Unidade 11: Estruturas de Decisão Múltipla

Prof. Daniel Caetano

Objetivo: Tomando decisões complexas no código de programação.

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

INTRODUÇÃO

Na aula passada vimos como implementar decisões na linguagem de programação C/C++. É claro que começamos com decisões bastante simples... mas o computador é capaz de decisões bem mais complexas, com base em várias comparações ao mesmo tempo.

Nesta aula, veremos exatamente como ensinar o computador fazer isso.

1. DECISÕES MÚLTIPLAS

Em algumas situações, como em um menu, queremos tomar uma atitude diferente dependendo do valor digitado pelo usuário. Por exemplo, considere que queremos fazer um programa que imprime o seguinte menu:

1) Saldo
2) Extrato
3) Saque

Para cada uma dessas opções, executaremos uma ação diferente e, caso nenhuma delas seja válida, queremos imprimir uma mensagem de erro.

Para resolver esse problema, basicamente temos que:

1) Imprimir o menu
2) Ler um número digitado pelo usuário
3) Verificar, com um IF, se a opção foi 1. Se foi, agir de acordo.
4) Verificar, com um IF, se a opção foi 2. Se foi, agir de acordo.
5) Verificar, com um IF, se a opção foi 3. Se foi, agir de acordo.

Esse código pode ser implementado como indicado a seguir.

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {
     int N;
     cout << "1) Saldo" << endl;
     cout << "2) Extrato" << endl;
     cout << "3) Saque" << endl;
     cin >> N;

     if ( N == 1 ) {
        cout << "Opção SALDO selecionada!" << endl;
        }
     if ( N == 2 ) {
        cout << "Opção EXTRATO selecionada!" << endl;
        }
     if ( N == 3 ) {
        cout << "Opção SAQUE selecionada!" << endl;
        }
getchar();
}
```

Esse programa faz o que pensamos... mas o que ocorre se o usuário digitar, digamos, um número maior que 3? Ou qualquer outro valor que não seja 1, 2 ou 3?

O programa não faz nada.... e o usuário fica sem saber o que aconteceu de errado!

Para evitar isso, é comum indicarmos uma mensagem de erro quando o usuário seleciona uma opção inexistente no menu. Para isso, precisamos verificar se o valor digitado é, **simultaneamente**, diferente de 1, 2 e 3. Isso pode ser conseguido **aninhando** os **if**s:

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

     cout << "1) Saldo" << endl;
     cout << "2) Extrato" << endl;
     cout << "3) Saque" << endl;
     cin >> N;

     if ( N == 1 ) {
        cout << "Opção SALDO selecionada!" << endl;
        }
     if ( N == 2 ) {
        cout << "Opção EXTRATO selecionada!" << endl;
        }
     if ( N == 3 ) {
        cout << "Opção SAQUE selecionada!" << endl;
        }
     if ( N != 1 ) {
        if ( N != 2 ) {
           if ( N != 3 ) {
              cout << "Opção inexistente!" << endl;
              }
           }
        }

getchar();
}
```

No código acima, o bloco rosa só será executado caso as condições N!=1, N!=2 e N!=3 sejam, simultaneamente, atendidas.

Será que não existe uma forma mais simples dee verificar se o valor de **N** é, ao mesmo tempo, diferente de vários números?

Ora, é claro que existe! Lembram-se dos **operadores lógicos**? Quando queremos verificar se várias coisas acontecem simultaneamente, usamos o operador **E**... afinal, no fundo queremos verificar isso:

se **N != 1 E N != 2 E N != 3**...

Não é? Se o resultado final dessa proposição for verdadeiro, é porque todas as proposições (N!=1, N!=2 e N!=3) são verdadeiras ao mesmo tempo! Assim, se o resultado final da proposição for verdadeiro, indicaremos uma mensagem de erro (afinal, o número digitado não foi nem 1, nem 2 e nem 3)! No código, isso ficaria algo como:

```
if ( N != 1 E N != 2 E N != 3 ) {

    }
```

Observe, porém, que se não fosse usada uma cor diferente para o operador lógico, a leitura da expressão ficaria **muito** prejudicada, porque as letras, em C/C++, são usadas, em geral, para nomes de variáveis. Assim, existem símbolos para especificar as operações lógicas:

Operador	Símbolo	Exemplo	Significado
OU	**||**	x == 2 || x == 9	verdadeiro se x for igual a 2 **OU** x for igual a 9.
E	**&&**	x > 2 && x < 9	verdadeiro se x for maior que 2 **E** x menor que 9.
NÃO	**!**	!(x == 2)	verdadeiro se x **NÃO** for igual a 2.

