



[HOME](#) [CONTESTS](#) [GYM](#) [PROBLEMSET](#) [GROUPS](#) [RATING](#) [API](#) [RCC](#)  [AIM TECH ROUND](#)  [CALENDAR](#)
[BLOG](#) [CONTESTS](#) [MEMBERS](#) [STATUS](#)

Kruskal

 By [bssanches](#), [history](#), 3 months ago,  

Kruskal é o algoritmo que resolve o problema de minimum spanning tree (MST)

Código (Só funciona em C++11):

Para compilar em C++11 `g++ a.cpp -std=c++11` ou `g++ a.cpp -std=c++0x`

```
vector<tuple<int,int,int> > edges;

int f(int x)
{
    if (g[x] == x)
        return x;
    return g[x] = f(g[x]);
}

int unio(int x, int y)
{
    x = f(x);
    y = f(y);
    if (x == y) return false;
    g[x] = y;
    return true;
}

int kruskal()
{
    sort(edges.begin(), edges.end());
    int mst = 0;
    for (int i = 0; i < edges.size(); ++i)
    {
        int peso,x,y;
        tie(peso, x, y) = edges[i];
        if (unio(x, y))
            mst += peso;
    }
    return mst;
}

int main()
{
    for (int i = 1; i <= n; ++i)
        g[i] = i;
    edges.push_back(make_tuple(peso, x, y));
    kruskal();
}
```

Problemas:

GEMA Bixos 2017

Private

Participant



→ About Group



[Group website](#)

→ Group ratings

- [Classificação](#)

→ Member management

You are the member of the group

Leave


http://olimpiada.ic.unicamp.br/pratique/programacao/nivel2/2011f2p2_rmapa

http://olimpiada.ic.unicamp.br/pratique/programacao/nivel2/2008f2p2_frete

[Read more »](#)



▲ 0 ▼

 [bssanches](#)

 3 months ago

 [0](#)

Dijkstra e Prim

By [bssanches](#), [history](#), 3 months ago, , 

Update:

Problema de prim easy:

OBI 2011 P2


```
void dijkstra(int curr, int dis[MAX])
{
    priority_queue<pair<int, int> > pq;
    for (int i = 0; i <= n; ++i)
    {
        dis[i] = INF;
    }
    pq.push(make_pair(0, curr));
    dis[curr] = 0;
    while (!pq.empty())
    {
        int v = pq.top().second;
        int peso = -pq.top().first;
        pq.pop();
        if (dis[v] < peso) continue;
        for (int i = 0; i < adj[v].size(); ++i)
        {
            int u = adj[v][i].first;
            int peso = adj[v][i].second;
            if (dis[u] > dis[v] + peso)
            {
                dis[u] = dis[v] + peso;
                pq.push(make_pair(-dis[u], u));
            }
        }
    }
}
```

Prim: <http://www.geeksforgeeks.org/greedy-algorithms-set-5-prims-minimum-spanning-tree-mst-2/>

[Read more »](#)



▲ 0 ▼

 [bssanches](#)

 3 months ago

 [0](#)

Problemas de Dijkstra na OBI

By [bssanches](#), 3 months ago, , 

OBI 2015 P2 (Facil)

OBI 2009 P2 (Facil)

OBI 2008 P2 (Médio)

[Read more »](#)



bssanches



3 months ago



0

BFS

By [bssanches](#), [history](#), 3 months ago,

Bfs resolve problemas do tipo: Qual o caminho mínimo entre A e B? Além disso BFS também tem varios outros usos (Assim como a DFS, você pode responder se existe um caminho de A até B)

Lembrem-se que a BFS só serve pra achar caminho mínimo se os pesos da aresta forem 1! (Ou se todas tiverem o mesmo peso X)

Arestas com pesos diferentes tem que usar um outro algoritmo que sera explicado na próxima semana :D

Abraços!

```
#define MAX 10000
int dis[MAX];
vector<int> adj[MAX];

void bfs(int curr)
{
    memset(dis, -1, sizeof dis);
    queue<int> q;
    q.push(curr);
    dis[curr] = 0;
    while (!q.empty())
    {
        int v = q.front(); q.pop();
        for (int i = 0; i < adj[v].size(); ++i)
        {
            int u = adj[v][i];
            if (dis[u] == -1)
            {
                dis[u] = dis[v] + 1;
                q.push(u);
            }
        }
    }
}
```

[Read more »](#)



bssanches



3 months ago



0

Problemas de DFS/BFS na OBI

By [bssanches](#), 3 months ago,

Alguns problemas da OBI pra quem quiser treinar mais grafos:

[OBI 2013 nivel 2 \(Facil\)](#)

[OBI 2012 nivel 2 \(Facil\)](#)

[OBI 2014 nivel 2 \(Dificil\)](#) (cuidado que a resposta usa long long!)

