S12 – STL

Alin Zamfiroiu

alin.zamfiroiu@csie.ase.ro

STL - string

- Înlocuiește lucrul cu char*.
- Este necesară includerea bibliotecii string.
- În interiorul clasei, folosește tot un char*, însă acest lucru este ascuns pentru cel care folosește clasa.

STL - string

- Declarare:
  - string sir_de_caractere;
- Inițializare:
  - sir_de_caractere = “ un sir de caractere”;
- Clasa string are supraîncărcați o serie de operatori pentru lucru cu șiruri de caractere, precum:
  - operator+
  - operator+=
  - operator<, >, ==
  - etc.

STL - string

- Pe lângă operatori clasa string implementează și o serie de metode:
  - length() – (sau size()) returnează lungimea șirului de caractere;
  - find(subsir, startP) – caută un subșir de caractere în cadrul șirului, pornind de la poziția startP;
  - substr(startP, nr_caractere) – extrage un subșir de caractere pornind de la poziția startP de lungimea dimensiune;
  - erase(startP, nr_caractere) - șterge substrigul de la pozitia startP de lungimea nr_caractere;
  - insert(startP, text) – insereaza subsirul text, incepand cu pozitia startP;

STL - Vector

- Pentru a utiliza clasa Vector trebuie inclusă în proiect librăria acestuia: #include<vector>

- Pentru declararea unui vector, trebuie să specificăm și tipul elementelor: vector<int> valori.

STL - Vector

- Pentru adăugarea unui element în vector se folosește metoda push_back().
- Pentru determinarea numărului de elemente din vector se folosește metoda size().
- Pentru parcurgere, vectorul se indexează ca oricare alt vector: valor[i].

STL - Vector

- Pentru parcurgere se poate folosi și un iterator.
- Iteratorul se declară pentru vectorul cu tipul de elemente creat: vector<int>::iterator it;
- Iteratorul reprezintă un pointer care se va deplasa între cei doi pointeri: cel de început și cel de sfârșit.

STL - Vector

- Iteratorul se inițializează cu pointerul de început al vectorului. Acesta se obține prin intermediul metodei begin();

- Adresa ultimului element din cadrul vectorului se obține prin intermediul metodei end().

STL - List

- Pentru utilizarea listei: list<int> lista_valori;
- Pentru listă există și metoda push_front(), care este folosită pentru adăugarea unul element în fața elementelor.
- Pentru listă se poate folosi și metoda insert(). Această metodă trebuie să primească un iterator pentru poziția unde se dorește inserarea.

STL - List

- Pentru list se poate folosi metoda sort(), care va sorta elementele din listă.

- Metoda reverse() – va schima ordinea elementelor în cadrul listei.

STL - Map

- Map se folosește atunci când se dorește o structură cu chei și valori.
- La declarare se stabilește tipul cheii și tipul valorii.
    - map<int, float> dictionar;
- Cheia poate fi un șir de caractere:
    - map<string, float> dictionar;

Algorithm și numeric

- Cele două librării conțin metode de prelucrare a containerelor STL.
  - » #include<algorithm>
  - » #include<numeric>

- Metode:
  - sort(begin, end);
  - sort(begin, end, metoda) – metoda este cea folosita pentru compararea elementelor;

Algorithm și numeric

- Metode:
  - accumulate(v.begin(),v.end(),startValue,[metoda]) – aduna toate elementele containerului v, startValue, este valoarea d la care incepe. Ultimul parametru este optional. Se poate define o metoda folosita pentru compunerea numerelor.
  - count(begin,end,valoare_cautata);
  - count_if(begin,end,metoda) – metoda primeste ca parametru un element si returneaza bool.
  - fill(begin,end,valoare);
  - fill_n(begin,n,valoare) – n reprezinta numarul de elemente initializate cu 0.

Tema

- Sa se realizeze o aplicatie in C++, care prin utilizarea STL sa realizeze gestiunea utilizatorilor unei aplicatii. Clasa Utilizator este create de fiecare cum doreste.
- Utilizarea containerelor si a metodelor din algorithm vor permite obtinerea de rapoarte cu privire la modul de accesare al aplicatiei de catre utilizatori.
- Rapoartele se fac in functie de atributele alese in clasa Utilizator.