Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

за 4 семестр

По дисциплине: «ООТиСП»

Тема: «СТАНДАРТНАЯ БИБЛИОТЕКА ШАБЛОНОВ»

Выполнил:

Студент 2 курса

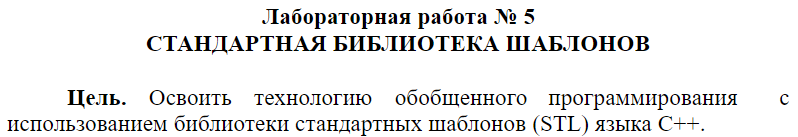
Группы ПО-4(2)

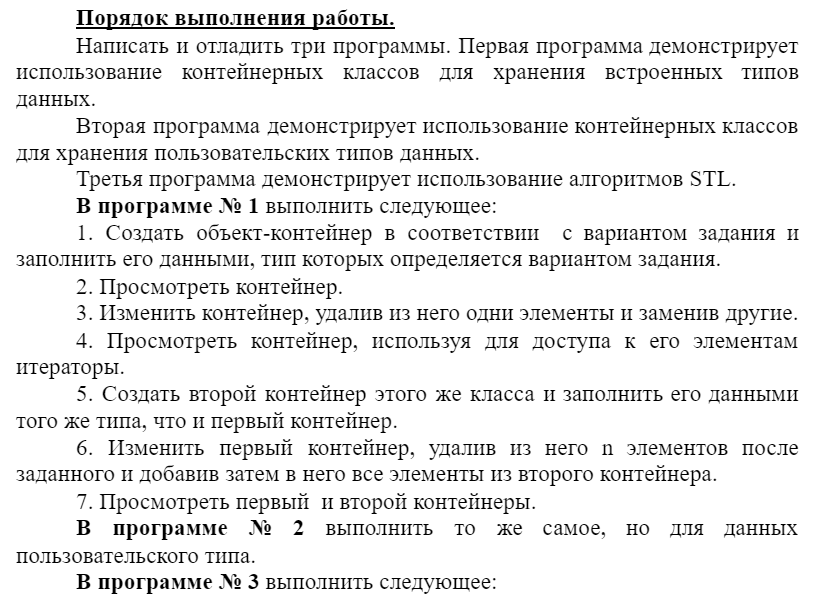
Кречко К.А.

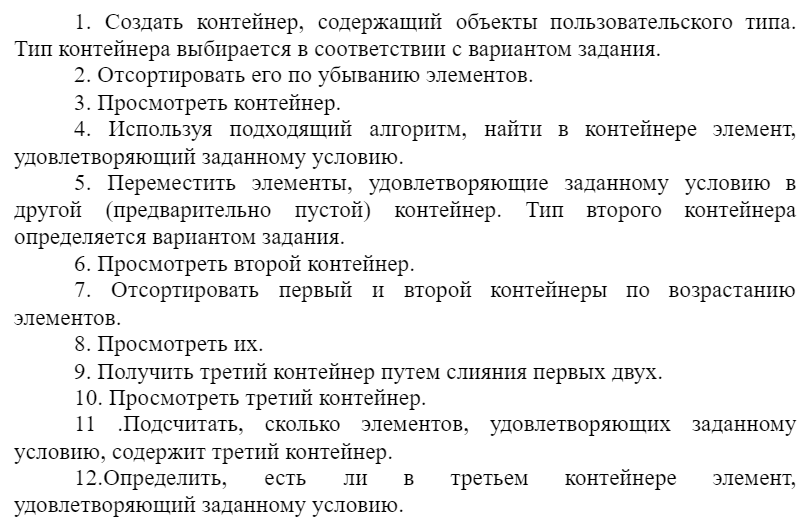
Проверил:

Миндер А. В.

Брест 2021







**Вариант – 14**

****

**Программа №1:**

#include <iostream>

#include <stack>

using namespace std;

void stack\_looking(stack <float> steck) {

const int size = steck.size();

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

cout << (steck.top()) << endl;

steck.pop();

}

}

int main() {

setlocale(LC\_ALL, "rus");

system("color f0");

// Заполнение

stack <float> main\_steck;

for (int i = 0; i < 5; i++) {

main\_steck.push((float)(rand()) / 1000);

}

// Просмотр

cout << "Исходный стек" << endl;

stack\_looking(main\_steck);

// Удаление элемента

stack <float> steck2 = main\_steck; // копируем исходный стек

stack <float> temp\_steck; // временный стек

cout << "\nУдаление элемента. Введите № элемента, который хотите удалить: ";

int n;

cin >> n;

if (n < 0 || n > sizeof(main\_steck) / sizeof(int)) return 0;

for (int i = 0; i < main\_steck.size(); i++) { // заполняем врем-й стек значениями, кроме выбранного

if (i == (n-1)) {

steck2.pop();

continue;

