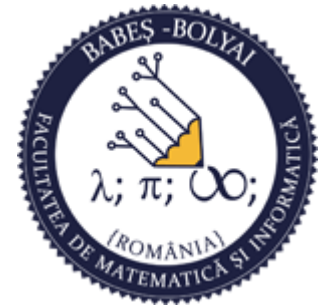




UNIVERSITATEA BABEŞ-BOLYAI
Facultatea de Matematică și Informatică



Programare orientată obiect

Curs 06

Laura Dioşan

POO

- *Standard Template Library (STL)*
 - Containeri & Iteratori
 - Algoritmi

Containeri - *Remember*

- Un container este un obiect care stochează o colecție de alte obiecte (elementele sale).
- Containerul:
 - gestionează spațiul pentru elemente
 - oferă funcții de acces la elemente, direct sau prin intermediul iteratorilor
- Unele containere au funcții comune și împart aceleași funcționalități
- Alegerea unui anumit tip de container depinde de:
 - funcționalitățile oferite de container
 - eficiența (complexitatea) acestor funcționalități

Containeri - *Remember*

- Containeri secvențiale
 - Vector
 - Listă
 - *Deque* (coadă cu 2 capete)

- Containerere adaptate
 - Stivă
 - Coadă
 - Coadă cu priorități

- Containeri asociative
 - Multime
 - Mulțime multiplă
 - Dicționar
 - Dicționar multiplu

Vectori

- Header *vector*
 - *#include <vector>*
 - *using namespace std;*

- Se folosesc vectori atunci cand elementele lor trebuie parcurse de mai multe ori

- Declarare
 - *vector<elemType> vectorName;*

- Metode
 - adăugare elem: *push_back, operator[], insert, assign, swap*
 - accesare elem: *operator[], at, front, back*
 - eliminare elem: *clear, pop_back, erase*
 - capacitate: *size, empty*
 - iteratori: *begin, end, rbegin, rend*

Iteratori

- Direcționali
 - Iteratori simpli
 - Iteratori bidirecționali
- Iteratori de I/O (*stream iterators*)
 - Iteratori pe *stream*-uri de intrare
 - Iteratori pe *stream*-uri de ieșire
 - Iteratori pe *stream*-uri de intrare cu *buffer*
 - Iteratori pe *stream*-uri de ieșire cu *buffer*
- Iteratori de inserție
 - Interatori de inserție
 - Iteratori de inserție la sfârșit (adăugare la sfârșit)
 - Iteratori de inserție la început (adăugare la început)

Vectori și iteratori - exemplu

□ 06/containers

- `vectorOfIntegers();`
- `vectorOfFlowers();`
- `vectorOfFlowerAddresses();`
- `vectorOfFlowersWithIterator();`
- `vectorWithBidirectionalIterators();`
- `vectorWithStreamIterators();`

- `vectorWithIteratorsAndPredicates();`
- `vectorWithIteratorsAndObjectFunctions();`

Deque

- ❑ Coadă cu două capete (*Double ended queue*)
- ❑ Header *deque*
 - *#include <deque>*
 - *using namespace std;*
- ❑ Declaraire
 - *deque<elemType> dequeName;*
- ❑ Metode
 - adăugare elemente: *push_front, push_back, operator[], insert, assign, swap*
 - accesare elemente: *operator[], at, front, back*
 - eliminare elemente: *clear, pop_front, pop_back, erase*
 - capacitate: *size, empty*
 - iteratori: *begin, end, rbegin, rend*

Deque - exemplu

- 06/containers
 - `dequeOfInteger();`
 - `dequeOfFlowers();`
 - `dequeOfFlowersWithIterator();`

Listă

- Header *list*
 - `#include <list>`
 - `using namespace std;`
- Listele se folosesc atunci când se inserează/elimină elemente de la o poziție dată
- Declarație
 - `list<elemType> listName;`
- Metode
 - adăugare elemente: `push_front`, `push_back`, `insert`, `assign`, `swap`
 - accesare elemente: `operator[]`, `at`, `front`, `back`
 - eliminare elemente: `clear`, `pop_front`, `pop_back`, `erase`
 - capacitate: `size`, `empty`
 - operații: `splice`, `remove`, `remove_if`, `unique`, `merge`, `sort`, `reverse`
 - iteratori: `begin`, `end`, `rbegin`, `rend`

Listă - exemplu

- 06/containers
 - listOfFlowersWithIterator();

