Luciano Moraes Da Luz Brum
Universidade Federal do Pampa – Unipampa – Campus Bagé
email: lucianobrum18@gmail.com

Encapsulamento de Código: Funções e Procedimentos

unipampa
Universidade Federal do Pampa

Engenharia de Computação

Sumário

- Introdução à Funções e Procedimentos
- Estrutura Básica de uma Função
- Declaração de Funções
- Invocação de Funções
- Passagem de Parâmetros para Funções
- Retorno de Valores das Funções
- Vantagens e Dicas
- Resumo

Introdução

- Para estruturação dos programas, utiliza-se o conceito de **encapsulamento de código**.

- Encapsular o código consiste em dividir o programa em partes que executam tarefas menores e depois são acopladas para formar o programa.

Introdução

- Existem duas formas de encapsulamento de código em C: **funções** e **procedimentos**.

- Definição (função): é um bloco de código que executa uma tarefa específica ao ser chamado. Ao final da execução, um valor é retornado.

- Definição (procedimento): é um bloco de código que executa uma tarefa específica ao ser chamado. Não são retornados valores.

Introdução

- Um exemplo de função da linguagem C é a função main.

- Funções e procedimentos são chamados dentro da função main ou de outras funções, na linguagem C.

unipampa
Universidade Federal do Pampa

Engenharia de Computação

Sumário

- Introdução à Funções e Procedimentos
- Estrutura Básica de uma Função
- Declaração de Funções
- Invocação de Funções
- Passagem de Parâmetros para Funções
- Retorno de Valores das Funções
- Vantagens e Dicas
- Resumo

Estrutura de uma função

```
Tipo_de_retorno  Nome_da_Função(Lista de parâmetros){

        Declarações

        Comandos

}
```

Estrutura de uma função

Exemplo em pseudocódigo:

```
função SomaValores (x, y: inteiro) : inteiro

        var soma: inteiro

        soma <- x + y

        retorna soma
```

Estrutura de uma função

Exemplo em pseudocódigo:

```
Procedimento SomaValores (x, y: inteiro) : vazio

        var soma: inteiro

        soma <- x + y

        escreva (soma)
```

unipampa
Universidade Federal do Pampa

Engenharia de Computação

Sumário

- Introdução à Funções e Procedimentos
- Estrutura Básica de uma Função
- Declaração de Funções
- Invocação de Funções
- Passagem de Parâmetros para Funções
- Retorno de Valores das Funções
- Vantagens e Dicas
- Resumo

Declaração de uma Função

Exemplo em linguagem C:

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

Declaração de uma Função

Exemplo em linguagem C:

Tipo de Retorno da função

```
int SomaValores (int x, int y) {

        int soma;

        soma = x + y;

        return soma;

}
```

Declaração de uma Função

Exemplo em linguagem C:


```
int SomaValores (int x, int y) {

        int soma;

        soma = x + y;

        return soma;

}
```

Declaração de uma Função

Exemplo em

Parâmetros x e y do tipo int, recebidos da função que chamou o SomaValores

```
int SomaValores (int x, int y) {

        int soma;

        soma = x + y;

        return soma;

}
```

Declaração de uma Função

Exemplo em linguagem C:

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

Declaração de uma Função

Exemplo em linguagem C:

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

Soma é a variável que contém o valor que deve ser retornado para a função que chamou SomaValores e return é a instrução que faz o retorno

Declaração de uma Função

Exemplo em linguagem

```
int SomaValores (int x,

        int soma;

        soma = x + y;

        return soma;

}
```

Por que é necessário o retorno de valores?

Declaração de uma Função

Exemplo em linguagem

```
int SomaValores (int x,

        int soma;

        soma = x + y;

        return soma;

}
```

Por que é necessário o retorno de valores?

Variáveis utilizadas numa função só existem dentro da função. No programa principal (main), as variáveis que foram usadas nas funções NÃO EXISTEM. Por isso, devemos retornar valores para o main, geralmente os resultados da função.

