Introdução à Programação
AULA 10 – Funções e Procedimentos

Profª. Glaucia M. M. Campos
glauciamelissa@uern.br

Universidade do Estado do Rio Grande do Norte - CaN

Funções

- Funções (rotinas ou sub-programas) são segmentos de programa que executam uma determinada tarefa específica.
  - **Funções de biblioteca**: sqrt(), pow(), getch() e printf()
- **Funções de usuário**: é permitido ao programador escrever suas próprias rotinas.
- **Segmentação/modularização** permite que cada segmento seja escrito e testado individualmente.
- Permite ainda que um programa seja escrito por vários programadores ao mesmo tempo, cada um escrevendo um segmento separado.

Declaração de uma função

- De modo formal, uma função é definida da seguinte forma:

```
tipo_de_retorno nome_da_função(tipo_1 arg_1, tipo_2 arg_2, ...){
    bloco_de_instruções;
}
```

- A primeira linha da função contém a **declaração** da função. Na declaração de uma função se define o **nome** da função, seu **tipo de retorno** e a **lista de argumentos** que recebe.
- Em seguida, dentro de chaves {}, definimos o bloco de instruções da função.
- O **tipo de retorno** da função especifica qual o tipo de dado retornado pela função. (int, float, void...)

Exemplo 1

```
float media2(float a, float b){
      float med;
      med = (a + b) / 2.0;
      return(med);
}
```

- "media2" recebe dois argumentos tipo float: a e b.
- A média destes dois valores é calculada e armazenada na variável med declarada internamente.
- A função retorna, para o programa que a chamou, um valor também do tipo float: o valor da variável med.
- Este retorno de valor é feito pela função return() que termina a execução da função e retorna o valor de med para o programa que a chamou.
- Depois de definimos um função, podemos usá-la dentro de um programa qualquer, fazendo uma **chamada** a função.

Exemplo 1 – Programa Principal

```
float media2(float a, float b){
      float med;
      med = (a + b) / 2.0;
      return(med);
}

int main(){
      float num_1, num_2, med;
      puts(”Digite dois números:”);
      scanf(”%f %f”, &num_1, &num_2);
      med = media2(num_1, num_2); // chamada a função
      printf(”\nA media destes números é %f”, med);
      return 0;
}
```

Exemplo 2

```
void mostra(char a){
     printf("%c", a);
}
```

- "mostra" recebe um argumento tipo char: a
- Esse caractere é impresso na tela por meio do comando **printf**
- A função não retorna nenhuma informação, pois o seu tipo de retorno é **void**
- O **resultado** da função (procedimento) vai ser impresso na tela

Exemplo 2 – Programa Principal

```
void mostra(char a){
        printf(“%c”, a);
}

int main(){
  printf(“Programa que imprime caractere:”);
  mostra('*'); //chamada ao procedimento
  return 0;
}
```

Exemplo 3

```c
#include <stdio.h>
main()
{
 int i;
 // Imprime linha com 10 *.
 for (i=0; i < 10; i++) printf("*");
 printf("\n");
 // Imprime linha com 10 -.
 for (i=0; i < 10; i++) printf("-");
 printf ("\n");
 // Imprime linha com 10 -.
 for (i=0; i < 10; i++) printf("-");
 printf ("\n");
 // Imprime linha com 10 *.
 for (i=0; i < 10; i++) printf("*");
 printf ("\n");
}
```

SEM MODULARIZAÇÃO

```c
#include <stdio.h>
void imprimeA()
{ int i;
   for(i=0;i < 10; i++) printf("*");
   printf ("\n");
}
void imprimeL()
{ int i;
   for(i=0; i < 10; i++) printf("-");
   printf ("\n");
}
main()
{imprimeA();
  imprimeL();
  imprimeL();
  imprimeA();}
```

COM MODULARIZAÇÃO

Localização das Funções

- Existem algumas posições possíveis para escrevermos o corpo de uma função:
  - No mesmo arquivo do programa principal
    - Antes do Programa Principal
    - Depois do Programa Principal
  - Em arquivo separado

Localização das Funções

- No mesmo arquivo **antes** do programa principal
- **Sintaxe:** Uma função escrita antes do programa principal:

```
tipo nomef(...){      // definição da função
  [corpo de função]
}
int main(){           // programa principal
  ...
  var = nomef(...)    // chamada da função
  ...
}
```

