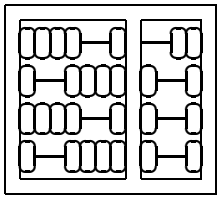

MC-102 Algoritmos e Programação de Computadores
IC-UNICAMP

Aula 15 - Aula sobre Funções e Procedimentos

1 Objetivos

Apresentar os conceitos de procedimentos e funções, suas vantagens e sua notação em C.

2 Motivação

- Permitir o reaproveitamento de código já construído (por você ou por outros programadores);
- Evitar que um trecho de código que seja repetido várias vezes dentro de um mesmo programa;
- Permitir a alteração de um trecho de código de uma forma mais rápida. Com o uso de uma função é preciso alterar apenas dentro da função que se deseja;
- Evitar que os blocos do programa fiquem grandes demais e, por conseqüência, mais difíceis de entender;
- Facilitar a leitura do programa-fonte;
- Separar o programa em partes(blocos) que possam ser logicamente compreendidos de forma isolada.

3 Procedimentos e Funções

Procedimentos são estruturas que agrupam um conjunto de comandos, que são executados quando o procedimento é chamado. Na verdade, procedimentos são subprogramas que são executados quando o programa principal invoca o procedimento.

No momento que o programa principal chama o procedimento, o fluxo de execução é desviado para o início do procedimento, todas as instruções do procedimento são executadas, e ao finalizar o procedimento, o fluxo de execução retorna ao comando seguinte a chamada do procedimento no programa principal. Ver figura 1.

Funções são semelhantes aos procedimentos, exceto que uma função sempre retorna um valor. Um exemplo de função seria o conjunto de instruções para calcular o fatorial de um número e após a função ser executada, ela deve retornar o fatorial do número pedido.

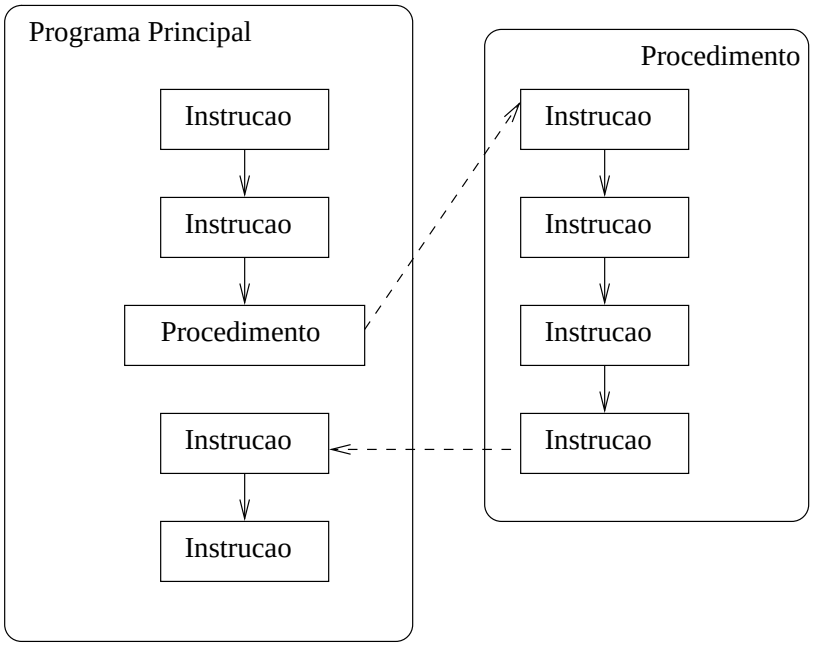

Figura 1: Fluxo de execução

3.1 Notação em C

A linguagem C possibilita apenas a definição de funções. A forma geral de uma função em C é:

```
<tipo_da_funcao> NomeDaFuncao ( <lista_de_parâmetros> )
{
    instrucoes;
}
```

Caso o tipo da função não seja fornecido, o C assume que a função é do tipo inteiro.

Para simular o comportamento de um procedimento em C, utiliza-se **void** no tipo da função. O *void* é um tipo nulo em C e portanto a função não necessita retornar nenhum valor.