[Read more »](#)



bssanches





3 months ago



0

DFS

By [bssanches](#), [history](#), 4 months ago,  

Definições sobre grafos

Vértice: Uma das extremidades da aresta

Aresta: O que liga os vértices

Grafo: Um conjunto de vértices e arestas que representa algum problema

Componente: O conjunto de vértices alcançáveis a partir de um nó

Caminho: Conjunto de arestas e vértices que ligam dois nós do grafo

Ciclo: Caminho que começa e termina no mesmo vértice

Aresta direcional: Aresta que sai de um vértice e vai para o outro mas não pode ser usado no sentido contrario

Aresta bidirecional: Aresta que possui as duas direções

DFS (Deep First Search) é um algoritmo de busca em grafos. Sua complexidade é $O(V + E)$ onde V é o número de vértices e E o número de arestas do grafo.

Código:

```
vector<int> adj[100010];
int vis[100010];
int dfs(int curr)
{
    for (int i = 0; i < adj[curr].size(); ++i)
    {
        int u = adj[curr][i];
        if (!vis[u])
        {
            vis[u] = 1;
            dfs(u);
        }
    }
}

int main()
{
    adj[2].push_back(3); // fala que o 2 tem como adjacente o 3
    adj[1].push_back(5); // fala que o 1 tem como adjacente o 5
    vis[1] = 1;
    dfs(1);
    return 0;
}
```

A complexidade utilizando matriz de adjacência é $O(V^2)$! Onde V é o número de vértices do grafo. Não utilize matriz ao menos que seja muito necessário!

[Read more »](#)

 0 



[bssanches](#)





4 months ago



0

STL

By [bssanches](#), [history](#), 4 months ago,  

À pedidos, segue um resumo de STL, vou colocar bastante código já que a maioria é exemplo prático Lembrando que todos os $O(\log)$ desse tutorial são na verdade $O(\log 2)$:p

cin/cout/vector/string

```
#include <bits/stdc++.h>

using namespace std;

int n;
string x;
vector<string> v;

int main()
{
    ios :: sync_with_stdio(false); //para deixar o cin e o cout mais rapidos
    cin >> n;
    for (int i = 0; i < n; ++i)
    {
        cin >> x;
        v.push_back(x); //adiciona a string x no fim do vector
    }

    for (int i = 0; i < v.size(); ++i) //v.size() eh o tamanho do vetor, 0(1)
    {
        cout << v[i] << "\n";
    }

    if (v[0] == v[1]) //compara se as duas primeiras strings sao iguais
    {
        cout << "sao iguais\n";
    }
    else
    {
        v[3] = v[4]; //copia a 4o string para a 3o posição
    }
    v.erase(v.begin() + 3); //apaga a 4o string do vector (0 indexado)
    v.erase(v.begin() + 0); //ou opcionalmente v.erase(v.begin());, apaga a primeira posicao

    vector<string> v2;
    //copia o v pro v2 0(n*m), pois tem n posições e no pior caso cada uma tera uma string de tamanho m
    v2 = v;
    v.clear(); // limpa o vector
    v.resize(100); //pre-aloca 100 posições
    v[5] = v2[0].substr(5, 10); //pega 10 caracteres a partir da posição 5 da primeira string em v2
    //e coloca em v[5]

    return 0;
}
```

