Linguagem C++

Estruturas de controle

Parte II – Estruturas de repetição

Prof. Bruno E. G. Gomes
IFRN

- Permitem o controle da sequência de execução de um programa

- Em C temos:
  - Estruturas de seleção (ou decisão)
  - **Estruturas de repetição (ou iteração)**
  - Instruções para interrupção e desvio

- Repetem a execução de um bloco de código
  - Expressão condicional controla o número de repetições

- Estruturas de C++:
  - **while** (enquanto)
  - **for** (para)
  - **do-while** (faça-enquanto ou “repita enquanto”)

Estrutura de Repetição *While* (enquanto)

Sintaxe: **while (**<*condição*>**) {**

<*instruções*>

}

<*condição*>: Expressão que retorne *verdadeiro* (***true*** ou diferente de 0) ou *falso* (***false*** ou igual a 0)

Funcionamento:

1. <*condição*> *é* ***verdadeira***:
   <instruções> são executadas e a <condição> é testada novamente

2. <*condição*> *é* ***falsa***:
   sai do ***while***, indo para o próximo comando após "**}**"

Condição de Teste do While

- É qualquer expressão que retorne *verdadeiro* ou *falso*
  - Deve ser verdadeira enquanto se quer continuar a repetição
  - Deve tornar-se falsa em algum momento

- É construída a partir de uma ou mais *variáveis auxiliares*
  - Variável tem que ser inicializada antes do primeiro teste do *while*
  - Variável deve ser modificada dentro do *while* (entre { }) para que o teste seja falso em algum momento

- Variável de controle geralmente é modificada por:
  - Um número que cresce ou decresce;
  - Um valor booleano que muda de verdadeiro para falso ou de falso para verdadeiro;
  - Um caractere ou texto que deve ser digitado

Exemplo: imprime os números de 1 a 100

```
#include <iostream>
using namespace std;

int main() {

    int n = 1; //variável auxiliar para controlar a repetição

    while ( n <= 1000 ) { //ou i < 999
        cout << n; //imprime cada número (valor de i)
        n = n + 1; //aumenta o valor de i de 1 em 1 até 1000
    }

    return 0;
}
```

Exemplo: repete 10 vezes uma frase

```
#include <iostream>
using namespace std;

int main()
{
    int i = 0;

    //repete 10 vezes a frase no corpo do while
    while ( i < 10 ) {
        i += 1;
        cout << i << " - ";
        cout << "Bem vindos novamente às aulas de C++\n";
    }

    return 0;
}
```

Exemplo: acumula 10 vezes o produto x*x

```
#include <iostream>
using std::cout;
using std::endl;

int main()
{
   int y, x = 1, total = 0;

   while ( x <= 10 ) {
      y = x * x;
      cout << y << endl;
      total += y;
      ++x;
   }

   cout << "Total: " << total << endl;
   return 0;
}
```

Exemplo: uso de '0' para *falso* e *não-zero* para *verdadeiro*

```
int i = 1;

while ( i ) {
    int x;

    cout << "Digite um valor inteiro: ";
    cin >> x;

    if (x % 2 == 0) {
        i = 0;
    }
}
```

Exemplo anterior modificado: variável *bool*

```
//Sai da repetição quando o número digitado for par. Uso de uma variável auxiliar booleana.
   bool ímpar = true;

   while ( ímpar ) {
      int x;

      cout << "Digite um valor inteiro: ";
      cin >> x;

      if (x % 2 == 0) {
         ímpar = true;
      }
}
```

Estrutura de Repetição *Do-While* (Repita)

- Sintaxe:

do {

<instruções>

} **while** (<*condição*>)**;**

- Testa a condição apenas ao final. Dessa forma, as <instruções> são executado ao menos uma vez;
- Assim como o **while,** repete a execução dos comandos enquanto a condição for **verdadeira**;
- O que pode ser feito com o **while**, pode ser feito com o **do-while**

Exemplo: leitura apenas de números não-negativos

```
#include <iostream>
using namespace std;

int main () {
    int num; //declaração da variável auxiliar

    do {
        cout << "Digite um número não-negativo: ";
        cin >> num;     //leitura do número
    } while (num < 0);    //teste da repetição

    return 0;
}
```

Exemplo

```
#include <iostream> #include <cstdlib> #include <ctime>
using namespace std;

int main() {
    /*inicializa o gerador de números aleatórios com o valor da "hora" atual*/
    srand(time(0));
    //gera um número aleatório no intervalo de 1 a 10
    int num_gen = (rand() % 10) + 1;
    int num;    //número fornecido pelo usuário

    do {
        cout << "Advinhe o número (1 a 10): ";
        cin >> num;
        if (num_gen < num)
            cout << "O número secreto é menor que " << num << endl;
        else if (num_gen > num)
            cout <<  "O número secreto é maior que " << num << endl;
     } while (num_gen != num);

       cout << "Parabéns, você advinhou o número!";
       return 0;
}
```