Exemplo de procedimento em C:

```
#include <stdio.h>

void Ola()              /* Declaracao da funcao - Preciso colocar os ()s*/
{
  printf("\n Ola!!! \n");     /* Corpo da funcao */
}


/* -------- Programa Principal --------- */
int main()
{
```

```
  printf("\n Programa Principal !!");

  Ola();     /* Chamada da funcao */

  printf("\n Posso chamar a funcao de novo:");

  Ola();      /* Chamada da funcao - Eh necessario chamar com os ()s */
  Ola();

  printf("\n Final do Programa!!");
  return 0;
}
```

Exemplo 2: Função soma em C, sem parâmetros.

```
#include <stdio.h>

int soma1010() /* declaracao da funcao */
{
  int A;
  A=10+10;       /* Corpo da funcao */

  return A;      /* comando que indica qual o valor que a funcao deve retornar*/
                 /* ao encontrar esse comando, a funcao e' ENCERRADA */
}


/* -------- Programa Principal --------- */
int main()
{
  int N;
  int Resultado;

  printf("\n Entre com N: ");
  scanf("%d",&N);
  Resultado = N*soma1010()+20*soma1010()+30*(20+20);
  printf("\n\n   O Resultadoé  %d ",Resultado);
}
```

O comando **return** indica a função qual o valor que ela deve retornar. No caso acima, o *return* indica que o resultado da função deve ser igual ao valor de A. Atenção uma função **encerra** assim que encontrar o comando *return*.

O comando *return* aceita como argumento constantes, variáveis, expressões, etc. desde que o tipo do argumento seja igual ao tipo da função. Exemplos de chamada do comando *return*:

```
return 1;           /* função do tipo inteiro */
return A;           /* função do mesmo tipo de A */
return (3.14);      /* função do tipo float */
```

```
return (i+(A-4)*3);  /* função do mesmo tipo do resultado da expressão */
```

A fim de tornar mais amplo o uso de funções e procedimentos, permite-se o uso de parâmetros. Estes parâmetros possibilitam que se defina sobre quais dados a função ou o procedimento devem operar.

Por exemplo, o programa acima poderia ser simplificado se a função soma pudesse somar quaisquer dois números, e não apenas 10 mais 10.

Exemplo 3: Função soma com parâmetros.

```
#include <stdio.h>

int soma(int x, int y) /* declaracao da funcao */
{
  int A;
  A=x+y;        /* Corpo da funcao */

  return A;       /* comando que indica qual o valor que a funcao deve retornar*/
                  /* AO encontrar esse comando, a funcao e ENCERRADA */
}


/* -------- Programa Principal --------- */
int main()
{
  int N;
  int Resultado;

  printf("\n Entre com N: ");
  scanf("%d",&N);
  Resultado = N*soma(10,10)+20*soma(10,10)+30*soma(20,20);
  printf("\n\n   O Resultadoé %d ",Resultado);
}
```

Notação em C: A lista de parâmetros em C, é declarada logo após o nome da função e deve estar parênteses. Cada parâmetro deve ser precedido pelo seu tipo e casa haja mais de um, eles devem ser separados por vírgula.

```
<tipo_funcao> nome_funcao (tipo_param param1, tipo_param param2,...)
```

Exemplos:

```
int soma (int a, int b);
float divide(int a, int b);
float sen(float x);
unsigned long potencia (int x, int y);
char maiuscula(char letra);
```

3.2 a função *main()*

Como já deve-se ter percebido, o programa principal do C é a função **main()** e assim como as outras funções em C, ela deve ter um *tipo* e no final deve retornar algum valor (*return*). Por isso, em todos os programas que fizemos até agora, era necessário declarar a função *main()* e no seu final encerrá-la com o comando *return*. O tipo padrão da função *main()* é o *int*, mas alguns compiladores aceitam o tipo *void*.

```
int main()  /* Declaração da função main() - Programa Principal do C */
{
  ...
  return 0;  /* Como e uma funcao deve retornar um valor */
}
```

3.3 Localização das Funções no Programa Fonte em C

A princípio podemos tomar como regra a seguinte afirmativa toda função deve ser declarada antes de ser usada.

A declaração de uma função em linguagem C não é exatamente o que fizemos até agora. O que estamos fazendo é a definição da função antes de seu uso. Na definição da função está implícita a declaração.

Alguns programadores preferem que o programa principal esteja no início do arquivo fonte. Para isto a linguagem C permite que se declare uma função, antes de defini-la. Esta declaração é feita através do **protótipo** da função. O protótipo da função nada mais é do que o trecho de código que especifica o nome e os parâmetros da função.

