
# Алгоритми

Лекція 2

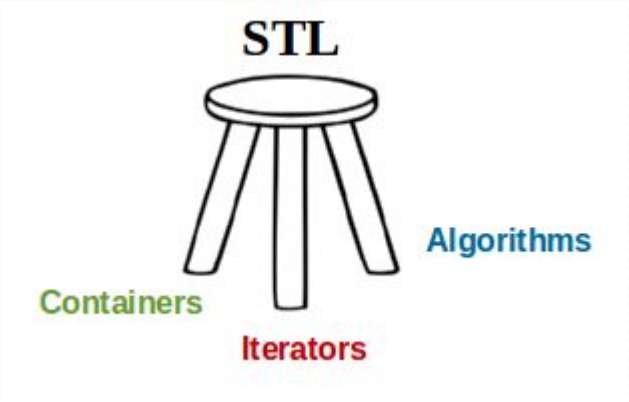
Контейнери в STL

# План лекції

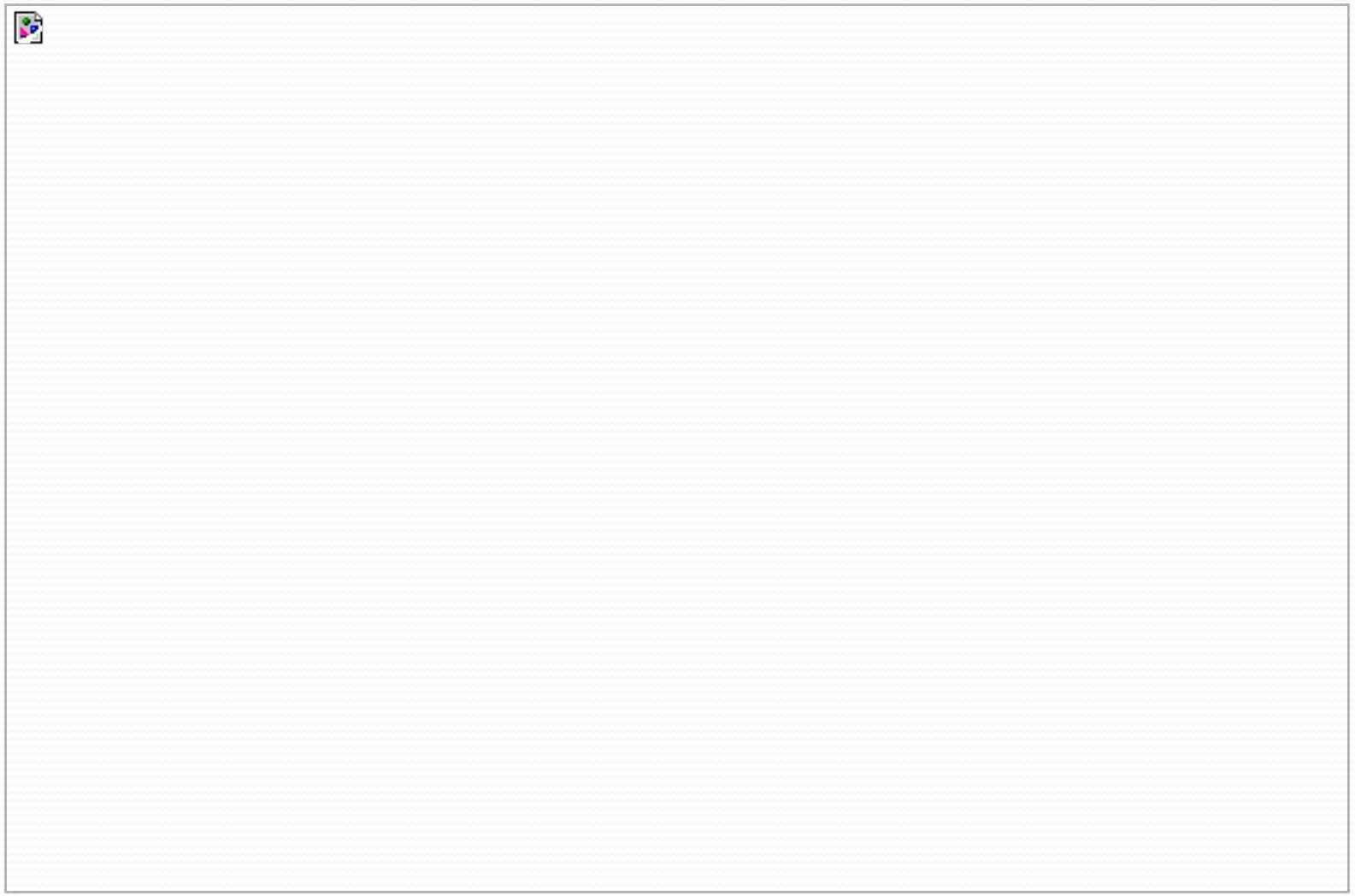
- Елементи STL
- Стек
- Черга

# Елементи STL

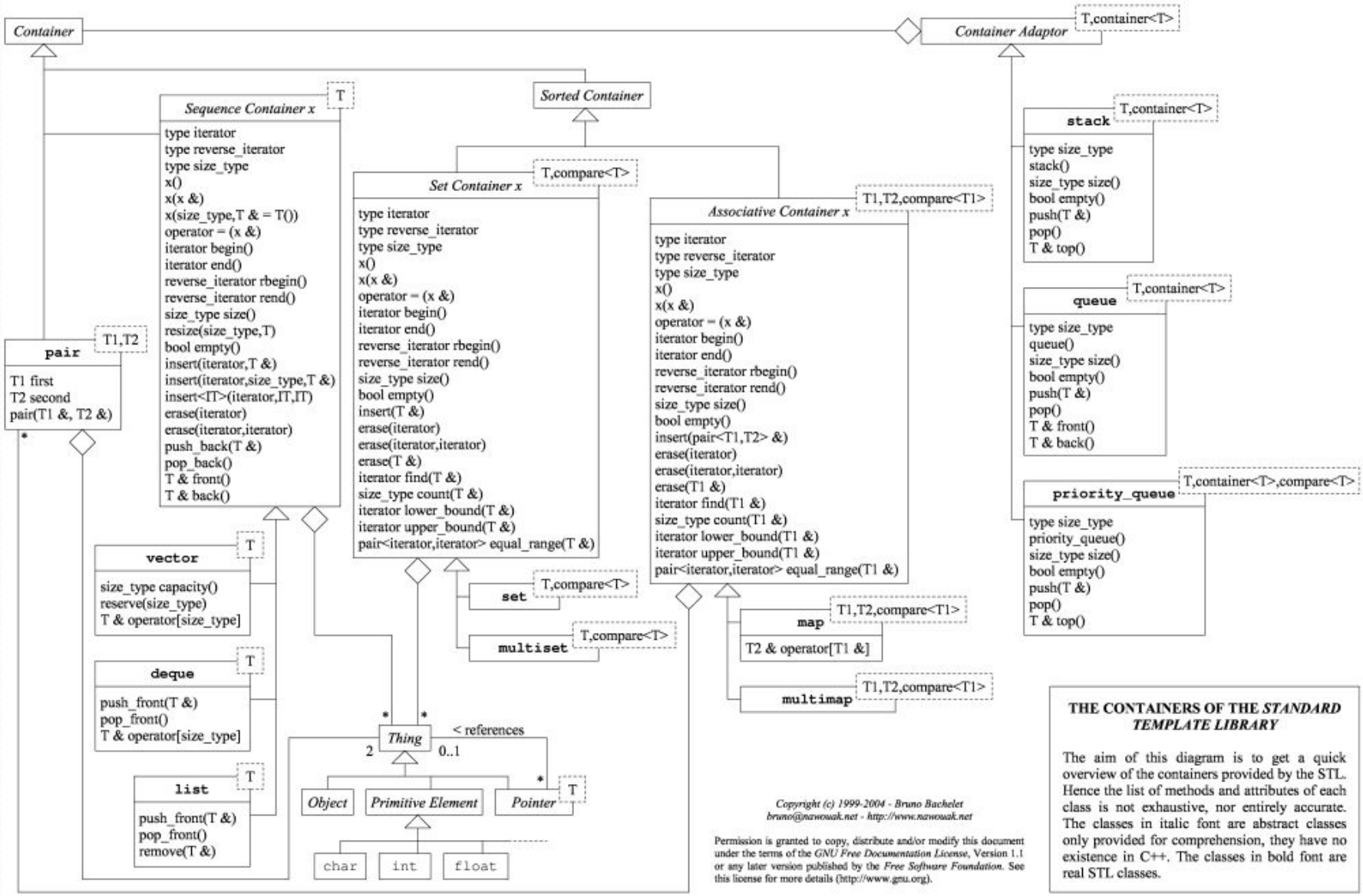
**Стандартна бібліотека шаблонів** (*Standard Template Library*; **STL**) —бібліотека для C++, що містить набір алгоритмів, контейнерів, засобів доступу до їхнього вмісту і різних допоміжних функцій.



