Estrutura de Repetição em C

Podemos classificar uma estrutura de repetição como:

- Repetição (laço) contada: quando se sabe previamente quantas vezes o comando composto no interior da construção será executado.

- Repetição condicional: quando não se sabe previamente o número de vezes que os comandos no interior do laço serão repetidos, pois este número depende de uma condição sujeita à modificação pelas instruções do interior do laço. Este tipo de repetição pode ser com interrupção no início ou no final.

Repetição Contada (for)

```
for (i = valor_inicial; condição; incremento ou decremento de i)
    comando ou bloco de comandos;
```

A primeira parte (i = valor_inicial) atribui um valor inicial à variável de controle i, que tem por objetivo controlar o número necessário de repetições.

A segunda parte (condição) corresponde a uma expressão relacional que, quando assumir valor falso, determinará o fim da repetição.

A terceira parte (incremento ou decremento de i) é responsável por alterar o valor da variável i com o objetivo de, em algum momento, fazer com que a condição assuma valor falso.

Deve-se lembrar que, se houver uma sequência de comandos a ser executada, esses comandos devem estar delimitados por { }.

Exemplo:

```
#include <iostream>

using namespace std;

main() {
```

```
        int contador =1;
        for (contador=1; contador<=10; contador++) {
                cout << contador << " ";
        }
}
```

Saída:

1 2 3 4 5 6 7 8 9 10

Repetição Com Interrupção No Início (while)

É uma estrutura de repetição utilizada quando não sabemos quantas repetições serão necessárias, isto é, a interrupção do laço depende de uma (ou mais) condição(ões). Neste caso, os comandos serão repetidos enquanto a condição for verdadeira.

Nesta estrutura, a verificação da condição é feita antes da execução dos comandos. Se logo no início a condição for falsa esta estrutura não será executada.

```
while (condição)
        comando ou bloco de comandos;
```

Lembre-se que pode existir mais de uma condição de controle. Além disso, se houver uma sequência de comandos a serem executados, tais comandos devem ser delimitados ({ }).

Para a execução terminar é necessário que a sequência de comandos altere a condição de verdadeira para falsa, senão o programa ficar num laço infinito (loop).

Exemplo:

```
#include <iostream>

using namespace std;

main() {

        int contador =1;

        while (contador <= 10) {

                cout << contador << " ";

                contador ++;

        }

}
```

Saída:

```
1 2 3 4 5 6 7 8 9 10
```

Repetição Com Interrupção No Final (do-while)

Este comando trabalha da mesma maneira que while, exceto pelo fato de que o valor da condição é testado após a execução da sequência de comandos, ao invés de ser testado antes. Isto implica que pelo menos uma vez esses comandos serão executados. Cada vez que o comando é executado a condição é testada. Se for verdadeira o processo é repetido, caso contrário é interrompido.

Este comando é útil quando a sequência de comandos deve ser executada uma vez antes que a condição seja avaliada.

```
do {

        comando ou bloco de comandos;

} while (condição);
```

Exemplo:

```
#include <iostream>

using namespace std;

main() {

        int contador =1;

        do {

                cout << contador << " ";

                contador ++;

        } while (contador <= 10);

}
```

Saída:

```
1 2 3 4 5 6 7 8 9 10
```

Interrupções com break e continue

A linguagem em C/C++ oferece ainda duas formas para a interrupção antecipada de um determinado laço.

O comando break tem duas utilizações: para terminar um case em um comando switch, ou para forçar o término imediato de um laço, evitando o teste condicional normal do laço. Quando utilizado dentro de um laço, interrompe e termina a execução do mesmo imediatamente. A execução prossegue com os comandos subsequentes ao bloco.

```
#include <iostream>

using namespace std;

main() {

        int i;
```

```
        for (i=0; i<10; i++) {
                if (i==5) break;
                cout << i << " ";
        }
        cout << "Fim" << endl;
}
```

A saída deste programa será: **0 1 2 3 4 Fim**, pois quando i tiver o valor 5, o laço será interrompido e finalizado pelo comando break, passando o controle para o próximo comando após o laço, no caso uma chamada final de cout.

O comando continue também interrompe a execução dos comandos de um laço. A diferença básica em relação ao comando break é que o laço não é automaticamente finalizado. O comando continue força que ocorra a próxima iteração do laço, pulando qualquer comando intermediário. Para o laço for, continue faz com que o teste condicional e a porção de incremento do laço sejam executados. Assim, o programa a seguir gera a saída: **0 1 2 3 4 6 7 8 9 Fim**.

```
#include <iostream>
using namespace std;
main() {
        int i;
        for (i=0; i<10; i++) {
                if (i==5) continue;
                cout << i << " ";
        }
        cout << "Fim" << endl;
}
```

Devemos ter cuidado com a utilização do comando continue nos laços while, pois nesses laços o controle do programa passa para o teste condicional. O programa a seguir é um programa **INCORRETO**, pois o laço criado não tem fim. Isto porque a variável i nunca terá valor

superior a 5, o teste será sempre verdadeiro. O que ocorre é que o comando continue “pula” os demais comandos do laço quando i vale 5, inclusive o comando que incrementa a variável i.

```
/* INCORRETO */
#include <iostream>
using namespace std;
main() {
        int i = 0;
        while (i<10) {
                if (i==5) continue;
                cout << i << " ";
                i++;
        }
        cout << "Fim" << endl;
}
/* INCORRETO */
```

Exercícios

01) Construir um programa para calcular o fatorial de um número N fornecido pelo usuário.

02) Faça um programa que leia um número indeterminado de linhas contendo cada uma a idade de um indivíduo. A última linha, que não entrará nos cálculos, contém o valor da idade menor ou igual a zero (flag). Calcule e mostre a idade média desse grupo de indivíduos.

03) Escreva um algoritmo para calcular a soma de dez números quaisquer fornecidos pelo usuário.

04) Escreva um programa que encontre o menor número inteiro positivo N que aceite as seguintes condições: N/3 dá resto 2, N/5 dá resto 3 e N/7 dá resto 4.

05) Dado um número N>0, construir um programa que mostre os N primeiros números ímpares.

06) Faça um programa que leia um valor x, calcule e mostre a série abaixo considerando os 10 primeiros termos.

$$S = \frac{x}{1} - \frac{x}{2} + \frac{x}{3} - \frac{x}{4} + ...$$

07) Escreva um programa que leia um conjunto de 100 números inteiros positivos e determine o maior deles.