Lab. 2

Variáveis, Operadores e Expressões

ECT2303 - T02 - 19.1

2.1 Variáveis

Toda variável armazenada na memória do computador tem um **nome**, um **valor**, um **tipo** e um **tamanho**.

Um nome de variável é qualquer *identificador* válido. Um *identificador* pode conter letras, dígitos, e o caractere sublinhado (_), que não começa com um digíto.

C++ é sensível a maiúsculas e minúsculas, ou seja, `n1` e `N1` são diferentes.

As declarações de variáveis podem ser colocadas em quase qualquer lugar dentro de uma função. No entanto, a declarção deve aparecer antes de a mesma ser usada no programa.

As variáveis são declaradas após a especificação de seus tipos.

Os **tipos de dados** mais comuns em C++ utilizados na declaração de variáveis são: **int** (para números inteiros), **float** (para números reais), **double** (números reais com dupla precisão), **bool** (valor lógico), **char** (para um caractere) e **void** (sem valor).

A partir desses tipos básicos, podem ser definidos novos tipos utilizando modificadores, tais como: **unsigned int**, **long int**, **unsigned long int**, **long double**, etc.

2.2 Operadores

- Os **operadores matemáticos** básicos são os mesmos que os disponíveis na maioria das linguagens de programação: **adição** (+), **subtração** (-), **divisão** (/), **multiplicação** (*) e **módulo** (%; produz o restante da divisão inteira). O operador de módulo não pode ser usado com números de ponto flutuante.
- Os **operadores lógicos** E (`&&`) e OU (`||`) produzem um verdadeiro ou falso com

base na relação lógica de seus argumentos. Lembre-se disso, em C++, uma declaração é verdadeira se tiver um valor diferente de zero e falsa se tiver um valor zero. O operador lógico **!** é usado para **negação**. Se você imprimir um **bool**, normalmente verá um '**1**' para **verdadeiro** e '**0**' para **falso**.

- O **operador sizeof** retorna o tamanho, em *bytes*, do tipo de dado durante a compilação do programa. Pode ser aplicado a qualquer nome de variável, nome de tipo, expressão ou valor constante. Por exemplo,

```
cout << "O tamanho de char = " << sizeof(char) << '
   \n' << sizeof('a') <<'\n' << sizeof(2+2);
```

- Os **operadores relacionais** (**>, <, >=, <=**) e de **igualdade** (**==, !=**) são utilizados para testar a relação entre duas expressões, retornando um valor **bool**. As condições em comandos **if** são comumente formadas usando-se operadores de igualdade e operadores relacionais. Por exemplo,

```
 1 #include <iostream>
 2
 3 using namespace std;
 4
 5 int main()
 6 {
 7   int a, b;
 8
 9   cout << "Digite dois inteiros:\n";
10   cin >> a >> b;
11
12   if ( a == b )
13     cout << a << " é igual a " << b << endl;
14
15   if ( a != b )
16     cout << a << " é diferente de " << b << endl;
17
18   if ( a < b )
19     cout << a << " é menor que " << b << endl;
20
21   if ( a > b )
22     cout << a << " é maior que " << b << endl;
23
24   if ( a <= b )
25     cout << a << " é menor que ou igual a " << b
26          << endl;
27   if ( a >= b )
28     cout << a << " é maior que ou igual a " << b
29          << endl;
30
31   return 0;
32 }
```

2.3 Expressões

C++ aplica os operadores nas expressões aritméticas em uma sequência determinada pelas seguintes regras de **precedência de operadores**.

1. Operadores em expressões dentro de pares de **parênteses** são calculados primeiro.
2. As operações de **multiplicação**, **divisão** e **módulo** são aplicadas em seguida, estão no mesmo nível de precedência e são aplicados da esquerda para a direita.
3. As operações de **adição** e **subtração** são aplicadas por último, estão no mesmo nível de precedência e são aplicados da esquerda para a direita.

O exemplo seguinte contém as operações módulo (`%`), multiplicação, divisão, adição e subtração.

```
y = a * b % c + d / e - f;
```

2.3.1 Expressões constantes

Variáveis constantes podem ser colocadas em qualquer lugar em que uma expressão constante é esperada.

Para declarar a variável constante, coloque o qualificador **`const`** antes do especificador de tipo na declaração da variável. Por exemplo,

```
const double PI = 3.14159;
const int x = 10000;
const int arraySize = 15;
```

Em C++, uma variável **`const`** deve sempre ser inicializada quando ela é declarada.

2.3.2 Conversões

Tipos **inteiros** e de **ponto flutuante** podem ser misturados livremente em atribuições e expressões.