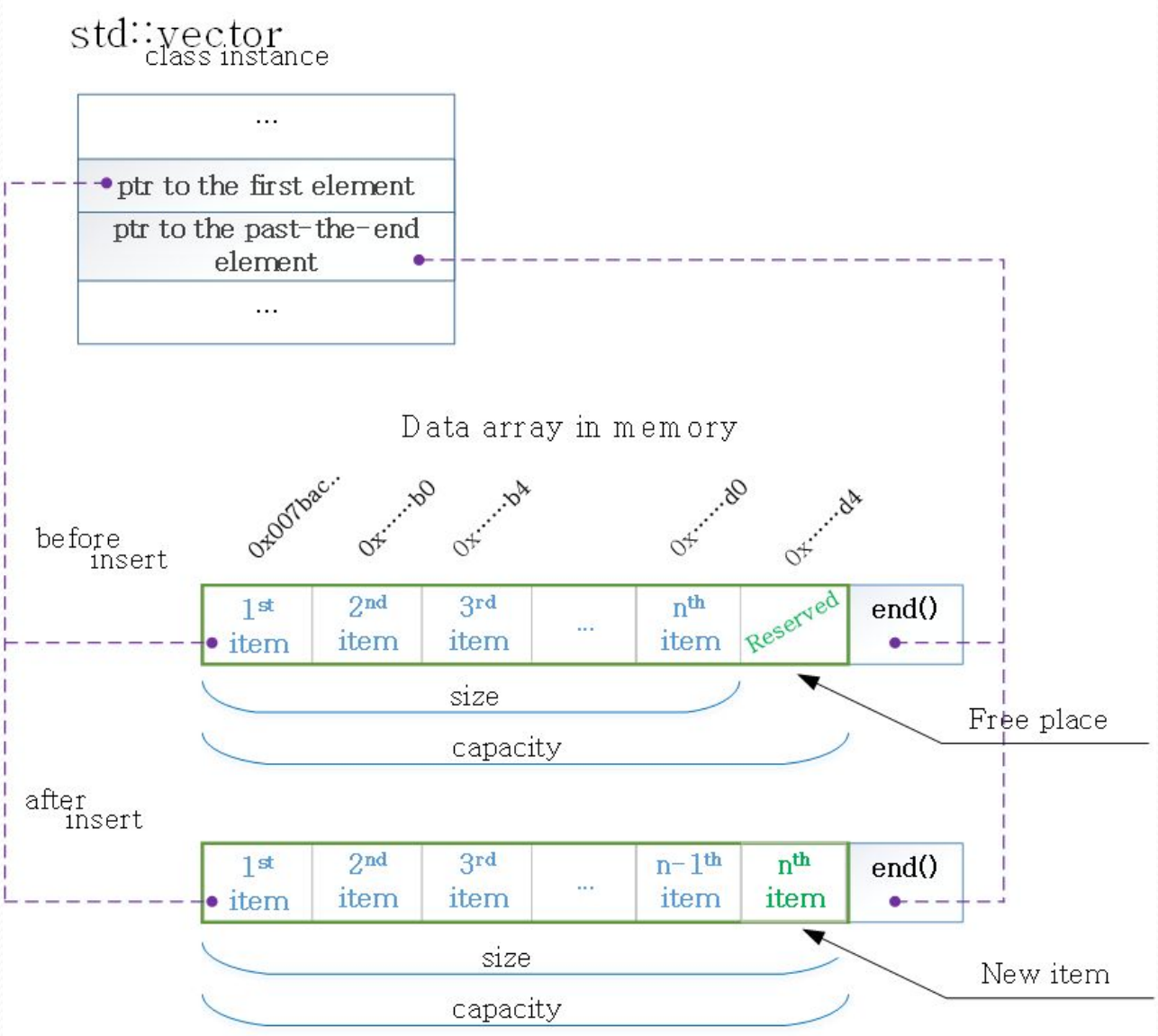
# Архітектура STL для чайників



# Архітектура STL для продвинутих



# Vector


std::vector
class instance
...
ptr to the first element
ptr to the past-the-end element
...
Data array in memory
0x007bac..
0x.....b0
0x.....b4
0x.....d0
0x.....d4
before insert
1st item
2nd item
3rd item
...
nth item
Reserved
end()
size
capacity
Free place
after insert
1st item
2nd item
3rd item
...
n-1th item
nth item
end()
size
capacity
New item

