> Um mosquito é uma pequena criação da natureza para nos fazer pensar melhor sobre as moscas.
>
> André Guillois

1 Trabalhando com constantes

Muitas vezes é desejável que além de uma variável possuir um valor pré-definido, quer-se que este valor não seja modificado em nenhum momento durante a execução de um jogo.

Uma *constante* é uma variável cujo valor nunca muda. Por exemplo, você escreveu um jogo de batalhas no espaço, onde cada espaçonave alienígena derrubado acrescente 150 pontos no *score* do jogador. Você pode definir uma constante *ALIENS_PONTOS* com o valor 150. Então, quando o jogador derrubar uma espaçonave, você utiliza esta constante em vez de somar o valor literal 150.

O uso de constantes traz 2 benefícios. Primeiro, o programa fica mais limpo e legível. Segundo, no caso de precisar modificar o valor dos pontos, basta modificar apenas o valor definido na constante. Observe o programa 1.

Programa 1: Uso de constantes.

```
1  // Uso de constantes
   // Outras formas para controlar o seu jogo
   // programa_001.cpp
   #include "biblaureano.h"

6  int main()
   {
     const int ALIEN_PONTOS=150; // definição
     int aliensMortos = 10;
     int score = aliensMortos * ALIEN_PONTOS;
11
     cout << "Score:" << score;

     // definindo outro tipo de constante
     enum DIFICULDADE {NOVATO, FACIL, NORMAL, DIFICIL, IMPOSSIVEL};
16   DIFICULDADE minhaDificuldade = NOVATO;

     enum ESPACONAVE {CACA=25, BOMBARDEIRO, CRUZADOR=50, DESTROIER=100};

     ESPACONAVE minhaNave = BOMBARDEIRO;
21
     cout << endl << "O upgrade da minha espaçonave para um Cruzador custará "
         << (CRUZADOR - minhaNave) << " pontos." << endl;
     return 0;
   }
```

1.1 Trabalhando com constantes

Definimos a constante *ALIEN_PONTOS* para representar o valor de pontos por espaçonave alienígena destruída:

```
const int ALIEN_PONTOS = 150;
```

Para que isto aconteça deve-se colocar o comando *const* antes da definição da variável, indicando ao compilador que quando detectar uma mudança de valor da variável, seja emitida uma mensagem de erro. Para facilitar a identificação de variáveis constantes, use a primeira letra do nome para identificar o tipo da variável e o restante em maiscúla.

Após a definição da constante, você poderá utiliza-la como se fosse uma variável qualquer:

```
int score = aliensMortos * ALIEN_PONTOS;
```

É possível definir constantes para qualquer tipo de variável (*int*, *float*, *char*, etc).

Não é possível atribuir um novo valor a uma constante durante a execução do programa. Isto causará um erro durante o processo de compilação.

1.2 Trabalhando com listas enumeradas

Uma *lista enumerada* nada mais é que uma lista em que o compilador para cada constante colocada irá atribuir um valor interno, caso o programador não tenha especificado este valor. Por exemplo, na definição:

```
enum DIFICULDADE {NOVATO, FACIL, NORMAL, DIFICIL, IMPOSSIVEL};
```

Será criada um novo tipo de dados (*DIFICULDADE*), cujos os únicos valores possíveis serão os que estão especificados entre { e } (*NOVATO,FACIL, NORMAL, DIFICIL, IMPOSSIVEL*).

Na prática, cada valor é um conjunto de *unsigned int*. Veja o programa 2.

Programa 2: Uso de listas enumeradas.

```
// Uso de constantes
// Alguns testes com listas enumeradas
// programa_002.cpp
#include "biblaureano.h"

int main()
{
  // como não foi especificado valores,
  // a lista será definida iniciando em 0
  enum DIFICULDADE {NOVATO, FACIL, NORMAL, DIFICIL, IMPOSSIVEL};
  DIFICULDADE minhaDificuldade = FACIL;

  cout << "Minha Dificuldade:" << minhaDificuldade << endl;

  cout << "Mudando a dificuldade para difícil.." << endl;
  minhaDificuldade = DIFICIL;
```

```
  cout << "Minha Dificuldade:" << minhaDificuldade << endl;

19

  enum ESPACONAVE {CACA=25, BOMBARDEIRO, CRUZADOR=50, DESTROIER=100};

  ESPACONAVE minhaNave = BOMBARDEIRO;

24 cout << endl << "O upgrade da minha espaçonave para um Cruzador custará "
       << (CRUZADOR - minhaNave) << " pontos." << endl;
  return 0;
}
```

No programa 2 fica evidente a utilização de listas enumeradas. Os enumeradores criados recebem um valor iniciando em 0 (zero). Logo, o primeiro enumerador (*NOVATO*) recebe o valor 0, o segundo (*FACIL*) recebe o valor 1 e assim sucessivamente. Na definição seguinte:

```
enum ESPACONAVE {CACA=25, BOMBARDEIRO, CRUZADOR=50, DESTROIER=100};
```

Foram especificados valores iniciais para os enumeradores, exceto o enumerador *BOMBARDEIRO* que, neste caso, recebe o valor o último enumerador definido $(CACA) + 1$.

