Lab. 3

Comandos de Seleção

ECT2303 - T02 - 19.1

Uma estrutura de seleção permite a escolha de um conjunto de ações a ser executado quando determinadas condições são ou não satisfeitas. Em C++, isto é feito usando um comando **`if`** ou um comando **`switch`**.

3.1 Comandos `if`, `if-else`, `else if`

Um comando **`if`** é a forma mais simples de seleção entre duas alternativa e será executado se a condição for verdadeira (**`true`**). Por exemplo,

```
if (condicao)
    comando;
```

Quando existe mais de um comando a ser executado, torna-se obrigatório a utilização de chaves.

```
if (condicao)
{
    comando1;
    comando2;
    ...
}
```

Um conjunto de comandos (instruções) entre chaves é chamado de **instrução composta** ou **bloco**. Esquecer uma ou ambas as chaves que delimitam um bloco pode levar a erros de sintaxe ou erros de lógica.

O comando **`if-else`** permite que um comando seja executado quando a condição é **`true`**, e um comando diferente quando a condição é **`false`**.

```
if (condicao)
  comando1;
```

```
else
  comando2;
```

Quando existe mais de um comando a ser executado, torna-se obrigatório a utilização de chaves.

```
if (condicao)
{
    comando1;
    comando2;
}
else
{
    comando3;
    comando4;
}
```

As instruções **if-else** aninhadas testam multiplas condições colocando comandos de seleção dentro de outros commandos de seleção. Por exemplo,

```
if (condicao1)
{
    comando1;
    comando2;
}
else if (condicao2)
{
    comando3;
    comando4;
}
else
    comando5;
```

3.2 Comandos switch - case

O comando de seleção **switch-case** é um comando de seleção múltipla que pode ou não executar determinadas instruções.

```
1   switch(valor)
2   {
3       case valor1:
4         comandoA1
5         comandoA2
6         ...
7         break;
8       case valor2:
9         comandoB1
10        comandoB2
```

```
11          ...
12          break;
13        ...
14        default:
15          comandoZ1
16          comandoZ2
17          ...
18    }
```

O comando **switch (valor)** avaliar o valor da expressão para decidir qual **case** será executado. Quando o valor da expressão não coincidir com aqueles especificados nos **case**, será executado então o **default**.

Exemplo 3.2.1. Programa que escreve por extenso na tela o nome dos dígitos de 0 até 2.

```
1  int main(){
2    int d;
3    cin >> d;
4    switch(d){
5        case 0:
6            cout << "zero" << endl;
7            break;
8        case 1:
9            cout << "um" << endl;
10           break;
11       case 2:
12           cout << "dois" << endl;
13            break;
14       default:
15         cout << "nao sei" << endl;
16         break;
17   }
18   return 0;
19 }
```

3.3 Exercícios de Fixação

1. As leis de De Morgan podem às vezes tornar mais conveniente para nós a forma de formular uma expressão lógica. Estas leis afirmam que a expressão `!(condição1 && condição2)` é logicamente equivalente à expressão `(!condição1 || !condição2)`. Da mesmo forma, a expressão `!(condição1 || condição2)` é logicamente equivalente a expressão `(!condição1 && !condição2)`. Use as leis de De Morgan para escrever expressões equivalentes a cada uma das expressões seguintes e, então, escreva um programa para mostrar que a expressão original e a nova expressão em cada caso, são equivalentes.

a) `!(x > 5) && !(y >= 7)`

b) `!(a == b) || !(g != 5)`

c) `!( (x <= 8) && (y > 4))`

d) `!( (i > 4) || (j <= 6))`

Exemplo de Execução:

```
Os valores das variáveis são:
x = 10, y = 1, a = 3, b = 3,
g = 5, Y = 1, i = 2, j = 9

!(x < 5) && !(y >= 7) é equivalente à !((x < 5) || (y >= 7))
!(a == b) || !(g != 5) é equivalente à !((a == b) && (g != 5))
!((x <= 8) && (Y > 4)) é equivalente à !((x <= 8) || (Y > 4))
!((i > 4) || (j <= 6)) é equivalente à !((i > 4) && (j <= 6))
```

2. Ler quatro valores referentes a quatro notas escolares de um aluno e imprimir uma mensagem dizendo que o aluno foi aprovado, se o valor da média escolar for maior ou igual a 7. Se o valor da média for menor que 7, solicitar a nota de exame, somar com o valor da média e obter nova média. Se a nova média for maior ou igual a 5, apresentar uma mensagem dizendo que o aluno foi aprovado em exame. Se o aluno não foi aprovado, indicar uma mensagem informando esta condição. Apresentar com as mensagens o valor da média do aluno, para qualquer condição.

