UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Linguagem de Programação

Comandos de Repetição

ECT2303

helton.maia@ufrn.br

Comandos de Repetição

Em C++, existem três tipos de comandos de repetição:

- while
- do-while
- for

Todos eles são usados para executar um bloco de código repetidamente até que uma determinada condição seja satisfeita.

Comandos de Repetição

O comando de repetição while executa o bloco de código enquanto uma determinada condição for verdadeira.
A sintaxe é a seguinte:

```
while (condicao) {
    // bloco de código a ser executado
}
```

Exemplo de utilização (while)

Escreva um programa que solicita que o usuário informe um número inteiro positivo e, em seguida, exibe os números pares menores ou iguais ao número informado pelo usuário.

Exemplo (while)

```
int main() {
    int num, i=1;

    cout << "Informe um numero inteiro positivo: ";
    cin >> num;
    cout << "Pares menores ou iguais a " << num << ":"<< endl;

    while (i <= num) {
        if (i % 2 == 0) {
            cout << i << " ";
        }
        i++;
    }
    return 0;
}
```

Comandos de Repetição

O comando de repetição do-while é semelhante ao while, mas executa o bloco de código pelo menos uma vez, independentemente da condição. A sintaxe é a seguinte:

```
do {
    // bloco de código a ser executado
} while (condicao);
```

Exemplo de utilização (do-while)

Escreva um programa para solicitar que o usuário informe uma senha, verificando se a senha informada é igual à senha cadastrada no sistema.

Exemplo (do-while)

```
int main() {
    const int senha_cadastrada = 1234;
    int senha_informada;

    do {
        cout << "Digite a senha: ";
        cin >> senha_informada;

        if (senha_informada != senha_cadastrada) {
            cout << "Senha incorreta. ";
            cout << "Tente novamente!" << endl;
        }

    } while (senha_informada != senha_cadastrada);

    cout << "Senha correta. Acesso permitido." << endl;
    return 0;
}
```

Comandos de Repetição

O comando de repetição for é usado para executar um bloco de código um número fixo de vezes.
A sintaxe é a seguinte:

```
for (inicializacao; condicao; atualizacao) {
    // bloco de código a ser executado
}
```

Exemplo de utilização (for)

Escreva um programa que solicite ao usuário um número inteiro positivo e exibe a soma dos números inteiros de 1 até o número informado.

Exemplo (for)

```
int main() {
    int num, soma = 0;

    cout << "Informe um numero inteiro positivo: ";
    cin >> num;

    for (int i = 1; i <= num; i++) {
        soma += i;
    }

    cout << "A soma dos numeros inteiros de 1 ate " <<
   num << " = " << soma << endl;

    return 0;
}
```

Comandos de Repetição

O comando *break* quando inserido em uma estrutura de repetição, **causa a saída imediata desta estrutura**. A execução do programa é então direcionada para a primeira instrução após o final do comando/bloco de código do laço(*loop*) de repetição.

```
int main(){
    int x = 1;

    while(x<=10){
       x++;
        if(x==9){break;}
        cout << x << " ";
    }
    cout << "x = " << x;
    return 0;
}
```

Estrutura de repetição utilizando pré e pós incrementos (Exemplo 1)

```
int main() {
  int x = 1;

  // Usando o operador ++x:
  cout << "Usando o operador ++x:\n";
  for (int i = 0; i < 5; i++) {
    cout << "x = " << ++x << endl;
  }

  x = 1; // resetando o valor de x

  // Usando o operador x++:
  cout << "\nUsando o operador x++:\n";
  for (int i = 0; i < 5; i++) {
    cout << "x = " << x++ << endl;
  }
  return 0;
}
```

Estrutura de repetição utilizando pré e pós incrementos (Exemplo 2)

```
int main() {
  int x = 1;
  // Usando o operador ++x:
  cout << "Usando o operador ++x:\n";
  for (int i = 0; i < 5; i++) {
    x = x + ++x; //x += ++x;
    cout << "x = " << x << endl;
  }

  x = 1; // resetando o valor de x

  // Usando o operador x++:
  cout << "\nUsando o operador x++:\n";
  for (int i = 0; i < 5; i++) {
    x = x + x++; //x += x++;
    cout << "x = " << x << endl;
  }
  return 0;
}
```

Exercício

Dado um número inteiro na entrada, crie um programa que verifica se esse número é ou não um número primo. Imprima na saída o resultado.

Obs: um número primo é aquele que é divisível somente por 1 e por ele mesmo.

Solução: Verificando o número é primo

```
int main() {
  int x; bool isPrime = true;
  cout << "Digite um número inteiro: ";
  cin >> x;

  for (int i = 2; i <= x / 2; i++) {
    if (x % i == 0) {
      isPrime = false;
      break;
    }
  }
  if (isPrime) {
    cout << x << " é um número primo." << endl;
  } else {
    cout << x << " não é um número primo." << endl;
  }
  return 0;
}
```

Comentando a solução:

- Neste exemplo, declaramos uma variável inteira `x` e uma variável booleana `isPrime` que será usada para armazenar se `x` é um número primo ou não. Em seguida, solicitamos que o usuário digite um número inteiro e armazenamos o valor em `x`.

- Em seguida, usamos um laço `for` para verificar se `x` é divisível por algum número entre 2 e `x/2`. Se `x` for divisível por algum desses números, sabemos que ele não é um número primo e atualizamos a variável `isPrime` para `false` e saímos do laço usando a instrução `break`.

- Finalmente, verificamos o valor da variável `isPrime`. Se ela for `true`, sabemos que `x` é um número primo e imprimimos uma mensagem indicando isso na tela. Caso contrário, imprimimos uma mensagem indicando que `x` não é um número primo.

?