Considerando esses símbolos, a expressão...

```
if ( N != 1 E N != 2 E N != 3 ) {

    }
```

...deve ser escrita, no código, desta maneira:

```
if ( N != 1 && N != 2 && N != 3 ) {

    }
```

Assim, a versão final do código fica:

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {
     int N;
     cout << "1) Saldo" << endl;
     cout << "2) Extrato" << endl;
     cout << "3) Saque" << endl;
     cin >> N;

     if ( N == 1 ) {
        cout << "Opção SALDO selecionada!" << endl;
        }
     if ( N == 2 ) {
        cout << "Opção EXTRATO selecionada!" << endl;
        }
     if ( N == 3 ) {
        cout << "Opção SAQUE selecionada!" << endl;
        }
     if ( N !=1 && N!=2 && N!=3 ) {
        cout << "Opção inválida!" << endl;
        }
getchar();
}
```

1.1. Exercício

Faça um programa que leia um número digitado pelo usuário e, **com um único if**, imprima a mensagem "número inválido" caso o número seja: "**menor que 10**" ou "**igual a 30**" ou "**um número par maior ou igual a 50**".

SOLUÇÃO:

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {
     float N;
     cout << "Digite um número: ";
     cin >> N;

     if ( N < 10 || N == 30 || ( N%2 == 0 && N >= 50 ) ) {
        cout << "Número inválido!" << endl;
        }
getchar();
}
```

2. ESTRUTURA DE ESCOLHA (SWITCH)

No início da aula falamos sobre a estrutura de menu, construída com **if**s. Uma vez que a necessidade de construir menus é muito comum, seja explicitamente (como em programas de computador, caixas eletrônicos, auto-atendimento telefônico etc.), seja implicitamente (como no software de controle do teclado de um celular ou MP3 player), criou-se um padrão e uma instrução para a elaboração de menus em C/C++.

O padrão é que para cada opção do menu (ou tecla de um equipamento) é associado um **número inteiro**, cujo valor indica uma opção... e, para implementar menus segundo esse padrão, foi criada a opção **switch** (em português, "switch" significa "escolha").

A sintaxe é simples (lembrando que as partes em cinza devem ser substituídas por código):

```
switch ( variável_inteira ) {
        case 1:
                Código da opção 1
                break;
        case 2:
                Código da opção 2
                break;
        }
```

O código do menu anterior, modificado para usar o **switch** fica:

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {
     int N;
     cout << "1) Saldo" << endl;
     cout << "2) Extrato" << endl;
     cout << "3) Saque" << endl;
     cin >> N;

     switch ( N ) {
        case 1:
           cout << "Opção SALDO selecionada!" << endl;
           break;
        case 2:
           cout << "Opção EXTRATO selecionada!" << endl;
           break;
        case 3:
           cout << "Opção SAQUE selecionada!" << endl;
           break;
        }
     if ( N !=1 && N!=2 && N!=3 ) {
        cout << "Opção inválida!" << endl;
        }
getchar();
}
```

O código fica bem mais limpo, não?

Sim... tirando aquele "if complicado" lá no final.

Pois então... a instrução **switch** tem mais uma vantagem! Esse último if **também** pode ser integrado ao **switch**. Além das opções case 1, case 2, ... , case N, o **switch** tem um caso especial chamado **default**, que é executado quando nenhum dos anteriores ocorreu:

```
switch ( variável_inteira ) {
        case 1:
                Código da opção 1
                break;
        case 2:
                Código da opção 2
                break;
        default:
                Código para todos os outros valores da variável inteira
        }
```

Com isso, o código anterior fica ainda mais enxuto. Observe!

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {
     int N;
     cout << "1) Saldo" << endl;
     cout << "2) Extrato" << endl;
     cout << "3) Saque" << endl;
     cin >> N;

     switch ( N )
        {
        case 1:
           cout << "Opção SALDO selecionada!" << endl;
           break;
        case 2:
           cout << "Opção EXTRATO selecionada!" << endl;
           break;
        case 3:
           cout << "Opção SAQUE selecionada!" << endl;
           break;
        default:
           cout << "Opção inválida!" << endl;
        }
getchar();
}
```

2.1. Exercício

Construa um programa que apresente um menu com 2 opções: 1) Multiplicar e 2) Dividir. Use **switch** e, para cada caso, faça um programa que peça um dos números e realize o cálculo adequado.

SOLUÇÃO

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {
     int N;
     float A, B, R;
     cout << "1) Multiplicar" << endl;
     cout << "2) Dividir" << endl;
     cin >> N;

     switch ( N )
        {
        case 1:
           cout << "Opção MULTIPLICAR selecionada!" << endl;
           cout << "Digite o primeiro número: ";
           cin >> A;
           cout << "Digite o segundo número: ";
           cin >> B;
           R = A * B;
           cout << "O resultado é: " << R << endl;
           break;
        case 2:
           cout << "Opção DIVIDIR selecionada!" << endl;
           cout << "Digite o primeiro número: ";
           cin >> A;
           cout << "Digite o segundo número: ";
           cin >> B;
           if ( B != 0 ) {
              R = A / B;
              cout << "O resultado é: " << R << endl;
              }
           else {
              cout << "Não é possível dividir por zero!" << endl;
              }
           break;
        default:
           cout << "A opção " << N << " não existe!" << endl;
        }
getchar();
}
```