Minicurso de C/C++ para Matemática

Pedro H A Konzen

17 de outubro de 2023

Conteúdo

1 Licença

2 Sobre a Linguagem

C e C++ são **linguagens** de programação **compiladas** de propósito geral. A primeira é **estruturada e procedural**, tendo sido crada em 1972 por Dennis Ritchie[1]. A segunda foi inicialmente desenvolvida por Bjarne Stroustrup[2] como uma extensão da primeira. Em sua mais recente especificação, a linguagem C++ se caracteriza por ser **multi-paradigma** (**imperativa**, **orientada a objetos** e **genérica**).

2.1 Instalação e Execução

Códigos C/C++ precisam ser compilados antes de serem executados. De forma simplificada, o **compilador** é um programa que interpreta e converte o código em um programa executável em computador. Há vários compiladores gratuitos disponíveis na web. Ao longo deste minicurso, usaremos a coleção de compiladores GNU GCC instalados em sistema operacional Linux.

2.1.1 IDE

Usar um **ambiente integrado de desenvolvimento** (IDE, em inglês, *integrated development environment*) é a melhor forma de capturar o melhor das linguagens C/C++. Algumas alternativas são:

- Eclipse
- GNU Emacs
- VS Code

2.2 Olá, mundo!

Vamos implementar nosso primeiro programa C/C++. Em geral, são três passos: 1. escrever; 2. compilar; 3. executar.

1. **Escrever o código**.

   Em seu IDE preferido, digite o código:

[1] Dennis Ritchie, 1941-2011, cientista da computação estadunidense. Fonte: Wikipédia.
[2] Bjarne Stroustrup, 1950, cientista da computação dinamarquês. Fonte: Wikipédia.

Código 1: ola.cc

```
1 #include <stdio.h>
2
3 int main()
4 {
5   printf("Olá, mundo!\n");
6   return 0;
7 }
```

2. **Compilar**.

   Para compilá-lo, digite no terminal de seu sistema operacional

```
1 $ gcc ola.cc -o ola.x
```

3. **Executar**.

   Terminada a compilação, o arquivo executável `ola.x` é criado. Para executá-lo, digite

```
1 $ ./ola.x
```

3 Elementos da Linguagem

3.1 Tipos de Dados Básicos

Na linguagem C/C++, **dados** são alocados em **variáveis** com tipos declarados[3].

Exemplo 3.1. Consideramos o seguinte código.

Código 2: dados.cc

```
1 /* dados.cc
2    Exemplo de alocação de variáveis.
3 */
4 #include <stdio.h>
5
6 int main()
```

[3] Consulte Wikipedia: C data type para uma lista dos tipos de dados disponíveis na linguagem

```
7  {
8    // var inteira
9    int i = 1;
10   // var pto flutuante
11   double x;
12
13   x = 2.5;
14   char s[6] = "i + x";
15   double y = i + x;
16   printf("%s = %f\n", s, y);
17   return 0;
18 }
```

Na linha 9, é alocada uma **variável do tipo inteira** com **identificador** `i` e **valor** `1`. Na linha 11, é alocada uma **variável do tipo ponto flutuante** (64 *bits*) com identificador `x`.

Na linha 14, é alocada uma **variável do tipo *string***[4]. Na linha 15, alocamos uma nova variável `y`.

Observação 3.1. (Comentários e Continuação de Linha.) Códigos C++ admitem **comentários** e **continuação de linha** como no seguinte exemplo acima. Comentários em linha podem ser feitos com `\\` e de múltiplas linhas com `\* ... */`. Linhas de instruções muito compridas podem ser quebradas em múltiplas linhas com a instrução de continuação de linha `\`.

Observação 3.2. (Notação científica.) Podemos usar **notação científica** em C++. Por exemplo 5.2×10^{-2} é digitado da seguinte forma `5.2e-2`.

Código 3: notacaoCientifica.cpp

```
1 #include <stdio.h>
2
3 int main()
4 {
5
6   int i = -51;
7   double x = 5.2e-2;
8
9   // inteiro
```

[4] Um arranjo de `char` (caracteres).

```
10    printf("inteiro: %d\n", i);
11    // fixada
12    printf("fixada: %f\n", x);
13    // notação científica
14    printf("científica: %e\n", x);
15    return 0;
16 }
```

Exercício 3.1.1. Antes de implementar, diga qual o valor de `x` após as seguintes instruções.

```
1 int x = 1
2 int y = x
3 y = 0
```

Justifique seu resposta e verifique-a.

Exercício 3.1.2. Implemente um código em que a(o) usuária(o) entra com valores para as variáveis `x` e `y`[5]. Então, os valores das variáveis são permutados entre si.

3.2 Operações Aritméticas Elementares

Os operadores aritméticos elementares são:

`+`, `-` : adição, subtração

`*`, `/` : multiplicação, divisão

`%` : módulo

Exemplo 3.2. Qual é o valor impresso pelo seguinte código?

```
1 #include <stdio.h>
2
3 int main()
4 {
5    printf("%f\n", 2+17%9/2*2-1 );
6    return 0;
7 }
```

[5] A entrada de valores via console pode ser feita com o método da biblioteca stdio.h.

