БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

ИИТ БГУИР

Программное обеспечение информационных технологий

Отчет

По лабораторной работе №2

«Классы и Объекты. Инкапсуляция.»

по курсу «ООТПиСП»

Выполнил студент группы 381064 Сорока А.А.

Проверил преподаватель: Лашкевич Е.М.

Минск 2015 г

**Цель работы** – познакомиться с созданием классов и объектов на языке C++. Изучить основные конструкции, используемые при построении классов и объектов. Разобраться с понятием инкапсуляции.

**Задание А4.** Состав класса **Person** (индивидуум): фамилия, возраст, пол. Подсчитать средний возраст и количество мужчин.

**Код программы:**

Main.h:

#include <iostream>

#include <string.h>

using namespace std;

class Person

{

public:

Person();

~Person();

string fio;

int age;

string gender;

private:

};

int ManCout(Person\* persons, int size);

double AverageAge(Person\* persons, int size);

Main.cpp:

#include "main.h"

int main(int argc, char \*argv[])

{

int size = 5;

Person \*persons = new Person[size];

persons[0].age = 20;

persons[0].fio = "Ivanov";

persons[0].gender = "male";

persons[1].age = 10;

persons[1].fio = "Ivanova";

persons[1].gender = "female";

persons[2].age = 30;

persons[2].fio = "Petrov";

persons[2].gender = "male";

persons[3].age = 40;

persons[3].fio = "Sidorova";

persons[3].gender = "female";

persons[4].age = 25;

persons[4].fio = "Genadiev";

persons[4].gender = "male";

cout << "man count: " << ManCout(persons, size) << endl;

cout << "average age: " << AverageAge(persons, size) << endl;

system("pause");

return 0;

}

double AverageAge(Person\* persons, int size)

{

double temp = 0;

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

{

temp += persons[i].age;

}

return temp / 5;

}

int ManCout(Person\* persons, int size)

{

int temp = 0;

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

{

if (persons[i].gender == "male") temp++;

}

return temp;

}

Person::Person()

{

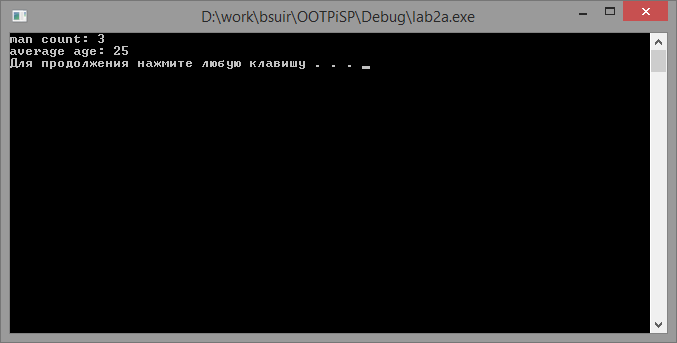
}

Person::~Person()

{

}

**Скриншот выполнения программы:**



**Задание В4.**Составить описание класса прямоугольников со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольников на плоскости, изменение размеров, построение наименьшего прямоугольника, содержащего два заданных прямоугольника, и прямоугольника, являющегося общей частью (пересечением) двух прямоугольников.

**Код программы:**

Main.h:

#include <iostream>

#include <string.h>

using namespace std;

struct Cord

{

int x;

int y;

};

struct Rectangle

{

Cord leftbottom;

Cord righttop;

};

class Rectangles

{

public:

Rectangles();

Rectangles(Rectangle rect1, Rectangle rect2);

~Rectangles();

void Move(int number, int moveX, int moveY); //двигаем указанный по номеру прямоугольник на х или у

void Resize(int number, int x1, int y1, int x2, int y2); //задаем новые координаты для выбранног опрямоугольника

Rectangle IntersectionRectangle(); //находим пересечение прямоугольников

Rectangle UnionRectangle(); //находим наименьший, что включает оба

void ViewRectangles(); //просмотр текущих прямоугольников

private:

Rectangle rectangles[2];

void SwapRectangles(); //меняем прямоугольники, что бы первый был левее

};

Main.cpp:

#include "main.h"

int main(int argc, char \*argv[])

{

bool ok = true;

Rectangles rect;

Rectangle temp;

while (ok)

{

int number, x, y, x1, x2, y1 ,y2;

system("cls");

cout << "1 - View rectangles" << endl;

cout << "2 - Move rectangle" << endl;

cout << "3 - Resize rectangle" << endl;

cout << "4 - Union Rectangle" << endl;

cout << "5 - Intersection Rectangle" << endl;

cout << "6 - Exit" << endl;

char a;

cin.get(a);

switch (a)

{

case '1' :

rect.ViewRectangles();

system("pause");

break;

case '2':

cout << "input number of changed rectabgle (1,2)" << endl;

cin >> number;

number--;

cout << "input x (+/-)" << endl;

cin >> x;

cout << "input y (+/-)" << endl;

cin >> y;

rect.Move(number, x, y);

system("pause");

break;

case '3':

cout << "input number of changed rectabgle (1,2)" << endl;

cin >> number;

number--;

cout << "input new leftbottom x and y" << endl;

cin >> x1 >> y1;

cout << "input new righttop x and y" << endl;

cin >> x2 >> y2;

rect.Resize(number, x1, y1, x2, y2);

system("pause");

break;

case '4':

cout << "Union Rectangle coord is: " << endl;

temp = rect.UnionRectangle();

cout << "leftbottom " << temp.leftbottom.x << " " << temp.leftbottom.y << endl;

cout << "rightbottom " << temp.righttop.x << " " << temp.righttop.y << endl;

system("pause");

break;

case '5':

cout << "Intersection Rectangle coord is: " << endl;

temp = rect.IntersectionRectangle();

if (temp.leftbottom.x == 0 && temp.leftbottom.y == 0 && temp.righttop.y == 0 && temp.righttop.x == 0)