Declaração de uma Função

Exemplo em linguagem C:

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

O valor em soma é perdido
toda vez que a função executa

Declaração de um Procedimento

Exemplo em linguagem C:

```
void SomaValores (int x, int y) {
        int soma;
        soma = x + y;
        printf (“A soma é: %d”, soma);
}
```

Declaração de um Procedimento

Exemplo em linguagem C:

Nenhum retorno

```
void SomaValores (int x, int y) {

        int soma;

        soma = x + y;

        printf (“A soma é: %d”, soma);

}
```

Declaração de um Procedimento

Exemplo em linguagem C:

```
void SomaValores (int x, int y) {
        int soma;
        soma = x + y;
        printf (“A soma é: %d”, soma);
}
```

Já que não retorna nenhum dado, o procedimento já mostra na tela o resultado da soma

unipampa
Universidade Federal do Pampa

Engenharia de Computação

Sumário

- Introdução à Funções e Procedimentos
- Estrutura Básica de uma Função
- Declaração de Funções
- Invocação de Funções
- Passagem de Parâmetros para Funções
- Retorno de Valores das Funções
- Vantagens e Dicas
- Resumo

Invocação de funções

- O programa principal se comunica com uma função ou procedimento através de uma chamada ou invocação da função.
- Uma chamada de função é especificada através do seu nome e passagem de argumentos dos quais a função precisa para realizar sua tarefa.
- Quando o programa principal chama a função, o controle do fluxo de execução do programa passa para a função.
- Funções não podem ser invocadas antes de serem declaradas.

Invocação de Funções

- Exemplo de invocação de função:

```
#include<stdio.h>

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

int main(void){
      int a, b, c;
      a = 10;
      b = 20;
      c = soma (a,b);
      printf("Soma:%d", c);
      return 0;
}
```

A função soma é chamada, enviando como parâmetros o conteúdo das variáveis a e b

Invocação de Funções

- Exemplo de invocação de função:

```
#include<stdio.h>

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

int main(void){
        int a, b, c;
        a = 10;
        b = 20;
        c = soma (a,b);
        printf("Soma:%d", c);
        return 0;
}
```

O fluxo de execução é transferido para a função soma. Ela executa até encontrar a instrução "return", enviando o resultado de volta para a função main.

Invocação de Funções

- Exemplo de invocação de função:

```
#include<stdio.h>

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

int main(void){
        int a, b, c;
        a = 10;
        b = 20;
        c = soma (a,b);
        printf("Soma:%d", c);
        return 0;
}
```

A variável c recebe o conteúdo da variável resultado, enviada pelo return da função soma. Por fim, o programa principal termina sua execução com o return 0.

Invocação de Funções

- Exemplo de invocação de procedimento:

```
#include<stdio.h>

    void imprime ( int s ) {
    printf (“O Resultado da soma é %d.”, s);
    }

    void soma(int x, int y){
        int resultado;
        resultado = x + y;
        imprime ( resultado );
    }
```

```
int main(void){
    int a, b;
    a = 10;
    b = 20;
    soma (a,b);
    retu n 0;
}
```

Aqui é feita a invocação do proc. SOMA

O fluxo de execução do programa passa agora para o proc.

Invocação de Funções

- Exemplo de invocação de procedimento:

```
#include<stdio.h>

    void imprime ( int s ) {
    printf ("O Resultado da soma é %d.", s);
    }

    void soma(int x, int y){
        int resultado;
        resultado = x + y;
        imprime ( resultado );
    }
```

```
int main(void){
        int a, b;
        a = 10;
        b = 20;
        soma (a,b);
        return 0;
}
```

O fluxo de execução do programa passa agora para o proc. IMPRIME

O procedimento executa e no final invoca outro proc., o IMPRIME

Invocação de Funções

- Exemplo de invocação de procedimento:

```
#include<stdio.h>

    void imprime ( int s ) {
    printf ("O Resultado da soma é %d.", s);
    }

    void soma(int x, int
        int resultado;
        resultado = x +
        imprime ( resu
    }

int main(void){
        int a, b;
        a = 10;
        b = 20;
        soma (a,b);
        return 0;
}
```

