Лабораторна робота №8. Стандартна бібліотека шаблонів STL, Алгоритми

МЕТА РОБОТИ: вміти використовувати алгоритми STL для роботи з шаблонними об’єктами цієї ж бібліотеки.

8.1. Програма роботи

8.1.1. Отримати завдання.

8.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

8.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

8.1.4. Оформити електронний звіт про роботу та захистити її.

8.2. Вказівки до виконання роботи

8.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 8.4 і записує його до звіту.

8.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 8.4.

8.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

8.2.4. Звіт має містити такі розділи:

мету роботи та завдання з записаною умовою задачі;

коди всіх використовуваних .h і .ccp файлів, а також пояснення до них;

результати реалізації програми;

* висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

8.3. Теоретичні відомості

Кожний алгоритм виражається шаблоном функції або набором шаблонів функцій. Таким чином, алгоритм може працювати з дуже різними контейнерами, що містять значення різноманітних типів. Алгоритми, які повертають ітератор, як правило, для повідомлення про невдачу використають кінець вхідної послідовності. Алгоритми не виконують перевірки діапазону на їхньому вході й виході. Коли алгоритм повертає ітератор, це буде ітератор того ж типу, що й був на вході. Алгоритми в STL реалізують більшість поширених універсальних операцій з контейнерами, такі як перегляд, сортування, пошук, вставка й видалення елементів.

Алгоритми визначені в заголовному файлі <algorithm.h>.

Нижче наведені імена деяких найбільш часто використовуваних функцій-алгоритмів STL.

**I. Операції, що не модифікують:**

1. **for\_earch() –** виконує операції для кожного елемента послідовності;
2. **find() –** знаходить перше входження значення в послідовності;
3. **find\_if()** – знаходить першу відповідність предикату в послідовності;
4. **count()** – підраховує кількість входжень значення в послідовність;
5. **count\_if()** – підраховує кількість виконань предиката в послідовності;
6. **search()** – знаходить перше входження послідовності як підпослідовності;
7. **search\_n()** – знаходить n – і входження значення в послідовності.

**II. Операції, що модифікують:**

1. **copy()** – копіює послідовність, починаючи з першого елемента;
2. **swap()** – міняє місцями два елементи;
3. **replace()** – заміняє елементи із зазначеним значенням;
4. **replace\_if()** – заміняє елементи при виконанні предиката;
5. **replace\_copy()** – копіює послідовність, заміняючи елементи із зазначеним значенням;
6. **replace\_copy\_if()** – копіює послідовність, заміняючи елементи при виконанні предиката;
7. **fill()** – заміняє всі елементи даним значенням;
8. **remove()** – видаляє елементи з даним значенням;
9. **remove\_if()** – видаляє елементи при виконанні предиката;
10. **remove\_copy()** – копіює послідовність, видаляючи елементи із зазначеним значенням;
11. **remove\_copy\_if() –** копіює послідовність, видаляючи елементи при виконанні предиката;
12. **reverse()** – міняє порядок проходження елементів на зворотний;
13. **random\_shuffle()** – переміщає елементи відповідно до випадкового рівномірного розподілу (“тасує” послідовність);
14. **transform()** – виконує задану операцію над кожним елементом послідовності;
15. **unique()** – видаляє рівні сусідні елементи;
16. **unique\_copy()** – копіює послідовність, видаляючи рівні сусідні елементи.

**III. Сортування:**

1. **sort()** – сортує послідовність із гарною середньою ефективністю;
2. **partial\_sort() –** сортує частину послідовності;
3. **stable\_sort()** – сортує послідовність, зберігаючи порядок слідування рівних елементів;
4. **lower\_bound()** – знаходить перше входження значення у відсортованій послідовності;
5. **upper\_bound()** – знаходить перший елемент, який є більшим за задане значення;
6. **binary\_search()** – визначає, чи є даний елемент у відсортованій послідовності;
7. **merge()** – зливає дві відсортовані послідовності.

**IV. Робота з множинами:**

1. **includes()** – перевірка на входження;
2. **set\_union() –** об'єднання множин;
3. **set\_intersection()** – перетинання множин;
4. **set\_difference()** – різниця множин.

**V. Мінімуми й максимуми:**

1. **min()** – менше із двох;
2. **max()** – більше із двох;
3. **min\_element()** – найменше значення в послідовності;
4. **max\_element()** – найбільше значення в послідовності.

**VI. Перестановки:**

1. **next\_permutation()** – наступна перестановка в лексикографічному порядку;
2. **pred\_permutation()** – попередня перестановка в лексикографічному порядку.

**Приклад програми**

Для пошуку мінімального і максимального елемента вектора можна використати алгоритм min\_element і max\_element, для пошуку заданого елемента – метод find ( ), для видалення елемента – метод remove ( ), для сортування елементів – алгоритм sort ( ):

#include <iostream>

#include <vector>

#include <algorithm>

#include <functional>

#include <cstdlib>

using namespace std;

void main ()

{

setlocale(LC\_ALL, "Ukranian");

vector <int> v, t(10);

vector <int> :: iterator pos, imin, imax, new\_end;

// Формування вектора

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

v.push\_back (rand() %100);

cout<< "Sformovanuj vector"<<endl;

for (pos=v.begin (); pos!=v.end (); ++pos)

cout<<\*pos<< " "<<endl;

// Пошук і виведення мінімального та максимального елементів

imin=min\_element (v.begin (), v.end ());

cout<< "Min element="<<\*imin<<endl;

imax=max\_element (v.begin (), v.end ());

cout<< "Max element="<<\*imax<<endl;

