**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 5**

Тема: Основы работы с коллекциями: итераторы

Студент: Курносов Максим Юрьевич

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

1. **Постановка задачи**

Разработать шаблоны классов согласно варианту задания. Параметром шаблона должен являться скалярный тип данных задающий тип данных для оси координат. Классы должны иметь публичные поля. Фигуры являются фигурами вращения, т.е. равносторонними (кроме трапеции и прямоугольника). Для хранения координат фигур необходимо использовать шаблон std::pair.

Реализовать программу, которая:

Позволяет вводить с клавиатуры фигуры (с типом int в качестве параметра шаблона фигуры) и добавлять в коллекцию;

Позволяет удалять элемент из коллекции по номеру элемента;

Выводит на экран введенные фигуры c помощью std::for\_each;

Выводит на экран количество объектов, у которых площадь меньше заданной (с помощью std::count\_if);Необходимо реализовать две шаблонных функции:

1.   Функция **print** печати фигур на экран std::cout  (печататься должны координаты вершин фигур). Функция должна принимать на вход std::tuple с фигурами, согласно варианту задания (минимум по одной каждого класса).

2.   Функция **square** вычисления суммарной площади фигур. Функция должна принимать на вход фигуру, согласно варианту задания - треугольник.

Коллекция должна содержать метод доступа:

доступ к элементу по оператору [ ];

При реализации шаблонных функций допускается использование вспомогательных шаблонов std::count\_if, std::shared\_ptr, std::for\_each.

Вариант задания №25:

|  |  |  |
| --- | --- | --- |
| 25.   25 | Треугольник | Динамический массив |

1. **Описание программы**

Программа принимает данные о фигурах, а именно, координаты двух вершин вашей фигуры в направлении против часовой стрелки (абсциссу и ординату первой, второй и т. д.). Затем программа ожидает ввода цифры (справка о вводе появится на экране), соответствующей своей команде. После чего выполняет нужное действие.

1. **Набор тестов**

Демонстрирует как вводить значения для создания фигур.

For file test\_01.txt:

1

1

2

3

4

1

54

23

7

8

1

0

0

-200

100

1

-3

-2

-200

100

3

4

100

0

For file test\_02.txt:

1

1

25

3

44

1

56

65

7

81

1

8

1

100

100

3

4

5

0

For file test\_03.txt:

1

1

2

3

4

1

5

6

7

8

1

0

0

100

100

3

4

100

0

1. **Результаты выполнения тестов**

root@g8-RM:/mnt/c/Users/g7/Desktop/коды/2sem/oop/5laboop# ./p < test\_01.txt

Для выполнения операции введите соответствующую цифру:

1 - Добавить фигуру

2 - Удалить фигуру

3 - Вывести все фигуры на экран

4 - Вывести все фигуры, у которых площадь меньше, заданной, на экран

0 - Выйти из программы

Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки.

Элемент вставлен.

Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки.

Элемент вставлен.

Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки.

Элемент вставлен.

Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки.

Элемент вставлен.

Фигура №1

Вершина 1 x = 1 y = 2

Вершина 2 x = 3 y = 4

Вершина 3 x = 0.268728 y = 4.73495

Фигура №2

Вершина 1 x = 54 y = 23

Вершина 2 x = 7 y = 8

Вершина 3 x = 43.4551 y = -25.2419

Фигура №3

Вершина 1 x = 0 y = 0

Вершина 2 x = -200 y = 100

Вершина 3 x = -186.84 y = -123.219

Фигура №4

Вершина 1 x = -3 y = -2

Вершина 2 x = -200 y = 100

Вершина 3 x = -190.07 y = -121.618

Введите площадь больше площадей только тех фигур , которые надо вывести

1

Выход из программы...

root@g8-RM:/mnt/c/Users/g7/Desktop/коды/2sem/oop/5laboop# ./p < test\_02.txt

Для выполнения операции введите соответствующую цифру:

1 - Добавить фигуру

2 - Удалить фигуру

3 - Вывести все фигуры на экран

4 - Вывести все фигуры, у которых площадь меньше, заданной, на экран

0 - Выйти из программы

Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки.

Элемент вставлен.

Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки.

Элемент вставлен.

Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки.

Элемент вставлен.

Фигура №1

Вершина 1 x = 1 y = 25

Вершина 2 x = 3 y = 44

Вершина 3 x = -14.4627 y = 36.2506

Фигура №2

Вершина 1 x = 56 y = 65

Вершина 2 x = 7 y = 81

Вершина 3 x = 17.59 y = 30.5535

Фигура №3

Вершина 1 x = 8 y = 1

Вершина 2 x = 100 y = 100

Вершина 3 x = -31.7044 y = 130.314

Введите площадь больше площадей только тех фигур , которые надо вывести

0

Выход из программы...

root@g8-RM:/mnt/c/Users/g7/Desktop/коды/2sem/oop/5laboop# ./p < test\_03.txt

Для выполнения операции введите соответствующую цифру:

1 - Добавить фигуру

2 - Удалить фигуру

3 - Вывести все фигуры на экран

4 - Вывести все фигуры, у которых площадь меньше, заданной, на экран

0 - Выйти из программы

Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки.

