

Introdução à linguagem C++

1) Linguagem C x Linguagem C++

a) Programação estruturada x Programação Orientado a Objetos

<pre>typedef struct{ char * where; int year; double rscale; } quake;</pre>	<pre>1 class Quake{ 2 private: 3 const string where; 4 const int year; 5 const double rscale; 6 }</pre>
--	---

Em relação à Programação Estruturada, POO serve muito bem para problemas grandes, modelagens complexas e representação de objetos da vida real, graças à sua bem estruturada **maneira de abstrair conceitos**.

b) Sintática

<pre>1 #include <iostream> 2 3 using namespace std; 4 5 int main(){ 6 int a=0; 7 cout << "This is a: " << a << endl; 8 cout << "Enter an integer, m8!: " << endl; 9 cin >> a; 10 return 0; 11 }</pre>	<pre>1 #include <stdio.h> 2 #include <stdlib.h> 3 4 5 int main(){ 6 int a=0; 7 printf("This is a: %d\n", a); 8 printf("Enter an integer, m8!: "); 9 scanf("%d", &a); 10 return 0; 11 }</pre>
--	---

Há muito da linguagem C na linguagem C++, justo porque o intuito da C++ é ser uma C com classes. Porém, há algumas palavras reservadas e maneiras de escrever código a mais, como, por exemplo:

- i) Entrada de dados: **cin >>**
- ii) Saída de dados: **cout <<**
- iii) Fim da linha: **endl**
- iv) Você pode usar a função **for** como:
for(int i = 0; i < size; i++)

- v) Finalmente você pode utilizar a classe **string**, e **não mais um ponteiro (vetor) de caracteres**

Mas não precisa se preocupar, pois muita coisa não mudou, como, por exemplo:

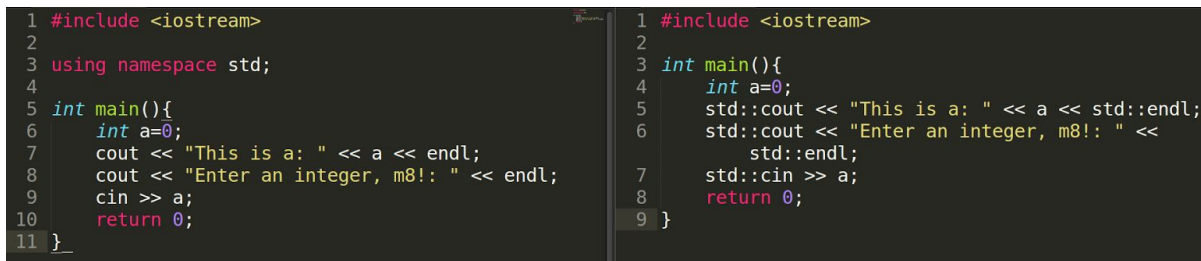
- A sintaxe das funções **condicionais** (como **if** e **switch**)
- Manipulação de arquivos
- **While, do-while, ponteiros, constantes, registros, vetores e matrizes**
- **Funções** (mais tarde na disciplina, verão que muda apenas a passagem de argumentos)
- **Bibliotecas** (cuidado com os nomes! Sempre é válido visitar www.cplusplus.com)

c) Operador de escopo:

Na primeira imagem vocês perceberam que há algo diferente no começo do código:

```
"using namespace std;"
```

Por hora, admita que isso define um escopo reservado **std** que contém vários objetos (**nesse caso, algumas palavras reservadas**). Caso não utilize essa linha no seu código, você deve explicitar de onde vem alguns objetos (palavras reservadas):



```

1 #include <iostream>
2
3 using namespace std;
4
5 int main(){
6     int a=0;
7     cout << "This is a: " << a << endl;
8     cout << "Enter an integer, m8!: " << endl;
9     cin >> a;
10    return 0;
11 }
    
```

```

1 #include <iostream>
2
3 int main(){
4     int a=0;
5     std::cout << "This is a: " << a << std::endl;
6     std::cout << "Enter an integer, m8!: " <<
7         std::endl;
8     std::cin >> a;
9     return 0;
10 }
    
```

cin e **cout** são objetos do **std**, ou seja, pertencem ao **escopo** de **std**. Isso é explicitado quando se escreve **"::"**. (Esses dois códigos fazem a mesma coisa)

Esse conceito ficará mais claro quando estiverem trabalhando com **classes**, quando tiverem que diferenciar vários componentes da classe e associá-los à classe desejada. (Confuso por enquanto? Talvez esse conceito esteja mais claro no futuro!)

d) Variáveis *static*

Outra coisa que C++ traz são as variáveis **static**, que possuem valor global, ou seja, mantém seu valor mesmo fora de seu escopo.

Guardem esse conceito para quando começarem a modelar classes que necessitam de qualquer tipo de armazenamento global à todas as instâncias (ual, quanto nome novo D: não se preocupem com todos esses termos por enquanto).

No **código** a seguir, temos que a variável ***and_i_will_be_one*** será **1** se o argumento for 0.

Vale lembrar que, após a execução da função, considerando-se o que foi visto de programação até agora, as variáveis perderão seus valores armazenados.

Observe que na linha **17** o inteiro 0 é passado como argumento, exibindo, assim, o número 1. Porém, na linha **18**, o inteiro 1 é passado como argumento. Mas na execução do programa **ainda é exibido o número 1!**

Isso acontece porque na linha anterior a variável ***and_i_will_be_one*** tem seu valor modificado uma vez após a declaração, tendo seu valor armazenado mesmo após a execução da função!

```

1  #include <iostream>
2
3  using namespace std;
4
5  int pass_me_zero(int i){
6      static int and_i_will_be_one = 0;
7      if(i == 0){
8          and_i_will_be_one = 1;
9          return and_i_will_be_one;
10     }
11     else
12         return and_i_will_be_one;
13 }
14
15 int main(){
16     int a=0;
17     cout << pass_me_zero(0) << endl;
18     cout << pass_me_zero(1) << endl;
19     return 0;
20 }

```

2) Exercícios:

A ideia, por enquanto, é **se acostumar com a linguagem C++**. Vale o conselho de fazer exercícios do <https://www.urionlinejudge.com.br/> (agora na linguagem C++). Claro, **não se importe em acertar outputs, mas sim a sintaxe**.

Entretanto, aqui vão alguns exercícios para serem resolvidos em C++:

1) Leia dois números e exiba:

- a) A soma destes valores
- b) O produto deles
- c) O quociente entre eles.

2) Faça um algoritmo para ler dois valores reais do teclado, calcular o MMC entre eles e imprimir na tela:

3) Leia 6 números e armazene-os em um vetor. Ordene-o em ordem crescente e exiba na tela. Em seguida, ordene-o em ordem decrescente e exiba na tela.

4) Leia oito números inteiros, calcule a soma dos números pares e a soma dos números ímpares.

5) Entre com dez números armazenando-os num vetor. Ache o menor e o maior.

6) Ler três valores e determinar o maior dentre eles. 8. Ler três valores inteiros e calcular a soma do menor valor com o maior.

7) Escreva um programa que gera um número inteiro aleatório de 0 a 42 e chama uma função que conta quantos números primos foram gerados até o usuário digitar "jv" no terminal. Essa função também deve calcular a quantidade de números gerados. Use variáveis **static**!