A verdade é uma mentira que ainda não foi desmascarada.

Beryl Bainbridge

1 Entendendo a verdade

A verdade é como o preto e o branco, não tem como enganar. Em C++ isto é um fato. Você pode representar a verdade (*true*) e a falsidade (*false*). É possível armazenar estes valores numa variável do tipo *bool*:

```
1  bool fato, ficcao;
   fato = true;
   ficcao = false;
```

As palavras chaves *true* e *false* representam na verdade duas situações em C++: um valor diferente de 0, representa verdadeiro. Um valor igual a 0, representa falso.

Os valores *true* e *false* normalmente estão associados a comparações. Para que ocorra comparações é necessário a utilização dos operadores relacionais (tabela 1). Os operadores relacionais são elementos que é fundamental conhecer dada a sua importância na elaboração de um programa. Em todos os programas são utilizadas expressões relacionais e lógicas para a tomada de decisões e consequente desvio do fluxo do programa. Os operadores relacionais servem para realizar a comparação de dois valores distintos.

Tabela 1: Operadores relacionais do C++.

Operador	em C++	Significado	Exemplo	Resultado
$<$	$<$	Menor que	$5 < 8$	*true*
			$8 < 5$	*false*
			$8 < 8$	*false*
$>$	$>$	Maior que	$5 > 8$	*false*
			$8 > 5$	*true*
			$5 > 5$	*false*
$\leq$	$<=$	Menor ou igual a	$5 <= 5$	*true*
			$5 <= 8$	*true*
			$8 <= 5$	*false*
$\geq$	$>=$	Maior ou igual a	$5 >= 5$	*true*
			$5 >= 8$	*false*
			$8 >= 5$	*true*
$=$	$==$	Igual a	$5 == 5$	*true*
			$5 == 8$	*false*
$\neq$	$!=$	Diferente de	$5! = 5$	*false*
			$5! = 8$	*true*

O programa 1 apresenta um rápido exemplo de uso dos operadores relacionais em C++. Você pode atribuir o resultado de uma expressão relacional em uma variável ou apresentar o resultado da expressão diretamente na tela. É claro que estas não são as únicas formas possíveis de utilização das expressões relacionais.

Programa 1: Um rápido exemplo de uso dos operadores relacionais.

```
// Operadores relacionais
// Como estabelecer a verdade
// programa_001.cpp
#include "biblaureano.h"

int main()
{
  bool resultado;


  resultado = (5==5); // produz um resultado
                      // do tipo bool

  cout << "5 == 5:" << resultado << endl;
  cout << "5 == 8:" << (5 == 8) << endl;

  resultado = (5>8);
  cout << "5 > 8:" << resultado << endl;
  cout << "5 < 8:" <<'' << endl;
  return 0;
}
```

É comum na programação surgirem situações em que a execução de uma ação ou sequência de sub-ações, está sujeita a uma certa condição. Esta condição é representada em programação por meio de uma *expressão lógica*. Denomina-se *expressão lógica* a expressão cujos operadores são lógicos e cujos operandos são relações, constantes e/ou variáveis do tipo lógico.

2 Utilizando a cláusula *if*

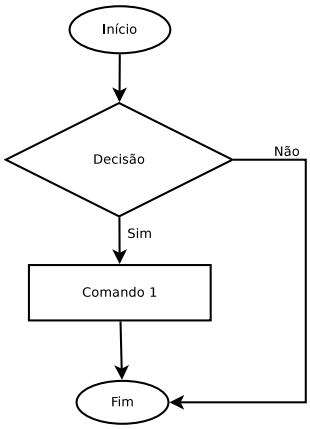
Até o momento você já entendeu o conceito de verdade e falsidade em C++. Agora vamos ver como utilizar estes conceitos para a tomada de decisões. Uma das primeiras formas de tomada de decisão em C++ é a *decisão com 1 alternativa* (figura 1). Em C++, a tomada de decisão ocorre com a utilização do comando *if*:

```
if (condição)
{
  bloco de comandos;
}
```

O comando *if* funciona da seguinte maneira. Primeiramente a expressão da condição é avaliada. Caso o resultado seja verdadeiro o bloco de comandos entre { e } (abre e fecha chaves) é executado. Caso a expressão resulte em falso o bloco de comandos não será executado. O programa 2 demonstra vários exemplos de verdades e falsidades combinados com o comando *if*.