pair/sort/lower_bound/upper_bound

```
#include <bits/stdc++.h>

using namespace std;

int n,x;
vector<pair<int,int> > v;

int main()
```

```

{
    ios :: sync_with_stdio(false); //para deixar o cin e o cout mais
    rapidos
    cin >> n;
    for (int i = 0; i < n; ++i)
    {
        cin >> x;
        v.push_back(make_pair(x, i)); //make pair cria um pair
    }

    sort(v.rbegin(), v.rend()); //Ordena o vetor de maneira
    decrescente

    sort(v.begin(), v.end()); //Ordena o vetor de pair de maneira
    crescente em O(nlog(n))
    //quando um pair eh ordenado, primeiro ele compara o primeiro
    elemento, se forem iguais o
    //desempate eh pelo segundo

    for (int i = 0; i < v.size(); ++i)
    {
        cout << v[i].first << " " << v[i].second << "\n";
    }
    //first eh o primeiro elemento do pair,

    //second o segundo

    //c++11, to procurando se o par (10, 2) existe dentro de v
    //lower_bound retorna o primeiro elemento >= a query, eh uma
    busca binaria portanto
    //complexidade = O(log(n))
    auto it = lower_bound(v.begin(), v.end(), make_pair(10, 2));
    if ((*it) == make_pair(10, 2))
    {
        cout << "achei\n";
    }

    //upper_bound retorna o primeiro elemento > que a query,
    O(log(n))
    auto it2 = upper_bound(v.begin(), v.end(), make_pair(10, 2));

    cout << it2 - it << "\n"; //quantos elementos tem entre it2 e it
    return 0;
}

```

queue/stack/deque

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
int n,x;
queue<int> q;
stack<int> s;
deque<int> dq;
```

```
int main()
```

```

{
    ios :: sync_with_stdio(false); //para deixar o cin e o cout mais
    rapidos
    cin >> n;
    for (int i = 0; i < n; ++i)
    {
        cin >> x;
    }
}

```

```

q.push(x); //coloca na fila O(1)
s.push(x); //coloca na pilha O(1)
dq.push_back(x); //coloca no fim do deque O(1)
dq.push_front(x); //coloca no começo do deque O(1)
}

while (!q.empty()) //enquanto a fila não estiver vazia O(1)
{
    int v = q.front(); //começo da fila O(1)
    cout << v << "\n";
    q.pop(); //tira do começo da fila O(1)
}

while (!s.empty()) //enquanto a pilha não estiver vazia O(1)
{
    int v = q.top(); //fim da pilha O(1)
    cout << v << "\n";
    q.pop(); //tira do fim da pilha O(1)
}

for (int i = 0; i < dq.size(); ++i)
{
    cout << dq[i] << "\n"; //O(1)
}
dq.pop_back(); //tira o ultimo elemento do deque
dq.pop_front(); //tira o primeiro elemento do deque
return 0;
}

```

map/set/piority_queue

```

#include <bits/stdc++.h>

using namespace std;

int n,x;
set<int> s;
map<int, int> m;
priority_queue<int> pq;

int main()
{
    ios :: sync_with_stdio(false); //para deixar o cin e o cout mais
    rapidos
    cin >> n;
    for (int i = 0; i < n; ++i)
    {
        cin >> x;
        s.insert(x); //coloca o x no set, lembrando que set n aceita
        elemento repetido
        //se vc tentar inserir um x que ja
        existe no set, ele vai soh ignorar
        // O(log(n))
        m[x] = 100; //O(log(n)) pois o acesso ao map atraves do "[" eh
        log
        pq.push(10); //O(log(n)), lembrando que a inserção na fila de
        prioridades
        //eh ordenada
    }

    if (s.count(23) == 1) //checa se o 23 existe dentro do set s
    O(log(n))
    {
        cout << "achei\n";
    }
}

```

```

    }
    if (m[23] > 0) //checa se o 23 existe no map m, lembrando que se
ele nao existir
        //o map vai adicionar o 23 e colocar 0 como
seu valor
    {
        cout << "achei\n";
    }

    s.erase(10); //remove o cara de valor 10 do set, lembrando q se
ele nao existir voce tomara
        //runtime error! O(logn)
    m.erase(50); //remove o cara de valor 50 do map O(logn)
    //c++11 only
    for (auto u : s)
    {
        cout << u << "\n"; //imprime todos os elementos dentro
de s
    }
    for (auto u : m)
    {
        cout << u << "\n"; //imprime todos os elementos dentro
de s
    }

    while (!pq.empty())
    {
        int v = pq.top(); //pega o primeiro elemento O(1)
        cout << v << "\n";
        pq.pop(); //Remove o primeiro elemento O(log(n))
    }

    cout << m.size() << " " << s.size() << " " << pq.size() << "\n";
    //pega o numero de elementos em m, s e pq, O(1)

    return 0;
}

```

[Read more »](#)


0



bssanches



4 months ago



0

Aula 5

By [bssanches](#), [history](#), 5 months ago,

Uma função em programação, funciona da mesma forma que na matemática. Por exemplo a função $F(x) = x^2$ pode ser escrita em c++ como

