**Aula 1**

**1. Quais ferramentas usaremos para programar:**

Visual Studio Code + Ubuntu - Wsl;

Ou algum compilador web (solução temporária só para o começo das aulas);

**2. Estrutura básica de um código c++ para programação competitiva:**  
Bibliotecas: São basicamente ferramentas que trazem funções e nos permitem ter uma programação mais simples, e não precisar escrever tudo do zero. As bibliotecas que usamos, para facilitar, depois explicaremos o porquê de usarmos elas, são “algorithm” e “bits/stdc++.h”.

Essas, então, serão sempre as duas primeiras linhas de um código para programação competitiva:

#include <algorithm>

#include <bits/stdc++.h>

Em seguida, temos o “using namespace std;”, usamos isso para não ter que escrever “std::” antes de toda a função padrão de c++, mas isso é considerado uma má prática fora da área de programação competitiva, cuidado;

using namespace std;

Agora teremos o início do programa:  
Sempre que você criar um programa, a máquina vai procurar isso no código e começar a executá-lo por ali. Chamamos de “main”, escrevemos desta forma:

int main() {

return 0;

}

E tudo que está entre essas chaves (“{“ e “}”) será o nosso código principal, o “return 0;” serve para mostrar para a máquina que o código foi executado sem erros, sempre deixe ele embaixo da main.

**3. Primeiro código em c++:**  
É uma tradição na programação fazer o primeiro código nosso escrever “Hello World”, segue o código.  
Começamos com a estrutura básica:

#include <algorithm>

#include <bits/stdc++.h>

using namespace std;

int main() {

return 0;

}

Aí usamos a nossa primeira função de biblioteca agora, ela chama “cout” e serve para escrever frases, variáveis, números, etc.

E sua estrutura básica é “cout << “o que deseja escrever;” “

Então escreveremos:  
“cout << “Hello World”;” (observação, sempre colocamos “;” no final das linhas)

Então o código fica assim:

#include <algorithm>

#include <bits/stdc++.h>

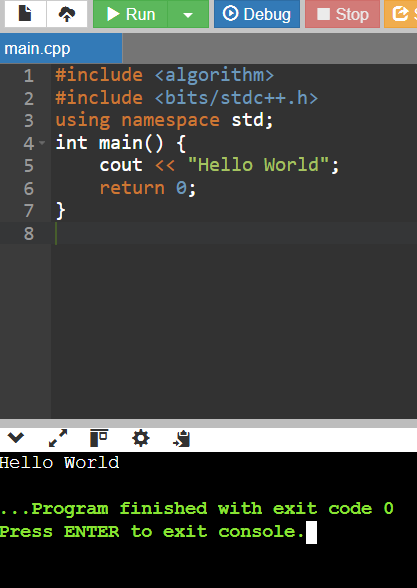
using namespace std;

int main() {

cout << "Hello World";

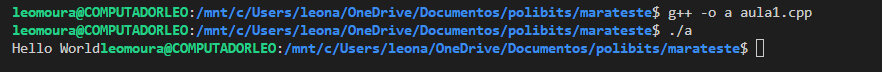
return 0;

}

E sua saída no compilador web Gdb fica assim:  


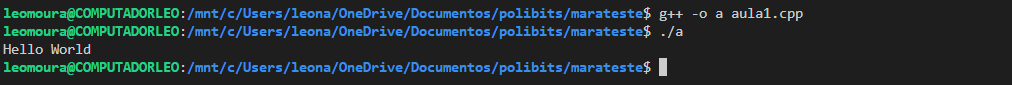
Veja que ele escreve isso no “terminal”.

Uma dica apenas é escrever “endl” no final do código, porque todos os sites de programação competitiva exigem isso, mais pra frente vai ficar claro o porquê, mas basicamente se eu rodar aquele código no meu terminal ele sai assim:



Parece que falta pular uma linha:

Então usamos outra função da nossa biblioteca, que é o “endl”, que serve para pular linha.

Com isso, o meu terminal fica assim:  


Muito melhor!!

Eu apenas troquei o código anterior para isso:

#include <algorithm>

#include <bits/stdc++.h>

using namespace std;

int main() {

cout << "Hello World" << endl;

return 0;

}

No próximo tópico vai ficar claro como usar essas “<<” corretamente.

**3.1 Comentários:**

Só um comentário, é possível comentar seu código, ou seja, deixar uma mensagem pra quem está lendo o código, o que não afeta a saída, usamos “//”, tudo depois disso até o fim da linha será um comentário:

#include <algorithm>

#include <bits/stdc++.h>

using namespace std;

int main() {

// estou escrevendo a palavra Amora no terminal

cout << "Amora" << endl; // Amora foi escrita

//hehe

return 0;

}

**4. Operadores aritméticos:**Podemos trabalhar com adição(+), subtração(-), multiplicação(\*), divisão(/) e resto da divisão (%);

O resto da divisão (%) é muito usado para saber se um número é par ou ímpar, por exemplo.

