UBERHUB
<CODE CLUB/>

Revisão e Ordenação

Aula 09

String

String é uma sequência de caracteres geralmente utilizada para representar palavras, frases ou textos de um programa.

Funções de String em c++

getline(cin,str) - lê a string até o \n

str.size() - retorna o tamanho de str

str.at(n) - retorna o char na posição n

str.push_back() - adiciona um único caractere ao final de str

Funções fora da main

Em C e C++, é permitido criar funções fora da int main, definindo as entradas, operações e valores retornados.

```
int f(int x) {
        return 2*x;
}
```

Funções em código

```cpp
#include <bits/stdc++.h>
using namespace std;
int f(int x) {
    return 2*x;
}
int main() {
    int a = 4;
    Cout << f(2) <<  " " << f(a) << " " << f(a+2) << endl;
```

Quando executarmos este código, será impresso:
4 8 12
Se referindo a f(2) = 4, f(4) = 8, f(4+2) = 12.

Praticar

Acessar o URI:

https://www.urionlinejudge.com.br

Passo a passo com o instrutor

String:

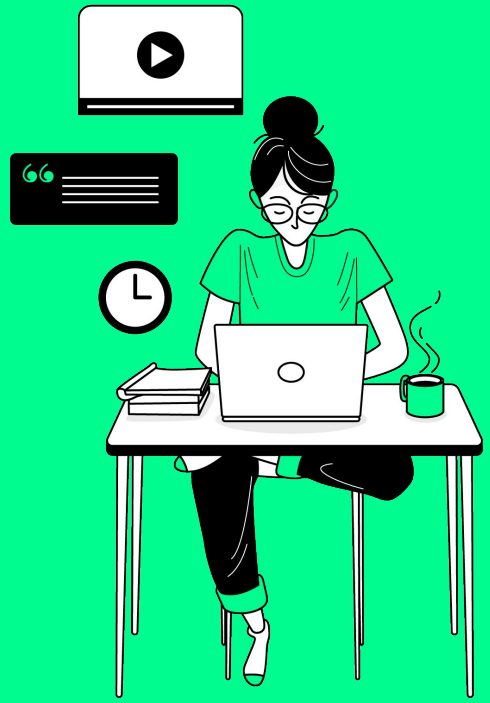
2694 2591
1168 1241

Função:

1555 1169

Consultar Fórum e Grupos para tirar dúvidas.

"O normal é ter muitas dúvidas, e superá-las com o esforço!"



Ordenação

É a operação de rearranjar os dados em uma determinada ordem.

sort()

A maior parte das vezes só queremos que o vetor seja ordenado, para isso utilizamos a função sort(), que pega um vetor ‘a’ e ordena de forma crescente ou decrescente.

```
#include<bits/stdc++.h>
using namespace std;

void show(int a[]){
    for(int i = 0; i< 10; ++i)
        cout << a[i] <<" ";
}
int main(){
    int a[10]= {1, 5, 8, 9, 6, 7, 3, 4, 2, 0};
    cout << "\n O vetor antes de ser ordenado eh: ";
    show(a);
    cout << "\n O vetor apos ser ordenado eh: ";
    sort(a, a+10);
    show(a);
}
```

sort()

