LABORATÓRIO DE PROGRAMAÇÃO I

Departamento de Ciência da Computação

AULA 5 - BUSCA BINÁRIA

Ordenação - binary_search

```
...
struct pessoa {
        int id;
        string nome;
        ...
};
bool cmp(pessoa i, pessoa j) { return (i.id < j.id || i.id == j.id && i.nome < j.nome); }
int main() {
        vector<pessoa> v;
        ...
        stable_sort (v.begin(), v.end(), cmp);
        pessoa j;
        ...
        bool r = binary_search(v.begin(), v.end(), j, cmp); // pode usar uma função de comparação, assim como o sort
}
```

Vetores - binary_search

Saiba mais em:

http://www.cplusplus.com/reference/algorithm/binary_search/

Ordenação - lower_bound

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std; // lound_bound -> std
int main() {
        vector<int> v;
        int i, j;
        for(i=0; i < 1000; i++) {
                cin >> j;
                v.push_back(j);
        }
        stable_sort (v.begin(), v.end());
        cin >> j;
        vector<int>::iterator it;
        it = lower_bound(v.begin(), v.end(), j);
}
// lower_bound - retorna iterator para o primeiro
// elemento que seja maior ou igual a j

// para saber a posição, faça it-v.begin()
```

Ordenação - upper_bound

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std; // upper_bound -> std
int main() {
        vector<int> v;
        int i, j;
        for(i=0; i < 1000; i++) {
                cin >> j;
                v.push_back(j);
        }
        stable_sort (v.begin(), v.end());
        cin >> j;
        vector<int>::iterator it;
        it = upper_bound(v.begin(), v.end(), j);
}
```

// upper_bound - retorna iterator para o primeiro elemento que seja maior que j

// para saber a posição, faça it-v.begin()

Vetores - {lower,upper}_bound

Saiba mais em:

http://www.cplusplus.com/reference/algorithm/lower_search/

http://www.cplusplus.com/reference/algorithm/upper_search/

Ordenação - binary_search

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std; // binary_search -> std
int main() {
        vector<int> v;
        int i, j;
        for(i=0; i < 1000; i++) {
                cin >> j;
                v.push_back(j);
        }
        stable_sort (v.begin(), v.end());
        cin >> j;
        if( binary_search(v.begin(), v.end(), j) )
                cout << “Tá lá!\n”;
}
// binary_search
// retorna true caso o elemento exista no vetor
// retorna false caso contrário
```