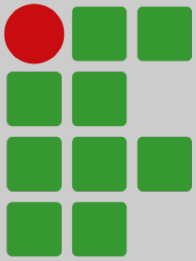


INSTITUTO FEDERAL

Campus Charqueadas

Programação

Funções e Procedimentos

Prof. Silvana Teodoro

Função X Procedimento

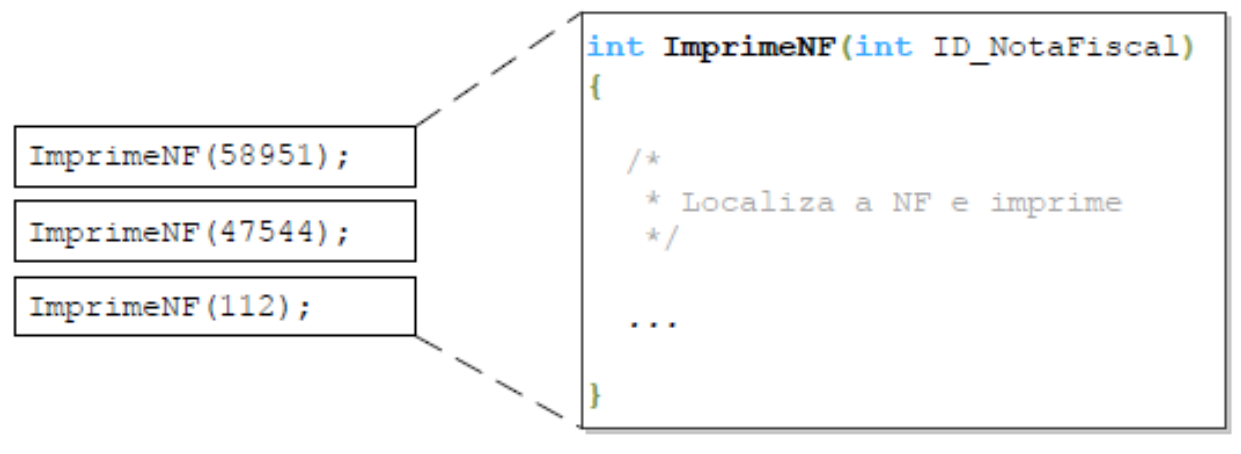
Função x Procedimento

- **Função:** Toda função que é criada, por obrigação tem que ter um retorno.
  - Utilizamos funções geralmente quando precisamos calcular algo, retornar algum valor verdadeiro, ou retornar algum teste.

- **Procedimento:** É quase igual a uma função, mas não precisa necessariamente ter retorno algum.
  - Geralmente utilizamos procedimentos quando precisamos utilizar alguns códigos que vamos ter que ficar repetindo, sendo assim, uma *procedure* concentra esses códigos e podemos a qualquer momento chamá-los.

Linguagem C
Funções

- **Implementação de funções**
  - Conjunto de operações a serem executados no programa
  - Reuso de código
  - Simplificação de programação
  - Parametrizável



Linguagem C
Funções

- **Funções**
  - Bloco de código que executa um conjunto de instruções
  - Pode ser chamado em qualquer parte do programa
  - Pode ser parametrizado
  - Pode ser função de retorno de valores, ou apenas processamento

Linguagem C
Funções

- **Definição de funções**
  - Tipo de retorno
    - char
    - int
    - float
    - void …
- **Lista de parâmetros**
  - Quantas forem necessárias
  - Separar por vírgula
  - Retorna valor com *return*

Sintaxe

```
1. tipo_retorno nome_função(lista_parametros) {
2.   // instruções da função
3. }
```

Exemplos

```
1. int soma(int x, int y) {
2.   return (x+y);
3. }
```

```
1. void espera() {
2.   int x;
3.   for(x=0; x<9999; x++);
4. }
```

Linguagem C
Funções

- Uma função pode retornar apenas um valor para o programa que a chamou, e este valor deve ser do **tipo** especificado no cabeçalho da função, é o **tipo da função**.
- O **nome** da função deverá sempre iniciar com uma letra, e normalmente é escrito em minúsculas.
- Os **parâmetros** são valores de variáveis que serão passados pelo programa para a função.
- Além da **variável de retorno** nenhum outro valor pode ser retornado pela função para o programa que a chamou.

Exemplos

```
1. int soma(int x, int y) {
2.    return (x+y);
3. }
```

Linguagem C
Funções

Exemplos

```
1. int soma(int x, int y) {
2.    x += y;
3.    return (x);
4. }

1. void main() {
2.    int x = 10;
3.    int y = 50;
4.    soma(x,y);
5. }
```

Linguagem C
Funções

- **FUNÇÃO SEM RETORNO**

Quando uma função não retorna um valor para a função que a chamou ela é declarada como **void**.

Exemplos

```
1. void espera() {
2.    int x;
3.    for(x=0; x<9999; x++);
4. }
```

Linguagem C
Funções

- **FUNÇÃO COM RETORNO**

Quando o programador quiser que a função envie um valor para o programa que a chamou, ele deverá declarar um **tipo** diferente de **void** para a função.

Exemplos

```
1. int soma(int x, int y) {
2.     return (x+y);
3. }
```

Linguagem C
Funções - Classes de Variáveis

- **Variáveis Locais**
  - As variáveis que são declaradas **dentro** de uma função são chamadas de **locais**.
  - Na realidade todas as variáveis declaradas dentro de um bloco { } podem ser referenciadas apenas dentro deste bloco.
  - Elas existem apenas durante a execução do bloco de código no qual estã declaradas.