Na figura 2 podemos observar o fluxograma do programa 2.

Figura 1: Decisão com 1 alternativa.



2.1 Testando *true* ou *false*

Na primeira condição testada no *if*, é testado o valor *true*. Bom, lembre-se que o bloco de comando será executado no caso da condição seja verdadeira.

```
if(true)
{
  cout << "A verdade está lá fora!" << endl;
}
```

Na próxima condição é testado o valor *false*. Como *false* não é *true*, o bloco de comandos não será executado.

```
if(false)
{
  cout << "Acho que nunca verei você!" << endl;
}
```

Lembre-se de não colocar um ; (ponto e vírgula) após o comando *if*. Isto indica que não haverá um bloco de comandos a ser executado, ou seja:

```
if(false);
{
  cout << "Acho que nunca verei você!" << endl;
}
```

É o mesmo que escrever:

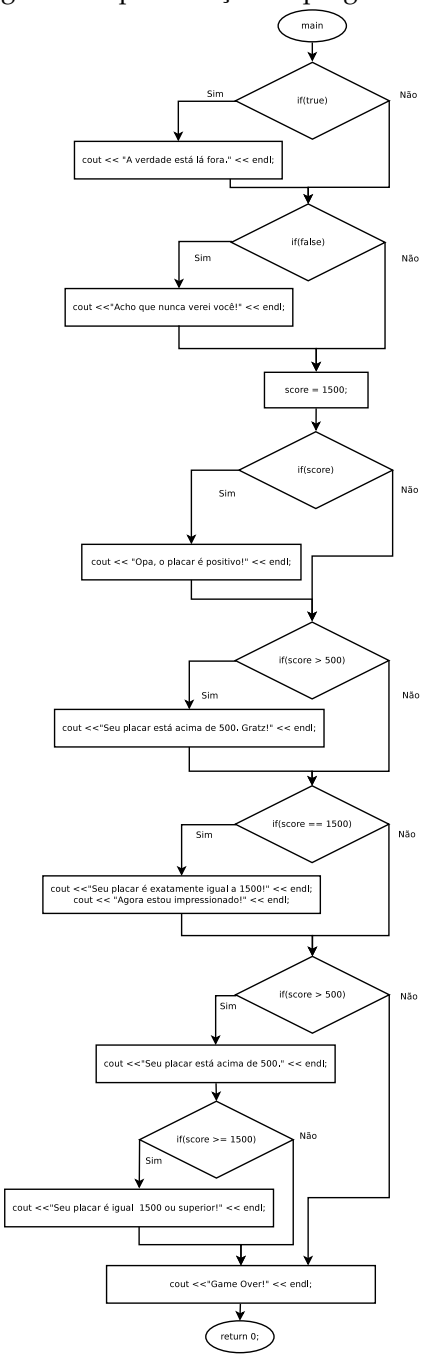
```
if(false);

cout << "Acho que nunca verei você!" << endl;
```

Altere o programa 2 e verifique você mesmo o resultado.

Programa 2: Primeiro exemplo do comando *if*.

```
// Comando if
// Como estabelecer a verdade
// programa_002.cpp
#include "biblaureano.h"

int main()
{
  if(true)
  {
    cout << "A verdade está lá fora." << endl;
  }

  if(false)
  {
    cout << "Acho que nunca verei você!" << endl;
  }

  int score = 1500;

  if(score) // lembre-se, um número diferente de
  {       // é considerado verdadeiro
    cout << "Opa, o placar é positivo!" << endl;
  }

  if(score > 500)
  {
    cout << "Seu placar está acima de 500. Gratz!" << endl;
  }

  if(score == 1500)
  {
    cout << "Seu placar é exatamente igual a 1500!" << endl;
    cout << "Agora estou impressionado!" << endl;
  }

  if(score > 500)
  {
    cout << "Seu placar está acima de 500." << endl;
    if(score >= 1500) //sim é possível utilizar um if dentro de outro
    {
      cout << "Seu placar é igual 1500 ou superior!" << endl;
    }
  }

  cout << "Game Over!" << endl;
  return 0;
}
```

Figura 2: Representação do programa 2.



