Lab. 1

Introdução ao *Code::Blocks*

Francisco Vidal - ECT2303 (T02 - 19.1)

Nesta prática de laboratório, veremos como digitar um programa em C++, compilá-lo e colocá-lo em execução usando o *Code::Blocks*.

1.1 Instalando o *Code::Blocks*

O *Code::Blocks* é um ambiente de desenvolvimento integrado (*IDE - Integrated Development Enviroment*), escrito em C++, de código aberto e multiplataforma.

Você pode fazer o download e instalar a parti de `http://www.codeblocks.org/downloads`.

No *Code::Blocks*, existe duas formas de gerar códigos-fonte:

1. Criando um projeto.
2. Criando um novo arquivo que será o código-fonte.

A criação de um projeto é mais recomendada, pois permite que o desenvolvedor tenha suporte a depuração de código e facilita o gerenciamento quando há vários códigos-fonte e/ou arquivos associados.

De um projeto, gera-se um único arquivo executável após a compilação.

Assim, cada atividade deverá ser criado um projeto diferente.

1.2 Criando e executando um programa

Os passos são:

1. Criar um novo projeto.
2. Entrar com o seu código fonte.
3. Construir um arquivo executável.
4. Executar o programa.
5. Salvar o programa.

1.2.1 Criar um novo projeto

Em *Code::Blocks* o conjunto de arquivos fontes (*.cpp*) e arquivos de biblioteca (*.h*) necessários para criar e executar um programa podem ser armazenados em um arquivo de projeto (**<name>.cbp**).

1. Abrir o IDE do *Code::Blocks*.
2. Abrir o menu **File**, apontar para **New** e clicar em **Project**.
3. Em **New from template**, selecionar **Console application** e clicar em **Go**.
4. No **Console Application Wizard**, selecionar **C++** e clicar em **Next**.
5. Na caixa de texto **Project title**, digite o nome do seu projeto, por exemplo, LAB1_PROJ1. A caixa de texto **Project Filename** automaticamente recebe o mesmo nome (*LAB1_PROJ1.cbp*).
6. Na caixa de texto **Folder to create**, escolha um diretório para o seu projeto e clique em **Next**.
7. Em **Compiler**, selecionar **GNU GCC Compiler**. Marque as opções **Debus** e **Release**.
8. Clicar em **Finish**.

O seu projeto aparecerá no painel **Management** (`<Shift + F2>`), loclizado na lateral esquerda do *Code::Blocks*.

1.2.2 Entrar com o seu código fonte

Para editar o arquivo do seu projeto, no painel **Management**, clique em **Sources** e, em seguida, dê dois cliques no arquivo **main.cpp**. O código a seguir será aberto no editor:

```
1 //Meu primeiro programa em C++
2 #include <iostream>
3
4 using namespace std;
5
```

```
 6 int main()
 7 {
 8    cout << "Hello world!\n";
 9    return 0;
10 } // Fim da main
```

Este é um programa clássico em C++. Ele escreve **Hello world!** na tela de seu monitor.

Vamos entender o que cada linha desse programa faz:

1. Qualquer coisa escrita após o símbolo **//** em uma linha é uma comentário. Comentários são ignorados pelo compilador e servem para descrever alguma informação útil sobre o código. Um outra maneira de comentar uma linha ou várias linhas de código começa com **/*** e termina com ***/**.

2. As linhas iniciadas com **#** são processadas pelo pré-processador antes do programa ser compilado. A diretiva **#include** geralmente tem o sufixo **.h**, chamado de arquivo de cabeçalho, e é utilizada para incluir o conteúdo do arquivo **<iostream >** que define os procedimentos de `entrada/saída`.

4. O comando **using namespace std** permite ao programador acessar todos os membros do **namespace** e escrever comandos

   ```
   cout << "Oi..."<< endl;
   ```

   em vez de

   ```
   std::cout << "Oi..."<< std::endl;
   ```

   Sem a linha 4, cada **cout** e **endl** teria que ser qualificado com **std::**.

6. Todo programa em C++ deve ter uma função **main** para indicar onde inicia a execução. A palavra-chave **int**, à esquerda de **main**, indica que **main** devolve um inteiro (estudaremos com mais detalhes na aula sobre *funções*).

7. A chave à esquerda, **{**, indica o início do corpo da função **main**.

8. A linha **cout << "Hello world!\n";** é chamada de **instrução**. Imprime na tela o *string* de caracteres, **Hello world!** contido entre as aspas, seguido de uma nova linha.

