Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра **«**Информационные технологии и автоматизированные системы**»**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №13**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Стандартные обобщенные алгоритмы библиотеки STL

Вариант 6.

Выполнил работу

Студент группы РИС-22-1Б

Поважный В. Е.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

г. Пермь-2023

**Введение**

Для выполнения лабораторной работы требуется сделать отчет программы. Создание программ – отличный способ практики программирования.

**Постановка задачи**

Задача 1:

1. Создать последовательный контейнер;

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции;

3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill());

4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy());

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort());

6. Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if());

7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) ;

8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Условия варианта:

1. Контейнер - список;

2. Тип элементов Pair (см. лабораторную работу №3).

Задача 2:

1. Создать адаптер контейнера;

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции;

3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill());

4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy());

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort());

6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if());

7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) ;

8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Условие варианта:

Адаптер контейнера – очередь с приоритетами.

Задача 3:

1. Создать ассоциативный контейнер;

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции;

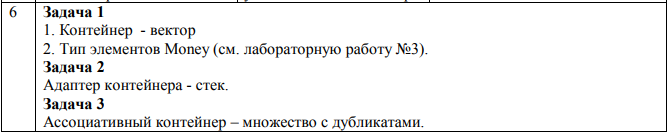
3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill());

4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy());

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort());

6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if());

7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each());

8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

**Код программы**

**Задача 1**

#include "Money.h"

#include "vector"

#include <iostream>

#include <conio.h>

#include <algorithm>

#include <list>

#include <iterator>

using namespace std;

typedef list<Money> TList;

TList make\_list(int n)

{

Money a;

TList v;

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

{

cin >> a;

v.push\_back(a);

}

return v;

}

void print\_list(TList v)

{

for (auto it = v.begin(); it != v.end(); ++it)

std::cout << \*it << " ";

std::cout << std::endl;

}

Money s;

Money p;

Money minmax(TList v)

{

TList::iterator x;

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

std::cout << "min =" << \*(x) << endl;

Money g = \*(x);

TList::iterator y;

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

std::cout << "max = " << \*(y) << endl;

Money h = \*(y);

Money p = h - g;

std::cout << "min - max = " << p << endl;

return p;

}

struct Greater\_s

{

bool operator()(Money t)

{

if (t > s) return true; else return false;

}

};

struct Comp\_less

{

public:

bool operator()(Money t1, Money t2)

{

if (t1 > t2)return true;

else return false;

}

};

struct Equal\_s

{

bool operator()(Money t)

{

return t == s;

}

};

void del(Money& t)

{

t = t + p;

}

void main()

{

int n;

cout << "Number: ";

cin >> n;

TList v;

v = make\_list(n);

print\_list(v);

TList::iterator i;

TList::iterator pos;

int num;

cout << "Add object " << endl;

cin >> s;

cout << "Pos for add: " << endl;

cin >> num;

i = find\_if(v.begin(), v.end(), Equal\_s());

pos = next(v.begin(), num);

if (i != v.end())

{

cout << "Object: " << \*(i) << endl << "Added for " << num << "pos" << endl;

v.insert(pos, s);

print\_list(v);

}

else

cout << "Not such element!" << endl;

cout << "Delete object " << endl;

cin >> s;

i = find\_if(v.begin(), v.end(), Equal\_s());

if (i != v.end())

{

cout << "Number:" << \*(i) << endl << "Deleting" << endl;

s = \*i;

i = remove\_if(v.begin(), v.end(), Equal\_s());

v.erase(i, v.end());

print\_list(v);

}

else

cout << "Not such element!" << endl;

cout << "Raznica min and max elementov" << endl;

p = minmax(v);

for\_each(v.begin(), v.end(), del);

print\_list(v);

}

**Задача 2**

**#include** "Money.h"

#include "deque"

#include "queue"

#include <iostream>

#include <conio.h>

#include <algorithm>

using namespace std;

typedef queue<Money> que;

typedef deque<Money> deq;

que make\_queue(int n)

{

que s;

Money t;

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

{

cin >> t;

s.push(t);

}

return s;

}

deq copy\_queue\_to\_deque(que s)

{

deq v;

while (!s.empty())

{

v.push\_back(s.front());

s.pop();

}

return v;

}

que copy\_deque\_to\_queue(deq v)

{

que s;

for (int i = 0; i < v.size(); i++)

s.push(v[i]);

return s;

}

void print\_queue(que s)

{

std::cout << endl;

deq v = copy\_queue\_to\_deque(s);

while (!s.empty())

{

std::cout << s.front() << endl;

s.pop();

}

s = copy\_deque\_to\_queue(v);

std::cout << endl;

}

Money s;

struct Greater\_s

{

bool operator()(Money t)

{

if (t > s)

return true;

else

return false;

}

};

struct Comp\_less

{

public:

bool operator()(Money t1, Money t2)

{

if (t1 > t2)

return true;

else

return false;

}

};

Money minmax(que s)

{

deq v = copy\_queue\_to\_deque(s);

deq::iterator x;

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

std::cout << "min = " << \*(x) << endl;

Money g = \*(x);

deq::iterator y;

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

std::cout << "max = " << \*(y) << endl;

Money h = \*(y);

Money p = h + g;

std::cout << "min + max = " << p << endl;

return p;

}

struct Equal\_s

{

bool operator()(Money t)

{

return t == s;

}

};

void del(Money& t)

{

t = t + s;

}

void main()

{

int n;

cout << "Number: ";

cin >> n;

que v;

v = make\_queue(n);

print\_queue(v);

deq vv;

deq::iterator i;

vv = copy\_queue\_to\_deque(v);

i = min\_element(vv.begin(), vv.end());

cout << "min = " << \*(i) << endl;

Money m = \*(i);

vv.push\_back(m);

v = copy\_deque\_to\_queue(vv);

print\_queue(v);

cin >> s;

i = find\_if(vv.begin(), vv.end(), Equal\_s());

if (i != vv.end())

{

cout << "Number:" << \*(i) << endl << "Deleting" << endl;

s = \*i;

i = remove\_if(vv.begin(), vv.end(), Equal\_s());

vv.erase(i, vv.end());

v = copy\_deque\_to\_queue(vv);

print\_queue(v);

}

else

cout << "Not such element!" << endl;

cout << "Adding" << endl;

s = minmax(v);

for\_each(vv.begin(), vv.end(), del);

v = copy\_deque\_to\_queue(vv);

print\_queue(v);

}

**Задача 3**

**#include** <iostream>

#include "Money.h"

#include <algorithm>

#include <set>

using namespace std;

typedef multiset<int> mst;

typedef multiset<int> ::iterator it;

mst make\_map(int n)

{

mst m;

int a;

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

{

cout << "?";

cin >> a;

m.insert(a);

}

return m;

}

void print\_map(mst v)

{

multiset <int> ::iterator it = v.begin();

cout << "Sorted: " << endl;

for (int i = 1; it != v.end(); i++, it++)

cout << \*it << " ";

}

int srednee(mst v)

{

int s = 0;

multiset <int>::iterator it = v.begin();

for (int i = 1; it != v.end(); i++, it++)

s += \*it;

return s / v.size();

}

Money s;

int main()

{

int n;

Money t;

cout << "Number: ";

cin >> n;

mst v = make\_map(n);

print\_map(v);

mst::iterator i;

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

s = srednee(v);

cout << "Middle = " << s << endl;

return 0;

}

**Вывод**

Программа выполняет свою задачу.

**Заключение**

Для решения задачи потребовались знания языка программирования, а конкретнее C++. Программа выполняет те условия, что были указаны в постановке задачи и работает без проблем.