No exemplo a seguir a função SOMA é prototipada antes de ser usada e assim pode ser chamada antes de ser definida.

```
#include <stdio.h>

int soma(int x, int y); /* PROTOTIPO da funcao */
                        /* A definicao da funcao esta depois do main() */


/* -------- Programa Principal --------- */
int main()
{
  int N;
  int Resultado;

  printf("\n Entre com N: ");
  scanf("%d",&N);
  Resultado = N*soma(10,10)+20*soma(10,10)+30*soma(20,20);
  printf("\n\n   O Resultadoé %d ",Resultado);
}
```

```
/* -----  Definicao da funcao --------  */
int soma(int x, int y) /* declaracao da funcao */
{
  int A;
  A=x+y;        /* Corpo da funcao */

  return A;       /* comando que indica qual o valor que a funcao deve retornar*/
                  /* AO encontrar esse comando, a funcao e ENCERRADA */
}
```

Portanto, procedimentos e funções simplificam a codificação e permitem uma melhor estruturação do programa, evitando que uma mesma seqüência de comandos seja escrita diversas vezes no corpo da função principal.

Por exemplo, suponha um programa para calcular o número $C(n,p) = \frac{n!}{p!(n-p)!}$ de combinações de n eventos em conjuntos de p eventos, $p \leq n$. Sem o conceito de função, teríamos que repetir três vezes as instruções para o cálculo do fatorial de um número x. Com o conceito de função, precisamos apenas escrever essas instruções uma única vez e substituir x por n, p e *(n-p)* para saber o resultado de cada cálculo fatorial.

```
#include <stdio.h>

/*Prototipo da Funcao */
double fatorial(int x);

/*Definicao da Funcao */
double fatorial(int x)
{
  double fat =1;
  int i;
  for (i=x;i>1;i--)
    fat=fat*i;

  return fat; /* Retorna o valor de fat */
}


/* ---------- Programa Principal ----------*/
int main()
{
  int n, p, C;
  scanf("%d %d",&n, &p);
  if ((p>=0)&&(n>=0)&&(p<=n)){
    /* Chamadas da çãfuno */
    C= (fatorial(n) / (fatorial(p) * fatorial (n-p)));
    printf("%d \n", C);
  }
  return 0;
```

}

3.4 Exercícios

1. Faça uma função que recebe por parâmetro o raio de uma esfera e calcula o seu volume ($v = 4/3 * \pi * R^3$). **Resposta:**

```
float volume_esfera(int raio) {
  return ((float) 4/3 * 3,14 * raio * raio * raio);
}
```

2. Escreva um procedimento 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 e se for P, a sua média ponderada (pesos: 5, 3 e 2). O procedimento deve imprimir a resposta.

   Resposta:

```
void media (float nota1, float nota2, float nota3, char opcao) {
  float media;
  if (opcao=='A')
      media=((nota1+nota2+nota3)/3);
  if (opcao=='P')
      media=((nota1*5+nota2*3+nota3*2)/10);
  printf("Media = %f \n",media);
}
```

3. Faça um procedimento que receba o IMC de uma pessoa e a classifique segundo a tabela abaixo:

```
        Condição   |    IMC em adultos
-------------------------------------------------
 Abaixo do Peso  |      < 18,5
 Peso Normal     |      18,5 <= IMC <= 25
 Acima do Peso   |        25 <  IMC <= 30
 Obeso           |      > 30
```

   Resposta:

```
void classifica(float IMC) {
  if (IMC < 18,5)
     printf("Abaixo do peso. \n");
  else
     if (IMC <= 25)
        printf("Peso normal. \n");
     else
        if (IMC <= 30)
           printf("Acima do Peso. \n");
        else
           printf("Obeso. \n");
}
```

4 Bibliografia Utilizada

Estas aulas foram baseadas nas notas de aula do **prof. Alexandre Falcão**. Disponível em: http://www.dcc.unicamp.br/ afalcao/mc102/notas-aula.pdf

Apostila do **prof. Flávio Keidi Miyazawa**.

Felipe Massia Pereira. MC102 Algoritmos e Programação. Disponível em:http://www.ic.unicamp.br/ massia/mc102/

Márcio Serolli Pinho. Laboratório de Programação I - Material de consulta de Linguagem C. Disponível em: http://www.inf.pucrs.br/ pinho/LaproI/

Fernando Lobo. Programação Imperativa, 2003/04. Disponível em :http://www.adeec.fct.ualg.pt/PI_flobo/

Material do Curso de C da UFMG - Disponível em:http://ead1.eee.ufmg.br/cursos/C/