2.2 Interpretando se um valor é *true* ou *false*

Como você interpreta um valor *true* ou *false* ? O C++ considera um valor diferente de zero como verdadeiro e igual a zero como falso. Isto pode ser observado no trecho de código:

```
if(score)
{
  cout << "Opa, o placar é positivo!" << endl;
}
```

Afinal, no momento da definição, foi atribuído o valor 1500 à variável *score*. Como 1500 é um valor diferente de zero, a condição é interpretada como *true*.

2.3 Utilizando os operadores relacionais

Na maioria dos casos, você utilizará operadores relacionais, para comparar 2 valores, associados ao comando *if*:

```
if(score > 500)
{
  cout << "Seu placar está acima de 500. Gratz!" << endl;
}
```

Como *score* é maior que 1500, o programa exibirá a mensagem na tela.

Na próxima condição, como *score* é igual à 1500, será exibido as mensagens. Se a variável *score* estivesse com outro valor, as mensagens não seriam exibidas:

```
if(score == 1500)
{
  cout << "Seu placar é exatamente igual a 1500!" << endl;
  cout << "Agora estou impressionado!" << endl;
}
```

Atenção: Cuidado ao utilizar o operador de igualdade (==). Caso você esqueça de colocar um sinal de igual, o C++ entenderá que você: 1º está atribuindo um valor a variável, e, 2º verificará se a condição é verdadeira. Observe:

```
int score = 200;
if(score = 500)
{
  cout << "Seu placar é exatamente igual a 500!" << endl;
}
```

Este tipo de erro não será apontado pelo compilador. Você pode ficar um bom tempo tentando descobrir o motivo do seu programa não estar funcionando adequadamente. Mas como consolo, saiba que este é um erro muito comum, até mesmo entre programadores experientes.

2.4 Trabalhando com *if* dentro *if*

É possível utilizar o comando *if* dentro de um bloco de comandos de outro *if*:

```
if(score > 500)
{
  cout << "Seu placar está acima de 500." << endl;
  if(score >= 1500)
  {
    cout << "Seu placar é igual 1500 ou superior!" << endl;
  }
}
```

Claro, o *if* mais interno somente será avaliado se a expressão do primeiro *if* for verdadeira.

Atenção: Você pode colocar quantos comandos *if* dentro de outros:

```
if(condição)
{
   if(outracondição)
   {
      if(maisumacondição)
      {
         cout << "Oi..." << endl;
      }
      if(chegadecondições)
      {
         if(argh)
         {
            cout << "Cuidado com uso excessivo de ifs um dentro do outro." << endl;
         }
      }
   }
}
```

Mas o múltiplo uso de comandos *if* dentro do outro pode deixar o seu código mais complicado de entender e propenso a erros.

3 Utilizando a cláusula *else*, seguindo por outro caminho

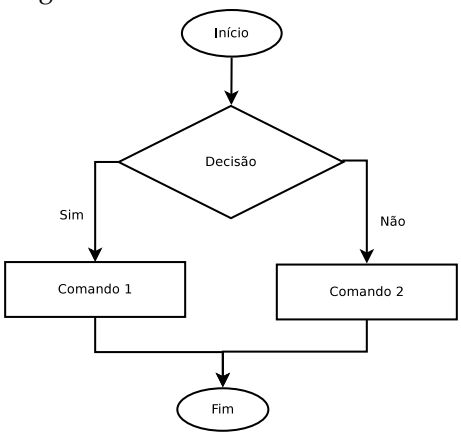
Como nós vimos até o momento, em C++, a tomada de decisão ocorre com a utilização do comando *if*. Mas o *if* fornece apenas um caminho para a tomada de decisão (quando a condição é *true*). Para os casos onde se deseja seguir outro caminho, devemos utilizar a cláusula *else*:

```
if (condição)
{
  bloco de comandos;
}
else
```

```
  {
    outro bloco de comandos;
  }
```

Neste caso, inicialmente testa-se a condição. Caso seja verdadeiro o *bloco de comandos* será executado. Caso a condição resulte em valor falso será executado o *outro bloco de comandos*. Portanto, utilizando *else* você tem a estrutura de *decisão com 2 alternativas* (figura 3).

Figura 3: Decisão com 2 alternativas.

