Пензенский государственный технологический университет

Факультет автоматизированных информационных технологий

Кафедра «Прикладная информатика»

ЕН.Ф.02

**УПРАВЛЕНИЕ ЗМЕЙКОЙ. ОБРАБОТКА НАЖАТИЯ КЛАВИШ**

Отчет о лабораторной работе № 8

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 5 лабораторной работы в класс Snake необходимо добавить метод для изменения направления движения змейки.
2. Добавить на экран точку описывающую еду.
3. Добавить в класс Snake метод для поедания еды.
4. Отладить программу.
5. Описать условия для определения столкновения змейки со стенами и своим хвостом, в результате выполнения которых игра будет проиграна

**2 Требования к программе**

Программа должна вывести границы экрана, управляемую “Змейку”, еду для змейки и реализовать механику поедания змейкой еды.

**3 Описание программы**

**3.1 Общие сведения**

Программа разработана в среде Visual Studio 2017

**3.2 Функциональное назначение**

Программа предназначена для вывода границ экрана, управляемой “Змейки”, еду для змейки.

**3.3 Описание логической структуры**

*Класс Point:*

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

Перегрузка конструктора Point.

Конструктор Point с тремя параметрами.

Конструктор Point с одним параметром.

Метод Draw служит для вывода символов на экран.

Метод IsHit используется для проверки точек на пересечения.

Метод Move используется для изменения положения змейки.

Метод Clear используется для удаления последней точки змейки.

*Класс Figure:*

Создается список pList для записи координат точек

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

Метод IsHit используется для проверки фигур на пересечение.

Метод IsHit используется для проверки точек на пересечения.

*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 Walls:*

Создается список wallList.

В методе Walls

Создаются экземпляры классов HorizontalLine, VerticalLine и передаются параметры.

Происходит добавление прямых в список wallList.

С помощью метода Draw происходит вывод фигур на экран.

*Перечисление Direction:*

Добавляем перечисление направлений.

*Класс Snake:*

Этот класс является наследником класса Figure.

В конструктор класса Snake будут передаваться следующие параметры: Точку с которой будет начинаться хвост, текущую длину змейки, направление движения змейки.

В цикле несколько раз будут создаваться точки являющиеся точной копией хвостовой точки, которая передана в конструкторе, затем эта точка будет сдвинута на заданной число позиций по направлению движения змейки и эта точка будет добавлена в список.

Метод Move.

В переменную tail класса Point записывается первый элемента списка pList с помощью метода First().

Удаление хвостовой точки из списка pList методом Remove().

Вызывается метод GetNextPoint для определения координат перемещения головной точки.

Добавление точки, полученной из метода GetNextPoint, в список pList.

Удаление хвостовой точки змейки из консоли.

Метод GetNextPoint:

Получение текущего положения головы змейки из списка pList с помощью метода Last.

Создается новая точка, которая является копией предыдущего положения головной точки.

Перемещение точки по направлению движения змейки.

Получение точки, которая является новым положением головы змейки.

Метод Handl определяется какая была нажата клавиша и соответственно в какую сторону повернет змейка.

Метод Eat, используется функция GetNextPoint для вычисления положения новой точки головы змейки.

Происходит проверка координат головной точки и “еды”. Если они пересекаются, то точка с “едой” заменяется на элемент змейки и змейка увеличивается в размерах.

*Класс FoodCreator:*

Создается конструктор в который передается ширина, высотка окна и символ.

Создается точка со случайными координатами в заданных рамках с символом “еды”.

*Класс Program:*

Метод Main (основной метод) в бесконечном цикле создается экземпляр класса Walls, и передаются параметры. Вызывается метод Draw класса Walls, для вывода границ экрана.

Создается переменная класса Point в нее передаются координаты начальной точки змейки и символ.

Создается переменная класса Snake и в нее передаются параметры начальной точки, длину змейки и направление движения.

Вызывается метод Draw для отрисовки змейки.

Создается переменная класса FoodCreator и в нее передаются параметры с координатами точки и символом.

Создается переменная класса Point и вызывается метод CreateFood класса FoodCreator.

Вызывается метод Draw для вывода “еды”.

В бесконечном цикле вызывается происходит вызов методов для проверки столкновения змейки со своим телом и с границами экрана.

Если столкновение произошло выводится игра останавливается и выводится меню.

Если столкновения не произошло происходит вызов метода проверки на пересечения головы змейки с едой. Если произошло пересечение, то вызывается метод CreateFood для реализации функции еды. Если не произошло, вызывается метод Move для дальнейшего движения змейки.

Происходит проверка на нажатие клавиши и вызов метода Handl.

**3 Описание применения**

После запуска программы выводятся 4 линии, которые образуют границы экрана и выводится перемещающаяся фигура “Змейка”, которая может поглощать фигуры и увеличиваться в размерах.

**Выводы**

Познакомились со способами обработки нажатия клавиш в консольном приложении.