# Vector

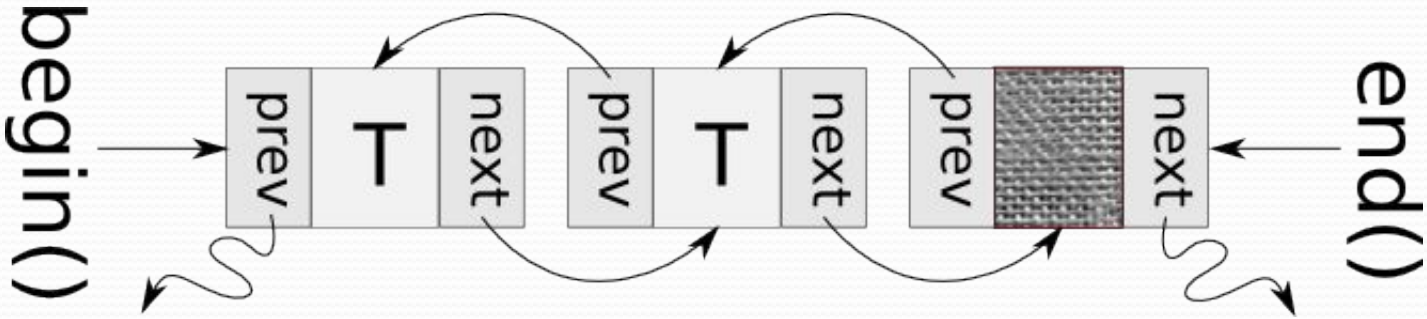
Вектор – послідовний контейнер який використовується для зберігання елементів, кількість яких невідома.

```
vector<int> myVector; // Порожній вектор із елементів типу int
vector<float> myVector(10); // Вектор із 10-и елементів типу float
vector<char> myVector(5, ' '); // Вектор містить 5 пробілів
int n = 10;
vector<T> myVector(n); // Вектор із 10-и елементів типу T
myVector.push_back (3);  // Запис числа 3 в кінець вектора
```

http://www.cplusplus.com/reference/vector/vector/

# List

Двозв’язний список призначений для послідовного зв’язку елементів. Використовується у випадку коли нема потреби у великій кількості проходів по всьому набору елементів.



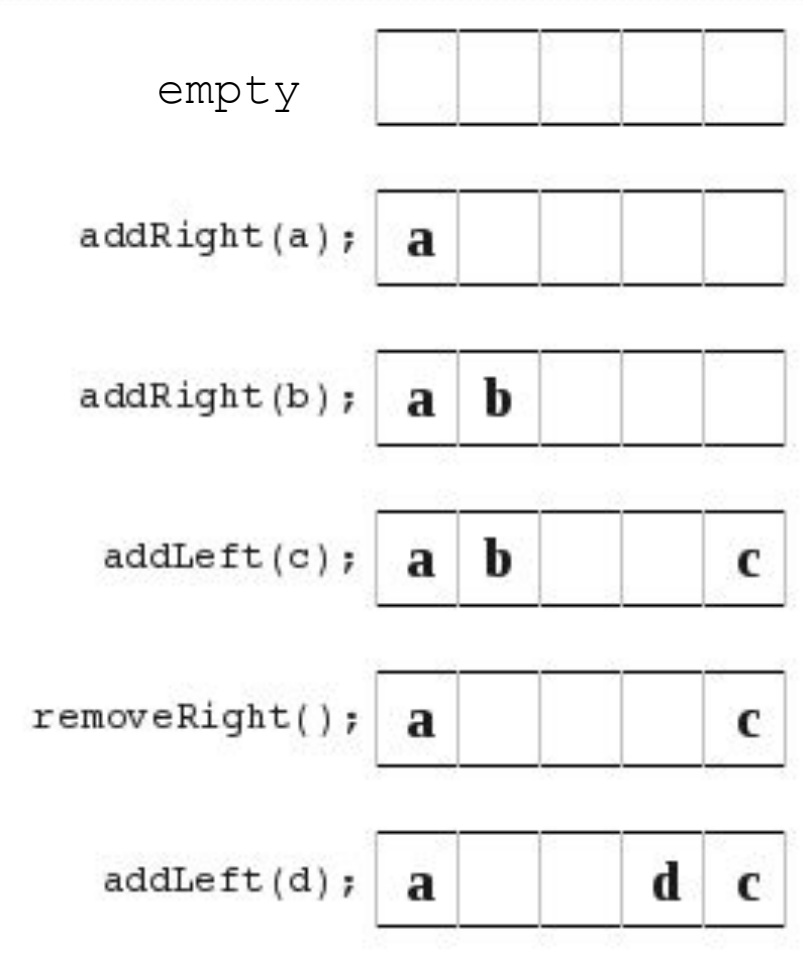
# List

```
std::list<int> mylist;

for (int i=1;i<=10;++i)
    mylist.push_back(i);

while (!mylist.empty())
{
     std::cout << mylist.front() << ' ';
     mylist.pop_front();
}
```

http://www.cplusplus.com/reference/list/list/

# Deque

Дек – це двостороння черга динамічного розміру. Таким чином елементи можуть додаватись та видалятись як в кінці так і в початку деку.



