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 № 18.11**

Дисциплина: Информатика

# Тема: «Объектно-ориентированное программирование. Последовательные контейнеры библиотеки STL»

Вариант 1

Выполнил работу:

студент группы РИС-20-1Б

Азмагулов Артём Вадимович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь

2021

**Цель работы**

* 1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
  2. Использование последовательных контейнеров библиотеки STL в ОО программе.

**Постановка задачи**

Задача 1.

* + 1. Создать последовательный контейнер.
    2. Заполнить его элементами стандартного типа (тип указан в варианте).
    3. Добавить элементы в соответствии с заданием
    4. Удалить элементы в соответствии с заданием.
    5. Выполнить задание варианта для полученного контейнера.
    6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 2.

1. Создать последовательный контейнер.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 3

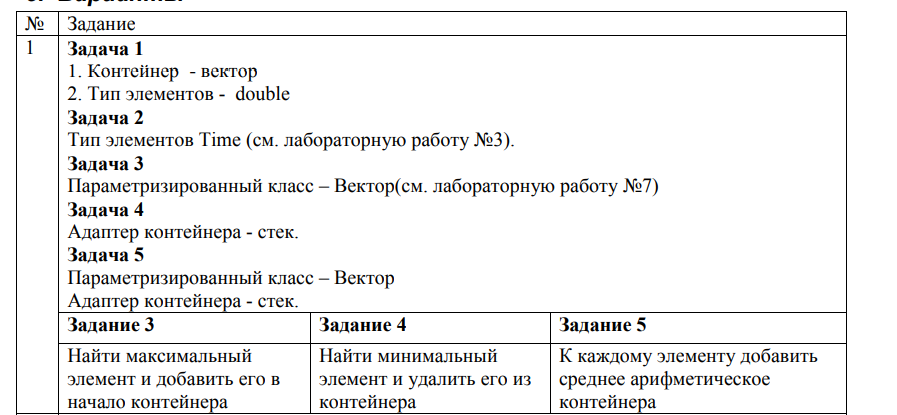
1. Создать параметризированный класс, используя в качестве контейнера последовательный контейнер.
2. Заполнить его элементами.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде методов параметризированного класса.

Задача 4

1. Создать адаптер контейнера.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 5

1. Создать параметризированный класс, используя в качестве контейнера адаптер контейнера.
2. Заполнить его элементами.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде методов параметризированного класса.



**Анализ задачи**

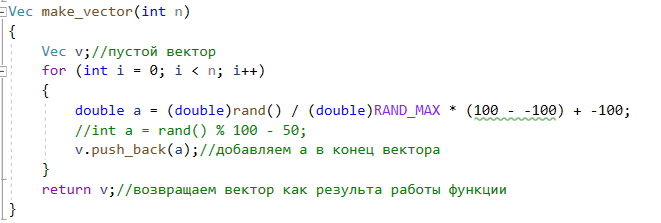
1. Необходимые действия

Задача 1

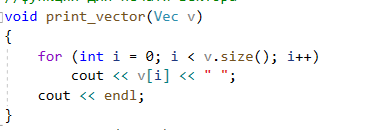
1. Ввести количество элементов в векторе



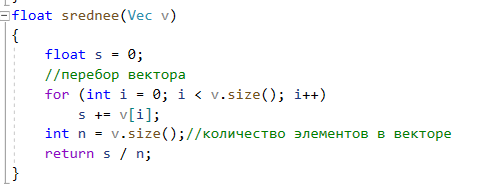
1. Написать функцию случайной генерации элементов в деке



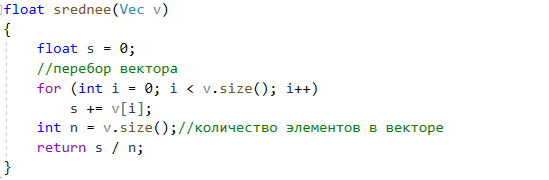
1. Вывести содержимое дека на консоль



1. Найти среднее значение вектора



1. Ввести значение и вставить элемент в вектор

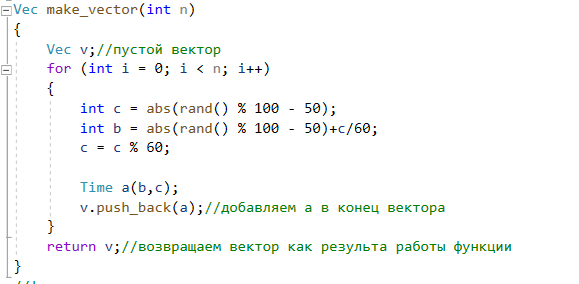


Задача 2

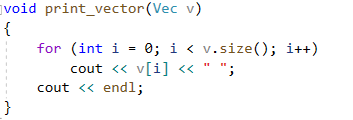
1. Ввести количество элементов в векторе

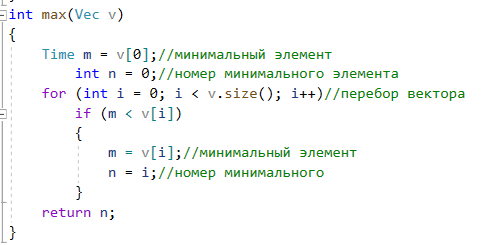


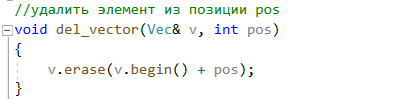
1. Написать функцию случайной генерации элементов в векторе



1. Вывести содержимое вектора на консоль



1. Найти максимальный элемент вектора
2. Ввести значение и удалить найденный элемент из вектора

