

Aula 6 – Introdução a o C++ Programação Sequencial

Algoritmos e Programação de Computadores

Profs: Ronaldo Castro de Oliveira – ronaldo.co@ufu.br

A linguagem C++

- A linguagem C foi desenvolvida no fim da década de 60;
- C++ começou na década de 70 e é uma extensão do C com diversas funcionalidades com orientação a objetos;
- A linguagem C++ é um super conjunto da linguagem C, ou seja, todo e qualquer programa em C também é um programa em C++, mesmo que o oposto não seja verdade.

Primeiro Programa

- O Algoritmo em linguagem C++, abaixo, descreve para o computador os passos necessários para se escrever a mensagem “Olá Mundo!” na tela do computador.
aspectos:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "Ola Mundo!" << endl;
8     return 0;
9 }
```

Área de um retângulo

- A área de um retângulo pode ser facilmente calculada caso você saiba o comprimento de sua base e de sua altura. Matematicamente, seja b o comprimento da base e a a altura. A função f equivalente à área do retângulo pode ser definida como: $f(a; b) = a * b$. Isto é, a função f tem dois parâmetros (a altura a e base b do retângulo) e calcula a área como sendo a multiplicação de a e b .

```
int f(int a, int b)
2 {
    return a * b;
4 }
```

Tipos primitivos da Linguagem C

- O código acima tem a limitação de só calcular a área de retângulos cujos lados tenham tamanhos inteiros.
- Para corrigir esta deficiência, vamos alterá-lo para que aceite números reais. Em computação, números reais são também chamados de números com pontos flutuantes e, em linguagem C, simplesmente de float.
- Podemos corrigir o programa simplesmente substituindo as ocorrências da palavra int por float.

```
float f(float a, float b)
2 {
    return a * b;
4 }
```

Organização do Código

- É possível perceber um padrão nos exemplos:
 - A linha definindo a função é seguida por uma linha contendo apenas um { que é alinhado com o início da linha acima.
 - A última linha da função contém apenas um }, alinhado com o { do início da função.
 - Todas as linhas entre o { inicial e o } final estão com alinhamento mais avançadas em relação às chaves.
 - Os { e } representam a especificação de um bloco de código (início e fim)

Comentários

- Algo que faltou nestes exemplos e que também serve ao propósito de facilitar o entendimento do código são os chamados comentários.

- `/*` → comentários por blocos

.....
.....*/

- `//` → comentários de linha

```
/*  
2  * A funcao a seguir calcula a area de um retangulo de base  
   * base e altura altura. Os parametros e resultado da funcao  
4  * sao do tipo float.  
   */  
6 float area_retangulo(float altura, float base)  
   {  
8     //Calcula e retorna a area do retangulo.  
     return altura * base;  
10 }
```

Saída de dados

- Um programa está executando a saída de dados quando envia para “fora” do programa tais dados. Exemplos comuns de saída de dados são a escrita em arquivo, o envio de mensagens na rede ou, impressão ou, mais comum, a exibição de dados na tela.
- Para enviar dados para a saída do C(++), usamos a expressão `cout <<`, seguido do dado a ser impresso na tela.

```
cout << "numero ";  
2 cout << 10;
```

- Imprime na tela:
numero 10

OBS: a palavra `numero` no programa aparece entre aspas duplas e `10` não. Isto ocorre por quê `numero` é um texto, e `10` é um número inteiro;

Saída de dados

- Variações de saída:

```
cout << "numero " << 10;
```

Imprime na tela:

numero 10

```
cout << "numero " << 10 << endl << "texto " << endl;
```

Imprime na tela:

numero 10

texto

```
cout << "sen(1)" << endl << sen(1);
```

Imprime na tela:

sen(1)

0

A função main()