```

int F(int x)
{
    return x * x;
}

```

O primeiro `int` representa o tipo de retorno da função, logo depois devemos definir o nome da função e seus parâmetros. O retorno por sua vez é o resultado calculado que queremos.

Um outro exemplo de função:

```

int F(int x, int y)
{
    return x + y;
}

```



```

int main()
{
    int a = F(2, 3); //Para chamar a função, quando chamamos ela assim
    x = 2 e y = 3
    return 0;
}

```

Vale notar que a `main` também é uma função.

Quando declaramos uma variavel dentro da função, ela só vale dentro da mesma, portanto os parâmetros da função não serão reconhecidos fora dela.

```

int x = 10;
void f(int x) //void é o tipo de retorno que não retorna nada :P, é
usado quando não temos retorno na função
{
    printf("x na função = %d\n", x);
}

int main()
{
    f(12345);
    printf("x na main = %d\n", x);
}

```

Podemos também chamar outra função de dentro da função:

```

int ehPar(int x)
{
    if (x%2 == 0)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

int calcula(int x) //funcao que se x for par retorna 2 * x e se x for
impar retorna x^2
{
    int retorno = ehPar(x);
    if (retorno == 1)
    {
        return 2 * x;
    }
    else
    {
        return x * x;
    }
}

int main()
{
    int retorno = calcula(10);
    return 0;
}

```

Utilizando esse pensamento, podemos também chamar a própria função dentro dela mesma. A isso damos o nome de recursão:

```

int fib(int x) //função que calcula fibonacci de x
{
    if (x == 0)
    {
        return 1;
    }
    else if (x == 1)
    {
        return 1;
    }
    else
    {
        return fib(x - 1) + fib(x - 2);
    }
}

```

Vale notar que quando definimos uma recursão, precisamos sempre lembrar de definir o caso base. O caso base é o caso que faz com que a recursão não entre em loop infinito, é o caso em que já sabemos a resposta e não precisamos chamar a recursão para frente (No exemplo a cima os casos base são `x == 0` e `x == 1`)

Um exemplo utilizando varias variaveis, o "escolhe" (ou combinatória)

```

int choose(int n, int k)
{
    if (n < k)
    {
        return 0;
    }
    else if (k == 0)
    {
        return 1;
    }
    else
    {
        return choose(n - 1, k) + choose(n - 1, k - 1);
    }
}

int main()
{
    int ret = choose(10, 5);
    printf("ret = %d\n", ret);
}

```

[Read more »](#)



+5



[bssanches](#)





5 months ago



[0](#)

Aula 3

By [bssanches](#), [history](#), 5 months ago,  

Olás, segue um resumo da aula

For aninhado

`for` são comandos que podem ser aninhados (Postos um dentro do outro), assim como os `if`. Um exemplo de como fazer isso:

```

for (int i = 0; i < 10; i++) //Esse for vai executar 10 vezes
{
    for (int j = 0; j < 5; j++) //Esse for vai executar 5 vezes pra cada
        iteração do for de cima
}

```

```
{  
    printf("%d %d\n", i, j); //Esse printf vai ser executado 50  
    vezes  
}  
}
```

Contar operações

Para contar operações só iremos considerar os `for`. Para calcular as operações precisamos considerar quantas vezes o que está dentro do `for` será executado. Por exemplo:

```
for (int i = 0; i < 100; i++) //Esse for vai executar 100 vezes  
{  
    //codigo que n seja outro for e será executado 100 vezes, por tanto  
    100 operações  
}
```

Outro exemplo:

```
for (int i = 0; i < 100; i++) //Esse for vai executar 100 vezes  
{  
    for (int j = 0; j < 5; j++) //Esse for vai executar 5 vezes  
    {  
        //codigo que n seja outro for e será executado 500 vezes, por  
        tanto 500 operações  
    }  
}
```

10^8 operações demoram 1s para serem executados! Calcule quantas operações seu código executa antes de submeter para não receber o veredito "Timelimit exceeded"

Matrizes

Uma matriz pode ser vista como um conjunto de vetores, da mesma forma que vetores podem ser vistos com um conjunto de variáveis.

	0	1	2	3	4
0					
1					
2					
3					
4					

O primeiro indice de uma matriz representa a linha, enquanto o segundo representa a coluna, por exemplo:

```
int mat[100][5]; //declara uma matriz de 100 linhas e 5 colunas

for (int i = 0; i < 100; i++)
{
    for (int j = 0; j < 5; j++)
    {
        scanf("%d", &mat[i][j]); //le o elemento na linha i coluna j
    }
}

for (int i = 0; i < 100; i++)
{
    for (int j = 0; j < 5; j++)
    {
        printf("%d\n", mat[i][j]); //imprime o elemento na linha i e
        coluna j
    }
}
```

Vale lembrar que `i++` é a mesma coisa que `i = i + 1`, assim como `i--` é a mesma coisa que `i = i - 1`

O tamanho de uma matriz é dado pela multiplicação de suas dimensões, por tanto a matriz acima tem 500 de tamanho, ou seja equivale a um vetor de 500 posições (Tentem manter suas matrizes/vetores com no máximo 10^6 de tamanho)

Struct

Uma `struct` é uma variavel customizada, no sentido de que podemos colocar o que quisermos dentro dela. Por exemplo

```
struct exemplo{
    int idade;
    char nome[100];
};

int main()
{
    exemplo ex;
    ex.idade = 100;
    ex.nome[0] = 'a';
    ex.nome[1] = 'b';
    ex.nome[2] = 'a';
    ex.nome[3] = '\\0';
    printf("nome = %s\\nidade = %d\\n",ex.nome, ex.idade);
    return 0;
}
```

O uso de structs irá ser mais util mais pra frente!

Lembrete: Sempre inicializem suas variaveis! Variaveis declaradas dentro da `main` possuem valores aleatórios (Depende do que estava executando na posição de memória em que seu programa está agora), variaveis declaradas fora da main inicializam automaticamente com 0!

Abraços!

[Read more »](#)

▲ +5 ▼



[bssanches](#)



5 months ago



0

Aula 2 (Inferno na terra)

By [bssanches](#), [history](#), 6 months ago,

Olás pessoas,

Segue um resumo da aula:

Comando "if"

Como utilizar o "se não" e o "se não se" (Quando a condição do `if` der falso, o "se não" e o "se não se" serão testados em ordem)

Exemplo:

```
#include <stdio>
```

```
int main()
{
    int A;
    scanf("%d", &A);
    if (A > 10) //testa primeiro esse
    {
        printf("A eh maior que 10");
    }
    else if(A < 10) //se o primeiro if for **falso** (A não é maior que 10), checa esse
    {
        printf("A eh menor que 10");
    }
    else //se o primeiro e o segundo forem falsos (A não é maior que 10 nem menor), então executa esse
    {
        printf("A eh igual a 10");
    }
}
```

```

    }
    return 0;
}

```

Lembrando que uma vez que a condição de um `if` é verdadeira, a execução entra dentro do `if` e não continua testando os outros `else if` e `else` (A não ser que um novo `if` apareça embaixo)

Exemplo:

```

#include <stdio>

int main()
{
    int A = 9;
    if (A > 10) //não entra nesse, A é menor que 10
    {
        printf("A eh maior que 10");
    }
    else if(A < 10) //Entra nesse, A é menor que 10
    {
        printf("A eh menor que 10");
    }
    else //Não entra nesse pois ja entrou no else if de cima
    {
        printf("A eh igual a 10");
    }

    if (A < 10) //entra nesse pois eh um novo if
    {
        printf("Mais um print");
    }
    return 0;
}

```

Tipo long long

O `long long` é outro tipo de variável que guarda números inteiros, assim como o `int`. A diferença entre o `int` e o `long long` está na quantidade de valores que eles conseguem armazenar.

Capacidade de armazenamento:

- `int` = $[-2 * 10^9, 2 * 10^9]$
- `long long` = $[-9 * 10^{18}, 9 * 10^{18}]$

(Valores aproximados)

Para ler um `long long` utilizem `%lld`

Para iniciar um valor fixo pra uma variável, utilizem o sufixo `LL`

Exemplo:

```

#include <stdio>

int main()
{
    long long a;
    long long b = 1000000000000LL;
    scanf("%lld",&a);
    printf("%lld\n",a + b);
    return 0;
}

```

Comando "while"

Como utilizar o `while` :

O `while` é um loop, ou seja, ele fica repetindo até que sua condição seja falsa (Uma forma de olhar o `while` é como um `if` que se repete)

exemplo:

```
//Programa que imprime todos os números de 0 a 9
#include <stdio>

int main()
{
    int i = 0; //não esqueçam de inicializar o contador
    while (i < 10) //Vai repetir o que ta dentro do while até que o i
seja igual a 10
    {
        printf("%d\n",i);
        i = i + 1; //incrementa o contador a cada vez que o loop executa
    }
    return 0;
}
```

outro exemplo:

```
//Programa que soma todos os pares até um dado n
#include <stdio>

int main()
{
    int n, sum = 0;
    scanf("%d",&n);
    int i = 0; //não esqueçam de inicializar o contador
    while (i <= n) //Vai repetir o que ta dentro do while até que o i
seja maior que n
    {
        sum = sum + i;
        i = i + 2; //Incrementa o contador a cada vez que o loop
executa, de dois em dois
    }
    printf("%d\n",sum);
    return 0;
}
```

Comando "for"

O `for` é um `while` de maneira mais organizada.

O formato é: `for(valor_inicial; condição_de_parada; valor_do_incremento)`

Exemplo:

```
#include <stdio>

int main()
{
    int n;
    scanf("%d",&n);
    //começa i com 0, a cada iteração soma 1 no i e para quando o i for
maior que n
    for (int i = 0; i <= n; i = i + 1)
    {
```

```

        printf("%d\n", i);
    }
    return 0;
}

```

Outro exemplo:

```

#include <stdio>

int main()
{
    int n;
    scanf("%d", &n);
    for (int i = 1; i <= n; i = i * 2) //começa i com 1, a cada iteração
        dobra o valor de i e para quando o i for maior que n
    {
        printf("%d\n", i);
    }
    return 0;
}

```

É possível também, omitir o valor inicial e declara-lo antes do `for`:

```

//programa que soma todos os números pares de 1 a n
#include <stdio>

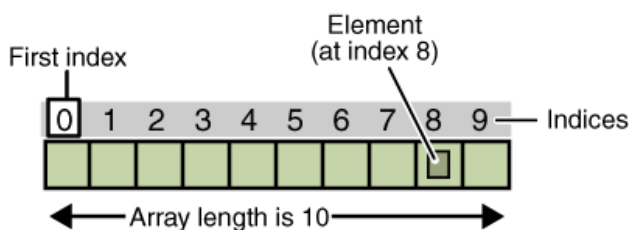
int main()
{
    int n;
    scanf("%d", &n);
    int i = 1;
    int soma = 0;
    //começa i com 1, a cada iteração aumenta i em 2 e para quando o i
    for maior que n
    for (; i <= n; i = i + 2)
    {
        soma = soma + i;
    }
    printf("%d\n", soma);
    return 0;
}

```

Não esqueçam de definir bem sempre as 3 coisas que fazem com que o `for` e o `while` funcionem: O valor inicial, a condição de parada e o valor do incremento a cada iteração.

Vetores

Vetores são varias variáveis declaradas de uma única vez, um exemplo gráfico de vetor a seguir (Cada bloquinho é como se fosse uma única váriavel, e juntos eles compõem o vetor)



Exemplo de manipulação de vetor:

```

#include <stdio>

int main()

```



```
{
    int v[4];
    scanf("%d%d%d%d",&v[0], &v[1], &v[2], &v[3]);
    printf("%d %d %d %d\n",v[0], v[1], v[2], v[3]);
    return 0;
}
```

Outro exemplo de manipulação de vetor:

```
//programa que le N números e imprime N números
#include <stdio>
```

```
int main()
{
    int n;
    scanf("%d",&n);
    int v[10000];
    for (int i = 0; i < n; i = i + 1)
    {
        scanf("%d", &v[i]);
    }
    for (int i = 0; i < n; i = i + 1)
    {
        printf("%d\n", v[i]);
    }
    return 0;
}
```

Strings

Strings: Em c++ vetores de chars também recebem o nome de strings, e existem maneiras especiais de lidar com eles:

- Uma forma especial de se ler e imprimir um vetor de char é utilizando o `%s`
- Vale lembrar que quando você lê uma linha da entrada com `%s`, a leitura vai parar no primeiro espaço ou `\n` encontrado!
- Quando se lê uma sequência de caracteres da entrada utilizando o `%s`, um caracter especial sera adicionado na ultima posição, o `\0`. Esse caracter indica o fim da string (Então lembrem de colocar um valor a mais no tamanho do vetor pra comportar o `\0` !)
- Não utilizar & no scanf quando for utilizar o `%s`
- Lembrar de incluir o (No windows funciona sem incluir, mas não ira funcionar no linux! Então incluam sempre)

```
#include <stdio>
#include <string>

int main()
{
    char v[4]; //cuidado, nesse vetor soh cabem 3 letras, uma vez que a
    ultima eh um \0
    scanf("%s", v);
    printf("%s\n",v);
    return 0;
}
```

É possível ler um vetor de chars, letra por letra também, mas dai para imprimir também sera necessário imprimir letra a letra (A não ser que você adicione um `\0` manualmente ao fim da string)

Exemplo:

```
#include <stdio>
#include <string>
```

```
int main()
{
    char v[4]; //cuidado, nesse vetor só cabem 3 letras, uma vez que a
    ultima eh um \0
    for (int i = 0; i < 3; i = i + 1)
    {
        scanf("%c", &v[i]);
    }

    printf("\nSem %s\n");
    for (int i = 0; i < 3; i = i + 1) //imprime letra a letra
    {
        printf("%c", v[i]);
    }
    printf("\n\n");

    v[3] = '\0'; //manualmente coloca o \0 ao fim da string
    printf("Com %s\n");
    printf("%s\n", v);

    return 0;
}
```


Existe também uma função chamada `strlen` para descobrir o tamanho da string:

```
#include <stdio>
#include <cstring>

int main()
{
    char v[4]; //cuidado, nesse vetor soh cabem 3 letras, uma vez que a
    ultima eh um \0
    scanf("%s", v);
    int sz = strlen(v);
    printf("Tamanho da string: %d\n", sz);
    for (int i = 0; i < sz; i = i + 1)
    {
        printf("%c\n", v[i]);
    }
    return 0;
}
```

Acho que é isso, qualquer duvida mandem mensagem no fb ou postem no grupo! Abraços!

[Read more »](#)

 **+11** 





[bssanches](#)



6 months ago



Aula 1 — Links complementares

By [Estevam](#), [history](#), 6 months ago, , 

Olás², pro pessoal que foi pra aula de sexta (ou quem mais quiser)

Deixo aqui pra vcs alguns links complementares, que passei na aula ou que estavam no slide:

Slide utilizado: [bit.ly/gema2017](https://uspbr-my.sharepoint.com/personal/estevam_arantes_usp_br/_layouts/15/WopiFrame.aspx?docid=1186bc7991e2c4d07bf1f705ae687eb7f&authkey=AU9r_AbLegNmiUBgCeuzVj8&action=view) (link por extenso caso alguém prefira: https://uspbr-my.sharepoint.com/personal/estevam_arantes_usp_br/_layouts/15/WopiFrame.aspx?docid=1186bc7991e2c4d07bf1f705ae687eb7f&authkey=AU9r_AbLegNmiUBgCeuzVj8&action=view)

Download do codeblocks: <http://www.codeblocks.org/downloads/26> (Baixar a versão codeblocks-16.01mingw-setup.exe)

[Cplusplus](#) — Site para referência de C e C++

Outros sites complementares para treino:

[Codcad](#) — Site com aulas de C++ e problemas extras (algumas aulas têm conteúdos diferentes do que ensinei hoje, mas é bom caso queiram complementar ou treinar um pouco mais).

[URI](#) — Problemas para treino (normalmente mais básicos)

[Hackerrank](#) — Outro site com aulas e mais exercícios para quem quiser revisar ou complementar.

Qualquer dúvida sintam-se à vontade pra me procurar

[Read more »](#)



[Estevam](#)





6 months ago



[1](#)

Aula 1

By [bssanches](#), [history](#), 6 months ago, , 

Olás,

Depois de toda aula faremos um blogpost com o resumo da aula (E mais algumas coisas que esqueci de falar).

Como começar um programa em c e c++

Em c++

