Laborator 2 - Standard Template Library (STL)

Standard Template Library (STL)

2.1 Standard Template Library (STL)

Biblioteca STL este compusa din clase de tip container, algoritmi si iteratori. Scopul acestora este sa implementeze structuri de date frecvent utilizate, precum si operatii si algoritmi specifici acestor structuri. Prin container intelegem clase care gestioneaza structuri de date precum vectori, liste, stive, cozi etc. Acestea ar trebui sa poata stoca orice tip de data. Din acest motiv utilizeaza sabloane (templates). Mai jos este prezentat un exemplu de utilizare al clasei *stack*.

Exemplul 2.1.

```
#include <iostream>

#include <stack>

using namespace std ;

int main ( )

{

  stack<int> myStack ;

  int i ;

   for ( i = 1 ; i <= 10 ; i++)

        myStack . push ( i ) ;

  cout<< " Scoatere elemente din stiva ... " ;

  cout<<endl ;

  while ( ! myStack . empty ( ) )
```

```
  {
     cout<<myStack . top()<<" " ;
     myStack . pop ( ) ;
  }
   cout<<endl ;
   return 0 ;
}
```

Pentru a accesa obiectele din cadrul containerelor se utilizeaza iteratori. Acestia generalizeaza notiunea de pointer, indicand un obiect din cadrul unui container. In exemplul urmator este creat un vector cu 10 elemente. Cu ajutorul unui iterator sunt afisate valorile acestora.

Exemplul 2.2.

```
#include <iostream>
#include <vector>
using namespace std ;
int main ( )
{
    vector<int> myVector ;
    int i ;
    for ( i = 0 ; i < 10 ; i++)
          myVector . push_back ( i * i ) ;
    vector<int >:: iterator it ;
    for ( it = myVector.begin ( ) ; it != myVector.end( ) ; it++)
              cout<<*it<<" " ;
   cout<<endl ;
   return 0 ;
}
```

Sunt implementati algoritmi clasici sub forma unor functii care prelucreaza datele memorate in containere. Modificand codul de mai sus, elementele vectorului sunt inversate inainte de a fi afisate.

Exemplul 2.3.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std ;
int main ( )
{
    vector<int> myVector ;
    int i ;
    for ( i = 0 ; i < 10 ; i++)
        myVector.push_back ( i * i ) ;
    vector<int >:: iterator it ;
    cout << "Elementele vectorului:"<< endl;
    for ( it = myVector.begin ( ) ; it != myVector.end ( ) ; it++)
            cout<<*it<<" " ;
    cout<<endl ;
    reverse (myVector.begin( ) , myVector.end( ) ) ;
    cout << "Elementele vectorului dupa inversare:"<< endl;
    for ( it = myVector.begin ( ) ; it != myVector.end ( ) ; it++)
            cout<<*it<<" " ;
     cout<<endl ;
    system( " pause " ) ;
    return 0 ;
```

```
}
```

Exercitiul 2.1.

Creati un program pentru implementarea unei stive, utilizand vectori in STL, plecand de la codul sursa de mai jos:

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector <int> exemplu;
   /* TODO
    -  adaugare 3 intregi in stiva( adaugare la finalul vectorului)
    -  afisare elemente
    -  verificare daca mai sunt elemente in vector
      - creare o alta stiva si introducere valori
    - testare daca cei doi vectori sunt egali
   -  stergerea unuia dintre  vectori
   */
    return 0;
}
```

Exercitiul 2.2.

Descarcati de aici fisierul AF-INFO1-5-2.cpp, si completati codul conforma instructiunilor.

Exercitiul 2.3.

Implementati codul sursa pentru o lista de structuri utilizand STL, avand definirea de mai jos:

```
struct Nod{
```

```
  int ID;
  char nume[50];
  int an;
};
```

Indicatii:

- se va utiliza biblioteca STL, astfel:

```
list <struct Nod> lista;
```

- introducere date in lista (la final):

```
Nod el;
el.ID=1;
strcpy(el.nume,"nume 1");
el.an=1;
lista.push_back(el); // introducere la final in lista
```

- afisare elemente lista

```
// Afisare elementelista
for(std::list<Nod>::iterator it = lista.begin(); it != lista.end(); it++) {
    cout << "ID: " << it->ID << " nume: " << it->nume << " an: "<< it->an << endl;
  }
```

- introducere date in lista (la inceput, la final)
- construiti programul principal.

Last modified: Monday, 27 February 2023, 9:28 AM