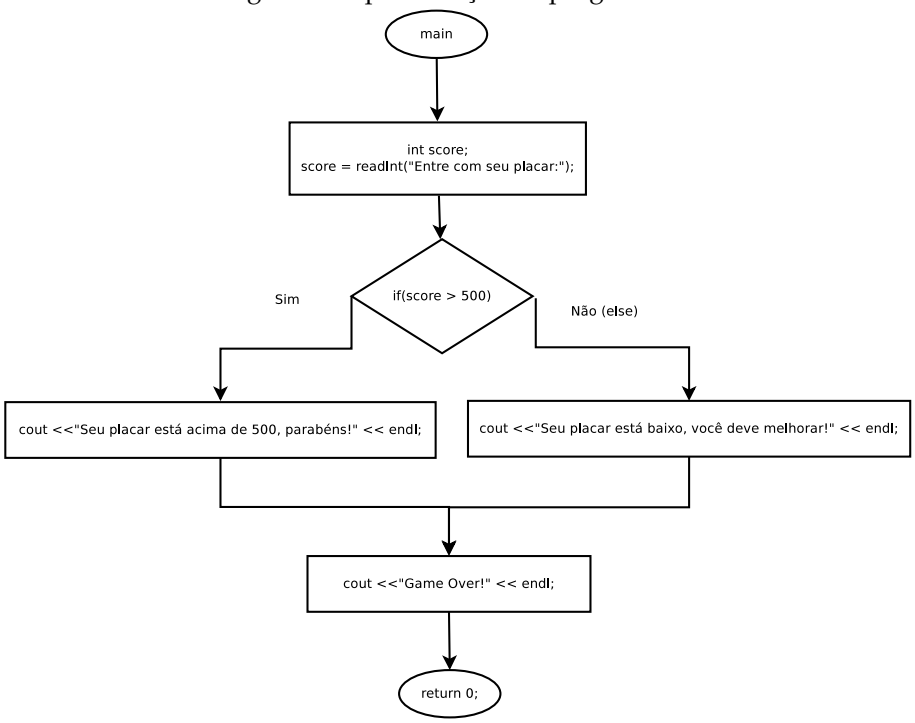


Programa 3: Outro exemplo utilizando *if* com *else*.

```
// Comando if com else
// Como estabelecer a verdade ou fugir dela
// programa_003.cpp
#include "biblaureano.h"

int main()
{
  int score;
  score = readInt("Entre com o seu placar:");

  if(score > 500)
  {
    cout << "Seu placar está acima de 500, parabéns!" << endl;
  }
  else
  {
    cout << "Seu placar está baixo, você deve melhorar!" << endl;
  }

  cout << "Game Over!" << endl;
  return 0;
}
```

Na figura 4 podemos observar o fluxograma do programa 3.

Figura 4: Representação do programa 3.



3.1 Duas alternativas para uma decisão

Como você deve ter percebido, o programa tem duas alternativas na tomada de decisão. Uma se o placar (*score*) for superior a 500:

```
if(score > 500)
{
  cout << "Seu placar está acima de 500, parabéns!" << endl;
}
```

Ou segue por outro caminho, se o placar (*score*) for menor ou igual a 500:

```
else
{
  cout << "Seu placar está baixo, você deve melhorar!" << endl;
}
```

3.2 Cuidados na tomada de decisão

Na vida real, uma decisão errada poderá trazer sérios infortúnios para você (subir no ônibus errado ou assistir a um filme bomba). Em programação isto também ocorre. Observe o programa 4:

Programa 4: Cuidados ao utilizar *if* ou *else*.

```
// Comando if com else
// As mensagens que nunca aparecem
// programa_004.cpp
#include "biblaureano.h"

int main()
{

  if(false)
  {
    if(true)
    {
      cout << "Será que verei esta mensagem ?" << endl;
    }
    else
    {
      cout << "Esta mensagem nunca será vista!" << endl;
    }
  }
  else
  {
    if(false)
    {
      cout << "O que se passa cabrón ?" << endl;
    }
    else
    {
      cout << "Agora sim! Oi gamer!" << endl;
    }
    cout << "Posso colocar outros comandos aqui..." << endl;
  }
  cout << "Game Over!" << endl;
  return 0;
}
```

No programa 4 fica claro que a escolha de condições erradas levam a tomada de decisões erradas. Muito cuidado para não cair na armadilha de colocar uma condição que nunca será verdadeira.

