Corso CP

- basi di C++
  - input / output
  - string
- algoritmi
  - complessità computazionale
  - sorting
  - binary search

- STL
  - sort()
  - binary_search()
- Data type
  - vector
  - set
  - map

C++

Input / output

```
#include <iostream>

using namespace std;

...

cin>>x;
cout<<x;
```

```
#include <fstream>

using namespace std;

ifstream in(“fileIn.txt”);
ofstream out(“fileOut.txt”);

...

in>>x;
out<<x;
```

C++

string

```
#include <string>

using namespace std;
...
string str1 = “enrico”;
string str2;
cin>>str2;

cout<<str1 + “ ” + str2<<endl;
cout<<str2.size()<<endl;
```

C++

string

Access

- []
- at()

Capacity

- size()
- length()

Search

- find_first_of()
- find_last_of()

Operations

- substr()

Operators

- +
- ==
- !=
- <
- >

Numeric conversions

- stoi()
- to_string()

algoritmi

Coplessità computazionale

$O(1)$

$O(\log N)$

$O(N)$

$O(N*\log N)$

$O(N^2)$

$O(N^3)$

$O(2^N)$

algoritmi

Sorting

6 4 9 0 1 3 2

0 1 2 3 4 6 9

algoritmi

Sorting

Selection sort

$O(n^2)$

algoritmi

Sorting

```
void ordina_array(int *arr, size_t n){

    for(size_t i=0; i<n; i++){

        int min = i;
        for(size_t j=i+1; j<n; j++){

            if(arr[j] < arr[min])
                min = j;
        }

        swap(arr[i], arr[min]);
    }
}
```

pausa

algoritmi

Binary search

target = 11

0 2 3 4 7 8 9 11 14

algoritmi

Binary search

target > 7

algoritmi

Binary search

algoritmi

Binary search

target > 9

algoritmi

Binary search

algoritmi

Binary search

target = 11

algoritmi

Binary search

```c
int binary_search(int *arr, size_t n, int target){

    size_t l = 0;
    size_t r = n-1;

    while (l <= r) {
        size_t mid = (l + r) / 2;

        if (arr[mid] == target)
            return mid;

        if (arr[mid] < target)
            l = mid + 1;
        else
            r = mid - 1;
    }

    return -1;
}
```

STL

Standard template library

cppreference

STL

sort

```
#include <algorithm>
...
void sort( RandomIt first, RandomIt last, Compare comp );
```

O(N*logN)

STL

sort

```
void ordina_array(int *arr, size_t n){

    sort(arr, arr+n);
}
```

STL

Binary search

```
#include <algorithm>

...

ForwardIt lower_bound(ForwardIt first, ForwardIt last, const T& value);
```

Ritorna un iteratore al primo elemento del range [first, last)
non minore di <value>

STL

Binary search

```
1  2  4  5  5  6
```

```
0  ->  0
1  ->  0
2  ->  1
3  ->  2
4  ->  2
5  ->  3
6  ->  5
7  ->  Not found
```

STL

Binary search

```
int binary_search(int *arr, size_t n, int target){

    int *result = lower_bound(arr, arr+n, target);

    if(result == arr+n){
        return -1;
    }else{
        return (*result == target) ? (result - arr) : -1;
    }
}
```

Data type

vector

Access

- []
- at()

Capacity

- size()
- empty()

Iterators

- begin()
- end()

Modifiers

- clear()
- insert()
- erase()
- push_back()

Operators

- ==
- !=
- swap()

Data type

vector

```cpp
int main(){

    int N;
    cin>>N;

    vector<int> v;

    for(int i=0; i<N; i++){
        int temp;
        cin>>temp;
        v.push_back(temp);
    }

    return 0;
}
```

Data type

set

Insieme ordinato di oggetti, senza duplicati

Data type

set

Capacity
- empty()
- size()

Iterators
- begin()
- end()

Modifiers
- clear()
- insert()
- erase()

Lookup
- count()
- find()

Data type

set

```cpp
int main(){

    set<int> s;

    s.insert(1);
    s.insert(8);
    s.insert(4);
    s.insert(2);
    s.insert(4);


    cout<<"elemento da cercare: ";
    int target;
    cin>>target;

    if(s.find(target) != s.end()){
        cout<<"elemento presente\n";
    }else{
        cout<<"elemento non presente\n";
    }

    return 0;
}
```

Data type

map

Struttura dati che associa ad una chiave un valore

```
["gennaio"]   ->   31
["febbraio"]  ->   28
["marzo"]     ->   31
["aprile"]    ->   30
```

Data type

map

Access
- []
- at()

Capacity
- empty()
- size()

Iterators
- begin()
- end()

Modifiers
- clear()
- insert()
- erase()

Lookup
- count()
- find()

problemi

`ois_isogram`