Funções em C

Docente: Gil Jader – gil.jader@gmail.com

Agenda – Parte 1

1 - CONCEITO

2 - Protótipo de função em C

3 - Função sem retorno

4 - Regras de Escopo

5 – Função Recursiva

6 – Referências

1. CONCEITO

Funções são trechos de código que podem ser referenciados por um **nome**. Quando o nome de uma função aparece no código, dizemos que a função foi “chamada". Quando uma função e chamada, ela “faz alguma coisa" e pode retornar (devolver) valores (resultado).

Quando a função não esta pronta em uma biblioteca, ela precisa ser criada (implementada).

Forma geral:

```
tipo_que_retorna   nome_da_função   (lista de parâmetros) {
    código da função
}
```

1. CONCEITO

Exemplo:

```
float   calculaMedia   (float a, float b) {
    código da função
    return(resultado);
}
```

A função se chama **calculaMedia**. Ela **retorna** (devolve) um valor real (**float**). Ela recebe dois argumentos do tipo **float** (*a* e *b*). Após calcular a média, o comando **return** (resultado); é responsável por retornar a média calculada para o programa que chamou a função.

Linguagem de Programação Estruturada – 2016.1

Exemplo de programa usando função em C

```c
#include<stdio.h>
#include<conio.h>
/*

   Nome da função: multiplica
   recebe como parâmetos dois valores inteiros (N1,N2)
   objetivo:  multiplicar os valores recebidos nos parâmetros.
   retorno:  um parâmetro inteiro (res) contendo o resultado
*/
int multiplica(int N1, int N2) //multiplica recebe N1,N2 e retorna um int
{
  int resultado;
  resultado = N1 * N2;
  return(resultado); //retornando o valor para main
}

/******************* função principal (main) **********************/

int main(void)
{
  int V1, V2, resultado;
  printf("Digite o primeiro valor:");
  scanf("%d", &V1);
  printf("Digite o segundo valor:");
  scanf("%d", &V2);

  //chama a função e recebe o retorno
  resultado = multiplica(V1,V2);
  printf("Resultado = %d\n", resultado);
  getch();
  return 0;
}
```

O que aconteceria se a função multiplica fosse declarada depois da função main?

```
Digite o primeiro valor:2
Digite o segundo valor:5
Resultado = 10
```

2. Protótipo de função em C

O protótipo de uma função é basicamente, uma declaração da interface da função, ou seja, deve especificar:

- Tipo da função;
- Nome da função;
- Lista de parâmetros que a função necessita;

2. Protótipo de função em C

Programa anterior reescrito usando protótipo:

```c
#include<stdio.h>
#include<conio.h>
/*   Protótipo da função  */
int multiplica(int N1, int N2);

int main(void)
{
  int V1, V2, resultado;
  printf("Digite o primeiro valor:");
  scanf("%d", &V1);
  printf("Digite o segundo valor:");
  scanf("%d", &V2);

  //chama a função e recebe o retorno
  resultado = multiplica(V1,V2);
  printf("Resultado = %d\n", resultado);
  getch();
  return 0;
}

int multiplica(int N1, int N2) //multiplica recebe N1,N2 e retorna um int
{
  int resultado;
  resultado = N1 * N2;
  return(resultado); //retornando o valor para main
}
```

2. Protótipo de função em C

Outro exemplo usando protótipo:

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

float calculaMedia(float, float);

int main (){

    float a , b , m;

    printf("\ nDigite o primeiro numero : ");
    scanf("%f", &a) ;
    printf("\ nDigite o segundo numero : ");
    scanf("%f" , &b);

    m = calculaMedia (a , b) ;

    printf("\nA media entre %.2f e %.2f eh: %.2f\n" , a , b ,m);

  system("pause");
  return (0);

}

float calculaMedia (float argA , float argB){
      float mediaCalculada;

      mediaCalculada = (argA + argB) / 2;

      return (mediaCalculada) ;
}
```

Linguagem de Programação Estruturada – 2016.1

```c
#include <stdio.h>
#include<conio.h>

void imprimeMatriz (float argMatriz[3][3]);

int main(){

    float matriz[3][3];
    int i , j;

    printf("\nDigite os elementos da matriz \n");
    for (i = 0; i < 3 ; i++){
        for ( j = 0 ; j < 3 ; j++ ){
            scanf("%f",&matriz[i][j]);
        }
    }

    imprimeMatriz(matriz);

   getch();
   return 0;

}
```

Outro exemplo usando protótipo:

Passando uma matriz como argumento para a função

```c
void imprimeMatriz (float argMatriz[3][3]){

    int i , j;

    printf("\nImpress~ao da matriz \n" );
    for(i = 0; i < 3; i++){
        for (j = 0; j < 3; j++){
            printf("%f\t", argMatriz[i][j]);
            printf("\n");
        }
    }
}
```

3. Função sem retorno

- Em C, é possível criar funções que não retornam nenhum valor.
- Neste caso, devemos usar *void* no tipo de retorno do cabeçalho da função.
- Se a função não recebe nenhum parâmetro, também colocamos *void* no local da listagem dos parâmetros.

exemplo:

```
void imprime_cabec(void)
{
  printf("*********************************\n");
  printf("*       LINGUAGEM C             *\n");
  printf("*********************************\n");

  return; /* retorno de uma função void  */
}
```

3. Função sem retorno

```c
#include<stdio.h>
#include<conio.h>

/******* Área dos protótipos ******/
void imprime_cabec(void);
int multiplica(int N1, int N2);
/******* fim dos protótipos *******/

/* ******* FUNÇÃO PRINCIPAL *******/
int main(void)
{
  int V1=0, V2=0, resultado=0;
  //Chamada da função imprime_cabec
  imprime_cabec();

  printf("Digite o primeiro valor:");
  scanf("%d", &V1);
  printf("Digite o segundo valor:");
  scanf("%d", &V2);

  //chama a função e recebe o retorno
  resultado = multiplica(V1,V2);
  printf("Resultado = %d\n", resultado);

  getch();
  return 0;
}
/* ******** FIM DA FUNÇÃO PRINCIPAL *******/
```

```c
/* --------- Corpo das funções --------- */
/*
***** Função imprime_cabec *****
Parâmetros de entrada: não tem (void)
Objetivo: imprimir cabeçalho na tela
Tipo de retorno: não tem (void)
*/
void imprime_cabec(void)
{
  printf("*********************************\n");
  printf("*       LINGUAGEM C             *\n");
  printf("*********************************\n\n");

  return;/* retorno de uma função void  */
}

/*
***** Função multiplica *****
Parâmetros de entrada: N1, N2 ambos int
Objetivo: multiplicar valores
Tipo de retorno: int (resultado);
*/
//multiplica recebe N1,N2 e retorna um int
int multiplica(int N1, int N2)
{
  int resultado;
  resultado = N1 * N2;

  //retornando o valor para main
  return(resultado);
}
```

4. Regras de Escopo

- Variáveis declaradas dentro de uma função são chamadas **variáveis locais**, ou seja, elas só podem ser usadas dentro da função.

- Quando o programa e executado, uma variável local e criada quando a execução do bloco e iniciada (**{**) e destruída quando a execução do bloco e encerrada (**}**).

```
...
int funcaoUm(int a){
    int x;
    x = a * a;
    return(x);
}
...
int funcaoDois(int m){
    int x;
    x = m * m * m;
    return(x);
}
```

5. Função Recursiva

- Uma função que é dita recursiva é aquela que invoca ela mesma.

- Porém também é possível que uma função chame ela mesma, mas é preciso um cuidado especial para não cairmos em um *looping* infinito.

- Geralmente, para que uma função não fique invocando ela mesma indefinidamente, devemos fazer umas alterações no argumento, ao invocar novamente a função ao passo que devemos definir, na função, testes condicionais sobre o parâmetro para saber onde devemos parar de invocar a função.

5. Função Recursiva

Exemplo:

Crie um programa em C que peça um número inteiro ao usuário e retorne a soma de todos os números de 1 até o número que o usuário introduziu ou seja: 1 + 2 + 3 + ... + n

Utilize recursividade.

Vamos criar uma função *soma(int n)*.

- Se n=5, essa função deve retornar: soma(5) = 5 + 4 + 3 + 2 + 1
- Se n=4, essa função deve retornar: soma(4) = 4 + 3 + 2 + 1
- Se n=3, essa função deve retornar: soma(3) = 3 + 2 + 1

5. Função Recursiva

Exemplo:

Crie um programa em C que peça um número inteiro ao usuário e retorne a soma de todos os números de 1 até o número que o usuário introduziu ou seja: 1 + 2 + 3 + ... + n

Utilize recursividade.

Veja que:

- soma(5) = 5 + 4 + 3 + 2 +1 = 5 + soma(4)

O mesmo para:

- soma(4) = 4 + 3 + 2 + 1 = 4 + soma(3)

A fórmula geral:
soma(n) = n + soma(n-1)

5. Função Recursiva

código em C:

```c
#include <stdio.h>

int soma(int n)
{
    if(n == 1)
        return 1;
    else
        return ( n + soma(n-1) );
}

int main()
{
    int n;
    printf("Digite um inteiro positivo: ");
    scanf("%d", &n);

    printf("Soma: %d\n", soma(n));
}
```

6. REFERÊNCIAS

- Márcio Alexandre Marques. Algoritmos - Lógica para Desenvolvimento de Programação de Computadores. 1ª Ed. Editora Érica, 2010.

- Sandra Rita. TREINAMENTO EM LOGICA DE PROGRAMAÇAO, Digerati Books, 1 ed. 2009.

- SIMÃO, DANIEL HAIASHIDA; REIS, WELLINGTON JOSÉ DOS. LOGICA DE PROGRAMAÇAO. São Paulo : EDITORA VIENA, 2015. 176p.

- Souza, Marco Antonio Furlan de *et. all*, Algoritimos e Lógica de Programação. 2 ed. São Paulo : Nobel, 2011.