МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №2

по курсу объектно-ориентированное программирование 3 семестр, 2021/22 уч. Год

Студент Абросимов Алексей Дмитриевич, группа М8О-207Б-20
Преподаватель Дорохов Евгений Павлович

Условие

Задание:Вариант 3: Прямоугольник

Задание Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий одну фигуру (колонка фигура 1), согласно вариантам задания. Классы должны удовлетворять следующим правилам:

Требования к классу фигуры аналогичны требованиям из лабораторной работы №1.

Классы фигур должны содержать набор следующих методов:

◦ Перегруженный оператор ввода координат вершин фигуры из потока std::istream (>>); Он должен заменить конструктор, принимающий координаты вершин из стандартного потока;

◦ Перегруженный оператор вывода в поток std::ostream (<<), заменяющий метод Print из лабораторной работы 1;

◦ Оператор копирования (=); ◦ Оператор сравнения с такими же фигурами (==).

 Класс-контейнер должен содержать объекты фигур “по значению” (не по ссылке);

Описание программы

Исходный код лежит в 6 файликах:

1.main.cpp — основная программа, направленная на взамодействие с пользователем.

2.rectangle.h — описание класса прямоугольник, который наследуется от фигуры.

3.rectangle.cpp — описание методов прямоугольника

4.tvector.h — описание класса вектора

5.tvector.cpp — описание методов вектора

6. figure.h — описание абстрактного класса фигуры

Дневник отладки

В процессе выполнения данной ЛР столкнулся с проблемой, что при добавлении фигуры в конец вектора программа аварийно завершалась. Оказалось, это было связано с неправильным конструктором: под динамический массив память не выделялась вообще. Во время добавления фигуры в конец вектора, создавался (если это необходимо) новый указатель , который указывал на область память с кол-вом фигур больше на единицу, фигуры переносились с старой области памяти и старая область памяти очищалась. Здесь и была проблема: при неинициализированной памяти нельзя очищать пустой указатель. Всё починилось при добавлении одной строчки кода в конструктор.

Результат вывода программы:

```
Rectangle #1coords is Rectangle coords (1,5) (2,6) (3,7) (4,1)
Rectangle #2coords is Rectangle coords (1,5) (2,6) (3,7) (4,2)
Rectangle #3coords is Rectangle coords (1,5) (2,6) (3,7) (4,3)
Rectangle #4coords is Rectangle coords (1,5) (2,6) (3,7) (4,4)
4
Rectangle coords (1,5) (2,6) (3,7) (4,4)

Rectangle #1coords is Rectangle coords (1,5) (2,6) (3,7) (4,1)
Rectangle #2coords is Rectangle coords (1,5) (2,6) (3,7) (4,3)
```

Недочёты

Выводы
Лабораторная работа укрепила мои навыки работы с классами. Мне показалось очень интересным и полезным написание класса динамического массива. Так же, я закрепил тему перегрузки операторов.

Ссылка на гитхаб: https://github.com/yungalexxxey/oop_labs/tree/main/lab2

Исходный код
figure.h
```
#ifndef FIGURE_H
#define FIGURE_H
#include <iostream>
class figure {
public:
virtual double Area()=0;
virtual size_t VertexesNumber()=0;
virtual bool isit()=0;
};
#endif // FIGURE_H
```
main.cpp
```
#include <iostream>
//#include "rectangle.cpp"
#include "figure.h"
#include "tvector.h"
int main()
{
TVector container;
Rectangle rec1(1,2,3,4,5,6,7,1);
Rectangle rec2(1,2,3,4,5,6,7,2);
Rectangle rec3(1,2,3,4,5,6,7,3);
Rectangle rec4(1,2,3,4,5,6,7,4);
container.InsertLast(rec1);
container.InsertLast(rec2);
container.InsertLast(rec3);
container.InsertLast(rec4);
std::cout<<container;
std::cout<< container.Length()<<std::endl;
container.Remove(2);
std::cout<<container.Last()<<std::endl;
std::cout<<container;
return 0;
}
```
rectangle.cpp
```
#include "rectangle.h"
#include <math.h>

Rectangle::Rectangle():x1(0),y1(0),x2(0),y2(0),x3(0),y3(0),x4(0),y4(0){

}
Rectangle::Rectangle(int x1,int x2,int x3,int x4,int y1,int y2,int y3,int y4){
this->x1=x1;
this->x2=x2;
this->x3=x3;
this->x4=x4;
this->y1=y1;
this->y2=y2;
this->y3=y3;
this->y4=y4;
};
size_t Rectangle::VertexesNumber(){
return 4;
```

```
}

bool Rectangle::isit(){
double perp;
double perp2;
perp=(x4-x1)*(x2-x1)+(y4-y1)*(y2-y1);
perp2=(x3-x4)*(x3-x2)+(y3-y4)*(y3-y2);
if((perp+perp2)==0) return true;
else return false;
}
double Rectangle::Area(){
double r1 = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
double r2 = sqrt((x2 - x3) * (x2 - x3) + (y2 - y3) * (y2 - y3));
double r3 = sqrt((x1 - x3) * (x1 - x3) + (y1 - y3) * (y1 - y3));
double p=(r1+r2+r3)/2;
double s= 2*sqrt((p * (p - r1) * (p - r2) * (p - r3)));
return s;
}

std::ostream& operator<<(std::ostream &out, const Rectangle &rec){
out << "Rectangle coords " <<"("<< rec.x1 << "," << rec.y1 << ")"<< " " <<"("<< rec.x2 << "," <<
rec.y2 << ")"<< " "<< "("<< rec.x3 << "," << rec.y3 << ")"<< " " << "("<<rec.x4 << "," << rec.y4 <<
")"<< std::endl;
return out;
}
std::istream& operator>>(std::istream &in,Rectangle &rec){
in >> rec.x1;
in >> rec.y1;
in >> rec.x2;
in >> rec.y2;
in >> rec.x3;
in >> rec.y3;
in >> rec.x4;
in >> rec.y4;
return in;
}
Rectangle& Rectangle::operator= (Rectangle &rec){
this->x1=rec.x1;
this->x2=rec.x2;
this->x3=rec.x3;
this->x4=rec.x4;
this->y1=rec.y1;
this->y2=rec.y2;
this->y3=rec.y3;
this->y4=rec.y4;
return *this;
}
```

rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H
#include "figure.h"
#include <iostream>

class Rectangle:public figure{
public:
Rectangle();
Rectangle(int x1,int x2,int x3,int x4,int y1,int y2,int y3,int y4);
size_t VertexesNumber();
double Area();
bool isit();
friend std::ostream& operator<<(std::ostream &out, const Rectangle &rec);
friend std::istream& operator>>(std::istream &in, Rectangle &rec);
Rectangle& operator= (Rectangle &rec1);
```

```
private:
double x1;
double y1;
double x2;
double y2;
double x3;
double y3;
double x4;
double y4;
};

#endif // RECTANGLE_H
```

tvector.cpp

```
#include "tvector.h"
#include "rectangle.h"
TVector::TVector() : length(1), count(0)
{
arr=new Rectangle[1];
}
TVector::~TVector()
{
delete[] arr;
}
int TVector::Length()
{
return this->length;
}
bool TVector::Empty()
{
return !count==0;
}

void TVector::Resize(int newlength)
{
std::cout << length << " " << count << std::endl;
if (newlength == length)
return;
if (newlength > length)
{
Rectangle *narr = new Rectangle[newlength];
for (int i = 0; i < length; i++)
narr[i] = arr[i];
delete[] arr;
arr = narr;
length = newlength;
}
else
{
Rectangle *narr = new Rectangle[newlength];
for (int i = 0; i < newlength; i++)
narr[i] = arr[i];
delete[] arr;
arr = narr;
count = newlength;
}
std::cout << length << " " << count << std::endl;
}
void TVector::InsertLast(Rectangle &newrec)
{
if (count == length)
{
length++;
```

```
count++;
Rectangle *narr = new Rectangle[length];
for (int i = 0; i < length - 1; i++)
narr[i] = arr[i];
narr[length - 1] = newrec;
delete[] arr;
arr = narr;
}
else if (count < length)
{
arr[count] = newrec;
count++;
}
}

void TVector::Clear()
{
delete[] arr;
length = 0;
count = 0;
}
const Rectangle TVector::Last()
{
Rectangle *narr = new Rectangle[length];
for (int i = 0; i < count - 1; i++)
{
narr[i] = arr[i];
}
Rectangle tmp = arr[count - 1];
count--;
length--;
delete [] arr;
arr = narr;
return tmp;
}

void TVector::Remove(int pos)
{
if (count == 0)
{
std::cout << "Container is empty" << std::endl;
return;
}
Rectangle *narr = new Rectangle[length - 1];
int current_index = 0;
for (int i = 0; i < count; i++)
{
if (i != pos - 1)
{
narr[current_index] = arr[i];
current_index++;
}
}
count--;
length--;
delete[] arr;
arr = narr;
}

//перегрузка операций
Rectangle &TVector::operator[](int i)
{
if (i >= 0 && i < this->length)
```

```
return this->arr[i];
else exit;
}

std::ostream &operator<<(std::ostream &out, TVector &cont)
{
for (int i = 0; i < cont.count; i++)
{
out << "Rectangle #" << i + 1 << "coords is " << cont[i];
}
return out;
}
```

tvector.h

```
#ifndef TVECTOR_H
#define TVECTOR_H
#include "rectangle.h"

class TVector
{
private:
int length;
int count;
Rectangle *arr;
public:
TVector();
~TVector();
TVector(const TVector& other);
void InsertLast(Rectangle &newrec);
const Rectangle Last();
int Length();
bool Empty();
void Resize(int nindex);
void Remove(int pos);
void Clear();

Rectangle& operator[] (int i) ;
friend std::ostream& operator<<(std::ostream &out, TVector &cont);
};

#endif // TVECTOR_H
```