}

temp\_steck.push(steck2.top());

steck2.pop();

}

int size = temp\_steck.size();

for (int i = 0; i < size; i++) { // реверсируем стек

steck2.push(temp\_steck.top());

temp\_steck.pop();

}

cout << "Результат: " << endl;

stack\_looking(steck2);

// Просмотр стека через иттераторы

//(в более новых версиях используем метод peek(). Т.к. его нет, реализуем стек через массив)

steck2 = main\_steck;

float steck[5];

cout << "\nОбращение по индексу. Исходный стек:" << endl;

for (int i = 0; i < 5; i++) {

steck[i] = steck2.top();

steck2.pop();

cout << steck[i] << endl;

}

cout << "Введите индекс элемента ";

cin >> n;

if (n < 0 || n > sizeof(steck) / sizeof(int)) return 0;

cout << "Результат: " << steck[n] << endl;

//Создание/заполнение второго контейнера

cout << "\nВторой контейнер: " << endl;

stack <float> main\_steck2;

for (int i = 0; i < 5; i++) {

main\_steck2.push((float)(rand()) / 1000);

}

stack\_looking(main\_steck2);

cout << "Введите индекс элемента, после которого будем удалять ";

int f;

cin >> f;

if (f < 0 || f > sizeof(main\_steck2) / sizeof(int)) return 0;

cout << "Введите количество удаляемых элементов: ";

cin >> n;

if (n < 0 || n > sizeof(main\_steck2) / sizeof(int)) return 0;

stack <float> temp\_stack;

size = main\_steck2.size();

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

if (i > f && i <= f + n) {

main\_steck2.pop();

continue;

}

temp\_stack.push(main\_steck2.top());

main\_steck2.pop();

}

size = temp\_stack.size(); // реверсируем стек

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

main\_steck2.push(temp\_stack.top());

temp\_stack.pop();

}

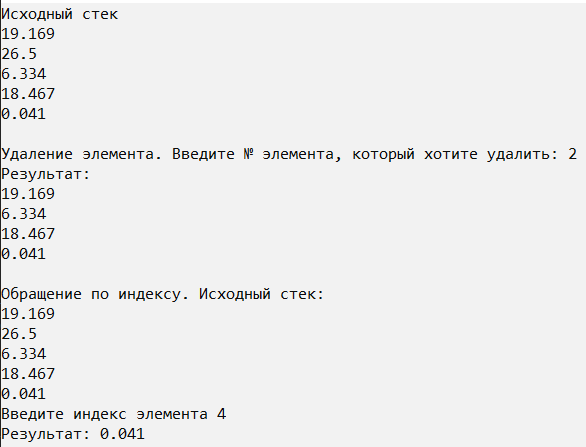
cout << "Результат: " << endl;

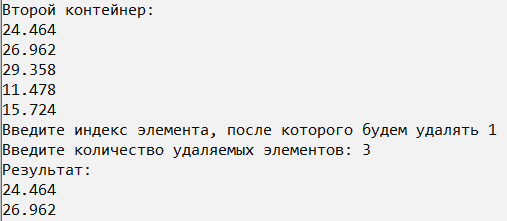
stack\_looking(main\_steck2);

return 0;

}

**Результат:**

****

****

**Программа №2:**

#include <iostream>

#include <stack>

using namespace std;

class Circle {

int x;

int y;

int R;

public:

Circle(int x = 0, int y = 0, int R = 1) {

this->x = x;

this->y = y;

this->R = R;

}

int Getx() {

return x;

}

int Gety() {

return y;

}

int GetR() {

return R;

}

friend std::ostream& operator<< (std::ostream& out, const Circle& point);

};

std::ostream& operator<<(std::ostream& out, const Circle& point) {

out << "Circle" << " ( " << point.x << ", " << point.y << ", " << point.R << " )";

return out;

}

void stack\_looking(stack <Circle> steck) {

const int size = steck.size();

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

cout << (steck.top()) << endl;

steck.pop();

}

}

int main() {

setlocale(LC\_ALL, "rus");

system("color f0");

// Заполнение

Circle a(5, 2, 1);

Circle b(1, 2, 3);

Circle c(22, 84, 50);

Circle d(-2, 4, 16);

Circle e;

stack <Circle> main\_steck;

main\_steck.push(a);

main\_steck.push(b);

main\_steck.push(c);

main\_steck.push(d);

main\_steck.push(e);

// Просмотр

cout << "Исходный стек" << endl;

stack\_looking(main\_steck);

