Unidade 15: Outras Estruturas de Repetição

Prof. Daniel Caetano

Objetivo: Realizando decisões de repetição no código de programação.

Bibliografia: ASCENCIO, 2007; MEDINA, 2006; SILVA, 2010; SILVA, 2006.

INTRODUÇÃO

Na aula anterior vimos como fazer repetições com o comando **while**. Nesta aula, veremos uma forma diferente de escrever a estrutura de repetição, que simplifica sua compreensão em casos comuns.

1. CONTAGEM COM O WHILE

Como vimos na aula passada, podeemos fazer um programa que realize a contagem de um número incial até um número final. Por exemplo: se quisermos que nosso programa conte de 3 a 17, basta escrever:

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

     int N;

     N = 3;
     while ( N <= 17 ) {
        cout << N << endl;
        N = N + 1;
        }

getchar();
}
```

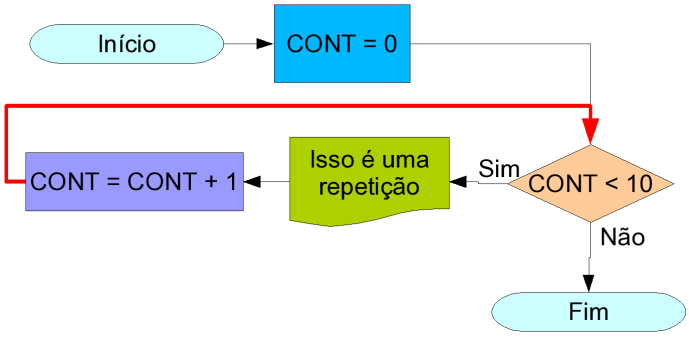
Esse programa inicializa o valor de N com 3 (o valor inicial) e, enquanto o valor de N for menor ou igual a 17, imprime N e, depois disso, soma 1 ao valor de N.

Simples, mas facilmente poderemos esquece de suas partes partes.

2. ESTRUTURA DE REPETIÇÃO DE CONTAGEM: FOR

Muitas vezes precisamos construir uma instrução de repetição que simplesmente sirva para executar alguma instrução um determinado número de vezes. Como vimos, a estrutura **while** é um pouco desajeitada. Vamos usar mais uma vez o **while**, agora para imprimir 10 vezes o texto "isso é uma repetição".

O fluxograma e código com while respectivos seriam os indicados a seguir.




```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

     int CONT;

     CONT = 0;
     while ( CONT < 10 ) {
        cout << "Isso é uma repetição" << endl;
        CONT = CONT + 1;
        }

getchar();
}
```

Observe a existência de QUATRO elementos fundamentais em uma contagem:

INICIALIZAÇÃO: Ajusta o valor inicial do contador
DECISÃO DE REPETIÇÃO: Verifica se bloco precisa ser repetido
BLOCO: Código a ser repetido
ATUALIZAÇÃO: Código que atualiza o contador (deve estar dentro do bloco!)

Sempre que um programador quiser repetir um bloco um certo número de vezes, ele precisará indicar essas quatro partes. Bem, mas os programadores não gostam de escrever muito... então foi inventada uma instrução que especifica a **INICIALIZAÇÃO**, a **DECISÃO DE REPETIÇÃO** e a **ATUALIZAÇÃO**, para que em seguida venha apenas o **BLOCO** a ser repetido. Essa instrução chama-se **for** (traduz-se como "para"). Observe como o código acima fica simplificado usando o **for**.

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

     int CONT;

     for (CONT = 0; CONT < 10; CONT = CONT + 1) {
        cout << "Isso é uma repetição" << endl;
        }

getchar();
}
```

E estrutura da instrução **for** é a seguinte:

```
for ( inicialização; condição de repetição; atualização ) {
  código a repetir enquanto a proposição for verdadeira
  }
```

Observe que o **for não acrescenta nenhuma funcionalidade**, é apenas um jeito mais cômodo de escrever o while anteriormente apresentado!

Exemplo:

```
for ( X=0; X < 7; X = X + 2 ) {
  cout << X << endl;
  }
```

Essa instrução pode ser lida assim:

Faça, a paritr de X = 0, enquanto X < 7 e contando de 2 em 2, a impressão de X.

3. EXERCÍCIO

a) Faça um programa que apresente os números de 50 a 75.

b) Modifique o programa do item (a) para que ele conte de 2 em 2.

c) Modifique agora o programa do item (b) para que ele imprima apenas os números divisíveis por 5.

