ESTRUTURA DE DADOS - 03A - 2022.2

Variáveis e operadores aritméticos

Programas em C++

Por enquanto todo seu código deve estar dentro da função `main`, como mostrado abaixo. Depois aprenderemos a criar outras funções.

```
int main()
{
        SEU CÓDIGO É COLOCADO AQUI.
}
```

Todo comando deve terminar com um `;`

Comandos podem ser substituídos por um bloco. Um bloco é um conjunto de comandos entre os símbolos `{` e `}`.

Comentários

Um comentário é um texto inserido no código sem afetar a compilação. É usado para documentar. Dispomos de comentários de linha e comentários de bloco, como ilustra o exemplo abaixo.

```
/* Comentários de bloco...
   ... podem ocupar ...
   várias linhas.
*/

int main()
{
        std::cout << "Hello world!";      // Comentários de linhas vão apenas até o final da linha.
}
```

Variável

Uma variável representa um espaço de memória utilizado pelo programa para armazenar alguma informação.

Em C++, toda variável deve ser explicitamente declarada. Na declaração indicamos seu tipo e seu nome. Opcionalmente podemos inicializar uma variável fornecendo um valor inicial para a variável.

```
<tipo> <nome> [= valor_inicial] ;

Exemplos:
```

```
int idade;        // Declaração sem inicialização
```

```
int idade = 18;   // Declaração com inicialização

int idade{ 18 };   // Outra forma de inicializar uma variável é usando chaves logo após a declaração do nome da variável
```

Regras para os nomes das variáveis:

- Tamanho do nome não limitado, mas apenas os 31 primeiros caracteres são considerados.

- Não pode ser uma palavra reservada da linguagem: if, while, int, double, ...
- Pode conter letras (sem acento/cedilha), dígitos ou '_', mas deve iniciar com letra ou '_'.
- Existe diferença de minúsculas e maiúsculas. Por exemplo, x e X são variáveis distintas.
- O nome deve sugerir o que é armazenado. Por exemplo, idade, altura, sexo, nome. Código com variáveis com nomes x, y, a1, a2,... são difíceis de ler.

Tipos de dados

Inteiros:

- **char:** 1 byte, geralmente usado para armazenar um caractere de um texto. Valores: de -128 até 127.
- **int:** normalmente 4 bytes (de -2.147.483.648 até 2.147.483.647).
- **short:** possivelmente menor que o int. Com 2 bytes vai de -32.768 até 32.767.
- **long:** possivelmente maior que o int. Normalmente 8 bytes.

Ponto flutuante (aproximação de números reais):

- **float:** 4 bytes, 7 dígitos significativos, ordem de grandeza de 10^{-38} até 10^{38}.
- **double:** 8 bytes, 15 dígitos significativos, ordem de grandeza de 10^{-308} até 10^{308}.

Os tipos inteiros podem ser precedidos pela palavra reservada `unsigned`. Neste caso todos os valores representados são positivos.

- **unsigned char:** de 0 até 255.
- **unsigned int:** de 0 até 4.294.967.295.
- **unsigned short:** de 0 até 65.535.
- **unsigned long:** de 0 até 2^{64}.

Comando de atribuição (=)

Serve para armazenar um valor em uma variável.

```
int main()
{
     int idade;
     idade = 18;
}
```

Literais

Valores que aparecem explicitamente no código.

- Todo inteiro que cabe em um int é um int. Caso contrário é um long.
- Para forçar literal long, usamos um L ou l no final. Ex.: 123L
- Zero antes do número indica que está em octal (base 8), e 0X ou 0x indica valor em hexadecimal (base 16). Ex.: 31 = 037 = 0x1F.
- Para constantes unsigned usamos U ou u no final. Ex.: 123U.
- Todo número com ponto decimal (ex.: 12.3) ou expoente (ex.: 1e-2) é considerado double. Para ser float, coloque F ou f no final.
- Um caractere é representado com aspas simples, produzindo assim seu valor na tabela ASCII. Ex.: 'A' tem valor 65 na tabela ASCII.
- Caracteres especiais: '\n' nova linha, '\t' tabulação, '\\' barra invertida, '\'' aspas simples, '\"' aspas duplas, '%%' caractere %, '\0' número 0.

Operadores Aritméticos

- Soma: +
- Subtração: -
- Multiplicação: *
- Divisão: /
- Resto da divisão: %
- Incremento: ++
- Decremento: --

Quando os operandos são inteiros, a divisão resulta em um número inteiro (parte fracionária é descartada).

Se algum operando é ponto flutuante, o resultado terá a parte fracionária. O operador de resto da divisão aceita apenas operandos inteiros.

```
int main()
{
    int x;
    double y;

    x = 7 + 2;        // x recebe 9
    x = 7 - 2;        // x recebe 5
    x = 7 * 2;        // x recebe 14

    x = 7 / 2;        // x recebe 3
    y = 7.0 / 2.0;    // y recebe 3.5

    x = 7 % 2;        // x recebe 1

    x = 5;
    x++;              // x agora armazena 6
    x--;              // x volta a armazenar 5

    int z = 3;
    x = -z;           // x recebe -3 (operador - unário)
}
```

Conversão de tipo

Se uma expressão tem mais de um tipo, todos os operandos são convertidos para o maior tipo (mais bytes).

É possível fazer conversão forçada de tipo. Para isso basta colocar o novo tipo entre parênteses.

```
int x = 2;
float z = 7.8;

float y = x;           // Neste caso a conversão é opcional (feita automaticamente). O valor de x é transformado em
float e armazenado em y
int w = (int) z;      // Neste caso, a conversão é forçada. O valor 7 é armazenado em w e a parte decimal é perdida
```

Última atualização: segunda, 3 mai 2021, 22:49

◄ Conceitos Básicos

Seguir para...

Entrada e Saída de dados ►