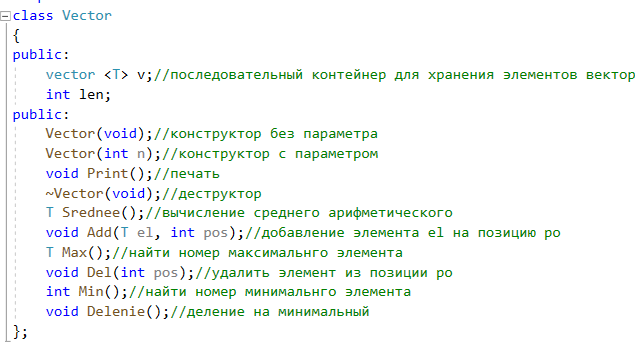


Задача 3

1. Создать вектор для работы с элементами целого типа



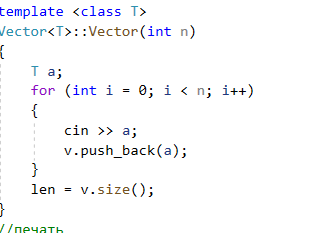
1. Создать шаблонный класс-вектор для работы с элементами любого типа



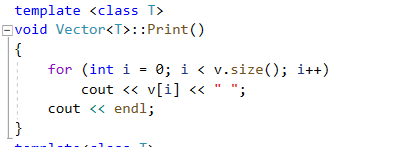
1. Ввести количество элементов в векторе



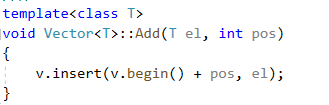
1. Написать функцию для добавления элементов в вектор



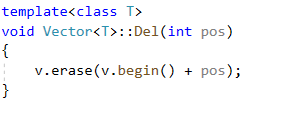
1. Вывести содержимое вектора на консоль

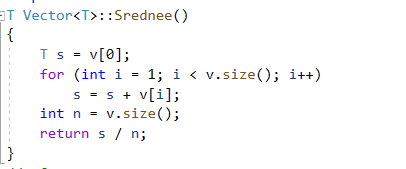


1. Реализовать добавление элементов в вектор



1. Ввести значение и удалить найденный элемент из вектора



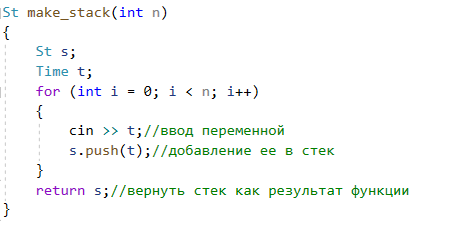
1. Найти среднее арифметическое элементов вектора 

Задача 4

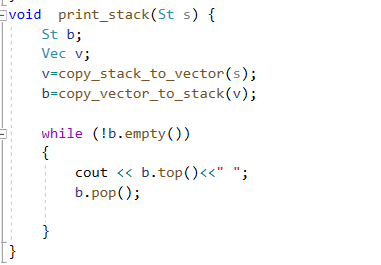
1. Ввести количество элементов в стеке



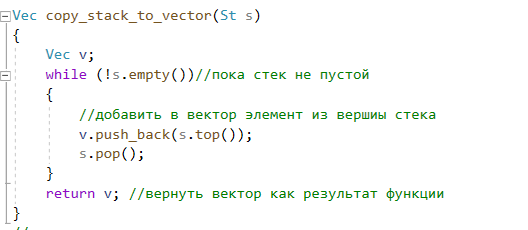
1. Написать функцию по созданию стека



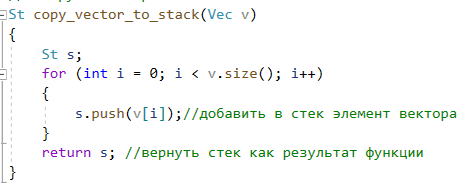
1. Вывести содержимое стека на консоль

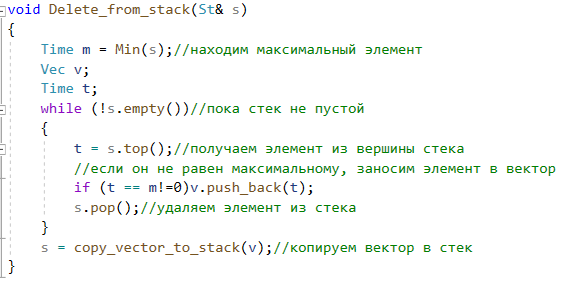
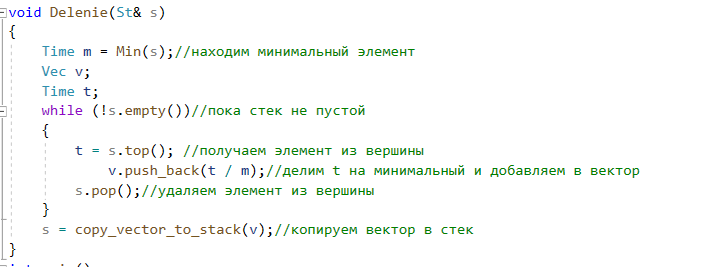
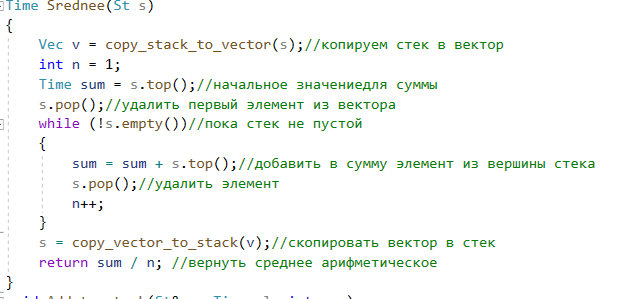


1. Реализовать функцию копирования стека в вектор



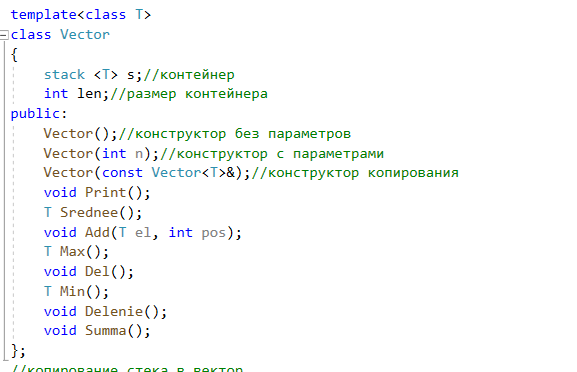
1. Реализовать функцию копирования вектора в стек