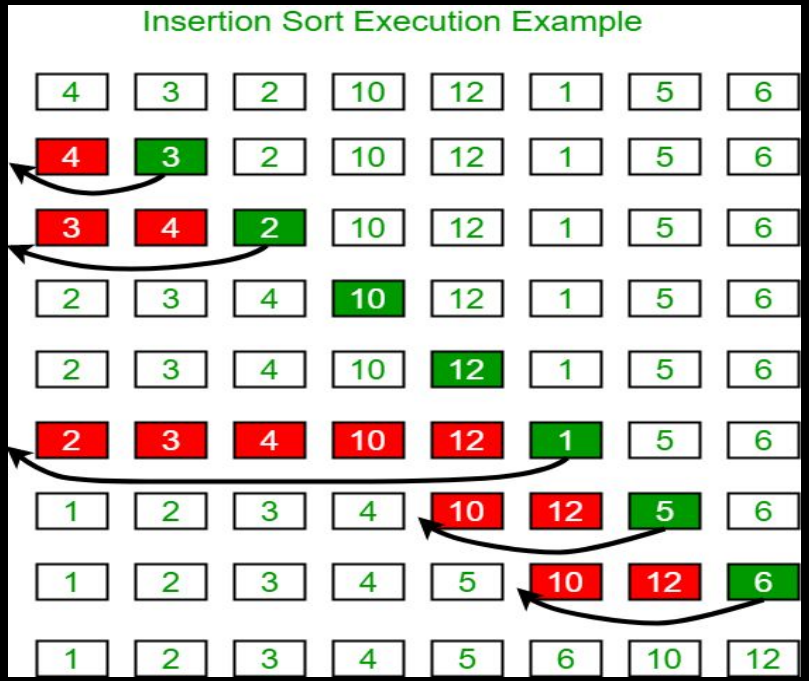
Vetor Ordenado de Forma decrescente.

```
#include<bits/stdc++.h>
using namespace std;

void show(int a[]){
    for(int i = 0; i< 10; ++i)
        cout << a[i] <<" ";
}
int main(){
    int a[10]= {1, 5, 8, 9, 6, 7, 3, 4, 2, 0};
    cout << "\n O vetor antes de ser ordenado eh: ";
    show(a);
    cout << "\n O vetor apos ser ordenado eh: ";
    sort(a, a+10, greater<int>());
    show(a);
}
```

Insertion sort

Insertion sort é um algoritmo que funciona da maneira que ordenamos cartas de um baralho na mão.



Insertion sort em código

```
void insertion_sort(std::vector<int> &vetor) {
    for (int i = 1; i < vetor.size(); i++) {
                int escolhido = vetor[i];
                int j = i - 1;
                while ((j >= 0) && (vetor[j] > escolhido)) {
                        vetor[j + 1] = vetor[j];
                        j--;}
                        vetor[j + 1] = escolhido;
                }
    }
```

Selection sort

A **ordenação por seleção** (do inglês, ***selection sort***) é um algoritmo de ordenação baseado em passar sempre o menor valor do vetor para a primeira posição (ou o maior dependendo da ordem requerida), depois o de segundo menor valor para a segunda posição, e assim é feito sucessivamente com os (n-1) elementos restantes, até os últimos dois elementos.

É composto por dois laços, um laço externo e outro interno. O laço externo serve para controlar o índice inicial e o interno percorre todo o vetor. Na primeira iteração do laço externo o índice começa de 0 e cada iteração ele soma uma unidade até o final do vetor e o laço mais interno percorre o vetor começando desse índice externo + 1 até o final do vetor. Isso ficará mais explícito no exemplo.

Exemplo: vetor = 9 - 7 - 8 - 1 - 2 - 0 - 4

1° 0 - 7 - 8 - 1 - 2 - 9 - 4

2° 0 - 1 - 8 - 7 - 2 - 9 - 4

3° 0 - 1 - 2 - 7 - 8 - 9 - 4

4° 0 - 1 - 2 - 4 - 8 - 9 - 7

5° 0 - 1 - 2 - 4 - 7 - 9 - 8

6° 0 - 1 - 2 - 4 - 7 - 8 - 9

Selection sort em código

```
void SelectionSort(int vetor[], int tam) {
    for (int indice = 0; indice < tam; ++indice) {
        int indiceMenor = indice;
        for (int indiceSeguinte = indice+1; indiceSeguinte < tam; ++indiceSeguinte) {
            if (vetor[indiceSeguinte] < vetor[indiceMenor]) {
                indiceMenor = indiceSeguinte;  } }
        int aux = vetor[indice];
        vetor[indice] = vetor[indiceMenor];
        vetor[indiceMenor] = aux; }}
```

Bubble sort

Este algoritmo percorre a lista de itens ordenáveis do início ao fim, verificando a ordem dos elementos dois a dois, e trocando-os de lugar se necessário. Percorre-se a lista até que nenhum elemento tenha sido trocado de lugar na passagem anterior.