O procedimento conclui sua execução e o fluxo volta para o programa principal

Invocação de Funções

- Exemplo de invocação de procedimento:

```
#include<stdio.h>

    void imprime ( int s ) {
    printf (“O Resultado da soma é %d.”, s);
    }

    void soma(int x, int y){
        int resultado;
        resultado = x + y;
        imprime ( resultado );
    }
```

```
int main(void){
        int a, b;
        a = 10;
        b = 20;
        soma (a,b);
        return 0;
}
```

unipampa
Universidade Federal do Pampa

Engenharia de Computação

Sumário

- Introdução à Funções e Procedimentos
- Estrutura Básica de uma Função
- Declaração de Funções
- Invocação de Funções
- Passagem de Parâmetros para Funções
- Retorno de Valores das Funções
- Vantagens e Dicas
- Resumo

Passagem de parâmetros para funções

- Funções e procedimentos podem não receber parâmetros, assim como podem receber vários parâmetros.

- Pode ser passado como parâmetro de funções:

  - Qualquer tipo básico de dado;
  - Vetores;
  - Matrizes;
  - Tipos estruturados;
  - Outras funções;

Passagem de parâmetros para funções

- Exemplos de passagem de parâmetros para funções:
  - c = potenciacao(x, y);
    - int potenciacao(int a, int b){ ...
  - imprime( );
    - void imprime(){ ...
  - y = sqrt (x);

Passagem de parâmetros para funções

- Exemplos de passagem de funções como parâmetro para funções:
  - c = soma(x, soma (y, z));
  - c = soma(soma(x, y), soma(w, z));

unipampa
Universidade Federal do Pampa

Engenharia de Computação

Sumário

- Introdução à Funções e Procedimentos
- Estrutura Básica de uma Função
- Declaração de Funções
- Invocação de Funções
- Passagem de Parâmetros para Funções
- Retorno de Valores das Funções
- Vantagens e Dicas
- Resumo

Retorno de valores das funções

- Funções sempre devem retornar um valor.
- Pode ser retornado qualquer tipo de valor, desde que isso esteja declarado na definição da função.
- Uma função só pode retornar um valor por vez.
- O return é o último comando a ser executado em uma função.

Retorno de valores das funções

```
#include<stdio.h>

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

int main(void){
      int a, b, c;
      a = 10;
      b = 20;
      c = soma (a,b);
      printf("%d", c);
      return 0;
}
```

Retorno de valores das funções

```
#include<stdio.h>

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

int main(void){
      int a, b, c;
      a = 10;
      b = 20;
      c = soma (a,b);
      printf("%d", c);
      return 0;
}
```

Retorno de valores das funções

```
#include<stdio.h>

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

int main(void){
      int a, b;
      a = 10;
      b = 20;
      printf(“%d”, soma (a,b));
      return 0;
}
```

unipampa
Universidade Federal do Pampa

Engenharia de Computação

Sumário

- Introdução à Funções e Procedimentos
- Estrutura Básica de uma Função
- Declaração de Funções
- Invocação de Funções
- Passagem de Parâmetros para Funções
- Retorno de Valores das Funções
- Vantagens e Dicas
- Resumo

Vantagens de Funções

- Ajudam a organizar o programa.
- Memória alocada para as variáveis das funções é liberada após a execução da função.
- Evitam repetição de código, diminuindo erros e facilitando alterações.
- Facilita a leitura e entendimento do código.