1. Удалить максимальный элемент из стека
2. Поделить каждый из элементов стека на минимальное значение 
3. Найти среднее арифметическое элементов стека 

Задача 5

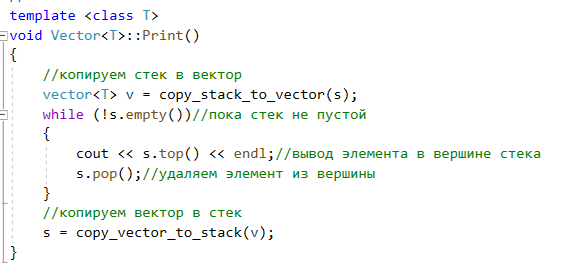
1. Реализовать вектор в виде пользовательского класса



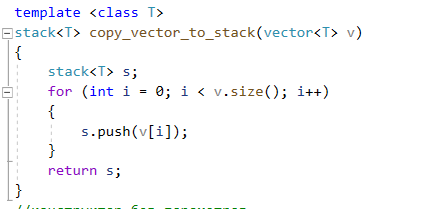
1. Ввести количество элементов в векторе



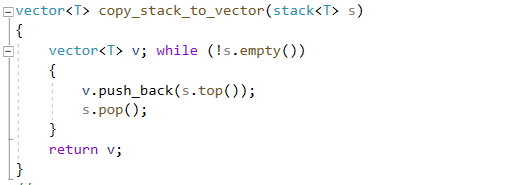
1. Вывести содержимое стека на консоль

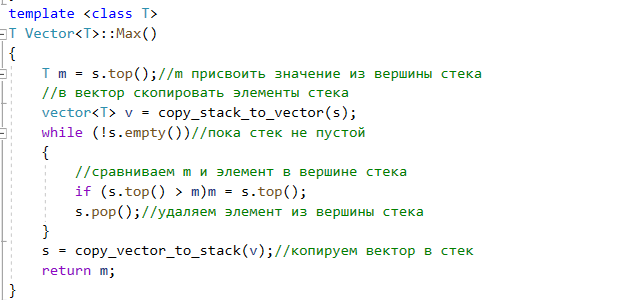


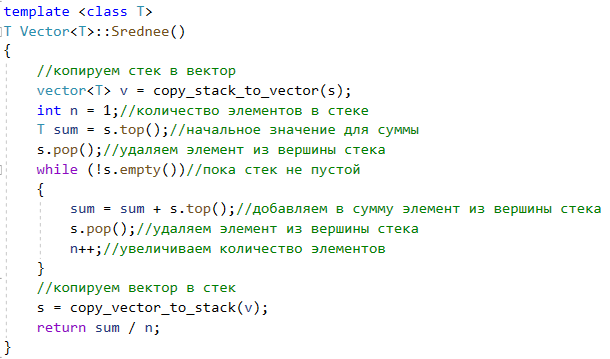
1. Реализовать функцию копирования стека в вектор



1. Реализовать функцию копирования вектора в стек



1. Найти максимальный элемент вектора
2. Найти среднее арифметическое элементов стека



1. Программный код

Задача 1

#include <iostream>

#include <vector>

#include <cstdlib>

using namespace std;

typedef vector<double>Vec;//определяем тип для работы с вектором

//функция для формирования вектора

Vec make\_vector(int n)

{

Vec v;//пустой вектор

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

{

double a = (double)rand() / (double)RAND\_MAX \* (100 - -100) + -100;

//int a = rand() % 100 - 50;

v.push\_back(a);//добавляем а в конец вектора

}

return v;//возвращаем вектор как результа работы функции

}

//функция для печати вектора

void print\_vector(Vec v)

{

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

cout << v[i] << " ";

cout << endl;

}

float srednee(Vec v)

{

float s = 0;

//перебор вектора

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

s += v[i];

int n = v.size();//количество элементов в векторе

return s / n;

}

void add\_vector(Vec& v, int el, int pos)

{

//добавляем на место pos элемент el

v.insert(v.begin() + pos, el);

}

int max(Vec v)

{

int m = v[0],//минимальный элемент

n = 0;//номер минимального элемента

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

if (m < v[i])

{

m = v[i];//максимальный элемент

n = i;//номер максимального

}

return n;

}

//удалить элемент из позиции pos

void del\_vector(Vec& v, int pos)

{

v.erase(v.begin() + pos);

}

int min(Vec v)

{

int m = v[0],//минимальный элемент

n = 0;//номер минимального элемента

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

if (m > v[i])

{

m = v[i];//минимальный элемент

n = i;//номер минимального

}

return n;

}

void delenie(Vec& v)

{

int r = min(v);

double a = v[r];

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

{

v[i] = v[i]/a;

}

}

//основная функция

void main()

{

try

{

vector<double> v;

vector<double>::iterator vi = v.begin();

//формирование вектора

int n;

cout << "N?"; cin >> n;

v = make\_vector(n);

print\_vector(v);

//вычисление среднего

float el = srednee(v);

cout << el;

//позиция для вставки

cout << "pos?";

int pos;

cin >> pos;

//генерируем ошибку, если позиция для вставки больше размера вектора

if (pos > v.size())throw 1;

//вызов функции для добавления

add\_vector(v, el, pos);

//печать вектора

print\_vector(v);

int n\_max = max(v);//найти номер максимального

cout << n\_max<<endl;

del\_vector(v, n\_max);//удалить элемент с этим номером

print\_vector(v);

delenie(v);//

print\_vector(v);

}

catch (int)//блок обработки ошибок

{

cout << "error!";

}

}

Задача 2

#include <iostream>

#include <vector>

#include <cstdlib>

#include "Time.h"

#include <math.h>

using namespace std;

typedef vector<Time>Vec;//определяем тип для работы с вектором

//функция для формирования вектора

Vec make\_vector(int n)