Exemplo – Antes do programa principal

```
float media2(float a, float b){
  float med;
  med = (a + b) / 2.0;
  return(med);
}
int main(){
  float num_1, num_2, med;
  puts(”Digite dois números:”);
  scanf(”%f %f”, &num_1, &num_2);
  med = media2(num_1, num_2);
  printf(”\nA media destes números é %f”, med);
  return 0;
}
```

Localização das Funções

- No mesmo arquivo **depois** do programa principal
- **Sintaxe:** Uma função escrita depois do programa principal:

```
int main(){          // programa principal
  tipo nomef(...);   // protótipo da função
  ...
  var = nomef(...)   // chamada a função
  ...
}
tipo nomef(...){     // definição da função
 [corpo de função]
}
```

Exemplo – Depois do programa principal

```
int main(){
  float media2(float,float);
  float num_1, num_2, med;
  puts(”Digite dois números:”);
  scanf(”%f %f”, &num_1, &num_2);
  med = media2(num_1, num_2);
  printf(”\nA media destes números e´ %f”, med);
  return 0;
}
float media2(float a, float b){  // função media2()
  float med;
  med = (a + b) / 2.0;
  return(med);
}
```

Localização das Funções

- Em arquivo **separado**
  - Usuário pode criar uma função em **um arquivo** e um programa que a chame em outro **arquivo distinto**
  - Criação de **bibliotecas de usuário**:

Um conjunto de arquivos contendo funções escritas pelo usuário. Esta possibilidade é uma grande vantagem utilizada em larga escala por programadores profissionais.

  - Local de **inserção** da biblioteca de usuário:

Biblioteca deve ser inserida no conjunto de arquivos de **compilação** do programa principal. Esta inclusão é feita com a diretiva #include.

Localização das Funções

- **Sintaxe:** A sintaxe de inclusão de funções de usuário é a seguinte:

```
#include "path"              // inclusão da função
void main(){                 // programa principal
...
  var = nomef(...)           // chamada a função
...
}
```

- Na diretiva #include, indicamos entre aspas duplas o caminho de localização do arquivo onde está definida a função chamada.

Exemplo – Em arquivo separado

```
#include "/Users/macbook/stat.h"
void main(){
  float num_1, num_2, med;
  puts(”Digite dois números:”);
  scanf(”%f %f”, &num_1, &num_2);
  med = media2(num_1, num_2);
  printf(”\nA media destes números e´ %f”, med);
}
```

- **Observação:** Um arquivo pode conter a definição de uma ou mais funções. Em geral, quando o arquivo possui apenas uma função ele é nomeado com o mesmo nome da função e extensão (*.c). Quando um arquivo possui a definição de mais de uma função, ele é nomeado com a extensão *.h

Exemplo – Em arquivo separado

- Criar um arquivo novo do tipo .h <stat.h>
- Declara o nome da função com o seu tipo e os seus argumentos
- Especifica o corpo da função

```
float media2(float a, float b);
float media2(float a, float b){
  float med;
  med = (a + b) / 2.0;
  return(med);
}
```

Exemplo 4 – Calculadora (funções)

```
#include<stdio.h>
#include<stdlib.h>

//conteudo do menu "soma”
float soma(int num1, int num2){
     return (num1+num2);
}

//conteudo do menu "subtrair”
float subtrai(int num1, int num2){
    return (num1 – num2);
}
```

```
//conteudo do menu "multiplicar"
float multiplica(int num1, int num2){
     return (num1*num2);
}

//conteudo do menu ”dividir"
float divide(int num1, int num2){
    if(num2!=0) {
         return (num1/num2);
    }else{
        printf(”Entre com valor positivo!");
        return 0;
   }
}
```

Exemplo 4 – Calculadora (funções)

```
int main(){
    //declaracao de variaveis
    float opc, num1, num2, result;
   //solicita dados
    printf("Entre com o primeiro numero: ");
    scanf("%d",&num1);
    printf("Entre com o segundo numero: ");
    scanf("%d",&num2);
    //solicita operacao
    printf("Escolha a operacao:");
    printf("\n[1] +\n[2] -\n[3] *\n[4] /\n");
    printf("Qual opcao voce escolhe? ");
    scanf("%d",&opc);
    getchar();
    switch(opc){
        case 1: result = soma(num1,num2);
                break;
        case 2: result = subtrai(num1,num2);
                break;
        case 3: result = multiplica(num1,num2);
                break;
        case 4: result = divide(num1,num2);
                break;
    }
    printf("\n\n\nResultado: %f\n",result);
    return (0);
}
```

