Aula 11 - Repetições Encaixadas

Já vimos os seguintes comandos de repetição:

`while`
`for`
`do while`

O formato geral destes comando é:

```
while (comparação)
    {c1; c2; ...; cn;}

for (inicialização; comparação; incremento)
     {c1; c2; ...; cn;}

do {c1; c2; ...; cn;}
while (comparação)
```

Quando há um só comando c1, não são necessárias as chaves { e }.

De uma maneira geral, esses três comandos são muito parecidos. É possível resolver a maior parte dos problemas que envolvam repetições usando qualquer um deles. A escolha do comando a ser usado depende do gosto de cada programador. Um critério que pode ser usado é a clareza na programação propiciada por cada uma das soluções.

Como c1, c2, ..., cn podem ser comandos quaisquer, em particular podem também ser os próprios comandos **`while`**, **`for`** e **`do while`**. Exemplos:

- `for` dentro de `for`:

  ```
  for (...;...;...) {
     ...
     for(...;...;...)
     ...
  }
  ```

- `while dentro de while:`

- `do while dentro de do while:`

E todas as outras combinações possíveis:

- `for` dentro de `while`:

  ```
  while (...) {
  ```

```
        ...
        for (...;...;...) {
        ...
        }
    }
```

- `while` dentro de `for`:
- `do while` dentro de `for`:
- Etc.

Quando um comando de repetição ocorre dentro de outro, dizemos que temos **repetições encaixadas**.

P33) Dado n e m maiores que zero e inteiros, imprimir uma tabela com os valores de x*y para x=1, 2, ..., n e y=1, 2, ... , m, da seguinte forma, supondo n=3 e m=5:

```
      1    2    3    4    5
  1   1    2    3    4    5
  2   2    4    6    8   10
  3   3    6    9   12   15
```

a) Solução com o comando `while`

```
#include <stdio.h>
#include <stdlib.h>
/* dado n e m maiores que zero e inteiros, imprimir a tabela de x*y
   para x=1, 2, ..., n e y=1, 2, ..., m */
int main() {
 int n,   /* valor lido */
     m,   /* valor lido */
     i,   /* contador */
     j;   /* contador */
 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);
 /* ler o m */
 printf("\ndigite o valor de m:");
 scanf("%d", &m);

 /* Variar i de 1 até n e para cada i variar j de 1 até m
    para cada par (i, j), imprimir i*j */

 /* imprimir linha de cabeçalho da tabela, isto é: 1, 2, ..., m */
 printf("     "); /* 5 brancos */
 i = 1;
 while (i <= m) {printf("%5d", i); i = i + 1;}

 /* variar i de 1 ate n */
 i = 1;
```

```
 while (i <= n) {
       printf("\n"); /* pula para a proxima linha */

       /* imprimir o cabeçalho de cada linha, ou seja i */
       printf("%5d", i);
       /* para cada i variar j de 1 ate m e imprima i * j */
       j = 1;
       while (j <= m) {printf("%5d", i*j); j = j + 1;}

       i = i + 1;  /* próxima linha */
 }
 system("pause");return 0;
}
```

b) Solução com o comando `for`

```
#include <stdio.h>
#include <stdlib.h>
/* Dado n e m maiores que zero e inteiros, imprimir a tabela de x*y
   para x=1, 2, ..., n e y=1, 2, ..., m */

int main() {
 int n,   /* valor lido */
     m,   /* valor lido */
     i,   /* contador */
     j;   /* contador */

 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);
 /* ler o m */
 printf("\ndigite o valor de m:");
 scanf("%d", &m);

 /* Variar i de 1 até n e para cada i variar j de 1 até m
    para cada par (i, j), imprimir i*j */

 /* imprimir linha de cabecalho da tabela, isto é: 1, 2, ..., m */
 printf("     "); /* 5 brancos */
 for (i = 1; i <= m; i = i + 1) printf("%5d", i);

 /* variar i de 1 até n */
 for (i = 1; i <= n; i = i + 1) {
       printf("\n"); /* pula para a proxima linha */
       /* imprimir o cabeçalho de cada linha, ou seja i */
       printf("%5d", i);
       /* para cada i variar j de 1 ate m e imprima i * j */
       for (j = 1; j <= m; j = j + 1) printf("%5d", i*j);
 }
 system("pause"); return 0;
}
```

Conforme já falamos anteriormente, quando a repetição é controlada por um contador como é o caso do problema acima, fica melhor usar o for.