3. Faça um programa que receba quatro valores: I, A, B, C. Desses valores I é inteiro e positivo, A, B e C são reais. Escreva os números A, B e C obedecendo à tabela a seguir:

Valor de I	Saída
1	A, B e C em ordem crescente
2	A, B e C em ordem descrescente
3	O Maior fica entre os outros dois números

4. Faça um programa que receba o código correspondente ao cargo de um funcionário e seu salário atual e mostre o cargo, o valor do aumento e seu novo salário. Os cargos estão na tabela a seguir.

5. O `switch-case` é comumente empregado na construção de *menus de opções*. Para verificar tal propriedade, crie um programa em C++ que simula uma máquina de bebidas. Para tanto, o programa deve seguir as seguintes etapas:

   (a) Primeiro, o programa deve solicitar um valor inteiro representando o código para o produto a ser desejado. A tabela abaixo mostra as opções de produtos disponíveis:

Código	Cargo	Percentual
1	Escrituário	50%
2	Secretário	35%
3	Caixa	20%
4	Gerente	10%
5	Diretor	Não tem aumento

Opção	Produto	Preço
1	Água mineral	R$ 2,00
2	Refrigerante	R$ 3,00
3	Suco	R$ 4,00
4	Energético	R$ 5,00

O código e os produtos disponíveis devem estar disponibilizados na tela para o usuário. Caso seja solicitado uma opção diferente das disponíveis, uma mensagem deve ser enviada ao usuário informando que a cédula solicitada não existe, e o programa deve ser encerrado.

(b) Depois, o programa deve solicitar que o usuário digite um valor inteiro positivo representando o número de unidades do produto selecionado. Caso o valor seja válido, o programa deve informar o valor a ser pago pelo número de unidades do produto selecionado. Caso o valor seja inválido (um valor nulo ou negativo, por exemplo), uma mensagem deve ser enviada ao usuário informando que a quantidade solicitada não pode ser atendida, e o programa deve ser encerrado.

(c) Por fim, o programa deve receber a quantia dada pelo usuário como pagamento, armazenada em um ponto flutuante. Caso a quantia dada seja menor que preço a pagar, uma mensagem deve ser enviada ao usuário informando que o valor é insuficiente para a operação, e o programa deve ser encerrado. Caso contrário, o programa deve informar o número de unidades obtidas, o troco a ser dado e uma mensagem sinalizando que a operação foi bem sucedida.

```
-- Exemplo 1: Execucao bem sucedida.
Digite o codigo do produto:
1 - Agua mineral
2 - Refrigerante
3 - Suco
4 - Energetico
1
Digite o numero de unidades do produto:
3
Valor a ser pago: R$ 6.00
Digite a quantia para o pagamento:
10
Foram enviadas 3 unidades do produto.
Troco: R$ 4.00.
```

```
Operacao realizada com sucesso!
-- Exemplo 2: Quantia insufuciente para a compra.
Digite o codigo do produto:
1 - Agua mineral
2 - Refrigerante
3 - Suco
4 - Energetico
4
Digite o numero de unidades do produto:
2
Valor a ser pago: R$ 10.00
Digite a quantia para o pagamento:
5
Quantia insuficiente para a compra!
-- Exemplo 3: Numero inadequado de unidades.
Digite o codigo do produto:
1 - Agua mineral
2 - Refrigerante
3 - Suco
4 - Energetico
3
Digite o numero de unidades do produto:
0
Numero de unidades invalido!
-- Exemplo 4: Opcao invalida.
Digite o codigo do produto:
1 - Agua mineral
2 - Refrigerante
3 - Suco
4 - Energetico
8
Opcao nao existente!
```

3.4 Referências Bibliográficas

1. MANZANO, J.A.; OLIVEIRA, J.F.; **Algoritmos - Lógica para Desenvolvimento de Programação**. Editora Erica.

2. ASCENCIO, A.F.G.; CAMPOS, E.A.V. **Fundamentos da Programação de Computadores - Algoritmos, Pascal e C/C++**. 3ed. Editora Pearson.

3. DEITEL, H. M.; DEITEL, P. J. **C++ Como Programar**. 3ed. Editora Bookman.