Estruturas de Dados I

Introdução

Prof. Bruno Azevedo

Instituto Federal de São Paulo

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Catanduva

A Declaração while

- A declaração 'if' oferece um desvio condicional. Mas ele ocorre apenas uma vez.
- Podemos querer repetir um conjunto de instruções múltiplas vezes, dada uma certa condição, ou condições.
- A declaração 'while' funciona do mesmo modo que a declaração 'if', mas permite a repetição de um conjunto de instruções enquanto as condições se mantiverem satisfeitas.
- Ou seja, o while manterá o fluxo de nosso programa dentro de um **desvio condicional repetitivo**, ou **laço**, até que as condições não sejam mais verdadeiras.
- A sintaxe da declaração while é a seguinte:

```
while (condicao) {
   // Código a ser executado enquanto a condição se mantiver verdadeira
}
```

- Quando todas as instruções dentro do bloco de código do while for executado, a condição será novamente testada.
- Se a condição continuar verdadeira, o bloco será executado novamente.
- Caso contrário, o bloco será ignorado e o programa continuará a partir da próxima linha após o bloco 'while'.

A Declaração while

- Assim como no 'if', podemos também conectar condições usando os operadores lógicos.
- Vamos ver um exemplo prático de um uso de 'while'.

```
#include <iostream>
using namespace std;
int main() {
    int numero, soma = 0, i = 1;
    cout << "Digite um número inteiro positivo:\n";
    cin >> numero;
    if (numero <= 0) {
        cout << "O número deve ser inteiro positivo.\n";
        return 1; // Encerra o programa com código de erro
    }
    while (i <= numero) {
        soma += i;
        i++;
    }
    cout << "A soma dos números de 1 a " << numero
         << " é: " << soma << endl;
    return 0;
}
```

goto, continue, e break

- goto: pula para um rótulo inserido pelo programador.
- continue: pula o restante do corpo do laço e inicia a próxima iteração.
- break: sai da estrutura antes que a condição normal de término seja satisfeita.
- **Evitem usar!**
- Podem resultar em códigos confusos, desorganizados, difíceis de entender e de depurar.
- Não é difícil inserir bugs com o uso dessas instruções.
- Existe uma exceção para o uso do break, que veremos futuramente.

A Declaração do. . .while

- Um 'while' que executa uma vez o bloco de código antes de verificar a condição.
- Ou seja, garante que o bloco de código seja executado pelo menos uma vez.
- A sintaxe da declaração do. . .while é a seguinte:

```
do {
   // Código a ser executado
} while (condição);
```

- Vamos ver um exemplo.

```
#include <iostream>
using namespace std;
int main() {
    int numero;
    do {
        cout << "Digite um número positivo:\n";
        cin >> numero;
    } while (numero <= 0);
    cout << "Você digitou o número positivo: " << numero << endl;
    return 0;
}
```

A Declaração do. . .while

- Lembram do exemplo do 'while'?
- Vocês podem utilizar o do. . .while para garantir que o usuário digite um número positivo sem que seja necessário sair do programa.

Exemplo Original.

```
#include <iostream>
using namespace std;
int main() {
    int numero, soma = 0, i = 1;
    cout << "Digite um número inteiro positivo:\n";
    cin >> numero;
    if (numero <= 0) {
        cout << "O número deve ser inteiro positivo.\n";
        return 1; // Encerra o programa com código de erro
    }
    while (i <= numero) {
        soma += i;
        i++;
    }
    cout << "A soma dos números de 1 a "
         << numero << " é: " << soma << endl;
    return 0;
}
```

Utilizando do. . .while.

```
#include <iostream>
using namespace std;
int main() {
    int numero, soma = 0, i = 1;
    do {
        cout << "Digite um número positivo: ";
        cin >> numero;
    } while (numero <= 0);
    while (i <= numero) {
        soma += i;
        i++;
    }
    cout << "A soma dos números de 1 a "
         << numero << " é: " << soma << endl;
    return 0;
}
```