Vantagens de Funções

- Elabore um programa que imprima a seguinte mensagem na tela:

```
+++

====

+++++

====

+++
```

Vantagens de Funções

- Sem uso de funções:

```
for( i = 0 ; i < 3 ; i++){
        printf(“+”);
}
printf(“\n”);
for( i = 0 ; i < 4 ; i++){
        printf(“=”);
}
printf(“\n”);
for( i = 0 ; i < 5 ; i++){
        printf(“+”);
}
printf(“\n”);
for( i = 0 ; i < 4 ; i++){
        printf(“=”);
}
printf(“\n”);
for( i = 0 ; i < 3 ; i++){
        printf(“+”);
}
```

Vantagens de Funções

- Com uso de procedimento:

```
#include<stdio.h>
void imprime ( int n, char ch ){
        int i;
        for ( i = 0; i < n; i++){
                printf (“%c”, ch);
        }
        printf(“\n”);
}
int main ( void ) {
        imprime (3 , ‘+’);
        imprime (4 , ‘=‘);
        imprime (5 , ‘+’);
        imprime (4 , ‘=‘);
        imprime (3 , ‘+’);
        return 0;
}
```

Vantagens de Funções

- Calcule o fatorial de um número.

Vantagens de Funções

- Calcule o fatorial de um número.

```
#include<stdio.h>
int main ( void ) {
        int x, y, i;
        ...
        y=1;
        for(i=1;i<=x;i++){
        y=y*i;
        }
        ...
        return 0;
}
```

Vantagens de Funções

- Calcule o fatorial de um número.

```
#include<stdio.h>
Int fact(int x){
        int i, r = 1;
        for(i=1;i<=x;i++){
        r=r*i;
        }
        return r;
}
int main ( void ) {
        int x, y, i;
        ...
        y = fact(x);
        return 0;
}
```

Vantagens de Funções

- Calcule o fatorial de um número.

```
#include<stdio.h>
Int fact(int x){
        int i, r = 1;
        for(i=1;i<=x;i++){
        r=r*i;
        }
        return r;
}
int main ( void ) {
        int x, y, i;
        ...
        y = fact(x);
        return 0;
}
```

Função fatorial para um número

Vantagens de Funções

- Agora faça um algoritmo que use a formula de combinação .

Vantagens de Funções

```
#include<stdio.h>
int fact(int x){
        int i, r = 1;
        for(i=1;i<=x;i++){
        r=r*i;
        }
        return r;
}
int main ( void ) {
        int n, y, k;
        scanf("%d%d", &n, &k);
        y = fact(n) / (fact(k) * fact(n-k));
        printf("Combinação de %d elementos tomados %d a %d e: %d",
n, k, k, y);
        return 0;
}
```

Função fatorial para um número

Fórmula da combinação

Dicas

- Não esqueça do retorno das funções.

- ERRADO: int funcao (int x , y; float a, b){...
- CORRETO: int funcao(int x, int y, float a, float b){...

- Evite usar os mesmos nomes para os argumentos da função e os parâmetros que definem a função.

- Escolha um bom nome de função.

unipampa
Universidade Federal do Pampa

Engenharia de Computação

Sumário

- Introdução à Funções e Procedimentos
- Estrutura Básica de uma Função
- Declaração de Funções
- Invocação de Funções
- Passagem de Parâmetros para Funções
- Retorno de Valores das Funções
- Vantagens e Dicas
- Resumo

Resumo

- Foram demonstradas as definições de funções e procedimentos.
- Foi demonstrada a estrutura básica de uma função em pseudocódigo e linguagem C.
- Foram demonstradas regras básicas para declaração e invocação de funções.
- Foi abordado o formato de envio de parâmetros e retorno de valores em funções.

Bibliografia

- ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi de. **Fundamentos da Programação de Computadores: Algoritmos, Pascal, C/C++ e Java.** São Paulo: Pearson Prentice Hall, 2008.

- FORBELLONE, André Luiz Vilar; EBERSPACHER, Henri Frederico. **Lógica de Programação: A construção de algoritmos e estruturas de dados**. 3 ed. São Paulo: Pearson Prentice Hall, 2005.

- CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. **Introdução a estruturas de dados**. Editora Campus, 2004.

- MANZANO, Jose Augusto N. G.; Oliveira, Jayr Figueiredo de. **Algoritmos: lógica para desenvolvimento de programação de computadores**. 17 ed. São Paulo: Érica, 2007.

unipampa

Universidade Federal do Pampa

Dúvidas?

email: lucianobrum18@gmail.com