// Визначення позиції мінімального та максимального елементів

imin=find (v.begin (), v.end (), \*imin);

cout<< "Pozuciya min elementa="<<imin - v.begin()<<endl;

imax=find (v.begin ( ), v.end ( ),\*imax);

cout<< "Pozuciya max elementa="<<imax - v.begin()<<endl;

// Видалення максимального елемента

new\_end=remove (v.begin (), v.end (), \*imax);

// Зміна розміру вектора після видалення

v.erase (new\_end, v.end ());

cout<< "Vector pislya vudalennya max elementa"<<endl;

for (pos=v.begin (); pos!=v.end (); pos++)

cout<< " "<<\*pos<<endl;

// Сортування вектора за спаданням

sort (v.begin (), v.end (), greater <int> ());

// Копіювання вектора

copy (v.begin (), v.end (), t.begin ());

// Розміщення елементів у зворотньому порядку

reverse (v.begin (), v.end ());

cout<< "Vuvedennya vidsortovanoho vectora y pryamomy poryadky"<<endl;

for (pos=t.begin (); pos!=t.end (); ++pos)

cout<<\*pos<< " "<<endl;

cout<< "Vuvedennya vidsortovanoho vectora y zvorotnomy poryadky"<<endl;

for (pos=v.begin (); pos!=v.end (); ++pos)

cout<<\*pos<< " "<<endl;

system ("pause");

}

8.4. Індивідуальні завдання

Розробити програму з використанням алгоритмів контейнеру algorithm бібліотеки STL згідно завдання.

1. В одновимірному масиві, що складається з n елементів, обчислити: суму від’ємних елементів масиву, добуток елементів масиву, розташованих між максимальним і мінімальними елементами; упорядкувати елементи масиву по зростанню.
2. В одновимірному масиві, що складається з n цілих елементів, обчислити: добуток елементів масиву з парними номерами, суму елементів масиву, розташованих між першим і останнім нульовими елементами.
3. В одновимірному масиві, що складається з n елементів, обчислити: суму додатніх елементів масиву, добуток елементів масиву, розташованих між максимальним по модулю мінімальним по модулю елементами, впорядкувати елементи масиву по спаданню.
4. В одновимірному масиві, що складається з n елементів, обчислити: суму елементів масиву з непарними номерами; суму елементів масиву, розташованих між першим і останнім від’ємними елементами, стиснути масив, видаливши з нього всі елементи, модуль яких не перевищує 1. Елементи, що звільнились в кінці масиву, заповнити нулями.
5. **В одновимірному масиві, що складається з n елементів, обчислити: максимальний елемент масиву; суму елементів масиву, розташованих до останнього додатнього елемента; стиснути масив, видаливши з нього всі елементи, модуль яких знаходиться в інтервалі [a,b]. Елементи, що звільнились в кінці масиву, заповнити нулями.**
6. В одновимірному масиві, що складається з n елементів, обчислити: номер мінімального по модулю елемента масиву, суму модулів елементів масиву, розташованих після першого від’ємного елемента, стиснути масив, видаливши з нього всі елементи, величина яких знаходиться в інтервалі [a,b]. Елементи, що звільнились в кінці масиву, заповнити нулями.
7. В одновимірному масиві, що складається з n цілих елементів, обчислити: номер максимального елемента масиву, добуток елементів масиву, розташованих між першим і другим нульовими елементами, перетворити масив таким чином, щоб у першій його половині розташовувались елементи, що стояли в непарних позиціях, а в другій половині — елементи, що стояли в парних позиціях.
8. В одновимірному масиві, що складається з n елементів, обчислити: номер мінімального елемента масиву, суму елементів масиву, розташованих між першим і другим від’ємними елементами; перетворити масив таким чином, щоб спочатку розташовувались всі елементи, модуль яких не перевищує 1, а потім — всі інші.
9. В одновимірному масиві, що складається з n елементів, обчислити: кількість елементів масиву, менших 3, суму цілих частин елементів масиву, розташованих після останнього від’ємного елемента, перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, що відрізняються від максимального не більше ніж на 20%, а потім — всі інші.
10. В одновимірному масиві, що складається з n цілих елементів, обчислити: мінімальний по модулю елемент масиву, суму модулів елементів масиву, розташованих після першого елемента, рівного нулю, перетворити масив таким чином, щоб у першій його половині розташовувалися елементи, що стояли в парних позиціях, а в другій половині — елементи, що стояли в непарних позиціях.
11. Дана прямокутна матриця. Визначити кількість від'ємних елементів в тих рядках, які містять хоча б один нульовий елемент.
12. Дана прямокутна матриця. Визначити номер рядка, в якому знаходиться найдовша серія з одинакових елементів.
13. У довільній матриці обчислити кількість елементів масиву, рівних нулю.
14. В одновимірному масиві обчислити суму модулів елементів, які розташовані після першого додатнього елемента.
15. У матриці обчислити суму елементів масиву, що розташовані між першим і другим додатними елементами.

8.5. Контрольні запитання

1. На які категорії поділяються алгоритми STL?
2. Яку бібліотеку слід підключити для роботи з алгоритмами STL?
3. Яка різниця між алгоритмами групи модифіковані операції з послідовностями і немодифіковані операції з послідовностями?
4. Які алгоритми сортування STL Ви знаєте? Наведіть приклад.
5. Яка різниця між алгоритмом **count\_if** і **count**?
6. Для чого призначений алгоритм **for\_each**?
7. Яка різниця між алгоритмами **iter\_swap**, **swap** і **swap\_ranges**?
8. Які алгоритми переміщення Ви знаєте? Наведіть приклад.