A Declaração for

- Quando usando laços while frequentemente nos encontramos:
  - Configurando uma condição inicial.
  - Testando se a condição é verdadeira
  - Incrementando, ou modificando de alguma forma, um contador dentro da condição.
- Neste caso, podemos utilizar a declaração for, que já possui estes elementos integrados em sua sintaxe.
- A sintaxe da declaração for é a seguinte:

```
for(inicialização;condição;atualização)
   // Código a ser executado enquanto a condição se mantiver verdadeira
}
```

- Um exemplo de uso da declaração for.

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 1; i <= 5; i++)
        cout << i << endl;
    return 0;
}
```

A Declaração for

- A variável de controle não precisa ser do tipo inteiro. Por exemplo:

```
#include <iostream>
using namespace std;
int main() {
    for(double i = 0.5; i <= 3.0; i += 0.5)
        cout << i << endl;
    return 0;
}
```

- E podemos usar múltiplas variáveis de controle. Por exemplo:

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 0, j = 10; i < 5 && j > 5; i++, j -= 2)
       cout << i << " e " << j << "\t";
    return 0;
}
```

- Sem testar, alguém pode me dizer o que será exibido com a execução do código acima?

A Declaração for

- A variável de controle não precisa ser do tipo inteiro. Por exemplo:

```
#include <iostream>
using namespace std;
int main() {
    for(double i = 0.5; i <= 3.0; i += 0.5)
        cout << i << endl;
    return 0;
}
```

- E podemos usar múltiplas variáveis de controle. Por exemplo:

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 0, j = 10; i < 5 && j > 5; i++, j -= 2)
       cout << i << " e " << j << "\t";
    return 0;
}
```

- Sem testar, alguém pode me dizer o que será exibido com a execução do código acima? 0 e 10 1 e 8 2 e 6.

Começando a Programar em C++

Exercícios (7).

⇒ Laços

- Escreva um programa que calcule o fatorial de um número inteiro não-negativo.
- Escreva um programa que solicite ao usuário um número positivo e exiba a sequência de Fibonacci até o número inserido pelo usuário. A sequência de Fibonacci é aquela em que cada número é a soma dos dois números anteriores na sequência: 0, 1, 1, 2, 3, 5, 8, 13, ...
- Escreva um programa que leia um conjunto de valores inteiros e positivos e determine qual o menor e maior valor do conjunto; a leitura de um valor negativo encerra a leitura do conjunto, e esse valor não será considerado.
- (Desafio) Escreva um programa que exiba o seguinte padrão, usando declarações for. Deve solicitar ao usuário que digite um número entre 1 e 9, definindo o número de linhas do padrão. No exemplo abaixo, o usuário digitou 5.

```
1         1
 2       2
  3     3
   4 4
    5
```

Um Primeiro Jogo

Exercícios (8).

⇒ Laços / Um Primeiro Jogo

- Escreva um programa que implemente o jogo "pedra, papel, e tesoura" contra o computador. O usuário deve digitar um número para escolher a opção (1 - Pedra, 2 - Papel, 3 - Tesoura, e 0 - Sair). O programa deve exibir qual foi a escolha do computador e quem venceu a partida. Quando o usuário escolher 0 (sair), o jogo deve exibir o número de vitórias do usuário, o número de vitórias do computador, e o número de empates. Dica: use srand(static_cast<unsigned int>(time(nullptr))) no início do código para inicializar o gerador de números aleatórios, e use rand() % 3 + 1 para sortear um número entre 1 e 3 para o computador.
- Para os mais avançados, tentem usar a biblioteca random que oferece um mecanismo mais robusto para geração de números aleatórios (descubram como).
- Usem `https://www.programiz.com/cpp-programming/online-compiler/` para testar o código. O Compiler Explorer não possui um bom suporte para programas muito interativos.