Элемент вставлен.

Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки.

Элемент вставлен.

Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки.

Элемент вставлен.

Фигура №1

Вершина 1 x = 1 y = 2

Вершина 2 x = 3 y = 4

Вершина 3 x = 0.268728 y = 4.73495

Фигура №2

Вершина 1 x = 5 y = 6

Вершина 2 x = 7 y = 8

Вершина 3 x = 4.26873 y = 8.73495

Фигура №3

Вершина 1 x = 0 y = 0

Вершина 2 x = 100 y = 100

Вершина 3 x = -36.5636 y = 136.748

Введите площадь больше площадей только тех фигур , которые надо вывести

2

Выход из программы...

1. **Листинг программы**

For file main.cpp:

#include <iostream>

#include <iterator>

#include <algorithm>

#include <memory>

#include <cmath>

using namespace std;

typedef double TP;

const int VECTOR\_EXTRA\_SIZE = 5;

template <typename T>

struct Figure

{

private:

int oir;

};

template <typename T>

struct Triangle : public Figure<T>

{

private:

bool MakeTr(TP x, TP y, TP x1, TP y1)

{

double angle = 3.14 / 180.0 \* (360 / 3);

for (int i = 0; i < 3; i++)

{

this->a[i].first = x;

this->a[i].second = y;

T nextX = ((x1 - x) \* cos(angle) + (y1 - y) \* -sin(angle)) + x1;

T nextY = ((x1 - x) \* sin(angle) + (y1 - y) \* cos(angle)) + y1;

x = x1;

y = y1;

x1 = nextX;

y1 = nextY;

}

}

public:

int count = 3;

std::pair<T, T> a[3];

Triangle()

{

this->MakeTr(0, 0, 1, 1);

}

Triangle(T x, T y, T x1, T y1)

{

this->MakeTr(x, y, x1, y1);

}

Triangle(Triangle<T> &tr)

{

this->a[0] = tr.a[0];

this->a[1] = tr.a[1];

this->a[2] = tr.a[2];

}

void Print()

{

for (int i = 0; i < 3; i++)

{

cout << "Вершина " << i + 1 << " x = " << this->a[i].first << " y = " << this->a[i].second << endl;

}

}

double Square()

{

double s, a, h;

a = sqrt((double)((this->a[0].first - this->a[1].first) \* (this->a[0].first - this->a[1].first) + (this->a[0].second - this->a[1].second) \* (this->a[0].second - this->a[1].second)));

//a = sqrt((x1-x2)^2+(y1-y2)^2)

h = a \* sin(3.14 / 3);

s = (a \* h) / 2;

return s;

}

void operator=(Triangle<TP> tr)

{

for (int i = 0; i < 3; i++)

{

this->a[i].first = tr.a[i].first;

this->a[i].second = tr.a[i].second;

}

}

~Triangle() {}

};

template <class T>

class vector

{

public:

int size;

int capacity;

T \*data;

vector()

{

this->size = 0;

this->capacity = 3;

this->data = new T[3];

}

vector(const vector<T> &v)

{

this->size = v.size;

this->capacity = v.capacity;

this->data = new T[v.capacity];

for (int i = 0; i < this->size; i++)

{

this->data[i] = v.data[i];

}

}

vector(int x)

{

this->size = 0;

this->capacity = x;

this->data = new T[x];

}

void PushEnd(T x)

{

if (this->size >= this->capacity)

{

int oldcap = this->capacity;

this->capacity += VECTOR\_EXTRA\_SIZE;

T \*buf = new T[this->capacity];

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

{

buf[i] = this->data[i];

}

delete[] this->data;

this->data = buf;

buf = 0;

}

this->data[this->size] = x;

this->size++;

}

void PopEnd()

{

if (this->size <= 0)

{

printf("Vector is empty.\n");

}

else

{

this->size--;

}

}

void Erase(int i)

{

if (this->size <= 0)

{

printf("Vector is empty.\n");

}

else

{

if ((i < this->size) && (i >= 0))

{

this->size--;

for (i; i < this->size; i++)

{

//if(i != )

this->data[i] = this->data[i + 1];

}

}

else

{

printf("В векторе нет элемента под индексом %d\n", i);

}

}

}

void insert(int i, T tr)

{

if ((i <= this->size) && (i >= 0))