P34) Dado n maior que zero e inteiro, imprimir uma tabela com os valores de x*y para x=1, 2, ..., n e y=1, 2, ... , n. Como a tabela será simétrica neste caso, imprimir apenas o triângulo inferior, da seguinte forma:

	1	**2**	**3**	**4**	**5**
1	1				
2	2	4			
3	3	6	9		
4	4	8	12	16	
5	5	10	15	20	25

```
#include <stdio.h>
#include <stdlib.h>
/* Dado n maior que zero e inteiro, imprimir a tabela de x*y
   para x=1, 2, ..., n e y=1, 2, ..., n
   so o triangulo inferior */

int main() {
 int n,   /* valor lido */
     i,   /* contador */
     j;   /* contador */
 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);

 /* Variar i de 1 ate n e para cada i variar j de 1 ate i
    para cada par (i, j), imprimir i*j */

 /* imprimir linha de cabecalho da tabela, isto e: 1, 2, ..., n */
 printf("     "); /* 5 brancos */
 for (i = 1; i <= n; i = i + 1) printf("%5d", i);

 /* variar i de 1 ate n */
 for (i = 1; i <= n; i = i + 1) {
       printf("\n"); /* pula para a proxima linha */

       /* imprimir o cabeçalho de cada linha, ou seja i */
       printf("%5d", i);
       /* para cada i variar j de 1 ate i e imprima i * j */
       for (j = 1; j <= i; j = j + 1) printf("%5d", i*j);
 }
 system("pause"); return 0;
}
```

P35) Idem, imprimindo apenas o triangulo superior, da seguinte forma:

	1	**2**	**3**	**4**	**5**
1	1	2	3	4	5
2		4	6	8	10
3			9	12	15
4				16	20
5					25

P36) Calcular os valores da função x^2-y^2+xy para x e y = -10, -9, ..., 9, 10. Imprimir da seguinte forma:

```
     x      y      x.y
   -10    -10      100
   -10     -9      109
     :      :        :
    10      9       90
    10     10      100
```

P37) Dado n > 0 inteiro, calcular o valor de x.y.z para x, y, z = 0, 1, 2, ... , n

P38) Dado n > 0 inteiro, imprimir o gráfico da função x^2+x+1 para $x=-n$ até n. Imprimir o gráfico usando como ordenadas o eixo horizontal e como abscissas o eixo vertical, da seguinte forma:

```
-5.....................*
-4............*
-3.......*
 :
 :
 5.....................*
```

```
#include <stdio.h>
#include <stdio.h>
/* Dado n maior que zero e inteiro, imprimir o gráfico da função
   para x**2 + x + 1 para x = -n ate n
   eixo das ordenadas na horizontal e
   eixo das abscissas na vertical */

int main() {
 int n,   /* valor lido */
     i,   /* contador */
     j;   /* contador */

 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);

 /* imprima o gráfico, variando i de -n até n */

 for (i = -n; i <= n; i = i + 1) {

     /* imprimir valor de i */
     printf("\n%5d", i);
     /* imprimir tantos pontos quantos for o valor da função */
     for (j = 1; j <= i*i + i + 1; j = j + 1) printf(".");
```

```
     /* imprime asterisco */
     printf("*");
 }
 system(“pause”); return 0;
}
```