{

cout << "Rectangles are not intersecting" << endl;

}

else

{

cout << "leftbottom " << temp.leftbottom.x << " " << temp.leftbottom.y << endl;

cout << "rightbottom " << temp.righttop.x << " " << temp.righttop.y << endl;

}

system("pause");

break;

case '6':

ok = false;

break;

}

}

return 0;

}

Rectangles::Rectangles()

{

rectangles[0].leftbottom.x = 1;

rectangles[0].leftbottom.y = 1;

rectangles[0].righttop.x = 5;

rectangles[0].righttop.y = 5;

rectangles[1].leftbottom.x = 3;

rectangles[1].leftbottom.y = 3;

rectangles[1].righttop.x = 7;

rectangles[1].righttop.y = 7;

}

Rectangles::~Rectangles()

{

}

void Rectangles::SwapRectangles()

{

if (rectangles[0].leftbottom.x > rectangles[1].leftbottom.x)

{

Rectangle temp = rectangles[1];

rectangles[1] = rectangles[0];

rectangles[0] = temp;

}

}

void Rectangles::Move(int number, int moveX, int moveY)

{

rectangles[number].leftbottom.x += moveX;

rectangles[number].leftbottom.y += moveY;

rectangles[number].righttop.x += moveX;

rectangles[number].righttop.y += moveY;

SwapRectangles();

}

void Rectangles::Resize(int number, int x1, int y1, int x2, int y2)

{

rectangles[number].leftbottom.x = x1;

rectangles[number].leftbottom.y = y1;

rectangles[number].righttop.x = x2;

rectangles[number].righttop.y = y2;

SwapRectangles();

}

Rectangle Rectangles::UnionRectangle()

{

Rectangle rect;

rect.leftbottom.y = (rectangles[1].leftbottom.y < rectangles[0].leftbottom.y ? rectangles[1].leftbottom.y : rectangles[0].leftbottom.y);

rect.leftbottom.x = rectangles[0].leftbottom.x;

rect.righttop.y = (rectangles[1].righttop.y > rectangles[0].righttop.y ? rectangles[1].righttop.y : rectangles[0].righttop.y);

rect.righttop.x = (rectangles[1].righttop.x > rectangles[0].righttop.x ? rectangles[1].righttop.x : rectangles[0].righttop.x);

return rect;

}

Rectangle Rectangles::IntersectionRectangle()

{

Rectangle rect;

rect.leftbottom.x = 0;

rect.leftbottom.y = 0;

rect.righttop.x = 0;

rect.righttop.y = 0;

//если прямоугольник не находиться правее

if (rectangles[0].righttop.x > rectangles[1].leftbottom.x)

{

//если прямогуольник выше

if (rectangles[0].righttop.y < rectangles[1].leftbottom.y)

{

return rect;

}

//если прямоугольник ниже

if (rectangles[0].leftbottom.y > rectangles[1].righttop.y)

{

return rect;

}

//х нижненго левог оугла всегда совпдает с х правого прямоугольника

rect.leftbottom.x = rectangles[1].leftbottom.x;

//находим самый высокий уровень по у для низ лево

rect.leftbottom.y = (rectangles[0].leftbottom.y > rectangles[1].leftbottom.y ? rectangles[0].leftbottom.y : rectangles[1].leftbottom.y);

//находим самый низкий уровень по у для верх право

rect.righttop.y = (rectangles[0].righttop.y < rectangles[1].righttop.y ? rectangles[0].righttop.y : rectangles[1].righttop.y);

//находим самый vfkymrbq уровень по х

rect.righttop.x = (rectangles[0].righttop.x < rectangles[1].righttop.x ? rectangles[0].righttop.x : rectangles[1].righttop.x);

}

return rect;

}

void Rectangles::ViewRectangles()

{

cout << "rectangle 1:" << endl;

cout << "leftbottom " << rectangles[0].leftbottom.x << " " << rectangles[0].leftbottom.y << endl;

cout << "righttop " << rectangles[0].righttop.x << " " << rectangles[0].righttop.y << endl << endl;

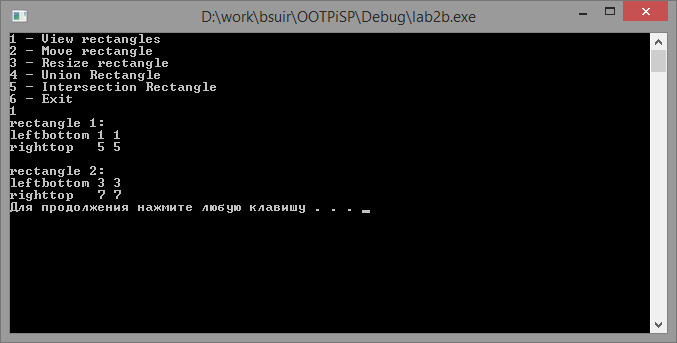
cout << "rectangle 2:" << endl;

cout << "leftbottom " << rectangles[1].leftbottom.x << " " << rectangles[1].leftbottom.y << endl;

cout << "righttop " << rectangles[1].righttop.x << " " << rectangles[1].righttop.y << endl;

}

**Скриншот выполнения программы:**



**Вывод**: В ходе выполнения лабораторной работы я познакомился с созданием классов и объектов на языке C++. Изучил основные конструкции, используемые при построении классов и объектов. Разобрался с понятием инкапсуляции