UNESPAR
Universidade Estadual do Paraná

Algoritmos e Técnicas de Programação

Aula 6 - Subrotina - Parte 1

Prof. Guilherme Guerino

Introdução

- Sub-rotinas também são conhecidas como **subprogramas** e podem ser definidas como **blocos de instruções** que realizam tarefas específicas.

- O código de uma sub-rotina pode ser executado quantas vezes for necessário.

UNESPAR
Universidade Estadual do Paraná

Introdução

- Um programa pode ser dividido em pequenas sub-rotinas, de modo a facilitar a organização.

- Na linguagem C, as sub-rotinas são implementadas por meio de **funções** (*functions*).

Introdução

- Um programa escrito em linguagem C possui no mínimo uma função.

- A **função main( )** é responsável por inicializar a execução de um programa escrito em C.

Introdução

- Existem outras funções da linguagem C que são muito utilizadas.

- Exemplos: scanf( ), gets( ), printf( ), strcpy( ),...

Introdução

- O programador pode desenvolver quantas funções achar necessário. **Não existe um número máximo permitido**.

- A quantidade de funções depende da ***complexidade*** do problema que está sendo resolvido.

Funções

- Sintaxe de uma função em linguagem C

```
tipo_de_retorno nome_da_funcao (parâmetros){
    declaracao de variaveis locais;
    instrucoes para realizar a tarefa;
    return valor; // instrução opcional (dependente da função)
}
```

Funções

- O valor de retorno da função é do mesmo tipo_de_retorno declarado.

Retornos de função

- Uma função em linguagem C pode retornar **um ou mais valores**.
- Quando a função **não retorna valor**, considera-se o tipo_de_retorno ***void***.
- Nesse caso, **não é necessário considerar a instrução return**.

Tipos de função

- Tipos de função considerados pela linguagem C
  - Função **sem** passagem de parâmetros e **sem** retorno
  - Função **com** passagem de parâmetros e **sem** retorno
  - Função **sem** passagem de parâmetros e **com** retorno
  - Função **com** passagem de parâmetros e **com** retorno

Tipos de função

- Tipos de função considerados pela linguagem C
  - **Função sem passagem de parâmetros e sem retorno**
  - Função **com** passagem de parâmetros e **sem** retorno
  - Função **sem** passagem de parâmetros e **com** retorno
  - Função **com** passagem de parâmetros e **com** retorno

Função sem passagem e sem retorno

- Este é o tipo **mais simples** de função, pois não recebe nenhum parâmetro.

- Também **não retorna nenhum valor** para a instrução de chamada, após a execução.

Exemplo 1

```
// Exemplo de função sem passagem de parâmetros e sem retorno
void soma(){
 int numero1, numero2;
 printf("\n Digite o numero 1 \n");
 scanf("%d", &numero1);
 printf("\n Digite o numero 2 \n");
 scanf("%d", &numero2);
 printf("\n\n Resultado da Soma: %d\n",
numero1+numero2);
}
```

Exemplo 1

```
#include <stdio.h>
//Função já implementada
void soma(){
 //código já apresentado
}
int main(){
 soma();
 return 0;
}
```

Exemplo 1 - Descrição

- No exemplo apresentado, a execução do programa inicia pela função main(). Esse é o fluxo de execução padrão.

- A função main() é executada até a instrução de chamada da função soma().

Exemplo 1 - Descrição

- Após a chamada, o fluxo de execução é desviado para a função soma(), que é executada até a última instrução.

- Ao final da execução da função, nenhum valor é retornado e o fluxo de execução volta para a função main().

Tipos de função

- Tipos de função considerados pela linguagem C
  - Função **sem** passagem de parâmetros e **sem** retorno
  - **Função com passagem de parâmetros e sem retorno**
  - Função **sem** passagem de parâmetros e **com** retorno
  - Função **com** passagem de parâmetros e **com** retorno

Função com passagem e sem retorno

- Essa função pode receber um ou mais parâmetros, na chamada da função.

- Após a execução da função, nenhum valor é retornado.

Exemplo 2

```
// Exemplo de função com passagem de parâmetros e sem retorno
void soma(int numero1, int numero2){
   printf("\n\n Resultado da Soma:
                              %d\n", numero1+numero2);
}
```

Exemplo 2

```
#include <stdio.h>
int main(){
 int numero1, numero2;
 printf("\n Digite o numero 1 \n");
 scanf("%d", &numero1);
 printf("\n Digite o numero 2 \n");
 scanf("%d", &numero2);
 soma(numero1, numero2);
 return 0;
}
```

Exemplo 2 - Descrição

- A execução do programa inicia pela função main() e continua até a chamada da função soma().

- Na chamada da função soma(), os parâmetros a serem utilizados são passados.

Exemplo 2 - Descrição

- O fluxo de execução é então desviado para a função soma(). A execução da função acontece, sem nenhum valor de retorno.

- O fluxo de execução volta para a função main() e permanece até a execução da última instrução.

Tipos de função

- Tipos de função considerados pela linguagem C
  - Função **sem** passagem de parâmetros e **sem** retorno
  - Função **com** passagem de parâmetros e **sem** retorno
  - **Função sem passagem de parâmetros e com retorno**
  - Função **com** passagem de parâmetros e **com** retorno

Função sem passagem e com retorno

- Essa função não considera valores de entrada, entretanto, devolve um ou mais valores de saída.
- A função não recebe um ou mais parâmetros (entrada).
- Ao final da execução, um ou mais valores são retornados (saída).

