Пензенский государственный технологический университет

Факультет автоматизированных информационных технологий

Кафедра «Прикладная информатика»

ЕН.Ф.02

**СОЗДАНИЕ КЛАССА «Змейки»**

Отчет о лабораторной работе № 7

по дисциплине «Информатика и программирование»

Выполнили: ст-ты гр.17СН1с

Тюняев А.В., Богданов М.О.

Проверил: ст. преп. каф. ПИ

Юранов В.С.

2018

**1 Формулировка задачи**

1. В приложение из 4 лабораторной работы в класс Point необходимо добавить методы для перемещения точки и очистки точки.
2. Добавить новый класс Snake описывающий змейку.
3. Добавить в класс Snake методы для движения змейки в заданном направлении.
4. В основной программе создать экземпляр класса змейки и бесконечный цикл, в котором проверить движение объекта змейки.

**2 Требования к программе**

Программа должна вывести границы экрана и “Змейку”

**3 Описание программы**

**3.1 Общие сведения**

Программа разработана в среде Visual Studio 2017

**3.2 Функциональное назначение**

Программа предназначена для вывода границ экрана и “Змейки”

**3.3 Описание логической структуры**

*Класс Point:*

В классе объявляются три поля, значение оси x, значение оси y, символ который будет выводится.

Конструктор Point ….

Конструктор Point…

Метод Draw служит для вывода символов на экран.

Метод IsHit ….

*Класс Figure:*

Создается список pList для записи координат точек

Метод Draw служит для вывода символов на экран используя список pList.

Метод IsHit …

*Класс VerticalLine:*

Этот класс является наследником класса Figure.

Метод VerticalLine принимает значения смещения по оси y, координату по оси x и символ.

Создается экземпляр класса List<Point> (списка).

В цикл передаются значения начальной и конечной точки по оси y.

В цикле создается экземпляр класса Point и в список добавляются координаты точек по осям x и y, и необходимый символ.

*Класс HorizontalLine:*

Этот класс является наследником класса Figure.

Метод HorizontalLine принимает значения смещения по оси x, координату по оси y и символ.

Создается экземпляр класса List<Point> (списка).

В цикл передаются значения начальной и конечной точки по оси x.

В цикле создается экземпляр класса Point и в список добавляются координаты точек по осям x и y, и необходимый символ.

*Класс Program:*

Метод Main (основной метод) в нем создается экземпляры классов VerticalLine и HorizontalLine, передаются данные в эти классы, вызывается метод Draw, для вывода рисунка.

**3 Описание применения**

После запуска программы выводятся 4 линии, которые образуют границы экрана и выводится перемещающаяся фигура “Змейка”.

**Выводы**

Познакомились со способами анимации в консольном приложении.

**Контрольные вопросы**

1. Для чего используется тип-перечисление?
2. Что такое конструктор класса?
3. Что такое метод класса?
4. В чем отличие модификатором доступа private и public?

ТЕКСТ ПРОГРАММЫ

Приложение А

(обязательное)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab5

{

class Point

{

int x;

int y;

char sym;

public Point()

{

}

public Point(int x, int y, char sym)

{

this.x = x;

this.y = y;

this.sym = sym;

}

public Point(Point p)

{

x = p.x;

y = p.y;

sym = p.sym;

}

public void Draw()

{

Console.SetCursorPosition(x, y);

Console.Write(sym);

}

public bool IsHit(Point p)

{

return p.x == this.x && p.y == this.y;

}

public void Check()

{

}

}

class Figure

{

protected List<Point> pList;

public void Draw()

{

foreach (Point p in pList)

{

p.Draw();

}

}

public bool IsHit(Figure figure)

{

foreach (var p in pList)

{

if (figure.IsHit(p))

return true;

}

return false;

}

private bool IsHit(Point point)

{

foreach (var p in pList)

{

if (p.IsHit(point))

return true;

}

return false;

}

}

class VerticalLine : Figure

{

public VerticalLine(int yUp, int yDown, int x, char sym)

{

pList = new List<Point>();

for (int y = yUp; y <= yDown; y++)

{

Point p = new Point(x, y, sym);

pList.Add(p);

}

}

}

class HorizontalLine : Figure

{

public HorizontalLine(int xLeft, int xRight, int y, char sym)

{

pList = new List<Point>();

for (int x = xLeft; x <= xRight; x++)

{

Point p = new Point(x, y, sym);

pList.Add(p);

}

}

}

class Program

{

static void Main(string[] args)

{

Point p = new Point(62, 2, '\*');

Point p2 = new Point(p);

//Point p2 = new Point(50, 2, '\*');

//p2.Draw();

Console.SetCursorPosition(55, 7);

if ((p.IsHit(p2)) == true)

Console.WriteLine("Точки совпадают");

else

Console.WriteLine("Точки не совпадают");

p.Draw();

p2.Draw();

Point p3 = new Point(80, 2, '\*');

p3.Draw();

Point p4 = new Point(85, 4, '\*');

p4.Draw();

Console.SetCursorPosition(75, 7);

if ((p3.IsHit(p4)) == true)

Console.WriteLine("Точки совпадают");

else

Console.WriteLine("Точки не совпадают");

HorizontalLine h1 = new HorizontalLine(1, 10, 2, '\*');

h1.Draw();

VerticalLine v1 = new VerticalLine(1, 5, 12, '\*');

v1.Draw();

Console.SetCursorPosition(1, 7);

if (h1.IsHit(v1) == true)

Console.WriteLine("Линии пересекаются");

else

Console.WriteLine("Линии не пересекаются");

HorizontalLine h2 = new HorizontalLine(35, 45, 2, '\*');

h2.Draw();

VerticalLine v2 = new VerticalLine(1, 5, 40, '\*');

v2.Draw();

Console.SetCursorPosition(30, 7);

if (h2.IsHit(v2) == true)

Console.WriteLine("Линии пересекаются");

else

Console.WriteLine("Линии не пересекаются");

Console.ReadKey();

}

}

}

РЕЗУЛЬТАТЫ ИСПЫТАНИЙ

Приложение Б

(обязательное)

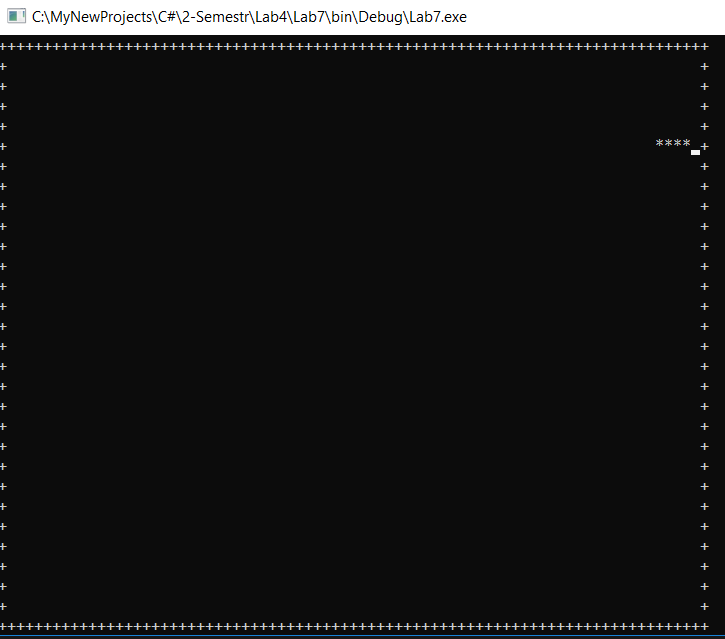


Рисунок 1 – Результат выполнения программы