Se um operador tem um operando do tipo **`double`**, usamos aritimética de ponto flutuante, produzindo um resultado **`double`**, caso contrário, usamos arimética inteira, produzindo um resultado **`int`**. Por exemplo:

```
double d = 2.5;
int i = 2;
double d2 = d/i; //d2 == 1.25
int i2 = d/i; //i2 == 1
```

As conversões implicitas que preservam valores são comumente referidas como *promoções*.

Um programador pode definir conversões para tipos definidos pelo usuário utilizando o operador **static_cast <tipo> (valor)**. A conversão ocorre apenas quando o programador requer explicitamente uma conversão. Por exemplo:

```
int i = 32767
float f;
f = static_cast<float>(i);
```

2.4 Exercícios de Fixação

1. Crie um programa que imprime o tamanho, em *bytes*, dos tipos básicos em C++, com e sem seus modificadores, pelo compilador na sua máquina (utilize o operador **sizeof**). Verifique se eles são, de fato, iguais aos valores mostrados em sala de aula.

2. Escreva um programa que mostra o valor de **x** após cada comando ser executado.

   (a) `x = 7 + 3 * 6 / 2 - 1;`

   (b) `x = 2 % 2 + 2 * 2 - 2 / 2;`

   (c) `x = (3 * 9 * ( 3 + ( 9 * 3 / ( 3 ))));`

3. Escreva um programa que solicita ao usuário inserir dois números, obtém os dois números do usuário e imprime a soma, produto, diferença e quociente dos dois números.

4. Escreva um programa que insere três números a partir do teclado e imprime a soma, a média, o produto, o menor e o maior desses números. O diálogo de tela deve se parecer com o seguinte:

   ```
   Entre com três valores inteiros: 10 15 5
   Soma: 30
   Média: 10
   Produto: 750
   O menor: 5
   O maior: 15
   ```

5. Escreva uma programa que lê um iteiro e determina e imprime se ele é ímpar ou par (**Dica:** Utilize o operador módulo. Um número par é múltiplo de dois. Qualquer múltiplo de dois deixa um resto de zero quando dividido por 2.).

6. Escreva uma programa que lê dois iteiros e determina e imprime se o primeiro é múltiplo do segundo (**Dica:** Utilize o operador módulo.).

7. Crie um programa que receba um valor em decimal do usuário, e retorna a parte inteira e a parte fracionária deste número. Assuma que o usuário pode digitar um

valor positivo ou negativo como entrada (**Dica:** Utilize a noção de conversão entre tipos para conseguir definir a parte inteira de um número em ponto flutuante.).

A saída do programa deve ser dada como a seguir:

```
-- Primeiro exemplo:
Digite um numero: 2.43
Parte inteira: 2
Parte fracionaria: 0.43
-- Segundo exemplo:
Digite um numero: -5.7602
Parte inteira: -5
Parte fracionaria: -0.7602
-- Terceiro exemplo:
Digite um numero: 0.25
Parte inteira: 0
Parte fracionaria: 0.25
```

8. Escreva um programa que insere um inteiro de cinco dígitos, separa o inteiro em seus dígitos individuais e imprime os dígitos separados entre si por três espaços cada (**Dica:** Utilize operadores de divisão de inteiros e módulo.). Por exemplo, se o usuário digitar 54321, o programa deve imprimir:

```
5   4   3   2   1
```

9. Escreva um programa que solicite ao usuário que digite o número de segundos como um valor inteiro (use o tipo `long` ou, se disponível, `long long`) e que exiba o tempo equivalente em dias, horas, minutos e segundos. Use constantes simbólicas para representar o número de horas no dia, o número de minutos em uma hora e o número de segundos em um minuto. A saída deve ser assim:

```
Entre com o numero de segundos: 31600000
31600000 segundos sao 365 dias, 17 horas, 46 minutos e 40 segundos.
```

10. A biblioteca `cmath` disponibiliza várias funções para que o usuário possa manipular com operações de funções matemáticas mais elaboradas. Utilizando essas funções, implemente programas para realizar as seguintes tarefas:

    (a) Crie um programa para recebe o valor de raio, e calcula o comprimento e a área do círculo de raio correspondente. Portanto, defina uma constante para o valor de π utilizando a seguinte equação:

    $$\pi = 4 * \arctan(1);$$

    (b) Crie um programa que recebe dois valores a e b, assumindo-os a parte real e a imaginária de um número complexo z. Ele deve converter este valor para a forma trigonométrica de um número complexo, que apresenta um módulo ρ e o ângulo θ, em radianos, utilizando as seguintes operações:

    $$\rho = \sqrt{a^2 + b^2};$$

$$\theta = \arctan\left(\frac{-b}{a}\right);$$

A saída do programa deve ser dada como a seguir:

```
Digite a parte real: 2
Digite a parte imaginaria: 2
Modulo da forma trigonometica: 2.82843
Angulo da forma trigonometrica: -0.785398
```

2.5 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**. 5ed. Editora Bookman.

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