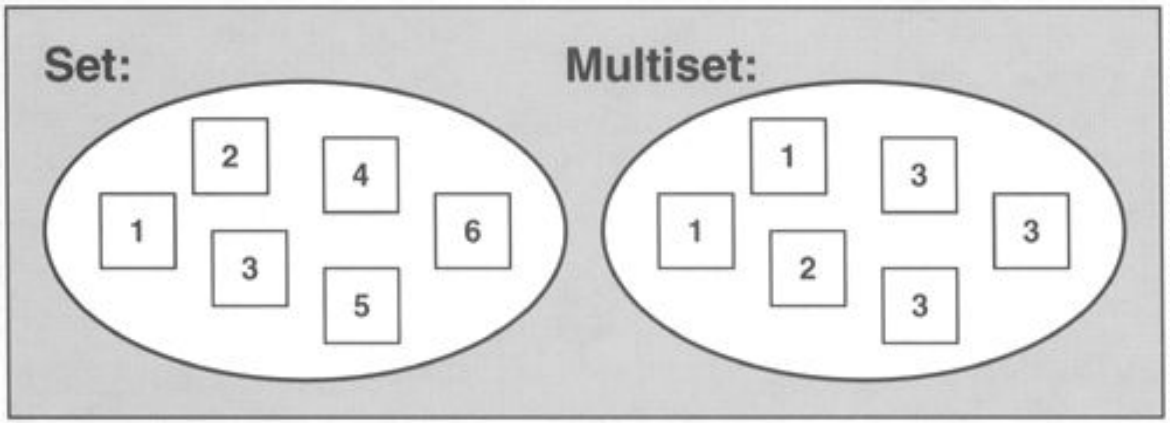
# Deque

```
std::deque<int> mydeque;

for (int i=1; i<=5; i++)
    mydeque.insert(mydeque.end(),i);

std::deque<int>::iterator it = mydeque.begin();
while (it != mydeque.end() )
       std::cout << ' ' << *it++;
```

http://www.cplusplus.com/reference/deque/deque/

# Set/Multiset

Set використовують для того, щоб зберігати тільки унікальні елементи. Відповідно multiset передбачає наявність повторень. Головним достоїнством цих контейнерів є те що вони містять уже відсортований набір даних.



# Set/Multiset

```
std::set<int> myset;
std::set<int>::iterator it;

for (int i=1; i<=5; i++) myset.insert(i*10);    // set: 10 20 30 40 50

it=myset.find(20);
myset.erase (it);

for (it=myset.begin(); it!=myset.end(); ++it)
    std::cout << ' ' << *it;
```

**OUTPUT**

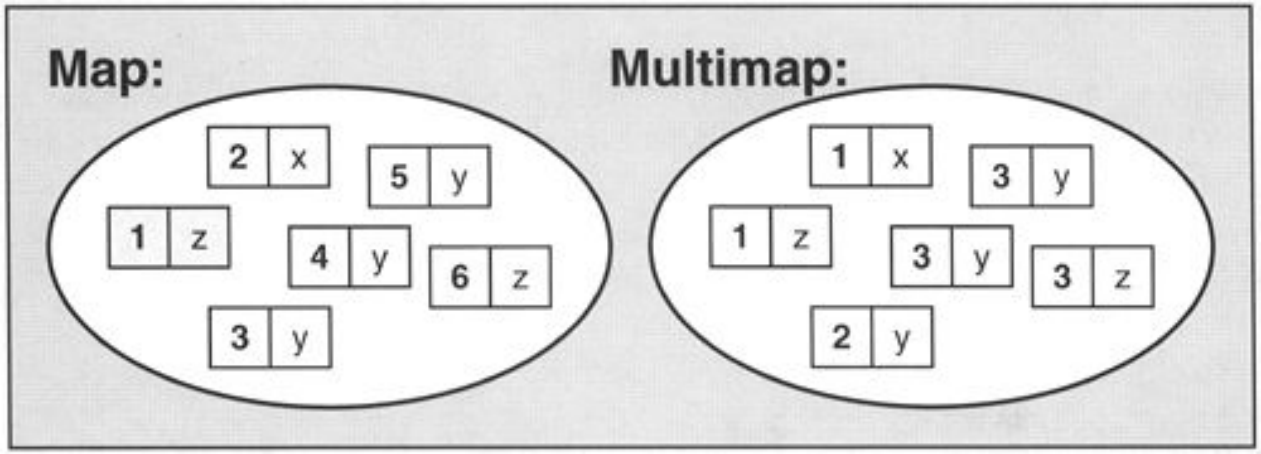
**myset contains: 10 30 40 50**

http://www.cplusplus.com/reference/set/set/

http://www.cplusplus.com/reference/set/multiset/

# Map/Multimap

Map зберігає пару <ключ, значення>, є зручним для зберігання таких пар даних у яких один з елементів є число, а інший – довільний.
Варто зазначити що Map/Multimap як і Set/Multiset зберігають уже відсортований набір даних.




http://www.cplusplus.com/reference/map/map/
http://www.cplusplus.com/reference/map/multimap/