{

this->PushEnd(tr);

for (int j = this->size - 1; j > i; j--)

{

this->data[j].Print();

this->data[j] = this->data[j - 1];

}

this->data[i] = tr;

}

}

class iterator

{

public:

int index = -1;

vector<T> \*vect;

//std::iterator\_traits<vector<T>::iterator> difference\_type = int;

using difference\_type = ptrdiff\_t;

using value\_type = T;

using reference = T&;

using pointer = T\*;

using iterator\_category = std::forward\_iterator\_tag;

std::shared\_ptr<T> ptr;

iterator()

{

index = 0;

}

iterator(const iterator &it)

{

this->vect = it.vect;

this->ptr = it.ptr;

this->index = it.index;

}

iterator(int ind, vector<T> \*vect)

{

this->ptr = std::make\_shared<T>((vect->data[ind]));

this->index = ind;

this->vect = vect;

}

iterator operator++()

{

//cout<<this<<" "<<this->index<<" "<<this->vect->size<<endl;

if ((this->index + 1) > this->vect->size)

{

//printf("Iterator cannot be incremented past the end of range.%d %d\n", this->index, this->vect->size);

throw std::out\_of\_range("Iterator cannot be incremented past the end of range.");

return \*this;

}

this->index++;

this->ptr = std::make\_shared<T>(this->vect->data[this->index]);

return \*this;

}

iterator &operator++(int)

{

if ((this->index + 1) > (this->vect->size))

{

//printf("Iterator cannot be incremented past the end of range.\n");

throw std::out\_of\_range("Iterator cannot be incremented past the end of range.");

return \*this;

}

this->index++;

this->ptr = std::make\_shared<T>(this->vect->data[this->index]);

return \*this;

}

iterator operator=(iterator x)

{

//printf("=\n");

this->ptr = x.ptr;

return \*this;

}

bool operator!=(const iterator &x)

{//printf("!=\n");

if (index != x.index)

return true;

//printf("false\n ");

/\* if(ptr != x.ptr) return true;

printf("\t !=\n "); \*/

return false;

}

bool operator>=(iterator x)

{

return this->ptr >= x.ptr;

}

int operator-(iterator x)

{

//printf("-\n");

return this->index - x.index;

}

bool operator==(const iterator &x)

{

//printf("\t ==\n ");

if (index == x.index)

return true;

if (ptr == x.ptr)

return true;

return false;

}

/\* bool operator==(iterator x){

return this->ptr == x.ptr;

} \*/

/\* bool operator>(iterator x){

return this->ptr > x.ptr;

}

bool operator<(iterator x){

return this->ptr < x.ptr;

} \*/

T operator\*()

{

return \*(this->ptr);

}

T \*operator->()

{

return this->ptr;

}

~iterator() {}

};

/\* struct iterator {

using value\_type = T;

using difference\_type = int;

using pointer = T\*;

using const\_pointer = const T\*;

using reference = T&;

using const\_reference = const T&;

using size\_type = std::size\_t;

bool compatible(self\_type const &other) const {

return container == other.container;

}

iterator(value\_type \*ptr, container\_type \*container, size\_type index) : container(container), index(index) {

this->ptr = ptr;

}

iterator() = default;

iterator(self\_type const &o) = default;

iterator &operator=(self\_type const &o) = default;

~iterator() = default;

self\_type &operator++() {

if (++index > container->size()) {

throw std::out\_of\_range("Iterator cannot be incremented past the end of range.");

}

return \*this;

}

self\_type operator++(vector<T>::iterator it) {

self\_type tmp = \*this;

++\*this;

return tmp;

}

bool operator==(self\_type const &other) const {

assert(compatible(other));

return index == other.index;

}

bool operator!=(self\_type const &other) const {

return !(\*this == other);

}

reference operator\*() const {

if (ptr == nullptr) {

throw std::bad\_function\_call();

}

return \*(ptr + index);

}

reference operator->() const {

if (ptr == nullptr) {

throw std::bad\_function\_call();

}

return \*(ptr->get() + index);

}

value\_type \*ptr = nullptr;

container\_type \*container = nullptr;

size\_type index = 0;

}; \*/

iterator begin()

{

vector<T>::iterator it(0, this);

return it;

}

iterator end()

{

vector<T>::iterator it(this->size, this);

return it;

}

void Printing()

{

printf(">>>>>>>>\n");

for (int i = 0; i < this->size; i++)

{

cout << this->data[i] << " ";

}

cout << endl;

printf(">>>>>>>>size %d | cap %d\n", this->size, this->capacity);

}

void Show()

{

printf(">>>>>>>>\n");

for (int i = 0; i < this->capacity; i++)

{

cout << this->data[i] << " ";

}

cout << endl;

printf(">>>>>>>>size %d | cap %d\n", this->size, this->capacity);