```
#include <cstdio> //Esse é o jeito c++
```

```
int main()
{
    return 0;
}
```

Em c

```
#include <stdio.h> //Esse é o jeito c
```

```
int main()
{
    return 0;
}
```

Lembrando que o jeito c também funciona em c++.

Como usar o `scanf()` e o `printf()` e a declarar variáveis

```
#include <cstdio> //Esse é o jeito c++
```

```
int main()
{
    int n; //declaração de um inteiro chamado n
    scanf("%d",&n); //lê o n
    printf("%d\n", n); //imprime o n e coloca um enter a mais no final
    return 0;
}
```

Tipos de variáveis:

- `int nome_da_variavel`
- `char nome_da_variavel`

- `double nome_da_variavel`

Lembrando que nomes de variáveis não podem começar com números, nem conter espaços nem possuir acentos

Mascaras do `printf` e `scanf` para ler e imprimir cada tipo de variável

- `int` = `%d`
- `char` = `%c`
- `double` = `%lf`

Como ler e imprimir mais de um número

```
#include <cstdio> //Esse é o jeito c++
```

```
int main()
{
    int n, m; //declaração de um inteiro chamado n e outro chamado m
    (Poderia ser um em cada linha se quisesse)
    scanf("%d%d", &n, &m); //lê o n e o m
    printf("%d %d\n", n + m, n - m); //imprime o n mais o m e imprime o
    n menos o m
    return 0;
}
```

Existe um comando condicional chamado `if`, com ele você pode realizar "perguntas" do tipo SE

```
#include <cstdio> //Esse é o jeito c++
```

```
int main()
{
    int n, m; //declaração de um inteiro chamado n e outro chamado m
    (Poderia ser um em cada linha se quisesse)
    scanf("%d%d", &n, &m); //lê o n e o m
    if (n + m == 10) //tudo que estiver dentro de {} depois do if só
    sera executado se n + m for igual a 10
    {
        printf("igual a 10\n");
    }

    if (n + m != 10) //tudo que estiver dentro de {} depois do if só
    sera executado se n + m for diferente 10
    {
        printf("diferente de 10\n");
    }

    if (n + m > 10)
    {
        printf("maior que 10\n");
    }
    return 0;
}
```

As operações que podem ser feitas dentro do `if` são:

- `A == B` compara se A é igual a B
- `A != B` compara se A é diferente de B
- `A > B` compara se A é maior que B
- `A < B` compara se A é menor que B
- `A >= B` compara se A é maior igual a B
- `A <= B` compara se A é menor igual a B

É possível além disso, utilizar mais de uma condição dentro do `if` utilizando as cláusulas `and` e `or`

```
if (A == 10 and B == 20) //se A for igual a 10 e B for igual a 20
{
    printf("A eh 10 e b eh 20");
}

if (A == 10 or B == 20) //se A for igual a 10 ou B for igual a 20
{
    printf("A eh 10 ou B eh 20");
}
```

Os operadores possíveis em c/c++:

- | | | |
|---|---|---|
| A | - | B |
|---|---|---|
- | | | |
|---|---|---|
| A | + | B |
|---|---|---|
- | | | |
|---|---|---|
| A | * | B |
|---|---|---|
- | | | |
|---|---|---|
| A | / | B |
|---|---|---|
- | | | |
|---|---|---|
| A | % | B |
|---|---|---|

```
int c = 10 + 20; //soma 10 e 20
int d = 10 - c; //pega 10 e subtrai c (que no momento vale 30)
double e = 10.0 / 3.0 //divide 10 por 3 (lembrando de colocar o .0 pra
transformar em double)
int f = 7;
int g = f * c; // multiplica o f e o c (ou seja 7 * 30)
int h = c%f; //pega o resto da divisão de c por f (30%7 = 2)
```

[Read more »](#)



[bssanches](#)



6 months ago



[0](#)