Stivă

- Header *stack*
 - *#include <stack>*
 - *using namespace std;*

- Declarare
 - *stack<elemType> stackName;*

- Metode
 - adăugare elemente: *push*
 - accesare elemente: *top*
 - eliminare elemente: *pop*
 - capacitate: *size, empty*
 - NU EXISTĂ ITERATORI

Stivă - exemplu

- 06/containers
 - `stackOfFlowers();`

Coadă

□ Header *queue*

- *#include <queue>*
- *using namespace std;*

□ Declarare

- *queue<elemType> queueName;*

□ Metode

- adăugare elemente: *push*
- accesare elemente: *front, back*
- eliminare elemente: *pop*
- capacitate: *size, empty*
- NU EXISTĂ ITERATORI

Coadă - exemplu

- 06/containers
 - `queueOfFlowers();`

Coadă cu priorități

□ Header *queue*

- *#include <queue>*
- *using namespace std;*

□ Declarație

- *priority_queue<elemType> queueName;*

□ Metode

- adăugare elemente: *push*
- accesare elemente: *top*
- eliminare elemente: *pop*
- capacitate: *size, empty*
- NU EXISTĂ ITERATORI

Coadă cu priorități – exemplu

- 06/containers
 - `priorityQueueOfFlowers();`

Mulțime

- Header *set*
 - *#include <set>*
 - *using namespace std;*

- Stochează elemente unice

- Declaraire
 - *set<elemType> setName;*

- Metode
 - adăugare elemente: *insert, swap*
 - eliminare elemente: *clear, erase*
 - capacitate: *size, empty*
 - iteratori: *begin, end, rbegin, rend*
 - observatori: *key_comp, value_comp*
 - operații: *find, count, ...*

Mulțime – exemplu

- 06/containers
 - setOfFlowers()

Colecție

- ❑ Mulțime cu chei multiple
- ❑ Header *set*
 - *#include <set>*
 - *using namespace std;*
- ❑ Permite chei multiple cu valori egale
- ❑ Declarație
 - *multiset<elemType> multisetName;*
- ❑ Metode
 - adăugare elemente: *insert, swap*
 - eliminare elemente: *clear, erase*
 - capacitate: *size, empty*
 - iteratori: *begin, end, rbegin, rend*
 - observatori: *key_comp, value_comp*
 - operații: *find, count, ...*

Colecție – exemplu

- 06/containers
 - multiSetOfFlowers()

Dicționar

- Header *map*
 - *#include <map>*
 - *using namespace std;*

- Container asociativ care stochează elemente formate din combinarea unei chei și a unei valori

- Declarație
 - *map<keyType, elemType> mapName;*

- Metode
 - adăugare elemente: *insert, swap*
 - eliminare elemente: *clear, erase*
 - capacitate: *size, empty*
 - iteratori: *begin, end, rbegin, rend*
 - observatori: *key_comp, value_comp*
 - operații: *find, count, ...*

Dicționar - exemplu

- 06/containers
 - mapOfFlowers()

Dicționar multiplu

- ❑ Dicționar cu chei multiple
- ❑ Header set
 - *#include <map>*
 - *using namespace std;*
- ❑ Permite chei multiple cu valori egale
- ❑ Declarare
 - *multimap<elemType> mapName;*
- ❑ Metode
 - adăugare elemente: *insert, swap*
 - eliminare elemente: *clear, erase*
 - capacitate: *size, empty*
 - iteratori: *begin, end, rbegin, rend*
 - observatori: *key_comp, value_comp*
 - operații: *find, count, ...*

Dicționar multiplu - exemplu

- 06/containers
 - multipleMapOfFlowers()

Algoritmi

□ Sortare

- crescător -> funcția *sort(...)*
- descrescător -> funcția *reverse(...)*

□ Interschimbare

- Funcția *swap(...)*

□ Copiere

- Funcția *copy(...)*
 - Dintr-un
 - container
 - *stream*
 - Într-un
 - alt container
 - alt *stream*
- Funcția *remove_copy_if(...)*

Predicate

- Un pointer spre:
 - O funcție booleană
 - Un obiect al cărui clasă supraîncarcă *operator()*

Algoritmi și predicate – exemplu

□ 06/containers

- `sortVectorOfIntegers();`
- `sortVectorOfFlowers();`
- `swapVector();`
- `copyVector();`
- `copyVectorCond();`
- `vectorWithStreamIterators();`
- `vectorWithIteratorsAndPredicates();`
- `vectorWithIteratorsAndObjectFunctions();`

Cursul următor

- Relația de moștenire