Um exemplo de uso seria :

#include <algorithm>

#include <bits/stdc++.h>

using namespace std;

int main() {

cout << 1 + 2 + 3\*4 << endl;

return 0;

}

E no terminal ele retorna 15, então ele se atenta à prioridade de expressões numéricas, mas é boa prática usar parêntesis:

#include <algorithm>

#include <bits/stdc++.h>

using namespace std;

int main() {

cout << (1 + 2 + 3)\*4 << endl;

return 0;

}

Aqui ele retorna 24.

**5. Introdução a tipos de variáveis:**  
Em c++, como em praticamente todas as linguagens, podemos trabalhar com os seguintes tipos de variáveis:

Variáveis binárias(pode ser 0 ou 1) : bool  
Variáveis inteiros: int , long int, long long int (autoexplicativo)

Variáveis de números quebrados: float, double

Variáveis de caracteres: char

Usaremos todas com o mesmo princípio:

Primeiro preciso “declarar” a variável, usando o nome do tipo da variável e depois o nome que você quer dar, pode ser x, quadrado, estrela, quase tudo o que você quiser, depois entraremos em mais detalhes.

Então, por exemplo:

int a;

double abelha;

char c;

Veja que essas variáveis estão sem valor associado a elas:

E se eu tentar escrever:

cout << a << endl;

Ele ou vai escrever zero, ou qualquer outro número aleatório, porque ele não foi definido.

Então é uma boa prática sempre definir a variável logo que declarou:

(a notação a = b significa que a recebe o valor de b, isso é muito importante).

int a = 5;

double abelha = 0.0007;

char c = 'e';

int ce = a + abelha;

A variável ce vai ter valor 5, pois é um inteiro, então arredonda tudo pra baixo.

Por exemplo:

int ce = a + 0.9;

Também é 5.

Percebe-se que o char, que é abreviação para caractere, é representado entre aspas simples, e palavras entre aspas duplas.

Para escrever os três, precisamos separá-los por “<<”, já que são, além de variáveis diferentes, variáveis com tipos diferentes, e por questão de estilo, é melhor separálos com espaços, que, como também são coisas diferentes das variáveis, precisamos separar por “<<”.

cout << a << " " << abelha << " " << c << endl;

No terminal fica:



Agora um certo “problema” de números inteiros na programação:

Se eu quiser fazer a conta “15 / 2” com números, assim como está escrito, o resultado é 7.

Fazendo de outra forma:

int a = 15;

cout << a / 2 << endl;

O resultado também é 7.

O que significa que “15 / 2 \* 2” é 14.

Isso geralmente não é um problema, mas é importante lembrar que inteiros não guardam a informação de que são, por exemplo, 7.5 , eles são inteiros sempre, por isso é bom tomar cuidado.

Fazendo:

double a = 15;

cout << a / 2 << endl;

A saída é 7.5 , como o esperado.

Outra forma de fazer 7.5 é explicitar

cout << 15.0 / 2 << endl;

**6. Como ler números:**

Por exemplo, se eu quiser fazer um programa para verificar se um número tal é par, como faço?

É necessário usar o “cin”, outra função da nossa biblioteca.

Então eu uso o cin para ler um número e retorno o resto por 2.

Fica assim:

#include <algorithm>

#include <bits/stdc++.h>

using namespace std;

int main() {

int a;

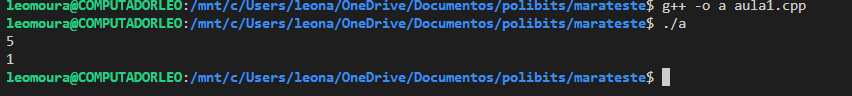
cin >> a;

cout << a % 2 << endl;

return 0;

}

No terminal:



O 5 foi eu quem digitei, poderia ter digitado qualquer inteiro.

**6.5 Operadores de comparação:**

Eu posso usar >, <, >= , <= , ==, !=; respectivamente, maior, menor, maior ou igual, menor ou igual, igual e diferente.

Eles verificam comparações, se as comparações forem verdadeiras eles retornam 1, se não retornam 0.

Exemplos:

#include <algorithm>

#include <bits/stdc++.h>

using namespace std;

int main() {

int a = 5;

int b = 3;

int c = 5;

cout << (a == b) << endl;

cout << (a != b) << endl;

cout << (a < c) << endl;

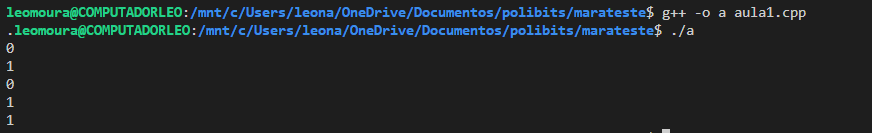
cout << (b != c) << endl;

cout << (a == c) << endl;

return 0;

}

Saída é:



**7. If e else:**

Ficou claro que eu não fiz um código que me respondia se um número era par ou não, ele apenas retornava o resto por 2, então, usando essa nova função if, é possível fazer isso:  
É literalmente a tradução em inglês, se o resto for 1, eu escrevo “esse numero eh par”, se não, eu escrevo “esse numero eh impar”.