{

Vec v;//пустой вектор

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

{

int c = abs(rand() % 100 - 50);

int b = abs(rand() % 100 - 50)+c/60;

c = c % 60;

Time a(b,c);

v.push\_back(a);//добавляем а в конец вектора

}

return v;//возвращаем вектор как результа работы функции

}

//функция для печати вектора

void print\_vector(Vec v)

{

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

cout << v[i] << " ";

cout << endl;

}

Time srednee(Vec v)

{

Time s(0,0);

//перебор вектора

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

s=s + v[i];

int n = v.size();//количество элементов в векторе

Time c= s / n;

return c;

}

void add\_vector(Vec& v, Time& el, int pos)

{

//добавляем на место pos элемент el

v.insert(v.begin() + pos, el);

}

int mini(Vec v)

{

Time m = v[0];//максимального элемент

int r = 0;//номер максимального элемента

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

if (m > v[i])

{

m = v[i];//максимальный элемент

r = i;//номер максимального

}

return r;

}

int max(Vec v)

{

Time m = v[0];//минимальный элемент

int n = 0;//номер минимального элемента

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

if (m < v[i])

{

m = v[i];//минимальный элемент

n = i;//номер минимального

}

return n;

}

void delenie(Vec& v)

{

int m = mini(v);

Time c = v[m];

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

v[i] = v[i] / c;

}

//удалить элемент из позиции pos

void del\_vector(Vec& v, int pos)

{

v.erase(v.begin() + pos);

}

//основная функция

void main()

{

try

{

vector<Time> v;

vector<Time>::iterator vi = v.begin();

//формирование вектора

int n;

cout << "N?"; cin >> n;

v = make\_vector(n);

print\_vector(v);

//вычисление среднего

Time el= srednee(v);

cout << el << endl;

int n\_min = mini(v);//найти номер максимального

cout << n\_min<< endl;

//позиция для вставки

cout << "pos?";

int pos;

cin >> pos;

//генерируем ошибку, если позиция для вставки больше размера вектора

if (pos > v.size())throw 1;

//вызов функции для добавления

add\_vector(v, el, pos);

//печать вектора

print\_vector(v);

delenie(v);//

print\_vector(v);

}

catch (int)//блок обработки ошибок

{

cout << "error!";

}

}

Задача 3

Vector.h

#pragma once

#include <vector>

#include <iostream>

using namespace std;

//шаблон класса

template<class T>

class Vector

{

public:

vector <T> v;//последовательный контейнер для хранения элементов вектора

int len;

public:

Vector(void);//конструктор без параметра

Vector(int n);//конструктор с параметром

void Print();//печать

~Vector(void);//деструктор

T Srednee();//вычисление среднего арифметического

void Add(T el, int pos);//добавление элемента el на позицию po

T Max();//найти номер максимальнго элемента

void Del(int pos);//удалить элемент из позиции po

int Min();//найти номер минимальнго элемента

void Delenie();//деление на минимальный

};

template <class T>

Vector<T>::Vector()

{

len = 0;

}

//деструктор

template <class T>

Vector<T>::~Vector(void) {

}

//конструктор с параметром

template <class T>

Vector<T>::Vector(int n)

{

T a;

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

{

cin >> a;

v.push\_back(a);

}

len = v.size();

}

//печать

template <class T>

void Vector<T>::Print()

{

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

cout << v[i] << " ";

cout << endl;

}

template<class T>

T Vector<T>::Srednee()

{

T s = v[0];

for (int i = 1; i < v.size(); i++)

s = s + v[i];

int n = v.size();

return s / n;

}

//добавление элемента

template<class T>

void Vector<T>::Add(T el, int pos)

{

v.insert(v.begin() + pos, el);

}

template <class T>

T Vector<T>::Max()

{

T m = v[0];

int n = 0;

for (int i = 1; i < v.size(); i++)

if (v[i] > m)

{

m = v[i];

n = i;

}

return m;

}

//удаление элемента из позиции pos

template<class T>

void Vector<T>::Del(int pos)

{

v.erase(v.begin() + pos);

}

template <class T>

int Vector<T>::Min()

{

T m = v[0];

int n = 0;

for (int i = 1; i < v.size(); i++)

if (v[i] < m)

{

m = v[i];

n = i;

}

return n;

}

template <class T>

//деление всех элементов вектора на минимальный элемент

void Vector<T>::Delenie()

{

int m = Min(); T min = v[m];

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

v[i] = v[i] / min;

}

/\*template <class T>

istream& operator>> (istream& in, Vector<T>& a)

{

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

in >> a.data[i];

return in;

}

template <class T>

ostream& operator<< (ostream& out, const Vector<T>& a)

{

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

out << a.data[i] << " ";

return out;

}\*/

18.11.(3).cpp

// 18.11(3).cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include "Vector.h"

#include <cstdlib>

#include "Time.h"

#include <math.h>

using namespace std;

void main()

{

/\*Vector<Time>vec(5);//создать вектор из 5 элементов

//печать вектора

vec.Print();

\*/

Vector<Time>vec(5);//создать вектор из 5 элементов

vec.Print();//печать вектора

Time s = vec.Srednee();//среднее ариметическое

cout << "Srednee=" << s << endl;

cout << "pos?";

int p;

cin >> p;//ввести позицию для добавления

vec.Add(s, p);//добавить элемент в вектор

vec.Print();//печать вектора

Time r = vec.Max();//найти максимальный элемент

vec.Add(r, 0);

vec.Print();//печать

vec.Delenie();//деление

vec.Print();

//печать

}

Задача 4

// 18.11(4).cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include "Time.h"

#include <stack>

#include <vector>

typedef stack<Time>St;//стек

typedef vector<Time>Vec;//вектор

using namespace std;

St make\_stack(int n)

{

St s;

Time t;

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

{

cin >> t;//ввод переменной

s.push(t);//добавление ее в стек

}

return s;//вернуть стек как результат функции

}

Vec copy\_stack\_to\_vector(St s)

{

Vec v;

while (!s.empty())//пока стек не пустой

{

//добавить в вектор элемент из вершиы стека

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

s.pop();

