("%d", res_soma);
}

int main (){
   soma(); //chamada da funcao
   return 0;
}
```

Regras de Escopo de Funções

INSTITUTO FEDERAL
PARANÁ

- O código que constitui o corpo de uma função é escondido do resto do programa e, a menos que use variáveis ou dados globais, não pode afetar ou ser afetado por outras partes do programa.
- Ao contrário das variáveis locais, as **variáveis globais** são reconhecidas pelo programa inteiro e podem ser usadas por qualquer pedaço de código.

Regras de Escopo de Funções

INSTITUTO FEDERAL
PARANÁ

- Além disso, elas guardam seus valores durante toda a execução do programa.
- As variáveis globais são criadas declarando-as fora de qualquer função.
- Elas podem ser acessadas por qualquer expressão independentemente de qual bloco de código contém a expressão.

Variáveis Globais

INSTITUTO FEDERAL
PARANÁ

```
int a, b, res_soma; //variaveis globais

void soma(){
  printf(“Digite dois numeros: ");
  scanf("%d %d", &a, &b);
  res_soma = a + b;
  printf("%d", res_soma);
}

int main (){
  soma();
  return 0;
}
```

Variáveis Globais

INSTITUTO FEDERAL
PARANÁ

```
int a, b; //variaveis globais

void soma(){
  int res_soma; //variaveis locais
  res_soma = a + b;
  printf("%d", res_soma);
}

int main (){
  printf(“Digite dois numeros: ");
  scanf("%d %d", &a, &b);
  soma();
  return 0;
}
```

Variáveis Globais

INSTITUTO FEDERAL
PARANÁ

- O armazenamento de variáveis globais encontra-se em uma região fixa da memória, separada para esse propósito pelo compilador C.
- Variáveis globais são úteis quando o mesmo dado é usado em muitas funções em seu programa.
- No entanto, você deve evitar usar variáveis globais desnecessárias.

Variáveis Globais

INSTITUTO FEDERAL
PARANÁ

- Elas ocupam memória durante todo o tempo em que seu programa está executando, não apenas quando são necessárias.
- Além disso, usar uma variável global onde uma variável local poderia ser usada torna uma função menos geral, porque ela conta com alguma coisa que deve ser definida fora dela.

Variáveis Globais

INSTITUTO FEDERAL
PARANÁ

- Uma das principais razões para uma linguagem estruturada é a compartimentalização ou separação de código e dados.
- Em C, esse isolamento é conseguido pelo uso de variáveis locais e funções.

Funções com retorno de valor

INSTITUTO FEDERAL
PARANÁ

- Comando **`return`**

```
return valor_de_retorno;
```

- Quando se chega a uma declaração **`return`** a função é encerrada imediatamente
- O valor de retorno tem que ser compatível com o tipo de retorno declarado

Funções com retorno de valor

INSTITUTO FEDERAL
PARANÁ

```
int a, b; //variaveis globais

int soma(){
  int res_soma; //variaveis locais
  res_soma = a + b;
  return res_soma; //retorno
}

int main (){
  int resultado; //variaveis locais
  printf(“Digite dois numeros: ");
  scanf("%d %d", &a, &b);
  resultado = soma(); //retorno da funcao para variavel
  printf("%d", resultado);
  return 0;
}
```

Funções com retorno de valor

INSTITUTO FEDERAL
PARANÁ

```
int a, b; //variaveis globais

int soma(){
  int res_soma; //variaveis locais
  res_soma = a + b;
  return res_soma; //retorno
}

int main (){
  printf("Digite dois numeros: ");
  scanf("%d %d", &a, &b);
  printf("%d", soma());  //retorno da funcao direto para escrita
  return 0;
}
```

Funções com retorno de valor

INSTITUTO FEDERAL
PARANÁ