}

~vector()

{

this->size = 0;

this->capacity = 0;

delete[] this->data;

}

T operator[](int i)

{

return this->data[i];

}

T \*operator++(int x)

{

return this->data[x + 1];

}

T \*operator--(int x)

{

return this->data[x - 1];

}

void operator=(vector<T> vec)

{

for (int i = 0; i < this->size; i++)

{

this->data[i] = vec.data[i];

}

}

};

void help()

{

printf("Для выполнения операции введите соответствующую цифру:\n ");

printf("\t 1 - Добавить фигуру\n ");

printf("\t 2 - Удалить фигуру\n ");

printf("\t 3 - Вывести все фигуры на экран\n ");

printf("\t 4 - Вывести все фигуры, у которых площадь меньше, заданной, на экран\n ");

printf("\t 0 - Выйти из программы\n ");

}

/\* void help()

void help()

{

printf("Для выполнения операции введите соответствующую цифру:\n ");

printf("\t 1 - Добавить фигуру в хранилище\n ");

printf("\t 2 - Удалить последнюю добавленную фигуру из хранилища\n ");

printf("\t 3 - Вывести Геом.Центр. для всех фигур\n ");

printf("\t 4 - Вывести Координаты всех вершин для всех фигур\n ");

printf("\t 5 - Вывести площади всех фигур\n ");

printf("\t 6 - Вычислить общую площадь всех фигур\n ");

printf("\t 7 - Удалить фигуру по индексу\n ");

printf("\t 8 - Закрыть программу\n ");

printf("\t 9 - Показать вектор\n ");

printf("\t 0 - Вывести справку\n ");

} \*/

TP Print(vector<Triangle<TP>> vec)

{

int j = 1;

cout << "Фигура номер " << j << endl;

for (int i = 0; i < 3; i++)

{

cout << "x " << vec.data[j].a[i].first << endl;

cout << "y " << vec.data[j].a[i].second << endl;

return 0;

}

}

int main()

{

int i = 1;

char enter = -1;

TP xl, yl, xr, yr;

int c;

vector<Triangle<TP>> vec;

help();

std::cin >> enter;

while (isdigit(enter))

{

switch (enter - 48)

{

case 1:

{

cout << "Введите координаты двух вершин вашей фигуры в направлении против часовой стрелки." << endl;

std::cin >> xl >> yl >> xr >> yr;

//Triangle<TP> tr(xl, yl, xr, yr);

Triangle<TP> \*ptr = new Triangle<TP>(xl, yl, xr, yr);

//\*ptr = tr;

vec.insert(vec.size, \*ptr);

//vec.PushEnd(\*ptr);

ptr = NULL;

delete ptr;

cout << "Элемент вставлен." << endl;

break;

}

case 2:

{

cout << "Введите номер удаляемой фигуры " << endl;

int numb;

std::cin >> numb;

vec.Erase(numb - 1);

cout << "Фигура удалена\n Текущий размер хранилища :Size = " << vec.size << endl;

break;

}

case 3:

{ //вывод всех фигур

for\_each(vec.begin(), vec.end(), [&](Triangle<TP> &&tr) {

printf("Фигура №%d\n", i);

i++;

tr.Print();

});

i = 1;

break;

}

case 4:

{

TP square = 0;

printf("Введите площадь больше площадей только тех фигур , которые надо вывести\n");

std::cin >> square;

//вывод если прлощадь меньше заданной

int r = std::count\_if(vec.begin(), vec.end(), [&](Triangle<TP> &&tr) { return tr.Square() < square; });

cout << r << endl;

break;

}

case 0:

{

cout << "Выход из программы..." << endl;

return 0;

}

default:

{

help();

break;

}

}

enter = 0;

std::cin >> enter;

}

cout << "Похоже введена буква, ну в другой раз не промахнётесь по клавиатуре ;-) \n\n"

<< endl;

return 0;

}

1. **Вывод**

В процессе выполнения лабораторной работы были закреплены навыки создания структур, работы с ними, работы с шаблонами, а также приобретены навыки в области работы с итераторами и стандартной библиотекой алгоритмов С++ .

**Список литературы**

1.Бьярне Страуструп. Программирование: принципы и практика использования C++, исправленное издание = Programming: Principles and Practice Using C++. — М.: «Вильямс», 2011. — С. 1248. — ISBN 978-5-8459-1705-8.

2.Канал #SimpleCode YouTube https://www.youtube.com/watch?v=O7ruEWCa7zc

3. Сайт stackoverflow.com Итераторы-и-перегрузка-операторов

<https://ru.stackoverflow.com/questions/514459/>Итераторы-и-перегрузка-операторов

4. Сайт cppreference.com

https://en.cppreference.com/w/cpp/iterator/iterator\_traits