Standard Template Library

Standard Template Library (*STL)*, is a library in C++ which contains container classes, iterators, algorithms.

The STL classes can be split in:
- Container classses (containers are objects which can contain other objects)
  - Secvențe
    - Vector
    - Deque
    - list
  - Associative containers
    - Set
    - map
  - Container adapters
    - Stack LIFO
    - Queue FIFO
    - Priority_queue
  - String
    - String
    - Rope
  - bitset
- Operations/utilities
  - Iterators (are equivalent to array indexes)
  - Algorithms (are template functions):
  - Auto_ptr

String – allows operations with strings

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
string s1 = "Hello ", s2 = " world!";
string s3 = s1+s2;
//overloading operator +=
s3 += "\n";
cout << s3;
//printing in C stile
printf("\ns3 = %s\n", s3.c_str());
//overloading operator ==
string raspuns;
```

```
cout << "Is this the laboratory 13 of C++?"; cin>>raspuns;
if (raspuns == "yes"){cout << "Correct answer\n";
} else
{
cout << "Wrong answer\n";
}
cin.ignore();
//handling strings
string name = "C++ Lecture\n";
cout << "Initial string: "<< name;
name.replace(0, 4, "Laboratory");
cout << "Replace: "<< name;
string s4 = name.substr(9, name.size());
cout << "Substring: " << s4;
getchar();
return 0 ;
}
```

Operations with lists

Example of handling a list of strings:

```
#include <iostream>
#include <list>
#include <string>
using namespace std;
int main()
{
//declaring a list of strings
list <string> name;
//adding informations in the list
name.push_back("middle name");
name.push_front("last name");
name.push_front("first name");
//displaying the informations stored in the list
cout << "The list is: ";
for (list<string>::iterator it = name.begin(); it != name.end();
it++)
{
cout << *it << ", ";
}
cout << endl;
//displaying the last element
cout << "The last elemet is: " << (string)name.back() << endl;
//find the dimension
cout << "The dimension of the list is: " << name.size() << endl;
```

```
//sorting the list
name.sort();
//displaying the informations stored in the list
cout << "The list is: ";
for (list<string>::iterator it = name.begin(); it != name.end();
it++)
{
cout << *it << ", ";
}
cout << endl;
getchar();
}
```

!!! Create a list with double elements handle the list (similar to the previous example).

Operations with map

The objects of the type map contain pairs of the form <key, value>.

Example:

```
#include <iostream>
#include <map>
#include <string>
using namespace std;
int main()
{
//declaring a mapping for days names
map <string, int> days;
//adding informations in map
days["Monday"] = 1;
days["Tuesday"] = 2;
days["Wednesday"] = 3;
days["Thursday"] = 4;
days["Friday"] = 5;
days["Saturday"] = 6;
days["Sunday"] = 7;
//display the informations from map
cout << "Day-Name\n";
map <string, int>::const_iterator it;
```

```
for (it = days.begin(); it != days.end(); it++)
{
cout << it->second << " " << it->first << "\n";
}
//ex: read a text from a file and replace the name of the day with
its number
getchar();
return 0;
}
```

Binding iterators with I / O

Example: Read from a file and write into another file the sorted words and eliminate the duplicates.

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <algorithm>
#include <iterator>
using namespace std;
int main()
{
string from, to;
cout << "The name of the file which is read : "; cin >> from;
cout << "The name of the file where we write : "; cin >> to;
//open the file
ifstream is(from.c_str());
//define file iterators
istream_iterator <string> isi(is);
istream_iterator <string> eos; //end of stream
//define vector which contains informations from the file
vector <string> v(isi, eos);
//sort words from the vector
sort(v.begin(), v.end());
//open file for writing
ofstream os(to.c_str());
//define iterator
ostream_iterator <string> osi(os, "\n");
//write in a file without duplications
unique_copy(v.begin(), v.end(), osi);
return (!is.eof() && !os);
}
```

Homework (use containers, iterators, algorithms defined in STL**)**:

1. Read from a text file. Display how many times a word <word, nb_occurrences> occurs. Read from the file words like (and, a, an, ...) eliminate them from the text and create a new file containing the new text.
2. Implement a program which calculates the exam grades obtained by a student. A student is characterized by: a name, practical_grade, exam_grade, number_of_absences. The program allows:
    - Saving the list of students into a file
    - Reading from file the list of students
    - Sorting the list of students by names
    - Searching by name for a student in the list
    - Displaying all the students which obtained a grade between x and y. x and y are read from the keyboard and are verified to be valid
    - Displaying the students which have less than 12 presences