// Удаление элемента

stack <Circle> steck2 = main\_steck; // копируем исходный стек

stack <Circle> temp\_steck; // временный стек

cout << "\nУдаление элемента. Введите № элемента, который хотите удалить: ";

int n;

cin >> n;

if (n < 0 || n > sizeof(main\_steck) / sizeof(int)) return 0;

for (int i = 0; i < main\_steck.size(); i++) { // заполняем врем-й стек значениями, кроме выбранного

if (i == (n - 1)) {

steck2.pop();

continue;

}

temp\_steck.push(steck2.top());

steck2.pop();

}

int size = temp\_steck.size();

for (int i = 0; i < size; i++) { // реверсируем стек

steck2.push(temp\_steck.top());

temp\_steck.pop();

}

cout << "Результат: " << endl;

stack\_looking(steck2);

// Просмотр стека через иттераторы

steck2 = main\_steck;

Circle steck[5];

cout << "\nОбращение по индексу. Исходный стек:" << endl;

for (int i = 0; i < 5; i++) {

steck[i] = steck2.top();

steck2.pop();

cout << steck[i] << endl;

}

cout << "Введите индекс элемента ";

cin >> n;

if (n < 0 || n > sizeof(steck) / sizeof(int)) return 0;

cout << "Результат: " << steck[n] << endl;

//Создание/заполнение второго контейнера

cout << "\nВторой контейнер: " << endl;

Circle a1(6, 3, 2);

Circle b1(3, 4, 5);

Circle c1(12, 854, 0);

Circle d1(-3, 74, 6);

Circle e1(-1, -2, 4);

stack <Circle> main\_steck2;

main\_steck2.push(a1);

main\_steck2.push(b1);

main\_steck2.push(c1);

main\_steck2.push(d1);

main\_steck2.push(e1);

stack\_looking(main\_steck2);

cout << "Введите индекс элемента, после которого будем удалять ";

int f;

cin >> f;

if (f < 0 || f > sizeof(main\_steck2) / sizeof(int)) return 0;

cout << "Введите количество удаляемых элементов: ";

cin >> n;

if (n < 0 || n > sizeof(main\_steck2) / sizeof(int)) return 0;

stack <Circle> temp\_stack;

size = main\_steck2.size();

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

if (i > f && i <= f + n) {

main\_steck2.pop();

continue;

}

temp\_stack.push(main\_steck2.top());

main\_steck2.pop();

}

size = temp\_stack.size(); // реверсируем стек

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

main\_steck2.push(temp\_stack.top());

temp\_stack.pop();

}

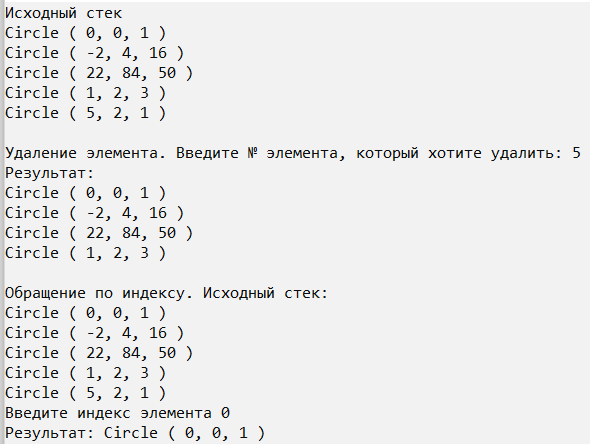
cout << "Результат: " << endl;

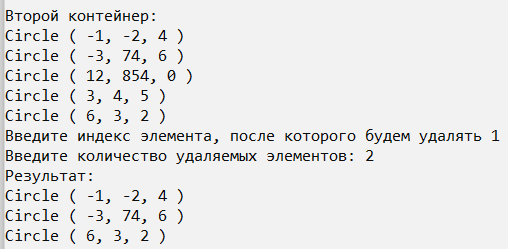
stack\_looking(main\_steck2);

return 0;

}

**Результат:**

****

****

**Программа №3:**

#include <iostream>

#include <vector>

#include <algorithm>

#include <stack>

using namespace std;

class Circle {

private:

int x;

int y;

int R;

public:

Circle(int x = 0, int y = 0, int R = 1) {

this->x = x;

this->y = y;

this->R = R;

}

int Getx() {

return x;

}

int Gety() {

return y;

}

int GetR() {

return R;

}

friend std::ostream& operator<< (std::ostream& out, const Circle& point);

};

ostream& operator<<(ostream& out, const Circle& point) {

out << "Circle" << " ( " << point.x << ", " << point.y << ", " << point.R << " )";

return out;

}

bool predicate\_function(Circle a) {

return a.Getx() >= 0 && a.Gety() >= 0;

}

void stack\_looking(stack <Circle> steck) {

const int size = steck.size();

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

cout << (steck.top()) << endl;

steck.pop();