4 Gerando números aleatórios - mais uma função da nossa biblioteca

Todo jogo tem um fator de aleatoriedade, seja para que um NPC (*non player character*) erre ou acerte um golpe durante uma luta ou erre um tiro num jogo de *first-person shooter* (jogo de tiro em primeira pessoa). No programa 5 podemos observar uma forma simples gerar números aleatórios.

Programa 5: Gerando números aleatórios.

```
// Números randômicos
// Um aliado do desenvolvedor de jogos
// e inimigo do jogador.
// programa_005.cpp
#include "biblaureano.h"
```

```
6
int main()
{
  int numeroAleatorio;

11  //inicializa os números randômicos
  //garante que o número gerado esteja
  //entre 1 e 10
  numeroAleatorio = randomico(1,10);

16  int numeroUsuario;
  numeroUsuario = readInt("Entre com o seu palpite:");

  if( numeroUsuario == numeroAleatorio )
  {
21    cout << "Parabéns, você acertou!" << endl;
  }
  else
  {
    cout << "Que pena, o número correto era ";
26    cout << numeroAleatorio << ". Melhor sorte na próxima!" << endl;
  }

  cout << "Game over!" << endl;

31  return 0;
}
```

Como você pode ter observado no programa 5, utilizamos uma função chamada *randomico* para que o computador retornasse uma número aleatório para utilizarmos no nosso jogo. Esta função pode ser utilizada de várias formas:

- *randomico()* - retornando um valor entre 0 e o máximo permitido pelo computador (acredite, é um número muito grade);
- *randomico(valorInicial)* - retornando um número entre o valor passado e o máximo permitido pelo computador;
- *randomico(valorInicial, valorFinal)* - retorna um número entre o valor inicial passado e o valor final passado (é o que mais utilizaremos);

O programa 6 demonstra os vários usos da função *randomico*:

Programa 6: Utilizando a função *randomico*.

```
// vários usos da função randomico
//programa_006.cpp
3 #include "biblaureano.h"

int main()
{
  cout << "Pegando um número entre 1 e 10:";
8   cout << randomico(1,10) << endl;

  cout << "Pegando qualquer número gerado:";
  cout << randomico() << endl;
```

```
  cout << "Pegando qualquer número gerado a partir do 500:";
  cout << randomico(500) << endl;

  cout << "Pegando qualquer número entre 100 e 300:";
  cout << randomico(100,300) << endl;

  cout << "Também posso atribuir para uma variável....";
  int numero = randomico(0,100);
  cout << "Numero gerado:" << numero << endl;
  return 0;
}
```

5 Gerando números aleatórios - no estilo C++

Para gerar um número randômico no estilo C++, é necessário inicializar uma *semente* (número inicial). A inicialização da *semente* é feita com a função *srand*. Esta função precisa de um número para utilizar como referência, sendo comum passar alguma informação variável do computador. Neste caso, está sendo utilizando a função *time* para retornar a quantidade de segundos. Este passo precisa ser feito apenas uma vez no início do programa. O programa 7 demonstra como gerar números aleatórios no estilo C++.

Programa 7: Utilizando as funções do C++ para gerar números aleatórios.

```
// Números randômicos
// Um aliado do desenvolvedor de jogos
// e inimigo do jogador.
// programa_007.cpp
#include "biblaureano.h"

int main()
{
  int numeroAleatorio;

  //inicializa os números randômicos
  srand(time(0));
  //garante que o número gerado esteja
  //entre 1 e 10
  numeroAleatorio = (rand()%10)+1;

  int numeroUsuario;
  numeroUsuario = readInt("Entre com o seu palpite:");

  if( numeroUsuario == numeroAleatorio )
  {
    cout << "Parabéns, você acertou!" << endl;
  }
  else
  {
    cout << "Que pena, o número correto era ";
    cout << numeroAleatorio << ". Melhor sorte na próxima!" << endl;
  }
```

```
    cout << "Game over!" << endl;

    return 0;
}
```

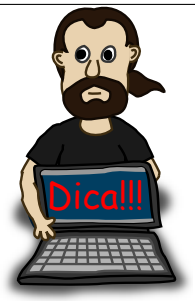
5.1 Entendendo o programa

Inicialmente é necessário inicializar a *semente* de números aleatórios:

```
srand(time(0));
```

Finalmente, é possível gerar o número aleatório com a função *rand*.

```
numeroAleatorio = (rand()%10)+1;
```



O número aleatório gerado pode ser muito grande para os seus objetivos. Uma forma de limitar o número gerado é pegando o resto da divisão entre o número gerado e o maior número que você deseja obter. Exemplo: Para garantir que os números gerados estejam entre 1 e 20:

```
numeroAleatorio = (rand()%21)+1;
```

6 Atividade

Bole um jogo onde seja necessário utilizar os conceitos vistos até o momento. Utilize todos os conhecimentos (inclusive das aulas passadas). Lembre-se: um bom jogo deve ter uma boa história para entreter o jogador.