# Map/Multimap

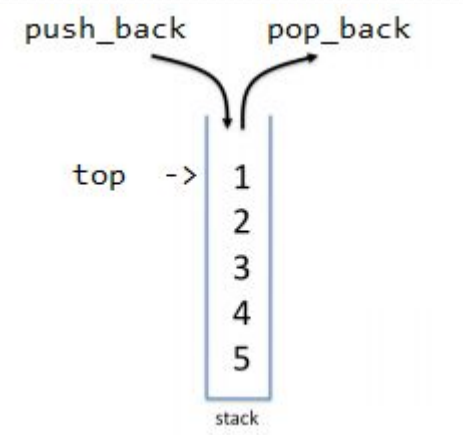
```
std::map<char,int> mymap;
// first insert function version (single parameter):
mymap.insert ( std::pair<char,int>('a',100) );
mymap.insert ( std::pair<char,int>('b',300) );
mymap.insert ( std::pair<char,int>('z',200) );

std::pair<std::map<char,int>::iterator,bool> ret;
ret = mymap.insert ( std::pair<char,int>('z',500) );
if (ret.second==false) {
  std::cout << "element 'z' already existed";
  std::cout << " with a value of " << ret.first->second << '\n';
}
// second insert function version (range insertion):
std::map<char,int> anothermap;
anothermap.insert(mymap.begin(),mymap.find('c'));
// showing contents:
std::cout << "mymap contains:\n";
for (it=mymap.begin(); it!=mymap.end(); ++it)
  std::cout << it->first << " => " << it->second << '\n';
std::cout << "anothermap contains:\n";
for (it=anothermap.begin(); it!=anothermap.end(); ++it)
  std::cout << it->first << " => " << it->second << '\n';
```

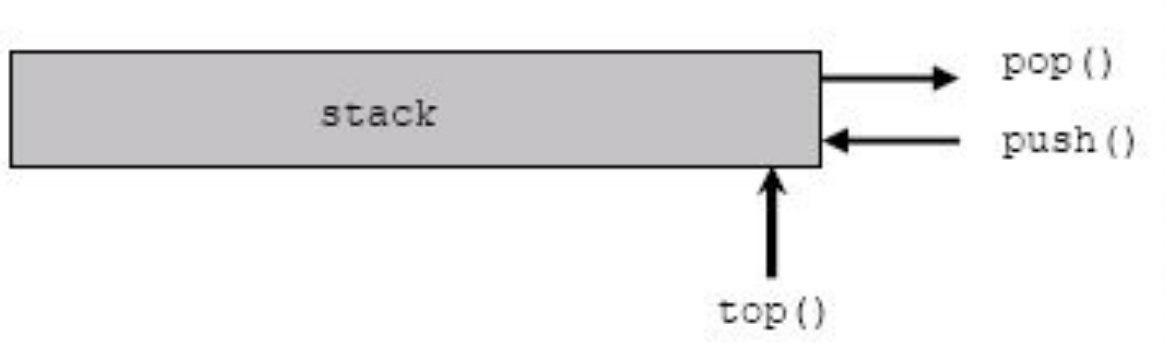
**OUTPUT:**

```
element 'z' already existed with a value of 200
mymap contains:
a => 100
b => 300
c => 400
z => 200
anothermap contains:
a => 100
b => 300
```

# Stack

Контейнер, що організований по принципу LIFO – last in first out.