Se o que estiver dentro do if retornar 1, ele entra no if, se não, ele entra no else.

Para eu saber se um número é igual ao outro, eu preciso usar “==”, que vai retornar 0 se forem iguais os valores dos números e 1 se não forem.

Esse “se não” ou “se não for isso”, é o else, é qualquer coisa que não satisfaça o if.

A sintaxe do if e else é assim:

if(algo){

algo acontece;

} else {

outra coisa acontece;

}

Esse “algo” é um binário 0 ou 1, 1 se for verdade e 0 se for falso.

Então o problema fica assim:

#include <algorithm>

#include <bits/stdc++.h>

using namespace std;

int main() {

int a;

cin >> a;

if(a % 2 == 0){

cout << "esse numero eh par" << endl;

} else {

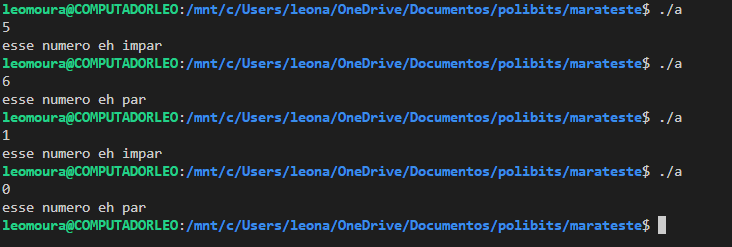
cout << "esse numero eh impar" << endl;

}

return 0;

}

No terminal fica assim:



O que resolve nosso problema:

**7.5 Operadores booleanos:**

Eu posso usar os operadores && (AND) e ||(OR), que são operadores que fazem isto:

Eu tenho duas afirmações e quero saber se as duas estão certas, então eu faço o seguinte

if(afirmacao1 && afirmacao2) tãnãnã

Ele só vai retornar 1 se ambas forem verdadeiras.

Exemplo:

int a = 5;

int b = 5;

if(a >= b && a < b) cout << 1 << endl;

else cout << 0 << endl;

//o cout dado foi 0

Esse foi o AND, o OR é assim:

if(afirmacao1 || afirmacao2)

Ele só vai entrar nesse if se alguma das afirmações for verdadeira, pode ser as duas ou só uma.

int a = 5;

int b = 5;

if(a >= b || a < b) cout << 1 << endl;

else cout << 0 << endl;

//o cout dado foi 1

**8. Observações:**

Tabela ASCII:

A tabela ascii é o que atrela cada caractere a um número, consulte este [site](https://www.ime.usp.br/~kellyrb/mac2166_2015/tabela_ascii.html), exemplos de coisas convenientes dessa tabela são que as letras maiúsculas estão juntas, as letras minúsculas estão juntas e números estão juntos:

Um exemplo do uso disso é como gravar a distância entre uma letra e a letra “a”:

char c = 'b';

char d = 'c';

cout << 'b' - 'a' << " " << 'c' - 'a' << endl; // os couts são 1 e 3, respectivamente

Também:

char c = '9';

char d = '6';

cout << c - d << endl; // o cout é 3

É possível criar uma variável que recebe uma expressão entre variáveis:

int a = b + c / d + 2;

É possível colocar outro if dentro do else e é possível fazer um if sem um else:

if(algo){

algo acontece;

} else {

if(outro algo){

outro algo acontece;

}

}

É possível fazer uma variável receber seu próprio valor mais algo, ou receber a sua metade:

Para fazer uma variável receber a si mesma mais um, é possível usar ++, se receber ela menos um usar -- , e receber ela mesma menos 3 pode usar a -= 3;

Então:

#include <algorithm>

#include <bits/stdc++.h>

using namespace std;

int main() {

int a = 9; // a == 9

a = a/2.0; // a == 4

a = a + 2; // a == 6

a++; // a == 7

a--; // a == 6

a -= 3; // a == 3

return 0;

}

Só um exemplo antes dos primeiros exercícios:  
Eu recebo dois números a e b, e vejo se são iguais, se forem eu escrevo a, se não eu escrevo b menos a soma dos dois.

#include <algorithm>

#include <bits/stdc++.h>

using namespace std;

int main() {

int a, b;

cin >> a >> b;

if(a == b){

cout << a << endl;

} else {

cout << -a << endl; //-a == b - (a + b)

}

return 0;

}

**9. Primeiros exercícios:**

1. Faça um programa que responde se um número é divisível pelo outro sem usar o operador (%). (testa um par, um impar , zero e um)
2. Faça um programa que recebe três números, e da cout no maior. (testa os trios {0, 1, 2}, {4, 1 , 3}, {3, 3, 3}, {3, 3, 1}, {1, 1, 3})
3. Faça um programa que recebe duas letras e, se for maiúsculo, deixa minúsculo, se for minúsculo, deixa maiúsculo. (consulta a tabela [ASCII](https://www.ime.usp.br/~kellyrb/mac2166_2015/tabela_ascii.html) se precisar) {testa ‘A’ e ‘b’, ‘D’ e ‘f’, ‘k’ e ‘i’}

**10. Loops (eita!!)**