MC102 - Algoritmos e Progração de Computador

Prof. Alexandre Xavier Falcão

7° Aula: Comandos de repetição.

1 Comando for

O comando **for** é uma simplificação do comando **while**, onde a inicialização da variável de controle, a expressão lógica envolvendo a variável de controle e a atualização da variável são especificadas no próprio comando. Sua implementação é feita com o comando while, portanto seu comportamento é o mesmo: após a inicialização, a expressão lógica é testada. Se for verdadeira, o bloco de comandos é executado. Após execução, a variável de controle é atualizada, a expressão lógica é verificada, e o processo se repete até que a expressão seja falsa.

```
for (inicialização; expressão; atualização)
{
bloco de comandos
}
```

Por exemplo, um programa para somar n números fica.

```
#include<stdio.h>

int main()
{
  int n,i;
  float soma,num;

  printf("Entre com a quantidade de números a serem somados: ");
  scanf("%d",&n);

  for (i=1, soma = 0.0; i <= n; i=i+1, soma = soma + num) {
    printf("Digite o %do. número:",i);
    scanf("%f",&num);
  }
  printf("O resultado da soma é %f\n",soma);

  return(0);
}
```

Uma observação interessante é que a sintaxe para incrementar/decrementar e multiplicar/dividir valores permite as seguintes variações.

```
soma += num; /* É o mesmo que soma = soma + num; */
prod *= num; /* É o mesmo que prod = prod * num; */
y /= 2;      /* É o mesmo que y = y / 2; */
i++;         /* É o mesmo que i = i + 1; */
i--;         /* É o mesmo que i = i - 1; */
```

Outro exemplo é um programa para calcular o maior entre n números lidos da entrada padrão.

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

int main()
{
  int i,n,num,maior;

  printf("Entre com a quantidade de números: ");
  scanf("%d",&n);

  maior = INT_MIN;
  for(i=1; i <= n; i++) {
    printf("Entre com um inteiro: ");
    scanf("%d",&num);
    if (num > maior)
      maior = num;
  }
  printf("O maior inteiro lido foi %d\n",maior);
  return(0);
}
```

Sabendo que o fatorial de um número inteiro n é $n \times (n-1) \times (n-2) \ldots 1$, faça um programa para calcular o fatorial de um número lido da entrada padrão usando o comando **for** e apenas duas variáveis.

O triângulo de Floyd é formado por n linhas de números consecutivos, onde cada linha contém um número a mais que a linha anterior. Para imprimir o triângulo de Floyd precisamos aninhar um for dentro do outro.

```
#include <stdio.h>

int main()
{
  int l,c,nl,i;

  printf("Entre com o número de linhas: ");
  scanf("%d",&nl);

  i = 1;
  for(l=1; l <= nl; l++) {
```

```
    for(c=1; c <= l; c++) {
      printf("%2d ",i);
      i++;
    }
    printf("\n");
  }

  return(0);
}
```

Outro exemplo que requer dois comandos for aninhados é a impressão de uma tabuada. Faça um programa para imprimir uma tabuada com n linhas e n colunas.

Observe que existe uma relação entre a integral de uma função contínua, sua aproximação pela somatória de uma função discreta, e a implementação da somatória usando o comando for. Por exemplo,

$$\int_{x_{\min}}^{x_{\max}} (ax+b)d_x \quad \approx \quad \sum_{x=x_{\min}+d_x/2}^{x=x_{\max}-d_x/2} (ax+b)d_x, \tag{1}$$

onde $x_{\min} < x_{\max}$, é uma aproximação para a integral da curva, a qual tem maior exatidão para valores menores de $d_x > 0$. Podemos implementar esta integral como:

```
#include<stdio.h>

int main()
{
  float a,b,xmin,xmax,x,dx,integral;

  printf("Entre com os coeficientes a e b da reta y=ax+b: \n");
  scanf("%f %f",&a,&b);

  printf("Entre com o intervalo [xmin,xmax] para cálculo de área: \n");
  scanf("%f %f",&xmin,&xmax);

  printf("Entre com o incremento dx: \n");
  scanf("%f",&dx);

  integral = 0.0;
  for (x=xmin+dx/2.0; x <= xmax-dx/2.0; x=x+dx)
    integral += (a*x+b)*dx;
  printf("integral %f\n",integral);

  return(0);
}
```