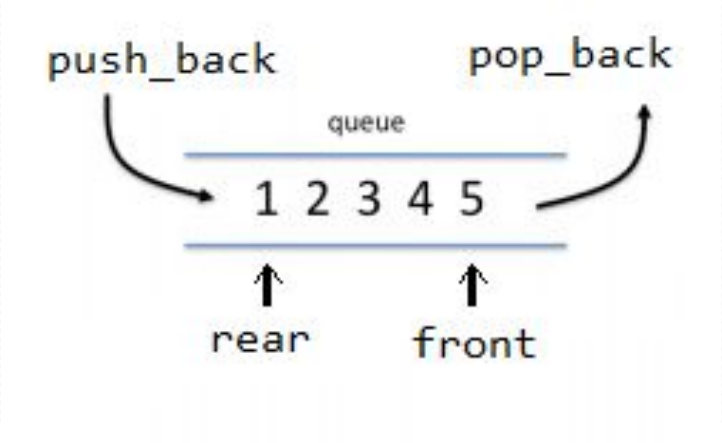


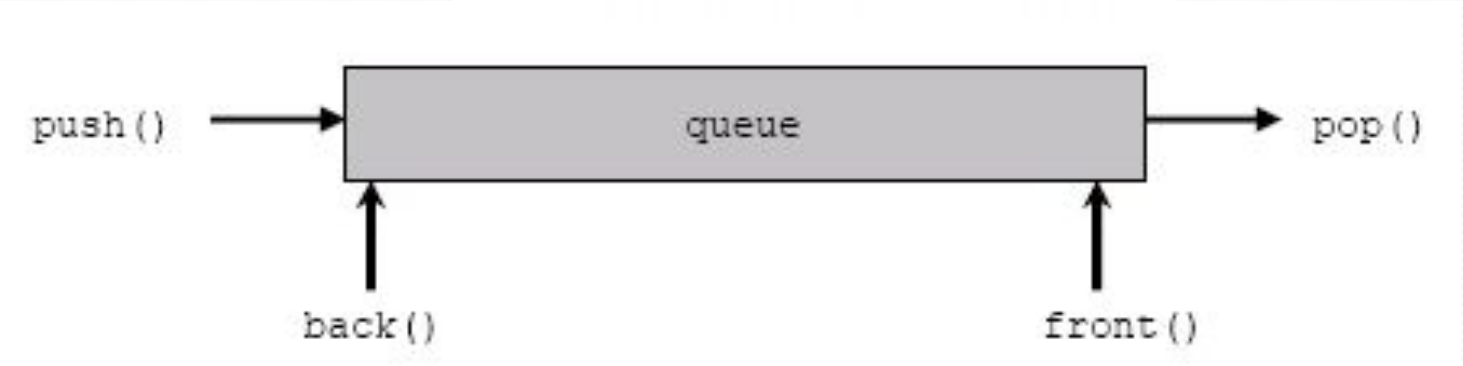




http://www.cplusplus.com/reference/stack/stack/

# Queue

Контейнер, що організований по принципу FIFO – first in first out.



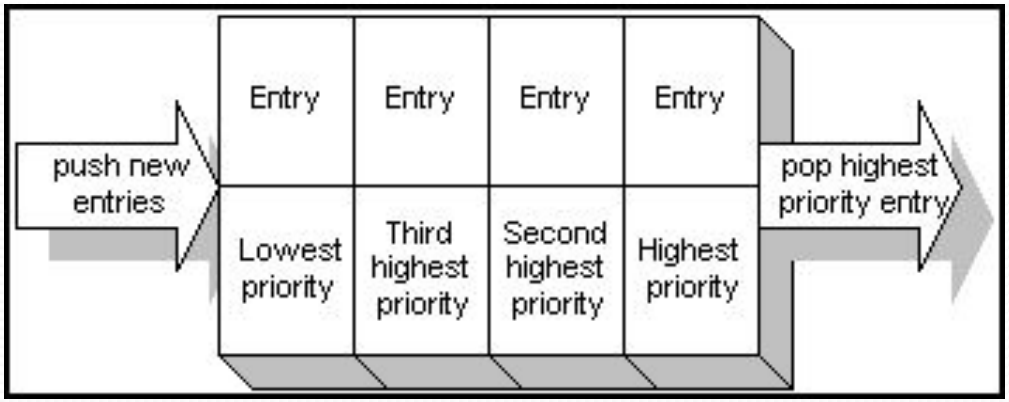




http://www.cplusplus.com/reference/queue/queue/

# Priority queue

Черга з пріоритетом має таку ж поведінку як і звичайна черга за виключенням операції видалення.

Вона відбувається не для того елемента який першим потрапив в чергу а для того, який має найбільший пріоритет (за певним критерієм) серед усіх елементів черги.