Exemplo 3

```
// Exemplo de função sem passagem de parâmetros e com retorno
int soma(){
 int numero1, numero2, resultado;
 printf("\n Digite o numero 1 \n");
 scanf("%d", &numero1);
 printf("\n Digite o numero 2 \n");
 scanf("%d", &numero2);
 resultado = numero1 + numero2;
 return resultado;
}
```

Exemplo 3

```
#include <stdio.h>
//Função já implementada
int soma(){
 //código já apresentado
}
int main(){
 int valor_da_soma;
 valor_da_soma = soma();
 printf("\n Resultado da Soma:
                          %d\n", valor_da_soma);
 return 0;
}
```

Exemplo 3 - Descrição

- A execução do programa começa pela função main() e continua até a chamada da função soma().
- Na chamada da função, parâmetros não são passados.
- O fluxo de execução é então desviado para a função soma().

Exemplo 3 - Descrição

- A execução da função soma() acontece e um valor de retorno é devolvido para a instrução que chamou a função.

- Na sequência, o fluxo de execução permanece na função main(), até a execução da última instrução.

Tipos de função

- Tipos de função considerados pela linguagem C
  - Função **sem** passagem de parâmetros e **sem** retorno
  - Função **com** passagem de parâmetros e **sem** retorno
  - Função **sem** passagem de parâmetros e **com** retorno
  - **Função com passagem de parâmetros e com retorno**

Função com passagem e com retorno

- Uma função com passagem e com retorno é o tipo de função mais utilizado em linguagem C.
- A função recebe um ou mais parâmetros.
- Ao final da execução da função, um ou mais valores são retornados.

Exemplo 4

```
// Exemplo de função com passagem de parâmetros e com retorno
int soma(int numero1, int numero2){
 int resultado;
 resultado = numero1 + numero2;
 return resultado;
}
```

Exemplo 4

```c
#include <stdio.h>

int main(){
    int numero1, numero2, valor_da_soma;
    printf("\n Digite o numero 1 \n");
    scanf("%d", &numero1);
    printf("\n Digite o numero 2 \n");
    scanf("%d", &numero2);
    valor_da_soma = soma(numero1, numero2);
    printf("\n Resultado da Soma: %d\n", valor_da_soma);
    return 0;
}
```

Exemplo 4 - Descrição

- A execução do programa começa pela função main() e continua até a chamada da função soma().
- Na chamada da função, os parâmetros são passados.
- O fluxo de execução é então desviado para a função soma().

Exemplo 4 - Descrição

- A execução da função soma() acontece e um valor de retorno é devolvido para a instrução que chamou a função.

- Na sequência, o fluxo de execução permanece na função main(), até a execução da última instrução da função().

Prática - Calculadora

- Desenvolva uma calculadora básica com as seguintes operações: adição, subtração, divisão e multiplicação.

- No desenvolvimento, pode-se considerar a passagem ou a não passagem de parâmetros, bem como o retorno ou não retorno de valores.

Prática - Calculadora

- Função de adição

```
//Função sem passagem de parâmetros e sem retorno
void adicao(){
 int numero1, numero2;
 printf("\n Digite os numeros 1 e 2 \n");
 scanf("%d %d", &numero1, &numero2);
 printf("\n Resultado adicao: %d\n", numero1+numero2);
}
```

Prática - Calculadora

- Função de subtração

```
//Função com passagem de parâmetros e sem retorno
void subtracao(int numero1, int numero2){
 int resultado = numero1 - numero2;
 printf("\n Resultado subtracao: %d\n", resultado);
}
```

Prática - Calculadora

- Função de divisão

```
//Função sem passagem de parâmetros e com retorno
int divisao(){
 int numero1, numero2, valor_da_divisao;
 printf("\n Digite os numeros 1 e 2 \n");
 scanf("%d %d", &numero1, &numero2);
 valor_da_divisao = (numero1/numero2);
 return valor_da_divisao;
}
```

Prática - Calculadora

- Função de multiplicação

```
//Funções com passagem de parâmetros e com retorno
int multiplicacao(int numero1, int numero2){
 int valor_da_multiplicacao;
 valor_da_multiplicacao = (numero1 * numero2);
 return valor_da_multiplicacao;
}
```

Prática - Calculadora

- Função main

```
int main(){
    int opcao, valor1, valor2, resultado;

    printf("\n Calculadora Basica \n");
    printf("<1> Adicao \n");
    printf("<2> Subtracao \n");
    printf("<3> Divisao \n");
    printf("<4> Multiplicacao \n");
    printf("\n Digite a sua opcao \n");
    scanf("%d", &opcao);
    ....
```

Prática - Calculadora

- Função main

```
int main(){
    ...
    switch(opcao){
        case 1:
            adicao();
            break;
        case 2:
            printf("\n Digite os numeros 1 e 2 \n");
            scanf("%d %d", &valor1, &valor2);
            subtracao(valor1, valor2);
            break;...
```

Prática - Calculadora

- Função main

```
case 3:
     resultado = divisao();
     printf("\n Resultado Divisao: %d\n", resultado);
     break;
case 4:
     printf("\n Digite os numeros 1 e 2 \n");
     scanf("%d %d", &valor1, &valor2);
     resultado = multiplicacao(valor1, valor2);
     printf("\n Resultado Multiplicacao: %d\n", resultado);
     break;
}
return 0;}
```

Dúvidas?