}

}

void vector\_looking(const vector <Circle> vec) {

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

cout << vec[i] << endl;

}

stack <Circle> stack\_sort(stack <Circle>& steck) {

const int size = steck.size();

vector <Circle> temp\_vector;

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

temp\_vector.push\_back(steck.top());

steck.pop();

}

sort(temp\_vector.begin(), temp\_vector.end(), [](Circle a, Circle b) {return a.GetR() > b.GetR(); });

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

steck.push(temp\_vector[i]);

return steck;

}

int main() {

setlocale(LC\_ALL, "rus");

system("color f0");

Circle a(5, 2, 1);

Circle b(1, 2, 3);

Circle c(22, -84, 50);

Circle d(-2, 4, 16);

Circle e;

Circle f(9, 4, 10);

// Создание/заполнение

vector <Circle> vec;

vec.push\_back(a);

vec.push\_back(b);

vec.push\_back(c);

vec.push\_back(d);

vec.push\_back(e);

vec.push\_back(f);

// Просмотр

cout << "Условие: окружности касаются I четверти\n";

cout << "Исходный вектор:\n";

vector\_looking(vec);

// Сортировка

sort(vec.begin(), vec.end(), [](Circle a, Circle b) {return a.GetR() > b.GetR();} );

cout << "\nСортировка по убыванию:\n";

vector\_looking(vec);

// Поиск элемента, удовлетворяющего условию

cout << "\nЭлемент, удовлетворяюий условию:\n";

auto result = find\_if(vec.begin(), vec.end(), predicate\_function);

if (result != vec.end())

cout << "Circle: (" << result->Getx() << ", " << result->Gety() << ", " << result->GetR() << ")\n";

else

cout << "Такого элемента нет" << endl;

// Перенос элементов в другой контейнер (стек)

stack <Circle> steck;

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

if (predicate\_function(vec[i]))

steck.push(vec[i]);

}

// Просмотр 2-го контейнера

cout << "\nСтек с объектами, подходящими под условие: \n";

stack\_looking(steck);

// Сортировка и просмотр контейнеров по возрастанию

sort(vec.begin(), vec.end(), [](Circle a, Circle b) {return a.GetR() < b.GetR();});

cout << "\nВектор по возрастанию: \n";

vector\_looking(vec);

stack\_sort(steck);

cout << "\nСтек по возрастанию: \n";

stack\_looking(steck);

// Создание, заполнение и просмотр 3-го контейнера

vector <Circle> vec1;

vec1.assign(vec.begin(), vec.end());

const int size = steck.size();

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

vec1.push\_back(steck.top());

steck.pop();

}

cout << "\nТретий контейнер\n";

vector\_looking(vec1);

// Подсчёт кол-ва элем-ов, удовл-х условию

cout << "\nКол-во элем-ов, удовл-х условию: " <<

count\_if(vec1.begin(), vec1.end(), predicate\_function) << endl;

// Определяем, есть ли элем-т, удовл-й условию

auto result2 = find\_if(vec1.begin(), vec1.end(), predicate\_function);

if (result2 != vec1.end())

cout << "Элемент, удовлетворяющий условию, есть" << endl;

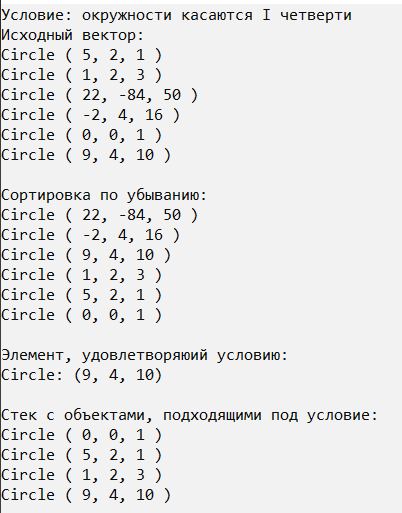
else

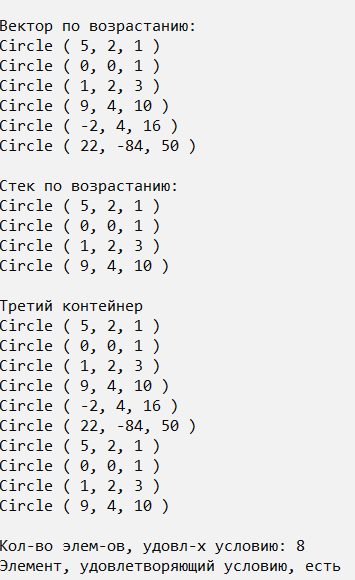
cout << "Такого элемента нет" << endl;

return 0;

}

**Результат:**

****

****

**Вывод:** Я освоил технологию обобщенного программирования с использованием библиотеки стандартных шаблонов (STL) языка C++.