Observamos que as operações `*` e `/` têm **precedência** maior que a operação `\%`. Esta, por sua vez, tem precedência maior que as operações `+` e `-`. Operações de mesma precedência seguem a ordem da esquerda para direita, conforme escritas na linha de comando. Usa-se parênteses para alterar a precedência entre as operações, por exemplo

```
1 printf("%f\n", ((2+17)%9))/2.*2-1 );
```

imprime o resultado `-1`. Sim, pois a divisão inteira está sendo usada. Para computar a divisão em ponto flutuante, um dos operandos deve ser `double`. Para tanto, podemos fazer um ***casting*** `double((2+17)\%9)/2*2-1` ou, simplesmente, `((2+17)\%9)/2.*2-1`.

Observação 3.3. (Precedência das Operações.) Consulte mais informações sobre a precedência de operadores em Wikipedia:Operators in C and C++.

Exercício 3.2.1. Escreva um programa para computar o vértice da parábola

$$ax^2 + bx + c = 0, \tag{1}$$

para $a = 2$, $b = -2$ e $c = 4$.

O operador `%` módulo computa o **resto** da divisão inteira, por exemplo, `5\%2` é igual a `1`.

Exercício 3.2.2. Use o Python para computar os inteiros não negativos q e r tais que

$$25 = q \cdot 3 + r, \tag{2}$$

sendo r o menor possível.

3.3 Funções e Constantes Elementares

A **biblioteca** C/C++ math.h disponibiliza várias funções e constantes elementares.

Código 4: mat.cc

```
1 #include <stdio.h>
2 #include <math.h>
3
```

```
4  int main()
5  {
6    printf("pi = %.9e\n", M_PI);
7    printf("2^(1/2) = %.5f\n", sqrt(2.));
8    printf("log(e) = %f\n", log(M_E));
9    return 0;
10 }
```

Observação 3.4. (Logaritmo Natural.) Notamos que `log` é a função logaritmo natural, i.e. $\ln(x) = \log_e(x)$. A implementação C/C++ para o logaritmo de base 10 é `log10(x)`.

Exercício 3.3.1. Compute

a) $\operatorname{sen}\left(\frac{\pi}{4}\right)$

b) $\log_3(\pi)$

c) $e^{\log_2(\pi)}$

d) $\sqrt[3]{-27}$

Exercício 3.3.2. Compute as raízes do seguinte polinômio quadrático

$$p(x) = 2x^2 - 2x - 4 \tag{3}$$

usando a fórmula de Bhaskara[6].

3.4 Operadores de Comparação Elementares

Os operadores de comparação elementares são

== : igual a

!= : diferente de

> : maior que

< : menor que

>= : maior ou igual que

<= : menor ou igual que

[6] Bhaskara Akaria, 1114 - 1185, matemático e astrônomo indiano. Fonte: Wikipédia.

Estes operadores retornam os **valores lógicos** `true` (verdadeiro, `1`) ou `false` (falso, `0`).

Por exemplo, temos

Código 5: opComp.cc

```
1 #include <stdio.h>
2
3 int main()
4 {
5   int x = 2;
6   bool res = x + x == 5;
7   printf("2 + 2 == 5? %d", res);
8 }
```

Exercício 3.4.1. Considere a circunferência de equação

$$c: \ (x-1)^2 + (y+1)^2 = 1. \tag{4}$$

Escreva um código em que a(o) usuária(o) entra com as coordenadas de um ponto $P = (x, y)$ e o código verifica se P pertence ao disco determinado por c.

Exercício 3.4.2. Antes de implementar, diga qual é o valor lógico da instrução `sqrt(3) == 3`. Justifique sua resposta e verifique!

3.5 Operadores Lógicos Elementares

Os operadores lógicos elementares são:

`&&` : `e` lógico

`||` : `ou` lógico

`!` : `não` lógico

Exemplo 3.3. (Tabela Booleana do `&&`.) A tabela booleana[7] do `e` lógico é

[7] George Boole, 1815 - 1864, matemático britânico. Fonte: Wikipédia.

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

O seguinte código, monta essa tabela booleana, verifique!

```
1  #include <stdio.h>
2
3  int main()
4  {
5    bool T = true;
6    bool F = false;
7    printf("A    | B    | A && B\n");
8    printf("%d    | %d    | %d\n", T, T, T&&T);
9    printf("%d    | %d    | %d\n", T, F, T&&F);
10   printf("%d    | %d    | %d\n", F, T, F&&T);
11   printf("%d    | %d    | %d\n", F, F, F&&F);
12 }
```

Exercício 3.5.1. Construa as tabelas booleanas do operador `||` e do `!`.

Exercício 3.5.2. Escreva um código para verificar as seguintes comparações

a) $1.4 <= \sqrt{2} < 1.5$.

b) $|x| < 1$, $x = \text{sen}(\pi/3)$.

c) $|x| > \frac{1}{2}$, $x = \cos(\pi * *2)$.

Exercício 3.5.3. Considere um retângulo $r:\ ABDC$ de vértices $A = (1, 1)$ e $D = (2, 3)$. Crie um código em que a(o) usuária(o) informa as coordenadas de um ponto $P = (x, y)$ e o código verifica cada um dos seguintes itens:

1. $P \in r$.
2. $P \in \partial r$.
3. $P \notin \overline{r}$.

Exercício 3.5.4. Implemente uma instrução para computar o operador `xor` (ou exclusivo). Dadas duas afirmações `A` e `B`, `A xor B` é `true` no caso de uma, e somente uma, das afirmações ser `true`, caso contrário é `false`.

Referências