Bubble sort em código

```
void BubbleSort( int A[],int n) {
   int temp, flag;
   for(int k=1;k<n;k++) {
         flag = 0;
               for(int i=0;i<n-k;i++) {
                     if(A[i]>A[i+1]) {
                           temp = A[i];
                           A[i] = A[i+1];
                           flag = 1;
                     }
               }
               if(flag == 0)
               Break;
   }
}
```

Praticar

Acessar o URI e NEPS:

https://www.urionlinejudge.com.br
https://neps.academy

Passo a passo, com o instrutor:

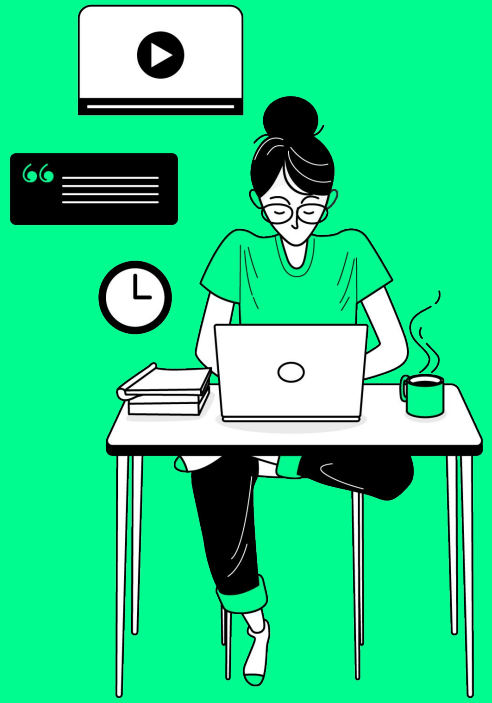
1548	2479
1244	1252

Praticar no NEPS:

400	8

Consultar Fórum e Grupos para tirar dúvidas.

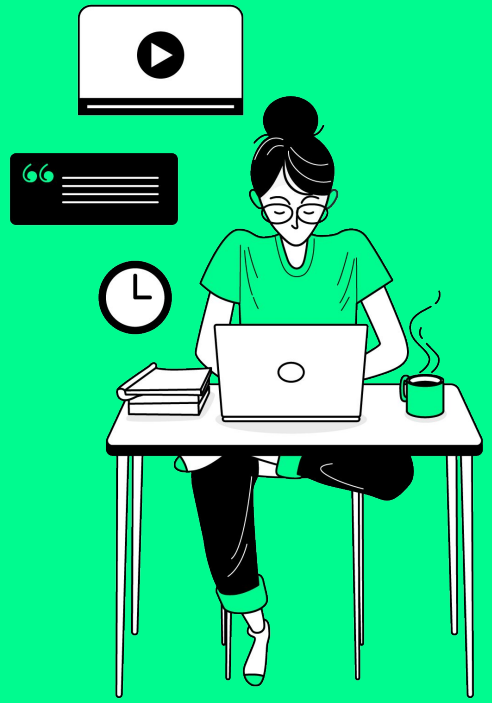
"O normal é ter muitas dúvidas, e superá-las com o esforço!"



Boas Práticas

Nas aulas presenciais, antes de sair do laboratório, favor:

- Desligar o Computador
- Desligar o Monitor
- Arrumar o Teclado e Mouse
- Limpar a Estação de Trabalho
- Recolher os Pertences
- Encostar a Cadeira



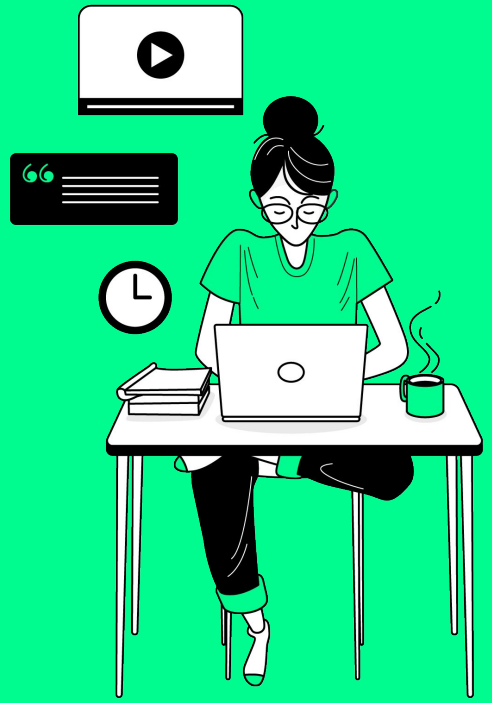
Links Úteis

UberHub Code Club:
https://uberhubcode.com.br/

Baixar CodeBlocks:
http://www.codeblocks.org/downloads/26

Exercícios:
https://www.urionlinejudge.com.br

Aulas Prof. André Backes:
https://www.youtube.com/user/progdescomplicada



UBERHUB
<CODE CLUB/>

Obrigado!

UberHub Code Club