SOLUÇÃO A

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

     int C;

     for ( C = 50; C <= 75; C = C + 1) {
        cout << C << endl;
        }

getchar();
}
```

SOLUÇÃO B

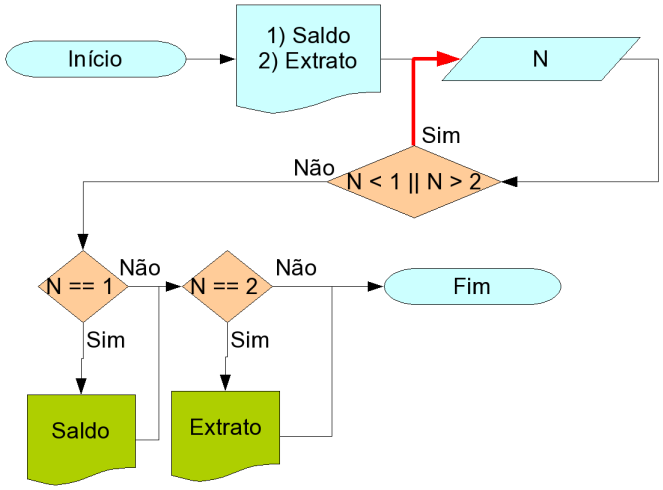
```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

     int C;

     for ( C = 50; C <= 75; C = C + 2) {
        cout << C << endl;
        }

getchar();
}
```

SOLUÇÃO C

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

     int C;

     for ( C = 50; C <= 75; C = C + 2) {
        if ( C%5 == 0 ) {
            cout << C << endl;
            }
        }

getchar();
}
```

4. ESTRUTURA DE REPETIÇÃO "PELO MENOS UMA VEZ": DO~WHILE

Vez ou outra precisamos criar uma estrutura **while** que seja executada pelo menos uma vez. Por exemplo: para escolher a opção de um menu, ignorando os valores inválidos:



Observe que, neste caso, a decisão é feita **após** o bloco. A forma de fazer isso na linguagem C/C++ é através da instrução do ~ while, que tem o seguinte formato:

do {
código a repetir enquanto a proposição for verdadeira
} while (proposição_lógica);

Isso, no código, pode ser implementado como se segue:

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {
     int N;

     cout << "1- Saldo" << endl;
     cout << "2- Extrato" << endl;
     do {
        cin >> N;
        } while (N < 1 || N > 2);

     if (N == 1) {
        cout << "Saldo!" << endl;
        }
     if (N == 2) {
        cout << "Extrato!" << endl;
        }

getchar();
}
```

5. EXERCÍCIO

Extenda o programa de menu apresentado para que ele contenha as opções:

1- Saldo
2- Extrato
3- Saque
4- Depósito

OBS: Quando uma opção for selecionada, o texto adequado deve ser impresso!

SOLUÇÃO

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {
     int N;

     cout << "1- Saldo" << endl;
     cout << "2- Extrato" << endl;
     cout << "3- Saque" << endl;
     cout << "4- Depósito" << endl;
     do {
        cin >> N;
        } while (N < 1 || N > 4);

     if (N == 1) {
        cout << "Saldo!" << endl;
        }
     if (N == 2) {
        cout << "Extrato!" << endl;
        }
     if (N == 3) {
        cout << "Saque!" << endl;
        }
     if (N == 4) {
        cout << "Depósito!" << endl;
        }

getchar();
}
```

6. EXERCÍCIO

A) Analise os dois códigos a seguir e diga qual dos dois usa **while** e qual usa **do~while**:

Código 1
a) Leia um número N;
b) N = N * 2;
c) Se N for menor que 32, volta para o passo (b);
d) Imprima N.

Código 2
a) Leia um número N;
b) Enquanto N for menor que 32, repita (c)
c) N = N*2;
d) Imprima N.

B) Implemente ambos os códigos.

C) Execute ambos para os seguintes números de entrada: 0, 20, 40. Os resultados foram sempre iguais? Por quê?

SOLUÇÃO A

O primeiro é com **do~while** e o segundo com **while**.

SOLUÇÃO B

Código 1

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

    float N;

    cout << "Digite um número: ";
    cin >> N;

    do {
       N = N * 2;
       } while (N < 32);

    cout << "Resultado: " << N << endl;

getchar();
}
```

Código 2

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

    float N;

    cout << "Digite um número: ";
    cin >> N;

    while (N < 32) {
       N = N * 2;
       }

    cout << "Resultado: " << N << endl;

getchar();
}
```

SOLUÇÃO C

Valor de Entrada:	Resultado Código 1	Resultado Código 2
0	Travou	Travou
20	40	40
40	80	40

A diferença é que no código 1 sempre ocorrerá pelo menos uma multiplicação, dado que o do~while obrigatoriamente executa o bloco uma vez; no caso do código 2, como a verificação é feita antes da execução do bloco, caso o valor digitado seja maior que 32, nenhuma multiplicação será executada.