}

return v; //вернуть вектор как результат функции

}

//копирует вектор в стек

St copy\_vector\_to\_stack(Vec v)

{

St s;

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

{

s.push(v[i]);//добавить в стек элемент вектора

}

return s; //вернуть стек как результат функции

}

void print\_stack(St s) {

St b;

Vec v;

v=copy\_stack\_to\_vector(s);

b=copy\_vector\_to\_stack(v);

while (!b.empty())

{

cout << b.top()<<" ";

b.pop();

}

}

Time Srednee(St s)

{

Vec v = copy\_stack\_to\_vector(s);//копируем стек в вектор

int n = 1;

Time sum = s.top();//начальное значениедля суммы

s.pop();//удалить первый элемент из вектора

while (!s.empty())//пока стек не пустой

{

sum = sum + s.top();//добавить в сумму элемент из вершины стека

s.pop();//удалить элемент

n++;

}

s = copy\_vector\_to\_stack(v);//скопировать вектор в стек

return sum / n; //вернуть среднее арифметическое

}

void Add\_to\_stack(St& s, Time el, int pos)

{

Vec v,c;

Time t;

int i = 1;//номер элемента в стеке

while (!s.empty())//пока стек не пустой

{

t = s.top();

v.push\_back(t);//добавить элемент из стека в вектор

s.pop();//удалить элемент из стека

}

s = copy\_vector\_to\_stack(v);//копируем вектор в стек

//print\_stack( s);

while (!s.empty())//пока стек не пустой

{

t = s.top();//получить элемент из вершины

//если номер равен номеру позиции, на котоую добвляем элемент

if (i == pos)c.push\_back(el);//добавить новый элемент в вектор

c.push\_back(t);//добавить элемент из стека в вектор

s.pop();//удалить элемент из стека

i++;

}

s = copy\_vector\_to\_stack(c);//копируем вектор в стек

}

Time Max(St s)

{

Time m = s.top();//переменной m присваиваем значение из вершины стека

Vec v = copy\_stack\_to\_vector(s);//копируем стек в вектор

while (!s.empty())//пока стк не пустой

{

if (s.top() > m)m = s.top();//срвниваем m и элемнт в вершине стека

s.pop();//удаляем элемент из стека

}

s = copy\_vector\_to\_stack(v);//копиуем вектор в стек

return m; //возвращаем m

}

Time Min(St s)

{

Time m = s.top();

Vec v = copy\_stack\_to\_vector(s);

while (!s.empty())

{

if (s.top() < m)m = s.top();

s.pop();

}

s = copy\_vector\_to\_stack(v);

return m;

}

//Удалить максимальный элемент из стека

void Delete\_from\_stack(St& s)

{

Time m = Min(s);//находим максимальный элемент

Vec v;

Time t;

while (!s.empty())//пока стек не пустой

{

t = s.top();//получаем элемент из вершины стека

//если он не равен максимальному, заносим элемент в вектор

if (t == m!=0)v.push\_back(t);

s.pop();//удаляем элемент из стека

}

s = copy\_vector\_to\_stack(v);//копируем вектор в стек

}

//деление на минимальный

void Delenie(St& s)

{

Time m = Min(s);//находим минимальный элемент

Vec v;

Time t;

while (!s.empty())//пока стек не пустой

{

t = s.top(); //получаем элемент из вершины

v.push\_back(t / m);//делим t на минимальный и добавляем в вектор

s.pop();//удаляем элемент из вершины

}

s = copy\_vector\_to\_stack(v);//копируем вектор в стек

}

int main()

{

/\* Time t;

St s;

int n;

cout << "n?";

cin >> n;

s = make\_stack(n);//создать стек

print\_stack(s);//печать стека

system ("pause");\*/

setlocale(LC\_ALL, "rus");

Time t;

St s;

int n;

cout << "n?";

cin >> n;

s = make\_stack(n);//создать стек

print\_stack(s);//печать стека

t = Srednee(s);//вычисляем среднее

cout << "Srednee=" << Srednee(s) << endl;

cout << "Add srednee:" << endl;

cout << "pos?";

int pos;

cin >> pos;//вводим позицию для добавления

Add\_to\_stack(s, t, pos);//добавление элемента

print\_stack(s);//печать стека

cout << "Delete Min:" << endl;

Delete\_from\_stack(s);

print\_stack(s);

cout << "Delenie:" << endl;

Delenie(s);

print\_stack(s);

}

Задача 5

Vector.h

##include <stack>

#include <vector>

#include <iostream>

using namespace std;

template<class T>

class Vector

{

stack <T> s;//контейнер

int len;//размер контейнера

public:

Vector();//конструктор без параметров

Vector(int n);//конструктор с параметрами

Vector(const Vector<T>&);//конструктор копирования

void Print();

T Srednee();

void Add(T el, int pos);

T Max();

void Del();

T Min();

void Delenie();

void Summa();

};

//копирование стека в вектор

template <class T>

vector<T> copy\_stack\_to\_vector(stack<T> s)

{

vector<T> v; while (!s.empty())

{

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

s.pop();

}

return v;

}

//копирование вектора в стек

template <class T>

stack<T> copy\_vector\_to\_stack(vector<T> v)

{

stack<T> s;

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

{

s.push(v[i]);

}

return s;

}

//конструктор без параметров

template <class T>

Vector<T>::Vector()

{

len = 0;

}

//конструктор с параметром

template <class T>

Vector<T>::Vector(int n)

{

T a;

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

{

cin >> a;

s.push(a);//добавить в стек элемент а

}

len = s.size();

}

//конструктор копирования

template <class T>

Vector<T>::Vector(const Vector<T>& Vec)

{

len = Vec.len;

//копируем значения стека Vec.s в вектор v

vector v = copy\_stack\_to\_vector(Vec.s);

//копируем вектор v в стек s

s = copy\_vector\_to\_stack(v);

}

//печать

template <class T>

void Vector<T>::Print()