   - **cout <<** : sintaxe para a saída de algum texto para a tela.
   - **"Hello world!\n"** : é uma cadeia de caracteres. Em C++, cadeias de caracteres são delimitadas por aspas (**""**).
   - **\n**: não são exibidos na tela. A barra invertida (****) é chamada de caractere de escape. A sequência de escape **\n** indica uma nova linha, fazendo com o que o cursor se mova para o começo da próxima linha. Outras sequências de escape são listadas na Tabela 1.1.

Sequência de Escape	Descrição
\n	Posiciona o cursor para o início da próxima linha.
\t	Tabulação horizontal.
\r	Posiciona o cursor no início da linha atual.
\a	Alerta sonoro.
...	...

Tabela 1.1: Sequência de escape

- **;** (*ponto-e-virgular*): todas as instruções em C++ terminan com um ponto-e-virgula. Omitir o ponto-e-vírgula no fim de uma instrução em C++ é um **erro de sintaxe**.

  OBS: As diretivas de pré-processador não terminam em ponto-e-vírgula.

9. É utilizada no final da função, o valor **0** indica que o programa terminou com sucesso (na aula sobre funções discutiremos em detalhes a instrução **return**).

10. A chave à direita, **}**, indica o fim do corpo da função **main**.

Agora você pode editar o arquivo e modificar o seu programa.

1.2.3 Construindo um programa executável

Quando concluir o código fonte do seu programa, vá para o menu **Build** e selecione **Build** (<Ctrl + F9>) ou use o íncone triangular que aponta para a direita na lista de ícones próximo do topo da janela do IDE.

Se tiver sucesso, deve aparecer na aba **Build log**, da janela **Log & others**, a seguinte mensagem

```
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))
```

Caso contrário, algumas mensagens de erro vão aparecer.

1.2.4 Executando o programa

Assim que todos os erros tenham sido eliminados, execute o programa indo para o menu **Build** e selecione **Build and run** (<F9>). A saída do seu programa deve aparecer na janela do console.

1.3 Adicionando Inteiros

O programa a seguir utiliza a instrução **cin >>** para receber dois inteiros digitados por um usuário, calcula a soma e imprime o resultado.

```
 1 #include <iostream>
 2
 3 using namespace std;
 4
 5 int main()
 6 {
 7    int n1, n2, soma;
 8
 9    cout << "Digite o primeiro inteiro: ";
10    cin >> n1;
11
12    cout >> "Digite o segundo inteiro: ";
13    cin >> n2;
14
15    soma = n1 + n2;
16
17    cout << "Soma = " << soma << endl;
18
19    return 0;
20 }
```

1.4 Exercícios de Fixação

1. Crie uma pasta dentro da qual serão armazenados todos os seus projetos. Sempre que você criar um novo projeto, o *Code::Blocks* criará automaticamente uma nova subpasta para o referido projeto dentro desta pasta. É importante que você crie a sua pasta de projetos e faça backup (via pendrive ou e-mail) após o final de cada aula.

2. Crie um novo projeto chamado LAB1_PROJ1. Configure o *Code::Blocks* para que o projeto seja armazenado na pasta criada no item anterior. Após a criação do projeto, um arquivo `main.cpp` contendo um algoritmo "*Hello World*" de exemplo deve ter sido adicionado automaticamente ao mesmo. Abra o referido arquivo para edição e verifique o seu conteúdo.

   (a) Compile e execute o algoritmo.

   (b) Modifique o arquivo para exibir a mensagem em várias linhas utilizando uma única instrução.

```
ECT2303 - Linguagem de Programação

TURMA 02 - 46M34
```

(c) Modifique o arquivo para imprimir os números 1 a 4 na mesma linha com cada par de números adjacentes separados por uma tabulação. Faça isso de várias maneiras:

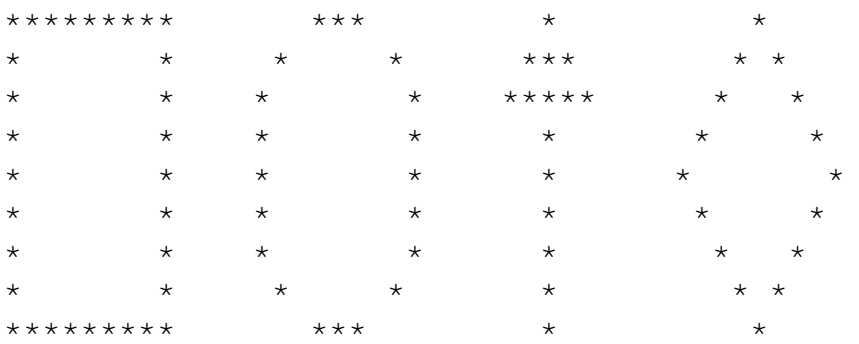
i. Utilizando uma instrução com um operador de inserção de fluxo (**<<**).

ii. Utilizando uma instrução com quatro operadores de inserção de fluxo.

