Programare Orientată Obiect

Laborator 11

Despre ce vorbim azi

- Biblioteca standard de șabloane (STL)

Standard Template Library

Are trei componente majore:

- Containere – corespunzătoare structurilor de date
- Iteratori – conțin modalități de acces
- Algoritmi – metode specifice fiecărui container

Containerele

Sunt de trei tipuri:

- Secvențiale (structuri de date liniare)
    - vector (alocat dinamic, poate fi redimensionat)
    - list (listă dublu înlănțuită)
    - deque (listă de vectori)

Containerele

- Asociative (păstrează chei sau asocieri cheie-valoare)
  - set (mulțime de elemente sortate și unice)
  - multiset (mulțime sortată)
  - map (mulțime de perechi cheie-valoare unice)
  - multimap (mulțime de perechi cheie-valoare)

Containerele

- Adaptive (adaptează alte tipuri de container, nu suportă iteratori)
    - stack (stivă ce adaptează vector, list, *deque*)
    - queue (coadă ce adaptează list sau *deque*)
    - priority_queue (coadă de priorități ce adaptează *vector* sau deque)

Iteratorii

- Nu sunt altceva decât pointeri către elemente
- Au implementări diferite în funcție de containerul pe care lucrează
- Cu toate acestea au un comportament similar: pot fi incrementați, decrementați, pot implementa comparații, etc

Algoritmii

- Funcții de prelucrare specifice containerelor
- Sunt generice și pot fi aplicate pe orice tip de container
- Ex: insert, erase, find, sort, size, swap, etc

Precizări

- Alegerea containerului, respectiv a algoritmilor este făcută de programator în funcție de realitatea modelată în program
- Majoritatea containerelor au avantaje și dezavantaje ce le fac potrivite doar pentru anumite tipuri de prelucrări
- Fiind clase template, containerele pot conține elemente de tip de bază (int, float, etc) sau de tip definit de utilizator (structură, clasă)
- Pentru a putea folosi elemente obiecte este necesar ca acele clase să implementeze anumite metode sau operatori. Este de datoria programatorului să scrie funcțiile necesare.

Vectorul șablon

- În fișierul **vector** din namespace-ul std
- Folosește un spațiu contiguu de memorie și se realocă automat atunci când își atinge limita maximă
- Așadar este eficient în contextul inserărilor și ștergerilor la sfârșit
- Implementează iteratori
- Metode specifice: push_back, size, begin, end, reserve, resize, capacity, erase

Lista șablon

- Listă dublu înlănțuită ce conține pointerii **previous** și **next** pentru accesul elementului precedent, respectiv următor
- Este o structură de date necontiguă
- Ideală pentru lucrul cu obiecte atunci când se fac des adăugări și ștergeri în interior (doar se refac niște legături)
- Metode specifice: push_back, push_front, insert, begin, end
- Acceptă iteratori ce pot deveni invalizi doar dacă se șterge obiectul pointat

Șablonul deque

- Ocupă memorie necontiguă formată din bucăți contigue (o putem privi ca pe o listă de vectori)
- Recomandată pentru prelucrări ce presupun operații la început și la sfârșit
- Permite iteratori
- Are toate metodele specifice vectorului plus: push_front, pop_front

Șabloanele set și multiset

- Sunt implementate prin arbori binari de căutare
- Permit iteratori
- Au metode ca: insert, key_comp, value_comp, find

Șabloanele map și multimap

- Map implementează o relație de tip 1:1 pe când multimap una de tip 1:M (ambele sunt implementate tot prin arbori binari de căutare)
- Elementele sunt de tip *pair<key, value>*
- Pot accesa cheia, respectiv valoarea prin intermediul atributelor **first** și **second** din cadrul clasei

Iteratorii

- Permit adresarea directă a unui element
- Permit manipulare unitară a unui tip de dată complex
- Există operatori predefiniți ca: ostream_iterator, istream_iterator, reverse_iterator, insert_iterator

Algoritmi

- copy – permite copierea datelor dintr-un container în altul
- for_each – permite parcugerea colecțiilor
- sort – algoritm de sortare
- find – căutare cu returnare de iterator
- transform – permite mutarea elementelor dintr-un tip de container în alt tip cu transformare (aplicarea unei funcții matematice)