{

//копируем стек в вектор

vector<T> v = copy\_stack\_to\_vector(s);

while (!s.empty())//пока стек не пустой

{

cout << s.top() << endl;//вывод элемента в вершине стека

s.pop();//удаляем элемент из вершины

}

//копируем вектор в стек

s = copy\_vector\_to\_stack(v);

}

template <class T>

T Vector<T>::Srednee()

{

//копируем стек в вектор

vector<T> v = copy\_stack\_to\_vector(s);

int n = 1;//количество элементов в стеке

T sum = s.top();//начальное значение для суммы

s.pop();//удаляем элемент из вершины стека

while (!s.empty())//пока стек не пустой

{

sum = sum + s.top();//добавляем в сумму элемент из вершины стека

s.pop();//удаляем элемент из вершины стека

n++;//увеличиваем количество элементов

}

//копируем вектор в стек

s = copy\_vector\_to\_stack(v);

return sum / n;

}

//добавление элемента el в стек на позицию pos

template <class T>

void Vector<T>::Add(T el, int pos)

{

vector <T>v;//вспомогательный вектор

T t;

int i = 1;

while (!s.empty())//пока стек не пустой

{

t = s.top();//получить элемент из вершины стека

//если номер элемента равен pos добавляем в вектор новый элемент

if (i == pos)v.push\_back(el);

v.push\_back(t);//добавляем t в вектор

s.pop();//удаляем элемент из вершины стека

i++;

}

s = copy\_vector\_to\_stack(v);//копируем вектор в стек

}

template <class T>

T Vector<T>::Max()

{

T m = s.top();//m присвоить значение из вершины стека

//в вектор скопировать элементы стека

vector<T> v = copy\_stack\_to\_vector(s);

while (!s.empty())//пока стек не пустой

{

//сравниваем m и элемент в вершине стека

if (s.top() > m)m = s.top();

s.pop();//удаляем элемент из вершины стека

}

s = copy\_vector\_to\_stack(v);//копируем вектор в стек

return m;

}

//удаление элемента из вектора

template <class T>

void Vector<T>::Del()

{

T m = Max();//поиск максимального

vector<T> v;

T t;

while (!s.empty())//пока стек не пустой

{

t = s.top();//получить элемент из вершины стека

//если t не равен максимальному, то добавить его в вектор

if (t ==m!=0)v.push\_back(t);

s.pop();//удалить элемент из стека

}

//копируем вектор в стек

s=copy\_vector\_to\_stack(v);

}

template <class T>

T Vector<T>::Min()

{

T m = s.top();

vector<T> v = copy\_stack\_to\_vector(s);

while (!s.empty())

{

if (s.top() < m)m = s.top();

s.pop();

}

s = copy\_vector\_to\_stack(v);

return m;

}

//деление всех элементов на минимальный

template <class T>

void Vector<T>::Delenie()

{

T m = Min();

vector<T> v;

T t;

while (!s.empty())

{

t = s.top();

v.push\_back(t / m);

s.pop();

}

s = copy\_vector\_to\_stack(v);

}

template <class T>

void Vector<T>::Summa()

{

T m = Srednee();

vector<T> v;

T t;

while (!s.empty())

{

t = s.top();

v.push\_back(t + m);

s.pop();

}

s = copy\_vector\_to\_stack(v);

}

18.11(5).cpp

#include <iostream>

#include <stack>

#include <vector>

#include "Vector.h"

#include "Time.h"

using namespace std;

//typedef vector<Time>Vec;

void main()

{

Vector<Time>v(3);

v.Print();

Time t = v.Srednee();

cout << "\nSrednee=" << t << endl;

v.Summa();

v.Print();

cout << "Add srednee" << endl;

cout << "pos?";

int pos;

cin >> pos;

v.Add(t, pos);

v.Print();

cout << "Max=" << v.Max() << endl;

cout << "Delete Max:" << endl;

v.Del();

v.Print();

cout << "Delenie na min" << endl;

cout << "Min=" << v.Min() << endl;

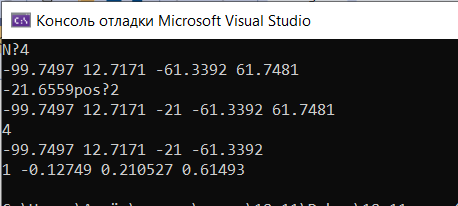
v.Delenie();

v.Print();