P40) Dado n >= 0 calcular os fatoriais dos números de 0 a n

```
#include <stdio.h>
#include <stdlib.h>
/* dado n>=0 calcular os fatoriais de 0 a n */
int main() {
 int n,    /* numero dado */
     fat,  /* fatorial */
     i,j;    /* contadores */
 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);
 /* calcula os fatoriais de 0 a n */
 for (i = 0; i <= n; i++) {
       /* calcula fatorial de i */
       fat = 1; /* inicia o valor de fat */
       /* multiplique por todos os números até n */
       for (j = 1; j <= i; j++) fat = fat * j;
       /* imprima o resultado de fatorial de i */
       printf("\nfatorial de %10d - %10d", i, fat);
 }
 system(“pause”); return 0;
}
```

P41) Dado n, calcular todos os primos entre 2 e n. Lembre-se que n é primo se não tem divisores entre 2 e raiz de n.

```
#include <stdio.h>
#include <stdlib.h>
 /* dado n>=0 calcular todos os primos de 2 a n */
 int main() {
 int n,    /* numero dado */
     d,    /* divisores */
     i;    /* contadores */
 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);
 /* verifica todos os números de 2 a n se são primos */
 for (i = 2; i <= n; i++) {
         /* verifica se i tem divisores entre 2 e raiz de n */
         for (d = 2; d*d <= i; d++)
               if (i%d == 0) d = i;  /* força a saída */
         /* verifica se encontrou algum divisor */
         /* se não saiu pela saida forçada então é primo */
         if (d != i+1) printf("\n%5d *** primo", i);
 }
 system("pause"); return 0;
}
```

Na solução acima, forçamos a saída do for, assim que encontramos um divisor.

Não é uma solução muito elegante, pois o que queremos é abandonar o for e para isso alteramos o valor da variável contadora. Uma forma melhor seria usar um comando específico para sair do for. Veja abaixo.

Os comando break e continue

O comando break é usado para sair forçadamente dos comandos `for`, `while`, `do while` e `switch` (que ainda não vimos). Sua ação e abandonar o laço mais interno. Tal laço está associado a um dos comandos anteriores.

Por exemplo, o P41 anterior ficaria:

```
#include <stdio.h>
#include <stdlib.h>
/* dado n>=0 calcular todos os primos de 2 a n */
int main() {
 int n,   /* numero dado */
     d,   /* divisores */
     i; /* contadores */
 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);
 /* verifica todos os números de 2 a n se são primos */
 for (i = 2; i <= n; i++) {
       /* verifica se i tem divisores entre 2 e raiz de n */
      for (d = 2; d*d <= i; d++)
               if (i%d == 0) break;  /* força a saida */
       /* verifica se encontrou algum divisor */
       /* se não saiu pela saida forçada então é primo */
      if (d*d > i) printf("\n%5d *** primo", i);
 }
 system("pause"); return 0;
}
```

O comando **`continue`** não sai do laço e sim faz com que a próxima repetição do laço, seja iniciada, ou de forma equivalente, desvia para o final do último comando do laço.

Outra versão do problema anterior:

```
#include <stdio.h>
#include <stdlib.h>
/* dado n>=0 calcular todos os primos de 2 a n */
int main() {
 int n,   /* numero dado */
     d,   /* divisores */
     i;   /* contadores */
 /* ler o n */
 printf("digite o valor de n:");
 scanf("%d", &n);
 /* verifica todos os números de 2 a n se são primos */
 for (i = 2; i <= n; i++) {
     /* verifica se i tem divisores entre 2 e raiz de n */
     for (d = 2; d*d <= i; d++)
               if (i%d != 0) continue;
               else break;  /* força a saida */
```

```
      /* verifica se encontrou algum divisor */
      /* se não saiu pela saida forçada então é primo */
      if (d*d > i) printf("\n%5d *** primo", i);
 }
 system("pause"); return 0;
}
```

P41a) Dado N, calcular a soma dos primos menores ou iguais a N.

P41b) Dado N, decompor N em seus fatores primos, ou seja imprimir a tabela:

Primo	Expoente
2	x1
3	x2
5	x3
7	x4
11	x5
...	...