Exemplos

```
1. int soma(int x, int y) {
2.     return (x+y);
3. }
```

Linguagem C
Funções - Classes de Variáveis

- **Variáveis Globais**
  - São conhecidas por todo programa e podem ser usadas em qualquer parte do código.
  - Permanecem com seu valor durante toda execução do programa.
  - Deve ser declarada fora de qualquer função e até mesmo antes da declaração da função main.

Exemplos

```
# include <stdio.h>
void func1( ), func2( );
int cont;
main( )
{
    cont = 100;
    func1( );
}

void func1( )
{
    int temp;
    temp = cont;
    func2( );
    printf ("temp é = %d", temp);
    printf ("cont é = %d", cont);
}
void func2( )
{
    int cont;
    for(cont =1; cont<10; cont++) printf(" . ");
}
```

Linguagem C
Funções - Classes de Variáveis

- **Variáveis Estáticas**
  - Funcionam de forma parecida com as variáveis globais, conservando o valor durante a execução de diferentes funções do programa.
  - No entanto só são reconhecidas na função onde estão declaradas.

Exemplos

```
#include <stdio.h>
main( )
{
   int i;
   static int x[10] = {0,1,2,3,4,5,6,7,8,9};
   for(i=0; i<10; i++) printf (" %d \n ", x[i] );
}
```

Linguagem C
Funções

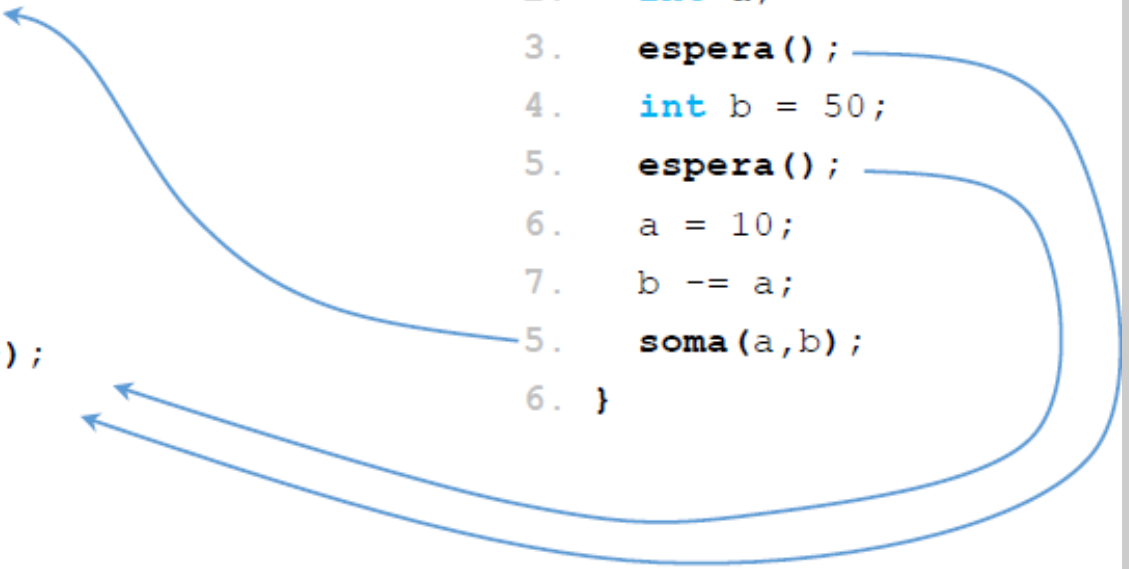
- **Definição de funções**

Exemplo

```
1. int soma(int x, int y) {
2.   return (x+y);
3. }
```

```
1. void espera() {
2.   int x;
3.   for(x=0; x<9999; x++);
4. }
```

```
1. void main() {
2.   int a;
3.   espera();
4.   int b = 50;
5.   espera();
6.   a = 10;
7.   b -= a;
5.   soma(a,b);
6. }
```



Linguagem C
Funções

- **Definição de funções convencionais**
  - Parâmetros passados a uma função não sofrem modificações

Exemplos

```
1. int soma(int x, int y) {
2.   x += y;
3.   return (x);
4. }

1. void main() {
2.   int x = 10;
3.   int y = 50;
4.   soma(x,y);
5. }
```

- Após a chamada da função **soma**, o valor de **x** continua em 10
- Compilador separa um registrador de memória para cada variável local
  - x local da **main**
  - x local da **soma** ≠

Linguagem C
Funções – Passagem de Vetor

- Para passar um **vetor** para uma função é necessário passar somente o **endereço** e não uma cópia do vetor.

Exemplos

```
#include <stdio.h>
void mostra (int num[ ] );
main( )
{
   int t[10], i;
   for ( i = 0; i < 10; i++) t[i] = i;
   mostra(t);
}
void mostra ( int num[ ] )
{
   int i;
   for( i = 0; i < 10; i++ ) printf ("%d", num[i]);
}
```

Exercícios – Parte 1

Exercício

Desenvolver um programa que leia do teclado dois valores. Após desenvolver 4 funções parametrizadas com esses dois valores lidos. Funções:

- Soma
- Subtração
- Divisão
- Multiplicação

Por fim, imprima o valor resultante de cada função em tela.