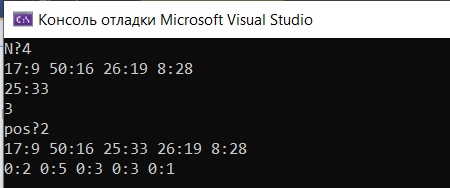
}

1. Демонстрация работы программы

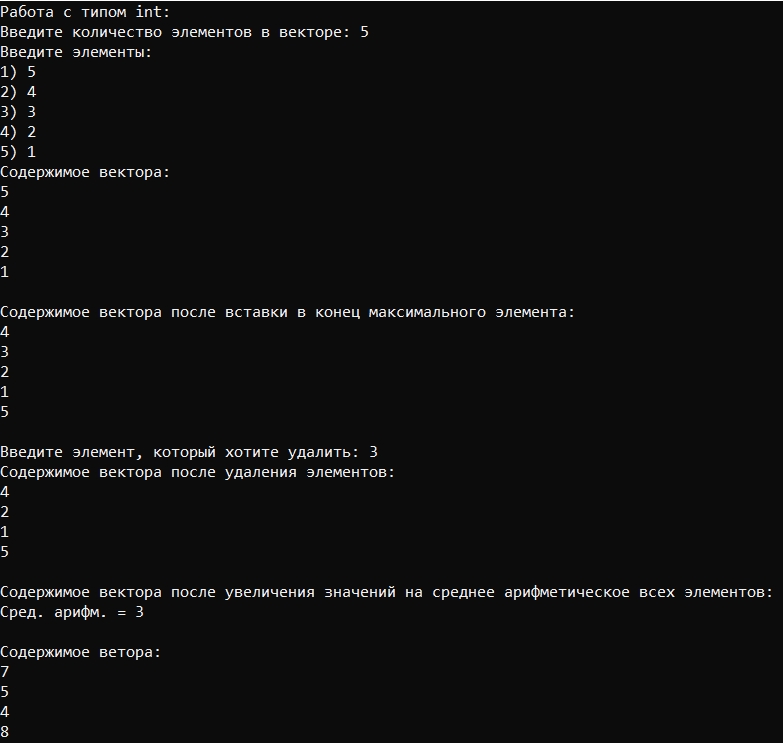
Задача 1

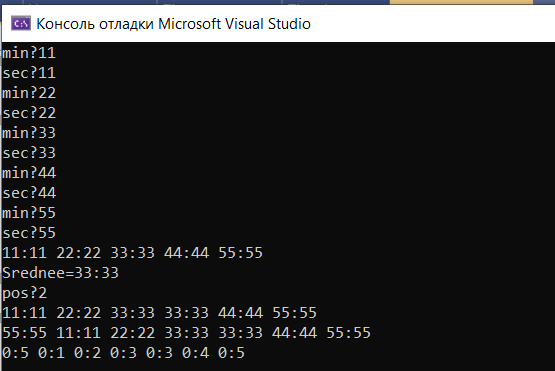


Задача 2

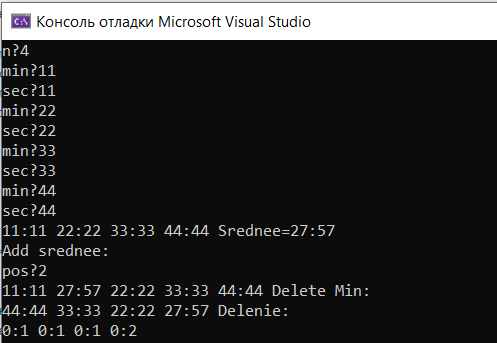


Задача 3

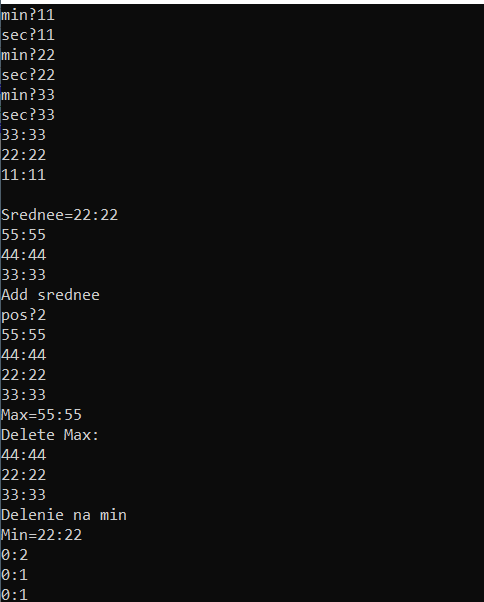




Задача 4



Задача 5



1. Типы переменных и представление данных

Задача 1

Deque<int> - дек для хранения целых чисел, int – целое число, которое помещается в дек; максимальное значение в деке; размер дека; сумма элементов в деке; среднее арифметическое

Задача 2

Vector<Money> - вектор для хранения объектов класса Money, int – хранение рублей и копеек, среднее арифметическое для рублей и копеек, размер вектора, сумма рублей и копеек, Money – класс для хранения значений рублей и копеек, максимальная сумма денег

Задача 3

Vector – класс для хранения элементов любого типа (int или double), vector<int> - вектор для хранения элементов целого типа, vector<double> - вектор для хранения элементов вещественного типа, vector<T> - вектор для хранения элементов любого типа (int и double), int – целое число, которое нужно удалить из вектора, размер вектора, double – вещественное число, которое нужно удалить из вектора, T – шаблонный тип

Задача 4

Int – значение стека, значение вектора, максимальное значение в стеке, максимальное значение в стеке, сумма элементов в стеке, среднее арифметическое, размер стека, размер вектора, число для удаления из стека, stack<int> - стек для хранения элементов целого типа, vector<int> - вектор для хранения элементов целого типа

Задача 5

Stack – пользовательский класс стек для хранения значений элементов, stack<int> - стек для хранения элементов целого типа, vector<int> - вектор для хранения элементов целого типа, int – значение стека, значение вектора, максимальное значение в стеке, максимальное значение в стеке, сумма элементов в стеке, среднее арифметическое, размер стека, размер вектора, число для удаления из стека

1. Контрольные вопросы
   1. Из каких частей состоит библиотека STL?

- стандартная библиотека шаблонов состоит из двух основных частей: набора контейнерных классов и набора обобщенных алгоритмов.

* 1. Какие типы контейнеров существуют в STL?

- последовательные и ассоциативные

* 1. Что нужно сделать для использования контейнера STL в своей программе?

- нужно подключить библиотеку с названием контейнера

* 1. Что представляет собой итератор?

- это обобщение концепции указателей: они ссылаются на элементы контейнера

* 1. Какие операции можно выполнять над итераторами?

- инкрементирование, присваивание одного итератора другому, сравнение итераторов на равенство и неравенство и разыменовывание

* 1. Каким образом можно организовать цикл для перебора контейнера с использованием итератора?

vector<int> myvector;

vector<int>::iterator iter;

for (iter = myvector.begin(); iter != myvector.end(); iter++)

cout<<\*iter<<endl;

* 1. Какие типы итераторов существуют?
* входные,
* выходные,
* прямые,
* двунаправленные итераторы,
* итераторы произвольного доступа
  1. Перечислить операции и методы общие для всех контейнеров.

-операции равенства и неравенства

-операция присваивания

-clear, insert, erase, size\_type size() const, size\_type max\_size() const, bool empty() const, iterator begin(),iterator end(),reverse\_iterator begin(),reverse\_iterator end()

* 1. Какие операции являются эффективными для контейнера vector? Почему?

[] и at – являются эффективными, так как не задействуются затратные по времени алгоритмы копирования или вставки элементов

* 1. Какие операции являются эффективными для контейнера list? Почему?