Exemplo 4 – Calculadora (procedimentos)

```
#include<stdio.h>
#include<stdlib.h>

//conteudo do menu "soma”
void soma(int num1, int num2){
     printf (“Resultado: %f”, num1+num2);
}

//conteudo do menu "subtrair”
void subtrai(int num1, int num2){
    printf (“Resultado: %f”, num1 – num2);
}
```

```
//conteudo do menu "multiplicar"
void multiplica(int num1, int num2){
     printf (“Resultado: %f”, num1*num2);
}

//conteudo do menu ”dividir"
void divide(int num1, int num2){
    if(num2!=0) {
         printf (“Resultado: %f”, num1/num2);
    }else{
        printf(”Entre com valor positivo!");
        return 0;
   }
}
```

Exemplo 4 – Calculadora (procedimentos)

```
int main(){
    //declaracao de variaveis
    float opc, num1, num2, result;
   //solicita dados
    printf("Entre com o primeiro numero: ");
    scanf("%f",&num1);
    printf("Entre com o segundo numero: ");
    scanf("%f",&num2);
    //solicita operacao
    printf("Escolha a operacao:");
    printf("\n[1] +\n[2] -\n[3] *\n[4] /\n");
    printf("Qual opcao voce escolhe? ");
    scanf("%f",&opc);
    getchar();
    switch(opc){
        case 1: soma(num1,num2);
                break;
        case 2: subtrai(num1,num2);
                break;
        case 3: multiplica(num1,num2);
                break;
        case 4: divide(num1,num2);
                break;
    }
    return (0);
}
```

Exercícios (procedimentos)

1. Escreva um programa que coloque na tela a seguinte saída, escrevendo uma linha com 20 asteriscos através de um laço for:

```
********************
Números entre 1 e 5
********************
1
2
3
4
5
********************
```

2. Reescreva o programa 1, usando uma função de nome linha.
3. Reescreva a função linha() tal que o número de asteriscos seja um argumento da mesma.

Exercícios (procedimentos)

4. Alterar o programa 3 de forma que a função linha() tenha como argumentos o número de vezes que um caractere será impresso e qual é o caractere a ser impresso.
5. Modifique o programa 4 para que seja pedido para o usuário, em main(), o número de caracteres num e o caractere impresso ch.

Exercícios (bibliotecas)

Criar as seguintes bibliotecas:

matematica.h

- Verificar quantas vezes um número é divisível por outro
- Verificar se um número é primo
- Calcular o fatorial
- Converter radianos para graus

minhastrings.h

- Recebe um caractere e retorna 1 se for consoante e 0 se for vogal
- Recebe um caractere e retorna 1 se for vogal minúscula e 0 se for maiúscula
- Recebe um caractere e retorna 1 se for consoante minúscula e 0 se for maiúscula

Exercícios (bibliotecas)

- Criar exemplos que utilizem as funções que foram definidas na biblioteca

Exercícios (funções)

1. Criar um programa que possui uma função Maior cuja lista de parâmetros contém dois números inteiros a e b e que retorna o resultado de max{a,b}.
2. Faça uma função que recebe por parâmetro um valor inteiro e positivo e retorna o valor lógico Verdadeiro caso o valor seja primo e Falso em caso contrário.
3. Faça uma função que recebe por parâmetro o raio de uma esfera e calcula o seu volume (v = 4/3.P .R3).
4. Escreva uma função que recebe as 3 notas de um aluno por parâmetro e uma letra. Se a letra for A o procedimento calcula a média aritmética das notas do aluno, se for P, a sua média ponderada (pesos: 5, 3 e 2) e se for H, a sua média harmônica. A média calculada também deve retornar por parâmetro.

Referências

- Medina, Marco; Fertig, Cristina. Algoritmos e Programação: teoria e prática. São Paulo: Novatec Editora, 2006.
- Lopes, Anita; Garcia, Guto. Introdução à Programação: 500 algoritmos resolvidos. Rio de Janeiro: Editora Campus, 2002.
- Mizrani, Victorine Viviane. Treinamento em Linguagem C, Módulo 1. Editora Makron Books.
- Transparências modificadas do professor Dr. Flavio Luiz Cardeal Pádua, do Centro Federal de Educação Tecnológica de Minas Gerais
- Transparências modificadas do professor Robson Fidalgo, da UFRPE.