```
int a, b; //variaveis globais

int soma(){
  int res_soma; //variaveis locais
  res_soma = a + b;
  return res_soma; //retorno
}

int main (){
  printf("Digite dois numeros: ");
  scanf("%d %d", &a, &b);
  printf("%d", soma());  //retorno da funcao direto para escrita
  return 0;
}
```

Evitar o uso de Variáveis Globais

Mas como ??

Passagem de Parâmetros

INSTITUTO FEDERAL
PARANÁ

```
tipo_de_retorno nome_da_função (lista_de_argumentos)
{
   código_da_função
}
```

- A declaração de parâmetros é uma lista:

```
tipo nome1, tipo nome2, ... , tipo nomeN
```

Passagem de Parâmetros

INSTITUTO FEDERAL
PARANÁ

- Os argumentos da função são variáveis, chamadas de **parâmetros formais da função**.
- Elas se comportam como quaisquer outras variáveis locais dentro da função e são criadas na entrada e destruídas na saída.
- É preciso assegurar-se de que os argumentos usados para chamar a função sejam compatíveis com o tipo de seus parâmetros.

Passagem de Parâmetros

INSTITUTO FEDERAL
PARANÁ

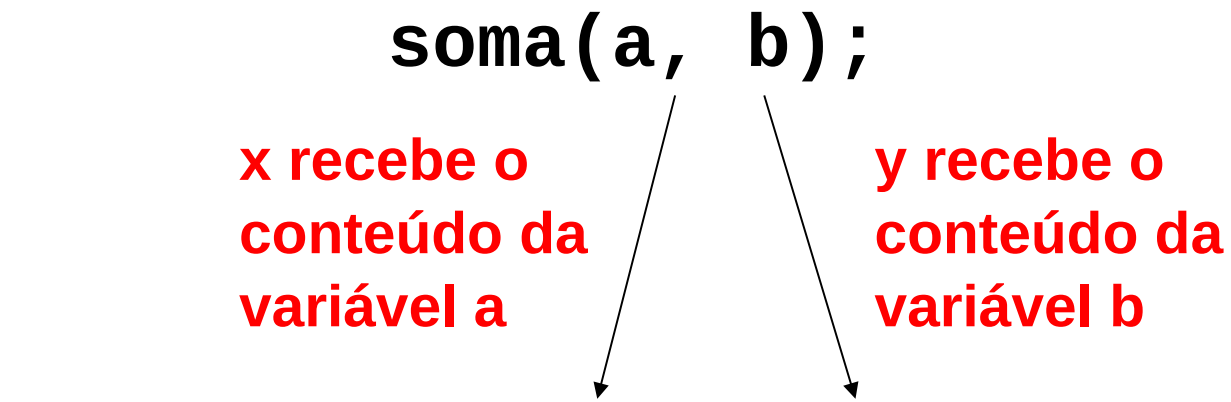
```
int soma(int x, int y){ //parametros formais
  int res_soma; //variaveis locais
  res_soma = x + y;
  return res_soma;
}

int main (){
  int a, b, resultado; //variaveis locais
  printf("Digite dois numeros: ");
  scanf("%d %d", &a, &b);
  resultado = soma(a, b); //chamada da funcao por valor
  printf("%d", resultado);
  return 0;
}
```

Passagem de Parâmetros

INSTITUTO FEDERAL
PARANÁ

- A passagem de parâmetros por valor copia o valor de um argumento no parâmetro formal da função.
- Assim, alterações feitas nos parâmetros da função não têm nenhum efeito nas variáveis usadas para chamá-la.

Passagem de Parâmetros




```
int soma(int x, int y){
    res_soma = x + y;
    return res_soma;
 }
```

➡ Passagem de parâmetros por valor

Passagem de Parâmetros

INSTITUTO FEDERAL
PARANÁ

```
#include <stdio.h>

int soma(int x, int y){
  return x + y;
}

int main (){
  int a, b;
  printf(“Digite dois numeros: ");
  scanf("%d %d", &a, &b);
  printf("%d", soma(a, b));
  return 0;
}
```