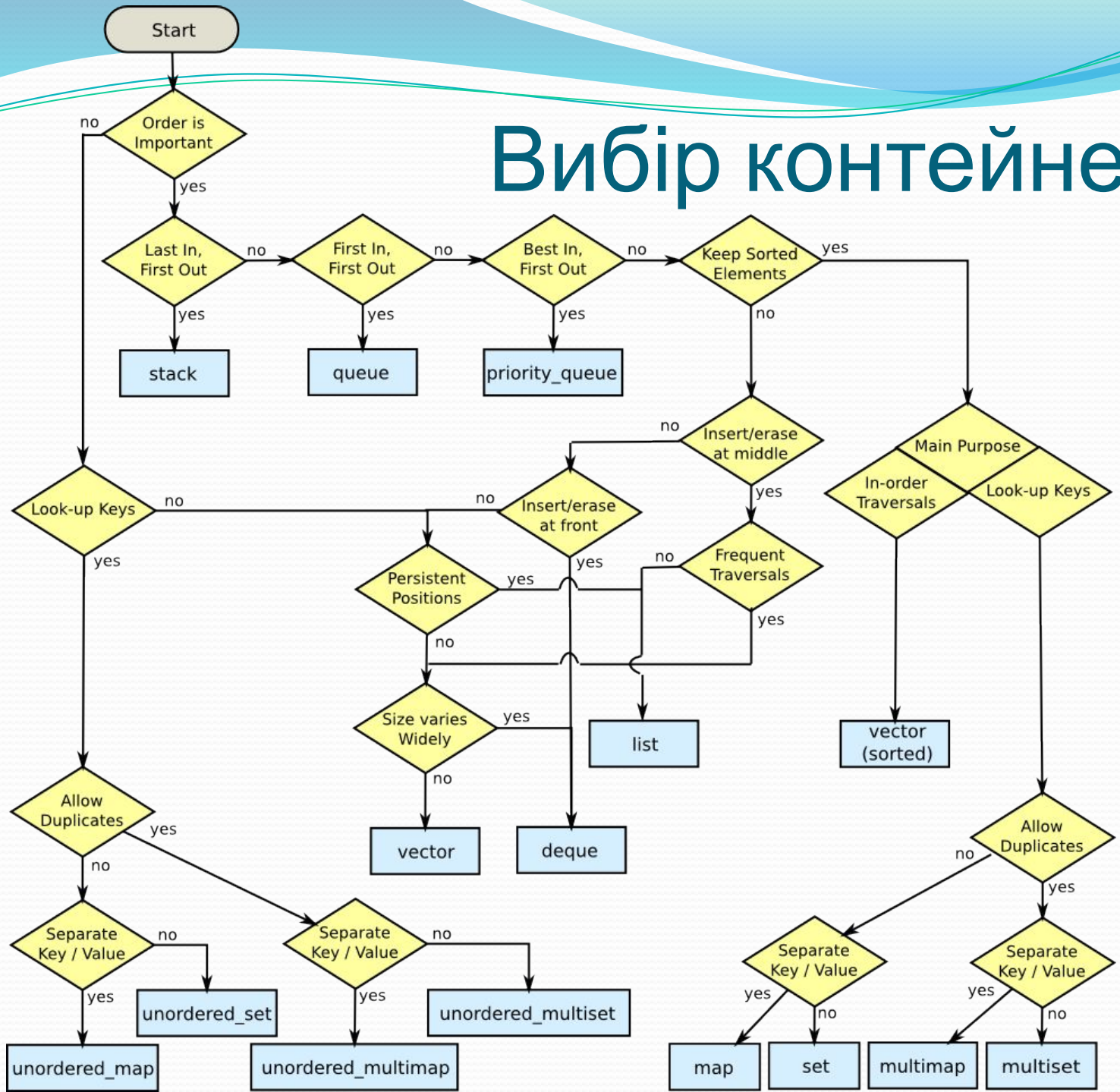
# Priority queue

```
std::priority_queue<int> mypq;
 mypq.push(30);
 mypq.push(100);
 mypq.push(25);
 mypq.push(40);
 std::cout << "Popping out elements...";
 while (!mypq.empty())
 {
    std::cout << ' ' << mypq.top();
    mypq.pop();
 }
```

**OUTPUT:**
**Popping out elements... 100 40 30 25**

http://www.cplusplus.com/reference/queue/priority_queue/

# Вибір контейнера



# Порівняльні характеристики контейнерів

| Container | Stores | [ ] | Iterators | Insert | Erase | Find | Sort |
|---|---|---|---|---|---|---|---|
| list | T | n/a | Bidirect'l | C | C | N | N log N |
| deque | T | C | Random | C at begin or end; else N/2 | C at begin or end; else N | N | N log N |
| vector | T | C | Random | C at end; else N | C at end; else N | N | N log N |
| set | T, Key | n/a | Bidirect'l | log N | log N | log N | C |
| multiset | T, Key | n/a | Bidirect'l | log N | d log (N+d) | log N | C |
| map | Pair, Key | log N | Bidirect'l | log N | log N | log N | C |
| multimap | Pair, Key | n/a | Bidirect'l | log N | d log (N+d) | log N | C |
| stack | T | n/a | n/a | C | C | n/a | n/a |
| queue | T | n/a | n/a | C | C | n/a | n/a |
| priority_ queue | T | n/a | n/a | log N | log N | n/a | n/a |

# Реалізація Stack

Необхідно реалізувати стек який би містив основні операції для роботи:

- stack();
- push();
- pop();
- top();
- empty();

# Реалізація Queue

Необхідно реалізувати чергу яка б містила основні операції для роботи:

- queue();
- push();
- pop();
- front();
- empty();

# Реалізація List

Домашнє завдання має містити наступні методи для роботи із списком:

1) **constructor** - Construct list

2) **empty** - Test whether container is empty

3) **insert** - Insert elements into given position

4) **erase** - Erase elements from given position

5 )Додатково можна реалізувати будь-який метод із std::list http://www.cplusplus.com/reference/list/list/

# Домашнє завдання:

1) Реалізувати однозв’язний/двозв’язний список (попередній слайд)
2) Розв’язати задачу «Атестація»