Unidade 5: Introdução à Programação com C/C++

Prof. Daniel Caetano

Objetivo: Explicitar os elementos básicos envolvidos na programação com a linguagem C/C++.

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

INTRODUÇÃO

Nas aulas anteriores estudamos os conceitos básicos de programação e, em algum detalhe, a linguagem Português Estruturado (ou Portugol). Nesta aula os mesmos conceitos serão apresentados com relação à linguagem C/C++.

Como será possível ver, a linguagem C/C++ não difere, essencialmente, do Portugol; entretanto, o C/C++ é uma linguagem mais burocrática, isto é, exige ainda maior atenção do programador durante a digitação.

Ao final desta aula, você saberá como declarar variáveis, como ler e escrever valores, como executar expressões aritméticas e armazenar seus resultados em variáveis, usando a linguagem C/C++, além de ser capaz de gerar um programa executável a partir deste código.

1. INTRODUÇÃO À LINGUAGEM C/C++

A linguagem *portugol* é importante para aprendermos, mas profissionalmente ela não é usada. A função do *portugol* é proporcionar uma transição suave entre os fluxogramas e outras linguagens de programação profissionais, como C/C++.

A grande maioria das regras permanece a mesma, de maneira que os conceitos vistos antreriormente para o *portugol* serão reapresentados aqui com o formato de C/C++.

A estrutura de um programam mínimo em C/C++ tem a seguinte "cara":

```
#include <iostream>
using namespace std;
main() {

      // Aqui será escrito nosso programa

}
```

Não se preocupe, neste instante, com o significado destas coisas todas; elas serão explicadas mais adiante. Observe, porém, o uso dos sinais de chaves: **{** e **}**. Estes sinais são fundamentais porque eles marcam onde nosso algoritmo começa e onde acaba: eles têm a função das palavras ALGORITMO e FIMALGORITMO, respectivamente.

Uma coisa **muito importante** neste momento é que **o C/C++ são sensíveis a maiúsculas e minúsculas**. Isso significa que os textos **devem ser digitados exatamente na forma como foram apresentados**.

NOTA: Os únicos lugares onde não é necessário uma preocupação maior com maiúsculas e minúsculas são nos comentários (linhas que se iniciam com **//**) ou em textos que serão impressos, representados entre aspas: **"tExTo"**.

Adicionalmente - e diferentemente do *portugol*, **toda linha de código deve ser finalizada com um sinal de ponto-e-vírgula: ; .**

NOTA: Nesse código apresentado, desenvolvido para o **Code::Blocks, não inclui nenhuma linha para "pausar" a saída final.**

Caso use o Dev-C++, inclua a linha abaixo antes do último **}**:

```
system("PAUSE");
```

Para conseguir o mesmo efeito no OpenWatcom, acrescente a linha no final do código:

```
getchar();
```

E, no início do programa, acrescente:

```
#include <stdio.h>
```

A função dessa linha é pedir ao sistema operacional que, ao final da execução, **NÃO** feche a janela até que apertemos uma tecla, para que possamos ver o resultado de nosso programa! No Code::Blocks isso não é necessário porque o próprio compilador mantém a janela aberta até que apertemos uma tecla!

2. VARIÁVEIS

Assim como no *portugol*, no C/C++ as variáveis precisam ser declaradas antes de serem usadas: isso significa que precisamos definir **um nome** e **um tipo** para cada uma delas. A sintaxe para isso no C/C++ é a seguinte:

```
tipo_da_variavel   nome_da_variavel;
```

Observe que o que separa o tipo da variável do nome da variável é um **espaço**, e **ele é importante**. Observe, ainda, que a linha termina com um sinal **;** e ele **também é importante.** Assim, domando como base o código base anterior, para declarar a variável idade, faríamos o seguinte:

```
#include <iostream>
using namespace std;
main() {

      int idade;

}
```

Observe que agora não usamos mais a palavra "inteiro". **Inteiro**, **real**, **caractere** e **logico** são declaradores de ***portugol***. No C/C++ usamos os seguintes equivalentes:

Portugol	**C/C++**	
Inteiro	int	long
Real	float	double
Caractere	char	
Literal	string	
Logico	bool	

A diferença entre **int** e **long** é o tamanho dos números que eles podem armazenar: int guarda números de -2 bilhões a +2 bilhões, aproximadamente; o long, por outro lado, armazena valores bem maiores.

Com relação ao **float** e o **double**, a diferença é similar: como ambos são números reais (isto é, com casas decimais), double armazena o **dobro** de casas decimais que o float. Em geral **int** e **float** são suficientes para a grande maioria dos usos.

NOTA: Podemos definir que uma variável int/long é **sem sinal** usando a palavra **unsigned**. Por exemplo:

unsigned int idade;

Isso significa que a idade pode variar entre 0 e 4 bilhões, ao invés de -2 bilhões a +2 bilhões.

Da mesma forma que em *portugol*, podemos declarar variáveis em várias linhas:

```
#include <iostream>
using namespace std;
main() {

      int idade;
      int dia;
      float nota;

}
```

Ou podemos fazê-lo com uma linha por tipo:

```
#include <iostream>
using namespace std;
main() {

	int idade, dia;
	float nota;

}
```

3. OPERADOR DE ATRIBUIÇÃO

Tudo que foi dito para o operador atribuição de *portugol* vale para C/C++, as únicas diferenças é que usaremos o sinal de igualdade **=** para representar a atribuição e a linha precisa ser finalizada com um ponto-e-vírgula:

nome_da_variavel **=** expressao**;**

Da mesma forma, o lado esquerdo só pode ser uma variável, e o lado direito é uma expressão que sempre será processada **antes** que a atribuição ocorra.

```
#include <iostream>
using namespace std;
main() {

	int idade;
	idade = 20;

}
```

4. OPERADORES MATEMÁTICOS

Os operadores matemáticos do C/C++ são, basicamente, os mesmos do portugol. Os símbolos usados para cada um deles estão indicados a seguir, bem como sua prioridade:

Operação	Sinal	Prioridade
Adição:	+	1
Subtração:	-	1
Multiplicação:	*	2
Divisão:	/	2
Resto da Divisão:	%	2

A divisão inteira e a exponenciação não existem na forma de operadores.

