|  |  |
| --- | --- |
|  | **Тема 13**  **STL бібліотека** |

Механізм шаблонів вбудований в компілятор C ++, щоб дати можливість програмістам робити свій код коротше за рахунок узагальненого програмування. Природно, існують і стандартні бібліотеки, реалізують цей механізм. STL є найефективнішою бібліотекою C ++ на сьогоднішній день.

**Колекції**

Для використання колекції в своєму коді використовуйте наступну директиву:

#include <T>,

де T - назва колекції

Найбільш часто використовуються:

* **vector** - колекція елементів, збережених в масиві, що змінюється в міру необхідності розміру;
* **list** - колекція, що зберігає елементи в вигляді двонаправленого пов'язаного списку;
* **map -** колекція, яка зберігає пари виду <const Key, T>, тобто кожен елемент - це пара виду <ключ, значення>, при цьому кожному ключу відповідає єдине значення, де ключ - деяка характеризує значення величина, для якої може бути застосована операція порівняння; пари зберігаються в відсортованому вигляді, що дозволяє здійснювати швидкий пошук по ключу, але за це, природно, доведеться заплатити: доведеться так реалізовувати вставку, щоб умова відсортованості не порушувалась;
* **set** - це відсортована колекція одних тільки ключів, тобто значень, для яких може бути застосована операція порівняння, при цьому унікальних - кожен ключ може зустрітися тільки один раз;
* **multimap** - map, в якому відсутня умова унікальності ключа, тобто якщо ви справите пошук по ключу, то отримаєте не єдине значення, а набір елементів з однаковим значенням ключа; для використання в коді використовується #include <map>;
* **multiset** - колекція з тим же відмінністю від set'а, що і multimap від map'а, тобто з відсутністю умови унікальності ключа; для підключення: #include <set>.

**Строкові потоки**

strstream - використовуються для організації STL-строкового збереження простих типів даних.

Строковий потік - це буфер з нуль-термінатором в кінці, тому при першій роздруківці в кінці рядка виявляється сміття, тобто отримати реальний кінець можна не за допомогою нуль-термінатора, а отримавши лічильник: pcount (). Потім «реальна частина» потоку копіюється в новий рядок, і ми отримуємо роздруківку вже без сміття.

#include <iostream>

#include <strstream>

#include <string>

using namespace std;

int \_tmain (int argc, \_TCHAR\* argv [])

{ strstream xstr;

for (int i = 0; i < 10; i++)

{ xstr << "Demo " << i << endl; }

cout << xstr.str ();

string str;

str.assign (xstr.str (), xstr.pcount ());

cout << str.c\_str ();

return 0; }

**ітератори**

Ітератор можна визначити як абстракцію, яка поводиться як покажчик, можливо, з якимись обмеженнями. Строго кажучи, итератор - більш загальне поняття, і є об'єктної обгорткою для покажчика, тому покажчик є ітератором.

Ітератори забезпечують доступ до елементів колекції

Для кожного конкретного класу STL ітератори визначаються окремо всередині класу цієї колекції.

Існують три типи ітераторів:

* **Forward iterator** - для обходу колекції від меншого індексу до більшого;
* **reverse iterator** - для обходу колекції від більшого індекс до меншого;

random access iterator - для обходу колекції в будь-якому напрямку.

**Методи колекцій**

Основними методами, які були присутні майже в усіх колекціях є такі:

empty - визначає, порожня чи колекція;

size - повертає розмір колекції;

begin - повертає прямий ітератор, який вказує на початок колекції;

end - повертає прямий ітератор, який вказує на кінець колекції, тобто на неіснуючий елемент, що йде після останнього;

rbegin - розраховує зворотній итератор на початок колекції;

rend - розраховує зворотній итератор на кінець колекції;

clear - очищає колекцію, тобто видаляє всі її елементи;

erase - видаляє певні елементи з колекції;

capacity - повертає місткість колекції, тобто кількість елементів, яке може вмістити ця колекція

**Алгоритми**

STL містить величезний набір оптимальних реалізацій популярних алгоритмів, що дозволяють працювати з STL-колекціями. Всі реалізовані функції можна поділити на три групи:

**Методи перебору** всіх елементів колекції і їх обробки: count, count\_if, find, find\_if, adjacent\_find, for\_each, mismatch, equal, search copy, copy\_backward, swap, iter\_swap, swap\_ranges, fill, fill\_n, generate, generate\_n, replace, replace\_if, transform, remove, remove\_if, remove\_copy, remove\_copy\_if, unique, unique\_copy, reverse, reverse\_copy, rotate, rotate\_copy, random\_shuffle, partition, stable\_partition

**Методи сортування колекції**: sort, stable\_sort, partial\_sort, partial\_sort\_copy, nth\_element, binary\_search, lower\_bound, upper\_bound, equal\_range, merge, inplace\_merge, includes, set\_union, set\_intersection, set\_difference, set\_symmetric\_difference, make\_heap, push\_heap, pop\_heap, sort\_heap, min, max, min\_element, max\_element, lexographical\_compare, next\_permutation, prev\_permutation

**Методи виконання арифметичних операцій над членами колекцій:** Accumulate, inner\_product, partial\_sum, adjacent\_difference