Posteriormente esta atividade deverá ser entregue para o professor em data a ser definida.

7 Exercícios

O objetivo geral desta sequência de exercícios é possibilitar a prática do uso de números aleatórios e principalmente trabalhar com condições (decisões).

1. Utilizando números aleatórios, apresente uma proposta de soma na tela e peça para o usuário digitar a resposta, em seguida o seu jogo deve dizer se o resultado informado está correto ou não. Exemplo de como pode ser a tela do jogo:

```
Bem vindo ao Calculator Tabajara!!
Qual o resultado para 5+7 ?
```

2. Utilizando o exercício anterior como base, faça que, além do programa gerar os números aleatórios, gere o sinal da operação. Dica: você pode gerar um número entre 1 e 4 utilizá-los para definir o sinal. Exemplo: se o número gerador for 1, então é uma soma. Se for 2, então é uma subtração e assim sucessivamente.
3. Escreva um programa que leia três números inteiros e positivos (A, B, C) e calcule a seguinte expressão:

   $D = \frac{R+S}{2}$, onde $R = (A+B)^2$ e $S = (B+C)^2$
4. Faça um programa que leia a idade de uma pessoa expressa em anos, meses e dias e mostre-a expressa apenas em dias.
5. Faça um programa que leia a idade de uma pessoa expressa em dias e mostre-a expressa em anos, meses e dias.
6. João Papo-de-Pescador, homem de bem, comprou um microcomputador para controlar o rendimento diário de seu trabalho. Toda vez que ele traz um peso de peixes maior que o estabelecido pelo regulamento de pesca do estado do Paraná (50 quilos) deve pagar um multa de R$ 7,00 por quilo excedente. João precisa que você faça um programa leia a variável P (peso de peixes) e verifique se há excesso. Se houver, gravar na variável E (Excesso) e na variável M o valor da multa que João deverá pagar. Caso contrário mostrar tais variáveis com o conteúdo ZERO.
7. A Secretaria de Meio Ambiente que controla o índice de poluição mantém 3 grupos de indústrias que são altamente poluentes do meio ambiente. O índice de poluição aceitável varia de 0,05 até 0,25. Se o índice sobe para 0,3 as indústrias do 1^{o}grupo são intimadas a suspenderem suas atividades, se o índice crescer para 0,4 as indústrias do 1^{o}e 2^{o}grupo são intimadas a suspenderem suas atividades, se o índice atingir 0,5 todos os grupos devem ser notificados a paralisarem suas atividades. Faça um programa que leia o índice de poluição medido e emita a notificação adequada aos diferentes grupos de empresas.
8. Calcular a quantidade dinheiro gasta por um fumante. Dados: o número de anos que ele fuma, o número de cigarros fumados por dia e o preço de uma carteira.
9. Ler um nome do teclado e ver se é igual ao seu nome. Imprimir conforme o caso: ”NOME CORRETO” ou ”NOME INCORRETO”.
10. Leia a velocidade máxima permitida em uma avenida e a velocidade com que o motorista estava dirigindo nela e calcule a multa que uma pessoa vai receber, sabendo que são pagos:
    a) 50 reais se o motorista estiver ultrapassar em até 10km/h a velocidade permitida (ex.: velocidade máxima: 50km/h; motorista a 60km/h ganha multa);
    b) 100 reais, se o motorista ultrapassar de 11 a 30 km/h a velocidade permitida;
    c) 200 reais, se estiver acima de 31km/h da velocidade permitida.
11. Sabendo que latão é constituído de 70% de cobre e 30% de zinco, indique a quantidade de cada um desses componentes para se obter uma certa quantidade de latão (requerida pelo usuário).
12. Desenvolva um programa que recebe do usuário o placar de um jogo de futebol (os gols de cada time) e informa se o resultado foi um empate, a vitória do primeiro time ou do segundo time.