Ao especificar uma lista enumerada, cria-se um tipo específico para aquela lista. Bastando então, utilizar este tipo para declarar uma variável deste tipo:

```
DIFICULDADE minhaDificuldade = FACIL;
...
ESPACONAVE minhaNave = BOMBARDEIRO;
```

Os únicos valores válidos para uma variável do tipo lista enumerada sãos os enumeradores definidos. No caso de tentar utilizar um outro valor não definido, ocorrerá um erro durante a compilação.

2 A fortuna perdida!!!

O programa 3 aplica alguns conceitos vistos até o momento. Também introduz alguns novos que serão explicados aqui.

Programa 3: Meu primeiro jogo (A fortuna perdida!).

```
// Juntando tudo
2 // Um jogo simples aplicando vários dos conceitos
// vistos até o momento
// programa_003.cpp
#include "biblaureano.h"

7 int main()
```

```
{
  const int PECAS_OURO = 900;
  int aventureiros, mortos, sobreviventes;
  string lider;

  //pegando as informações
  cout << "Bem vindo ao 'Fortuna Perdido!'.\n\n";
  aventureiros = readInt("Por favor entre com a quantidade de aventureiros:");

  mortos = readInt("Entre com um número, menor que o anterior:");
  sobreviventes = aventureiros - mortos;
  lider = readString("Entre com o seu último nome:");

  //contando a história
  cout << "\n\tUm grupo de " << aventureiros << " bravos aventureiros ";
  cout << "recebeu uma missão - encontrar e recuperar um tesouro guardado ";
  cout << "pelo homem do saco. ";
  cout << "Este grupo era liderado pelo famoso " << lider;
  cout << ". Um grande mestre das armas.";

  cout << "\n\tAo longo do caminho, vários sacis atacaram o grupo. ";
  cout << "Os aventureiros, seguindo os comandos do seu bravo líder " << lider;
  cout << ", derrotaram vários dos atacantes. Mas houve um custo. ";
  cout << "Dos " << aventureiros << " aventureiros, " << mortos;
  cout << " deram suas vidas pela missão. ";
  cout << "Os " << sobreviventes << " restantes continuaram fortes e unidos ";
  cout << " para completar a missão.";

  cout << "\n\tO grupo estava cheio de esperança e cansados, afinal, ";
  cout << "o ouro prometido compensaria todo o esforço. ";
  cout << "Após uma última e árdua luta, o homem do saco foi derrotado. ";
  cout << "Os aventureiros acharam as " << PECAS_OURO << " peças de ouro.";
  cout << "O líder " << lider << " deu a cada aventureiro ";
  cout << (PECAS_OURO / sobreviventes) << " peças de ouro.";
  cout << "As peças restantes (" << (PECAS_OURO%sobreviventes) << ") ele guardou ";
  cout << " para si (junto com as que já ganhou na divisão é claro).";

  cout << "\n\tDepois de cumprirem a brava missão, o grupo seguiu seu caminho em paz.";

  return 0;
}
```

2.1 O que temos de novo

No programa 3, as novidades que aparecem: utilização de *strings* e uma nova sequência de caracteres para a quebra de linha e tabulação.

2.1.1 Strings

Estamos utilizando uma *string* para ler o nome do líder da aventura:

```
string lider;
```

Em C++, uma *string* é uma classe com vários métodos (calma pequeno gafanhoto, logo chegaremos lá), próprio para a manipulação de cadeias de caracteres.

A simples utilização do comando *cin* para ler uma *string* nos traz um problema. Não é possível ler nomes completos, ou seja, separados por espaços em branco. Para resolver esta questão utilizamos a função *readString()* que é fornecida na biblioteca.

2.1.2 Nova sequência de caracteres

Como você deve ter observado no programa 3, em nenhum momento foi utilizado o operador *endl* para a quebra de linhas. Em compensação foram utilizados um novo caracter para cumprir esta função (\n). No no trecho de programa é possível observar a utilização de dois novos caracteres:

```
cout << "\n\tUm grupo de " << aventureiros << " bravos aventureiros ";
```

Estes caracteres, considerados constantes, tem um significado especial para a linguagem C++ (tabela 1).

Tabela 1: Algumas constantes do C++ (existem outras).

Constante	Significado
\n	Nova linha (*new line*)
\t	Tabulação horizontal (*tab*)
\\	Barra invertida

3 Atividade

Escreva um programa que conte uma história e aplique **todos** os conceitos vistos até o momento. Esta atividade poderá ser realizada em dupla e deverá ser entregue para o professor na data estipulada.