Sintaxe:

for (<inicialização>**;** <condição>**;** <incremento>) **{**
    <instruções>
}

inicialização: valor inicial da variável que controla a repetição

condição: Expressão que determina quando a repetição acaba. <comandos> são repetidos enquanto a condição for verdadeira

incremento: variação da variável de controle *após* cada repetição

Estrutura *For* - Funcionamento

```
1. inicialização;

2. if (<condição>) {
3.     <instruções>;
4.     <incremento>;
5.     "volte para 2"
}
```

1. A inicialização é feita antes da primeira repetição;
2. A condição é testada. Se for **falsa**, a repetição termina;
3. Se for **verdadeira**, as instruções são executados;
4. O incremento da variável de controle é feito;
5. Volta para o *passo 2* (teste).

Exemplo

```
//imprime os números de 1 até 10
for (x = 1; x <= 10; x++) {
    cout << x;
}
```

- **x** é a variável de controle da repetição
- **x** é inicializada com 1 (*x = 1*) antes da primeira repetição
- A repetição é feita enquanto *x <= 10*
- **x** é incrementada por 1 após a execução de cada impressão do seu valor.

```
1. for (x = 100; x != 65; x -= 5) {
2.     z = x*x;
3.     cout << “O quadrado de ” << x << “é” << z;
4. }
```

- A linha 2 e a linha 3 serão executadas enquanto o valor de x for diferente de 65.

Variações do laço for

- Podem existir expressões de inicialização e incremento para mais de uma variável;
- Há apenas uma expressão de condição

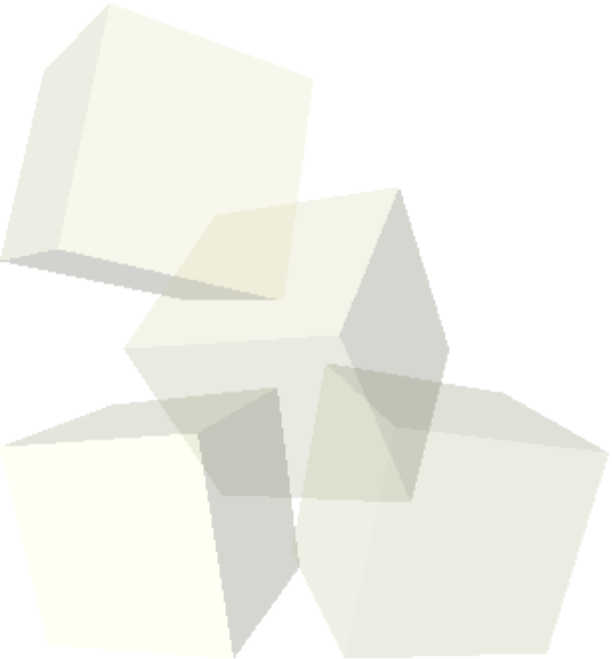
```
for (i = 0, j = 10; i < 10; i++, j--) {
    cout << i + j;
}
```

Inicialização e incremento fora do cabeçalho do **for**

```
int x;
x = 1;
for ( ;x < 10; ) {
    cout << x;

    ++x;

}
```



Variações do for: “*loop* infinito”

- O **for** será executado para sempre quando a condição de teste for omitida.
  - O mesmo que fazer:
    - while (true) { }

    - do {
      } while (true);

- No exemplo do slide seguinte, a saída do laço é feita com uma instrução de interrupção (break).

Exemplo de *loop* infinito

```
#include <iostream>
//biblioteca onde está definida a função getchar()
#include <cstdio>

using namespace std;

int main()
{
    char ch = '0';
    for (;;) {
        ch = getchar();

        if (ch == 'a') break; //força a saída do laço
    }

    return 0;
}
```

Variações do for: for sem corpo

- É possível criar laços for que não tenham corpo.
- Podem ser utilizados, por exemplo, para se criar um “atraso de tempo” no programa, como no exemplo abaixo:

```
for (t = 0; t < 100000; t++);
```

O Comando de interrupção **break**

- Utilizado para forçar a saída de uma estrutura de controle
- Exemplo:

```
int t = 0;
while (t < 100) {
    cout << t;
    if (t == 10) { break; }
    t++;
}
```

  - Escreve os números de 1 até 10. Quando t == 10, a condição do **if** é verdadeira e o comando **break** é executado**,** o que faz o controle do programa ir para depois da estrutura **while** (sai da estrutura).

O comando de interrupção **continue**

- Em vez de forçar a saída da estrutura, como o **break**, o **continue** sai apenas da repetição atual.
  - Ou seja, pula todos os comandos depois dele, mas a repetição ainda continua a executar.

- Exemplo (Imprime os números de 0 a 99, com exceção do número 10.)

```
for (t = 0; t < 100; t++) {
        if (t == 10) {
            continue;
        }

        cout << t;
}
```