MC102 - Algoritmos e Progração de Computador

Prof. Alexandre Xavier Falcão

16° Aula: Funções

1 Funções

Vimos que diversos comandos (e.g. scanf(), printf(), strlen(), strcmp(), sin(), cos(), etc) realizam tarefas mais complexas sobre valores de entrada. Esses comandos são denominados **funções**, pois consistem de agrupamentos de instruções (i.e. seqüências de comandos com uma determinada finalidade) assim como a função principal, **main()**, que agrupa as instruções do programa. Uma função, portanto, é uma seqüência de comandos que pode ser executada a partir da função principal (**ou de qualquer outra função**).

As funções simplificam a codificação e permitem uma melhor estruturação do programa, evitando que uma mesma seqüência de comandos seja escrita diversas vezes no corpo (**escopo**) 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 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>

double fatorial(int x); /* protótipo da função */

/* escopo da função */

double fatorial(int x)
{
  double fat=1;
  int i;

  for (i=x; i > 1; i--)
    fat = fat * i;

  return(fat);
}

/* função principal */

int main()
```

```
{
  int n,p,C;

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

A função **fatorial** recebe o valor de uma variável da função principal, armazena em uma variável x do seu escopo, e retorna o cálculo do fatorial de x para o programa principal. As variáveis x, i, e fat declaradas na função fatorial são denominadas **variáveis locais** desta função. Elas só existem na memória enquanto a função fatorial estiver sendo executada (no exemplo, elas são alocadas e desalocadas três vezes na memória), e só podem ser usadas no escopo desta função. A mesma observação é válida com relação às variáveis locais n, p, e C da função principal, porém essas permanecem na memória durante toda a execução do programa.

Um programa pode ter várias funções e uma função pode conter chamadas a outras funções do programa. Cada função deve ser declarada da seguinte forma:

```
tipo funcao(tipo var1, tipo var2,..., tipo varn); /* protótipo */


tipo funcao(tipo var1, tipo var2,..., tipo varn)
{
  /* escopo */

  return (valor);
}
```

O tipo do valor retornado pela função deve ser compatível com o tipo da função. O tipo **void** é usado quando a função não retorna valor. Os valores de entrada e saída de uma função são denominados **parâmetros**. Esses parâmetros podem ser de qualquer tipo válido, incluindo registros, apontadores, cadeias de caracteres, vetores, etc.

2 Parâmetros passados por valor e por referência

No caso da função fatorial, o valor de n na chamada fatorial(n) é passado para uma cópia x da variável n. Qualquer alteração em x não afeta o conteúdo de n no escopo da função principal. Dizemos então que o parâmetro é passado por **valor**. Isto nos permite, por exemplo, simplificar a função para:

```
double fatorial(int x)
{
  double fat=1;
```

```
  while (x > 1){
    fat = fat * x;
    x--;
  }

  return(fat);
}
```

Porém, em várias situações desejamos alterar o conteúdo de uma ou mais variáveis no escopo da função principal. Neste caso, os parâmetros devem ser passados por **referência**. Isto é, a função cria uma cópia do endereço da variável correspondente na função principal em vez de uma cópia do seu conteúdo. Qualquer alteração no conteúdo deste endereço é uma alteração direta no conteúdo da variável da função principal. Por exemplo, o programa acima requer que $p \leq n$. Caso contrário, podemos trocar o conteúdo dessas variáveis.

```
#include <stdio.h>

double  fatorial(int x);
void troca(int *x, int *y); /* x e y são apontadores para endereços de
                               memória que guardam valores do tipo int */

double fatorial(int x)
{
  double fat=1;
  while (x > 1){
    fat = fat * x;
    x--;
  }
  return(fat);
}

void troca(int *x, int *y)
{
  int aux;

  aux = *x;  /* conteúdo de x é atribuído ao conteúdo de aux */
  *x  = *y;  /* conteúdo de y é atribuído ao conteúdo de x */
  *y  = aux; /* conteúdo de aux é atribuído ao conteúdo de y */
}

int main()
{
  int n,p,C;

  scanf("%d %d",&n,&p);
```

```
  if (p > n)
    troca(&p,&n); /* passa os endereços de p e de n */

  if ((p >= 0)&&(n >= 0)){
    C = (int)(fatorial(n)/(fatorial(p)*(fatorial(n-p))));
    printf("%d \n",C);
  }
  return 0;
}
```

3 Exercício

Reescreva programas das aulas anteriores utilizando funções.