iii. Utilizando quatro instruções.

```
1 2 3 4
1 2 3 4
1 2 3 4
```

(d) Modifique o arquivo para imprimir uma caixa, uma oval, uma seta e um losango da seguinte maneira:



3. Sem fechar o projeto criado no item anterior, crie um novo projeto chamado LAB1_PROJ2, configurando o *Code::Blocks* para que ele também seja armazenado na pasta criada no item 1. Verifique que, para este projeto, também foi criado um arquivo de exemplo chamado `main.cpp`. Abra o referido arquivo para edição e modifique-o para lê o raio de um círculo como um inteiro e imprime o diâmetro, a circunferência e a área do círculo. Utilize o valor constante 3.14159 para π. Faça todos os cálculos em intrução de saída. Em seguida, compile e execute o projeto.

4. Sem fechar o projeto criado no item anterior, crie um novo projeto chamado LAB1_PROJ3, configurando o *Code::Blocks* para que ele também seja armazenado na pasta criada no item 1. Verifique que, para este projeto, também foi criado um arquivo de exemplo chamado `main.cpp`. Abra o referido arquivo para edição e modifique-o para conter o seguinte algoritmo. Em seguida, compile e execute o projeto.

```
1  #include <iostream>
2  #include <iomanip>
3
4  using namespace std;
5  //teste
6  int main() {
7     cout << setiosflags(ios::fixed);
8     cout << setiosflags(ios::showpoint);
9     cout << setprecision(2);
10    cout << setfill('.');
```

```
11      cout << "Constantes" << setw(20) << "Valor" <<
           endl;
12      cout << "Pi " << setw(20) << 3.141592653589793 <<
           endl;
13      cout << "Euler " << setw(20) << 2.718281828459045
           << endl;
14      cout << "Aureo " << setw(20) << 1.618033988749895
           << endl;
15      cout << "Unidade " << setw(20) << 1. << endl;
16
17      return 0;
18  }
```

5. O exemplo mostrado no projeto LAB1_PROJ2 utiliza uma biblioteca extra chamada <iomanip>, que permite manipular o formato da saída do `cout`. A descrição dso comandos utilizados é dada abaixo:

   - **`setioflags()`**: comando utilizado para configurar o modo de apresentação de um número (em ponto flutuante, em notação científica, etc.). Para fazer a alteração, ele utiliza *flags*, que são constantes internas utilizadas para sinalizar uma opção. No exemplo, são utilizadas as seguinte opções:
     - **`ios::fixed`**: Para exibir os valores utilizando notação em ponto fixo (o habitual é em ponto flutuante);
     - **`ios::showpoint`**: Para sempre o ponto decimal, mesmo em caso de valores inteiros;
   - **`setprecision(n)`**: Para exibir um número em ponto flutuante com n casas decimais;
   - **`setw(n)`**: Para definir que o próximo elemento a ser impresso tenha tamanho n. Caso o elemento tenha tamanho menor, espaços em branco são inseridos a direita dela. Caso seja maior, será utilizado o tamanho do elemento ao invés de n;
   - **`setfill(c)`**: Para definir que os espaços em branco do comando **`setw`** sejam preenchido com o caractere indicado por **`c`**;

   Para verificar os efeitos destes comandos, faça as seguintes modificações no programa, uma de cada vez:

   (a) Do código original, troque o valor do comando **`setprecision`** para **`10`**. Então, compile e execute;

   (b) Do código original, troque o caractere do comando **`setfill`** para **`-`**. Então, compile e execute;

   (c) Do código original, comente o comando **`cout << setiosflags(ios::fixed);`**. Então, compile e execute;

   (d) Do código original, comente o comando **`cout << setiosflags(ios::showpoint);`**. Então, compile e execute.

   Sugestão: Crie um projeto a parte para fazer as modificações. Assim, para obter o código original, basta copiar do projeto LAB1_PROJ2 para este.

6. Para exercitar a possibilidade de se trabalhar em dois projetos simultaneamente, mantendo o projeto LAB1_PROJ2 aberto, ative o projeto LAB1_PROJ1, recompile e execute-o novamente. Certifique-se de que a mensagem "*Hello World*" foi exibida na tela. Em seguida, mantendo o projeto LAB1_PROJ1 aberto, volte a ativar o projeto LAB1_PROJ2, recompile e execute-o novamente.

7. Fecheo *Code::Blocks* e verifique o conteúdo da pasta criada por você no item 1 deste laboratório. Verifique quais são aastas e arquivos existentes no diretório e pesquise para que eles servem.

1.5 Referências Bibliográficas

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

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

3. BJARNE STROUSTRUP. **Princípios e Práticas de Programação com C++**. 4ed. Editora Bookman.