Вставка и удаление на любую позицию являются эффективными операциями, так как работа происходит с указателя без необходимости копирования элементов в другой список

* 1. Какие операции являются эффективными для контейнера deque? Почему?

Вставка и удаление первого элемента являются эффективными операциями, так как для этого не нужно копировать все элементы в другой дек, достаточно лишь поменять значения ссылок

* 1. Перечислить методы, которые поддерживает последовательный контейнер vector.

Push\_back(), pop\_back(), insert(), erase(), [], at, swap(), clear()

* 1. Перечислить методы, которые поддерживает последовательный контейнер list.

Push\_back(), pop\_back(), push\_front(), pop\_front(), insert(), erase(), swap(), clear(), splice()

* 1. Перечислить методы, которые поддерживает последовательный контейнер deque.

Push\_back(), pop\_back(), push\_front(), pop\_front(), insert(), erase(), [], at,

* 1. Задан контейнер vector. Как удалить из него элементы со 2 по 5?

vector<int> myvector = { 1, 2, 3, 4, 5 };

vector<int>::iterator iter;

iter = myvector.begin();

myvector.erase(++iter, iter+5);

* 1. Задан контейнер vector. Как удалить из него последний элемент?

vector<int> myvector = { 1, 2, 3, 4, 5 };

myvector.erase(--myvector.end());

* 1. Задан контейнер list. Как удалить из него элементы со 2 по 5?

list<int> mylist = { 1, 2, 3, 4, 5 };

list<int>::iterator iter1, iter2;

iter2 = mylist.begin();

iter2++;

for (int i = 0; i < 4; i++) {

iter1 = iter2;

iter2++;

mylist.erase(iter1);

}

* 1. Задан контейнер list. Как удалить из него последний элемент?

list<int> mylist = { 1, 2, 3, 4, 5 };

list<int>::iterator iter1;

iter1 = mylist.end();

iter1--;

mylist.erase(iter1);

* 1. Задан контейнер deque. Как удалить из него элементы со 2 по 5?

deque<int> mydeque = { 1, 2, 3, 4, 5 };

deque<int>::iterator iter1;

iter1 = mydeque.begin();

iter1++;

mydeque.erase(iter1, iter1+4);

* 1. Задан контейнер deque. Как удалить из него последний элемент?

deque<int> mydeque = { 1, 2, 3, 4, 5 };

deque<int>::iterator iter1;

iter1 = mydeque.end();

mydeque.erase(--iter1);

* 1. Написать функцию для печати последовательного контейнера с использованием итератора.

void print\_deque(deque<int> mydeq) {

deque<int>::iterator iter = mydeq.begin();

for (; iter != mydeq.end(); iter++)

cout << \*iter << endl;

}

* 1. Что представляют собой адаптеры контейнеров?

Контейнерные классы, реализованные на основе вектора, двунаправленной очереди, двунаправленного списка

* 1. Чем отличаются друг от друга объявления stack<int> s и stack<int, list<int> > s?

Стек является адаптером, по умолчанию он создается на основе дека (в первом случае). Во втором случае основой для стека будет являться двунаправленный список

* 1. Перечислить методы, которые поддерживает контейнер stack.

Push(), pop(), top(), empty(), size()

* 1. Перечислить методы, которые поддерживает контейнер queue.

Push(), pop(), front(), back(), empty(), size()

* 1. Чем отличаются друг от друга контейнеры queue и priority\_queue?

Очередь с приоритетами отличается от обычной очереди тем, что для извлечения выбирается макси- мальный элемент из хранимых в контейнере. Поэтому после каждого изменения состояния очереди максимальный элемент из оставшихся сдвигается в начало контейнера.

* 1. Задан контейнер stack. Как удалить из него элемент с заданным номером?

void delete\_element(stack<int> &mystack, int index) {

if (mystack.size() == 0)

return;

stack<int> temp\_stack;

for (int i = 0; i < index; i++) {

temp\_stack.push(mystack.top());

mystack.pop();

}

mystack.pop();

int temp\_stack\_size = temp\_stack.size();

for (int i = 0; i < temp\_stack\_size; i++) {

mystack.push(temp\_stack.top());

temp\_stack.pop();

}

}

* 1. Задан контейнер queue. Как удалить из него элемент с заданным номером?

void delete\_element(queue<int> &myqueue, int index) {

if (myqueue.size() == 0)

return;

queue<int> temp\_queue;

for (int i = 0; i < index; i++) {

temp\_queue.push(myqueue.front());

myqueue.pop();

}

myqueue.pop();

int myqueue\_size = myqueue.size();

for (int i = 0; i < myqueue\_size; i++) {

temp\_queue.push(myqueue.front());

myqueue.pop();

}

int temp\_queue\_size = temp\_queue.size();

for (int i = 0; i < temp\_queue\_size; i++) {

myqueue.push(temp\_queue.front());

temp\_queue.pop();

}

}

* 1. Написать функцию для печати контейнера stack с использованием итератора.

vector<int> copy\_stack\_to\_vector(stack<int> st)

{

vector<int> vec;

while (!st.empty())

{

vec.push\_back(st.top());

st.pop();

}

return vec;

}

void print\_stack(stack<int> st) {

vector<int> temp\_vec = copy\_stack\_to\_vector(st);

vector<int>::iterator iter;

for (iter = temp\_vec.begin(); iter != temp\_vec.end(); iter++)

cout << \*iter << endl;

temp\_vec.clear();

}

* 1. Написать функцию для печати контейнера queue с использованием итератора.

vector<int> copy\_queue\_to\_vector(queue<int> qu)

{

vector<int> vec;

while (!qu.empty())

{

vec.push\_back(qu.front());

qu.pop();

}

return vec;

}

void print\_queue(queue<int> qu) {

vector<int> temp\_vec = copy\_queue\_to\_vector(qu);

vector<int>::iterator iter;

for (iter = temp\_vec.begin(); iter != temp\_vec.end(); iter++)

cout << \*iter << endl;

temp\_vec.clear();

}