```
1 #include <iostream>
2
3 using namespace std;
4
5 /*
6  * A funcao a seguir calcula a area de um retangulo de base
7  * base e altura altura. Os parametros e resultado da funcao
8  * sao do tipo float.
9  */
10 float area_retangulo(float altura, float base)
11 {
12     //Calcula e retorna a area do retangulo.
13     return altura * base;
14 }
15
16 int main()
17 {
18     //Calculemos a area de alguns retangulos.
19     cout << area_retangulo(3.3, 2.0) << endl;
20     cout << area_retangulo(2.0, 2.0) << endl;
21
22     //Lembre-se, todo numero inteiro tambem e um numero real.
23     cout << area_retangulo(4, 2) << endl;
24
25     return 0;
26 }
```

A função main()

- Algumas observações importantes sobre a função main:
 - A função main tem sempre um resultado do tipo inteiro e seu resultado é sempre o (return 0;);
 - Só pode haver uma Função main para cada programa;
 - Esta regra vale para toda e qualquer função, ou seja, não se pode ter nomes repetidos de funções;
 - Finalmente, a função area_retangulo aparece antes da função main no programa. Isto deve ser verdade para todas as funções do seu programa. Isto ocorre por quê, antes de executar a função main, o computador precisa aprender sobre a existência das outras funções.

Compilação e Execução

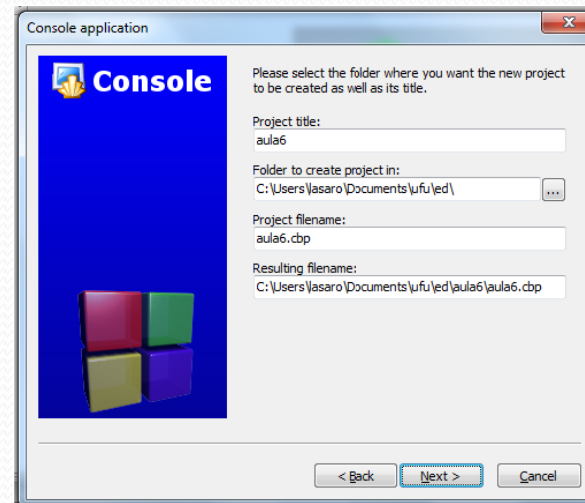
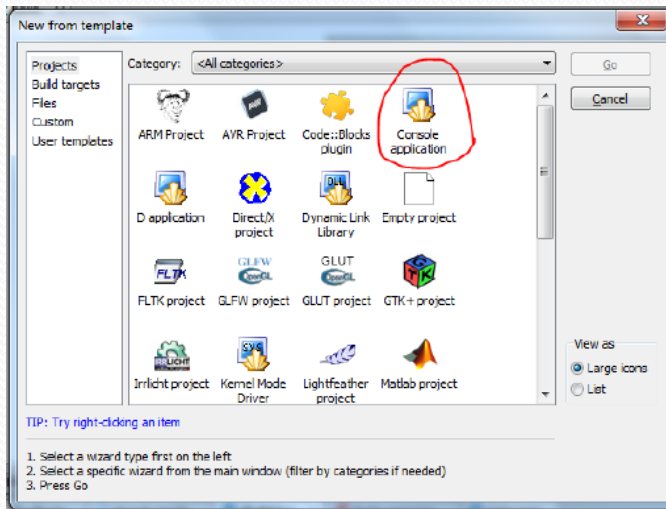
- Para colocarmos nossos algoritmos em execução, o primeiro passo é escrevê-los, usando um editor de textos qualquer que salve arquivos em texto puro, como o notepad, vim, gedit, etc. A este arquivo com o código chamaremos código fonte ou simplesmente fonte (extensão . Cpp).
- A sequência de passos que compõem a compilação é a seguinte:
 - **Código Fonte → Pré-processador → Fonte Expandido → Compilador → Arquivo Objeto → Ligador → Executável**
- A compilação traduz o código que você escreveu para uma linguagem inteligível ao computador, salvando-o em um arquivo chamado arquivo objeto. Por exemplo, a compilação transformaria o código “Olá Mundo!” escrito acima em algo como:

```
...  
CALL write(0x1,0x400623,0xe)  
GIO fd 1 "Olá Mundo!"  
RET  
...
```

OBS: para um primeiro programa:
primeiroProg.cpp
primeiroProg.obj
primeiroProg.exe

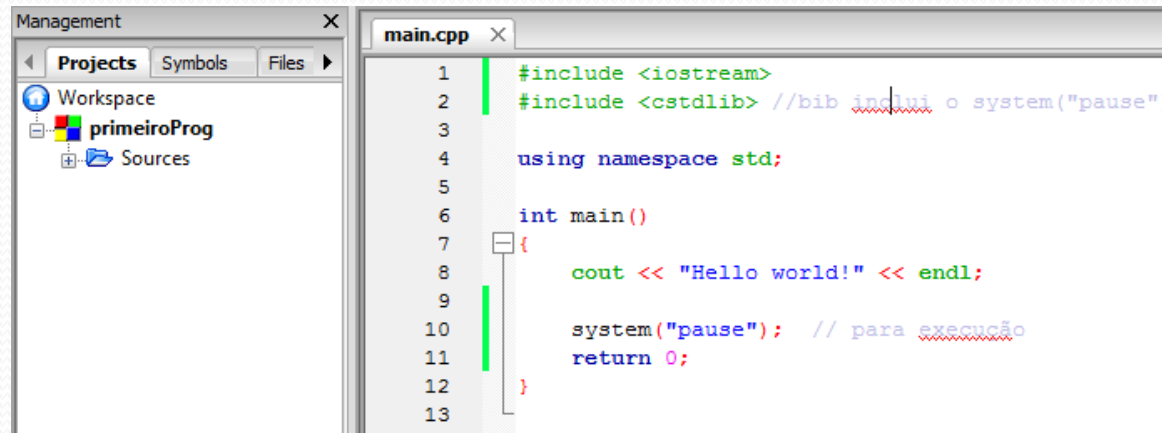
A IDE Code::Blocks

- Criando um Projeto: clique em **File** e, em seguida, **New, Project**;
- Escolha **Console Application** e então clique em **Go**;
- Escolha **C++** e clique em **Next**;
- Em **Project title** escreva algo como **teste1**; em **Folder to create the project in**, clique no botão com **...** e escolha uma pasta para salvar o projeto. Pode ser a pasta Meus Documentos ou uma pasta qualquer em um pen drive. Clique então **Next** e, na tela seguinte, clique em **Finish**.



A IDE Code::Blocks

- Seu projeto foi criado. Agora abra o arquivo main.cpp, que está na pasta sources, dando um clique duplo no nome do arquivo. Observe que o Code::Blocks criou automaticamente um programa básico.

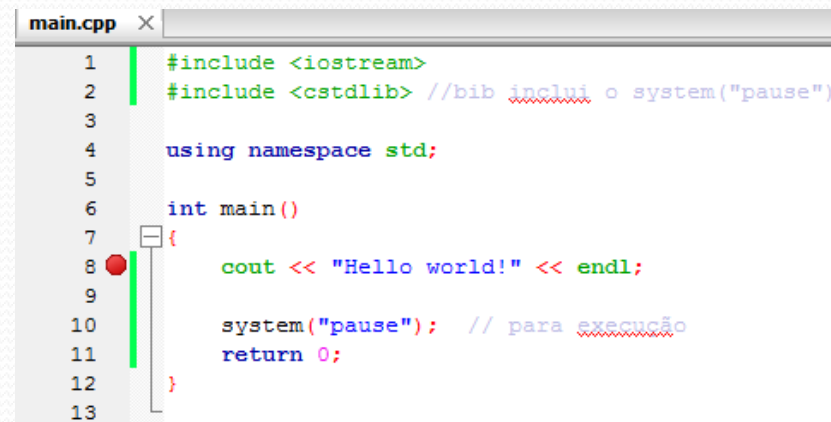


```
1 #include <iostream>
2 #include <cstdlib> //bib inclui o system('pause')
3
4 using namespace std;
5
6 int main()
7 {
8     cout << "Hello world!" << endl;
9
10    system("pause"); // para execução
11    return 0;
12 }
13
```

- Clique em em build and run  . Parabéns, você acaba de executar seu primeiro programa.

Depuração

- Todo programa é comum encontrar erros (bugs) de codificação e de lógica. Uma das formas de achar os bugs do seu programa é fazer com que o computador execute seu programa passo a passo, isto é, linha a linha, e acompanhar esta execução verificando se o programa faz o que você espera.



```
main.cpp x
1  #include <iostream>
2  #include <cstdlib> //bib inclui o system("pause")
3
4  using namespace std;
5
6  int main()
7  {
8  cout << "Hello world!" << endl;
9
10     system("pause"); // para execução
11     return 0;
12 }
13
```

- Para depurar, clique ao lado direito do número 8 (oitava linha do programa), até que uma bolinha vermelha apareça, como na figura. A bolinha vermelha é, na verdade, um sinal de pare, e diz ao computador que deve, ao executar seu programa, parar ali.
- Clique no menu **Debug** e então em **Start** ou, alternativamente, pressione a tecla **F8** (▶). Observe que a execução parou onde você esperava.
- Agora, clique em **Debug** e **Next Line** ou aperte **F7** (🔍), no teclado, sucessivamente para ver o que acontece. Observe que cada linha é executada passo a passo.

Declaração de Variáveis

- Na linguagem C, toda variável deve ser declarada (isto é, criada) no início do corpo da função que a contém. A declaração de uma variável tem pelo menos duas partes:
 - Tipo: tipo de dado, ou seja, se é um número, ou uma palavra, ou uma caractere, etc;
 - Nome: usado para referenciar a variável quando se precisa ler ou escrever a mesma;
- Algumas regras simples devem ser seguidas na hora de se nomear uma variável:
 - o nome só pode conter os caracteres [a-z], [A-Z], [0-9] e o “_”;
 - o nome não pode começar com números.
- Tipos básicos:
 - **int** - representando um número inteiro, como por exemplo 3, 4 e -78;
 - **float** - representando um número real, com casas decimais separadas por ponto “.” como por exemplo 3.1416 e -1.2;
 - **char** - representando um caractere (letra, dígito, sinal de pontuação) identificado por apóstrofes. Exemplo ‘5’, ‘a’, ‘Z’, ‘.’, ‘e’, ‘-’.
- Exemplo:

São exemplos de declarações de variáveis válidas:

```
1 int nota1, nota2;  
2 float media;  
3 char _caractere;
```

São exemplos de declarações inválidas:

```
1 int 1nota, 2nota;  
2 float #media;  
3 char nome completo;
```


Atribuição e uso de variáveis

```
1 int inteiro1, inteiro2;  
  float real;  
3  
  inteiro1 = 0;  
5 inteiro2 = 10;  
  real = 10.0;
```

```
int inteiro1 = 0,  
    inteiro2 = 10;  
float real = 10.0;
```

Parâmetros são variáveis:

```
float area_retangulo(float altura, float base)  
2 {  
    //Calcula e retorna a area do retangulo.  
4    return altura * base;  
    }  
6  
int main()  
8 {  
    float area;  
10    area = area_retangulo(2.0, 2.0);  
    cout << area;  
12  
    return 0;  
14 }
```

Entrada de dados

- De forma semelhante ao **cout**, há um comando para leitura denominado **cin**. Este comando permite ler valores digitados pelo usuário atribuindo a variáveis definidas por meio do conector **>>**.

```
char letra;  
2 int idade;  
  
4 cout << "Informe a letra inicial de seu nome e sua idade: ";  
  // a seguir eh feita a leitura  
6 cin >> letra >> idade;  
  cout << "A letra eh " << letra;  
8 cout << " e sua idade eh " << idade << endl;
```

Saída de dados

- Imprimindo conteúdos de variáveis:

```
1 char letra = 'a';  
2 int num = 2;  
3 cout << "letra = " << letra << endl << "num = " << num << endl;
```

Imprime na tela:

```
letra = a  
num = 2
```

- Programa completo:

```
1 float area_retangulo(float altura, float base)  
2 {  
3     //Calcula e retorna a area do retangulo.  
4     return altura * base;  
5 }  
6  
7 int main()  
8 {  
9     float area,  
10         b,  
11         a;  
12     cout << "Qual a altura do retangulo?" << endl;  
13     cin >> a;  
14  
15     cout << "Qual a base do retangulo?" << endl;  
16     cin >> b;  
17  
18     area = area_retangulo(b, a);  
19     cout << "A area do retangulo de base " << b << " e altura "  
20         << a << " eh " << area << endl;  
21  
22     return 0;  
23 }
```

Imprime na tela:

```
Qual a altura do retangulo?  
5  
Qual a base do retangulo?  
7  
A area do retangulo de base 7  
e altura 5 eh 35
```

Formatação de impressão

- Em algumas ocasiões há necessidade de formatar a saída para, por exemplo, garantir que os dados fiquem alinhados, imprimir uma tabela, ou simplesmente por estética (**setw(<valor>)**, **right**, **left** e **setfill(<caracter>)**).

```
1 #include <iostream>
2 #include <iomanip>
3
4 using namespace std;
5
6 float volume_cubo(float aresta)
7 {
8     return aresta*aresta*aresta;
9 }
10
11 int main()
12 {
13     float a, v;
14     cout << "Entre valor da aresta do cubo:" << endl;
15     cin >> a;
16     v = volume_cubo(a);
17     cout << setw(30) << left << "O volume do cubo eh: " << v << endl;
18     cout << setfill('-') << setw(30) << left << "O volume do cubo eh: " << v << endl;
19     cout << setw(30) << left << "O volume do cubo eh: " << setw(20) << v <<
20     endl;
21     cout << setw(30) << "O volume do cubo eh: " << setw(20) << right
22     << v << endl;
23     cout << setw(30) << left << "O volume do cubo eh: " << v << endl;
24     return 0;
25 }
```

Imprime na tela:

Entre valor da aresta do cubo:

2.5

O volume do cubo eh: 15.625

O volume do cubo eh: -----15.625

O volume do cubo eh: -----15.625-----

O volume do cubo eh: -----15.625

O volume do cubo eh: -----15.625

Formatação de impressão

- Para formatação de números reais (float e double), o exemplo a seguir mostra alguns comandos para formatação:

```
1 #include <iostream>
2 #include <iomanip>
3
4 using namespace std;
5
6 float volume_cubo(float aresta)
7 {
8     return aresta*aresta*aresta;
9 }
10
11 int main()
12 {
13     float a, v;
14     cout << "Entre valor da aresta do cubo:" << endl;
15     cin >> a;
16     v = volume_cubo(a);
17     cout << "O volume do cubo eh: " << v << endl;
18     cout << fixed << setprecision(2);
19     cout << "O volume do cubo eh: " << v << endl;
20     cout << fixed << setprecision(4);
21     cout << "O volume do cubo eh: " << v << endl;
22     return 0;
23 }
```

Imprime na tela:

Entre valor da aresta do cubo:

4

O volume do cubo eh: 64

O volume do cubo eh: 64.00

O volume do cubo eh: 64.0000

Operadores

- Matemáticos:
 - = (igual), + (soma), - (subtração), * (multiplicação), / (divisão) e % (resto da divisão)
 - OBS: $a += b \rightarrow a = a + b$; $x *= y \rightarrow x = x * y$
- Relacionais:
 - == (teste de igualdade), <> (diferente), > (maior que), < (menor que), >= (maior ou igual) e <= (menor ou igual)
- Lógicos:
 - && (and), || (or), ! (not)
- Funções
 - abs(X): obtém o valor absoluto de X;
 - sqrt(X): calcula a raiz quadrada de X;
 - log(X): calcula o logaritmo de X;
 - mod(X,Y): obtém o resto da divisão de X por Y;
 - trunca(X): obtém a parte inteira de X;
 - round(X): arredonda o valor de X;
 - sen(X): calcula o valor do seno de X;
 - cos(X): calcula o valor do cosseno de X;
 - tan(X): calcula o valor da tangente de X.

Escopo de Variáveis

```
1 float area_retangulo(float altura, float base)
2 {
3     //Calcula e retorna a area do retangulo.
4     return altura * base;
5 }
6
7 int main()
8 {
9     float area,
10    b,
11    a;
12
13    cout << "Qual a altura do retangulo?" << endl;
14    cin >> a;
15
16    cout << "Qual a base do retangulo?" << endl;
17    cin >> b;
18
19    area = area_retangulo(b, a);
20    cout << "A area do retangulo de base " << b << " e altura "
21        << a << " eh " << area << endl;
22
23    return 0;
24 }
```

Posso chamar as variáveis **float a, b;** de **float altura, base;**?

Esta mudança afeta alguma coisa na função **area_retangulo**?

Estas mudanças não afetaram a execução do programa. Isto acontece por que as variáveis tem **escopos bem definidos** em C++. A **variável altura** da função **main** não é a mesma **variável/parâmetro altura** da função **area_retangulo**; cada uma só existe dentro do corpo da função em que foi declarada. Quando a função **area_retangulo** é invocada passando-se como parâmetro a variável **altura** da função **main**, o **valor desta variável é copiado para o parâmetro altura** da função invocada.

OBS: definição de constantes

#define PI 3.141559

Deve ser escrito no começo do programa antes depois dos includes e antes do código.

Lendo e imprimindo Strings

- Declarando uma string:

`char nome[30];` → vetor de caracteres

- Lendo uma string:

`cin >> nome;`

→ lê somente uma
única palavra

`cin.getline (nome, 30);`

→ lê uma frase até
30 caracteres

- Imprimindo uma string:

`cout << nome;`

→ imprime nome
palavra ou frase

```
main.cpp x
1  #include <iostream>
2  #include <cstdlib> //bib inclui o system("pause")
3
4  using namespace std;
5
6  int main()
7  {
8      char nome[30];
9      int P1, P2, P3;
10     float media;
11     cout << "Calculo de media de notas de um aluno." << endl;
12     cout << "Digite o nome do aluno: ";
13     cin.getline(nome, 30);
14     cout << "Digite a nota 1 do aluno: ";
15     cin >> P1;
16     cout << "Digite a nota 2 do aluno: ";
17     cin >> P2;
18     cout << "Digite a nota 3 do aluno: ";
19     cin >> P3;
20     media = (P1 + P2 + P3)/3;
21     cout <<"A media de notas do aluno " << nome << " eh: " << media << endl;
22
23     system("pause"); // para execucao
24     return 0;
25 }
```