OPCIONAL: Existem, porém, alguns operadores especiais, usados em situações específicas:

Operador	Uso	Equivalência
+=	x += y	x = x + y
-=	x -= y	x = x -y
*=	x *= y	x = x*y
/=	x /= y	x = x/y
%=	x %= y	x = x%y
++	x++	x = x+1
++	y = x++	y = x; x = x+1
++	y = ++x	x = x + 1; y = x
--	x--	x = x+1
--	y = x--	y = x; x = x - 1
--	y = --x	x = x - 1; y = x

Exemplos de operações:

X = 37 % 7; X = A + B * 2; X = (A + B) * 2;

```
#include <iostream>
using namespace std;
main() {

	int idade;
	idade = 20;
	idade = idade + 10;

}
```

5. SAÍDA DE DADOS

No C/C++ existem diversas formas de realizar a saída de dados. Inicialmente trabalharemos com "comando" **cout** . A sintaxe é a seguinte:

cout **<<** dado_a_ser_escrito**;**

O **dado a ser escrito** pode ser um número:

```
#include <iostream>
using namespace std;
main() {

	cout << 10;

}
```

Um texto (lembrando que texto deve SEMPRE vir entre aspas):

```
#include <iostream>
using namespace std;
main() {

	cout << "Um texto qualquer";

}
```

Ou, ainda, pode ser um nome de variável, situação na qual **o valor da variável** será impresso:

```
#include <iostream>
using namespace std;
main() {

	int idade;
	idade = 20;
	idade = idade + 10;
	cout << idade;

}
```

O comando cout aceita, ainda, que indiquemos uma expressão matemática, quando então ele **imprime o resultado da expressão**:

```
#include <iostream>
using namespace std;
main() {

	cout << (10+20)*3;

}
```

Podemos compor uma linha com várias instruções **cout**:

```
#include <iostream>
using namespace std;
main() {

	int idade;
	idade = 20;
	cout << "Minha idade é: ";
	cout << idade;

}
```

Ou podemos pedir que um único **cout** imprima várias coisas:

```
#include <iostream>
using namespace std;
main() {

	int idade;
	idade = 20;
	cout << "Minha idade é: " << idade;

}
```

Observe a repetição do sinal **<<** entre os diferentes dados a imprimir.

NOTA: Para "pular uma linha" após o texto escrito, use o código **endl**:

cout << "Primeira Linha" << endl << "Segunda Linha";

6. ENTRADA DE DADOS

A entrada de dados no C++ é feita com o "comando" **cin**. Ele tem a seguinte sintaxe:

cin **>>** nome_da_variavel **;**

O valor digitado pelo usuário **deve** ser do mesmo tipo da variável. Observe o programa abaixo:

```
#include <iostream>
using namespace std;
main() {

	int idade;
	cout << "Digite sua idade: ";
	cin >> idade;
	cout << "Sua idade é: " << idade;

}
```

Como idade é um inteiro, se o usuário digitar algo como **20,5** ou mesmo escrever a palavra **vinte** , o programa provavelmente vai parar, indicando um erro.

7. EXEMPLO

Como exemplo, vamos apresentar o programa que calcula a média de dois números, feito na aula passada, no formato em C/C++:

```
#include <iostream>
using namespace std;
main() {

      int n1, n2, m;
      cout << "Calcula a média de dois números" << endl;
      cout << "Digite o primeiro número: ";
      cin >> n1;
      cout << "Digite o segundo número: ";
      cin >> n2;
      m = (n1+n2)/2;
      cout << "A média é: " << m;

}
```

Observe que este programa, desta forma, dá uma responsta truncada se fizermos, por exemplo, a média entre 2 e 3 (que deveria ser 2,5, mas o programa mostra como 2). Isso ocorre porque declaramos as variáveis como **int**.

Vamos modificar o programa, declarando-as agora como **float** e vejamos o que ocorre:

```
#include <iostream>
using namespace std;
main() {

      float n1, n2, m;
      cout << "Calcula a média de dois números" << endl;
      cout << "Digite o primeiro número: ";
      cin >> n1;
      cout << "Digite o segundo número: ";
      cin >> n2;
      m = (n1+n2)/2;
      cout << "A média é: " << m;

}
```

O resultado apareceu certo agora, não? Pois é! A seleção dos **tipos das variáveis** é uma decisão importante!

8. BIBLIOGRAFIA

ASCENCIO, A.F.G; CAMPOS, E.A.V. **Fundamentos da Programação de Computadores**. 2ed. Rio de Janeiro, 2007.

MEDINA, M; FERTIG, C. **Algoritmos e Programação: Teoria e Prática**. 2ed. São Paulo: Ed. Novatec, 2006.

SILVA, I.C.S; FALKEMBACH, G.M; SILVEIRA, S.R. **Algoritmos e Programação em Linguagem C**. 1ed. Porto Alegre: Ed. UniRitter, 2010.