ТЕКСТ ПРОГРАММЫ

Приложение А

(обязательное)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Snake

{

class Walls

{

List<Figure> wallList;

public Walls(int mapWidhtm, int mapHeight)

{

wallList = new List<Figure>();

HorizontalLine line1 = new HorizontalLine(0, mapWidhtm-2, 0, '+');

HorizontalLine line2 = new HorizontalLine(0, mapWidhtm-2, mapHeight-1, '+');

VerticalLine line3 = new VerticalLine(0, mapHeight-1, 0, '+');

VerticalLine line4 = new VerticalLine(0, mapHeight-1, mapWidhtm-2, '+');

wallList.Add(line1);

wallList.Add(line2);

wallList.Add(line3);

wallList.Add(line4);

}

internal bool IsHit (Figure figure)

{

foreach (var wall in wallList)

{

if (wall.IsHit(figure))

return true;

}

return false;

}

public void Draw()

{

foreach (var wall in wallList)

{

wall.Draw();

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Snake

{

class FoodCreator

{

int mapWidth;

int mapHeigth;

char sym;

Random random = new Random();

public FoodCreator (int mapWidth, int mapHeigth, char sym)

{

this.mapWidth = mapWidth;

this.mapHeigth = mapHeigth;

this.sym = sym;

}

public Point CreateFood()

{

int x = random.Next(2, mapWidth - 2);

int y = random.Next(2, mapHeigth - 2);

return new Point(x, y, sym);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Snake

{

enum Direction

{

LEFT,

RIGHT,

UP,

DOWN

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Snake

{

class Snake : Figure

{

Direction direction;

public Snake(Point tail, int lenght, Direction direction)

{

this.direction = direction;

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

{

Point p = new Point(tail);

p.Move(i, direction);

pList.Add(p);

}

}

internal void Move()

{

Point tail = pList.First();

pList.Remove(tail);

Point head = GetNexPoint();

pList.Add(head);

tail.Clear();

head.Draw();

}

public Point GetNexPoint()

{

Point head = pList.Last();

Point nextPoint = new Point(head);

nextPoint.Move(1, direction);

return nextPoint;

}

internal bool IsHitTail()

{

var head = pList.Last();

for (int i = 0; i < pList.Count - 2; i++)

{

if (head.IsHit(pList[i]))

return true;

}

return false;

}

public void Handl(ConsoleKey key)

{

if (key == ConsoleKey.LeftArrow)

direction = Direction.LEFT;

else if (key == ConsoleKey.RightArrow)

direction = Direction.RIGHT;

else if (key == ConsoleKey.DownArrow)

direction = Direction.DOWN;

else if (key == ConsoleKey.UpArrow)

direction = Direction.UP;

}

internal bool Eat (Point food)

{

Point head = GetNexPoint();

if (head.IsHit(food))

{

food.sym = head.sym;

pList.Add(food);

return true;

}

else

return false;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Snake

{

class Figure

{

protected List<Point> pList = new List<Point>();

public void Draw()

{

foreach (Point p in pList)

{

p.Draw();

}

}

internal 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;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Snake

{

class Program

{

static void Main(string[] args)

{

Console.SetWindowSize(80, 25);

Console.SetBufferSize(80, 25);

Console.CursorVisible = false;

while (true)

{

Walls walls = new Walls(80, 25);

walls.Draw();

Point p = new Point(4, 5, '\*');

Snake snake = new Snake(p, 4, Direction.RIGHT);

snake.Draw();

FoodCreator foodCreator = new FoodCreator(80, 25, '$');

Point food = foodCreator.CreateFood();

food.Draw();

while (true)

{

if (walls.IsHit(snake) || snake.IsHitTail())

{

Menu menu = new Menu();

menu.Mn();

break;

}

if (snake.Eat(food))

{

food = foodCreator.CreateFood();

food.Draw();

}

else

{

snake.Move();

}

Thread.Sleep(100);

if (Console.KeyAvailable)

{

ConsoleKeyInfo key = Console.ReadKey();

snake.Handl(key.Key);

}

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Snake

{

class Menu

{

public void Mn()

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.SetCursorPosition(30, 10);

Console.Write("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.SetCursorPosition(31, 11);

Console.Write("ИГРА ЗАКОНЧЕНА");

Console.SetCursorPosition(30, 12);

Console.Write("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.SetCursorPosition(30, 14);

Console.Write("(Enter) Повторить");

Console.SetCursorPosition(33, 15);

Console.Write("(Esc) Выход");

Console.ForegroundColor = ConsoleColor.White;

ConsoleKeyInfo cki = new ConsoleKeyInfo();

while (true)

{

if (Console.KeyAvailable == true)

{

cki = Console.ReadKey(true);

if (cki.Key == ConsoleKey.Enter)

{

Console.Clear();

break;

}

}

else if (Console.KeyAvailable == true)

{

cki = Console.ReadKey(true);

if (cki.Key == ConsoleKey.Escape)

{

Environment.Exit(0);

}

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Snake

{

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);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Snake

{

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);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Snake

{

class Point

{

public int x;

public int y;

public char sym;

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 Move(int offset, Direction direction)

{

if (direction == Direction.RIGHT)

{

x = x + offset;

}

else if (direction == Direction.LEFT)

{

x = x - offset;

}

else if (direction == Direction.DOWN)

{

y = y + offset;

}

else if (direction == Direction.UP)

{

y = y - offset;

}

}

public bool IsHit (Point p)

{

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

}

public void Draw()

{

Console.SetCursorPosition(x, y);

Console.Write (sym);

}

public void Clear()

{

sym = ' ';

Draw();

}

РЕЗУЛЬТАТЫ ИСПЫТАНИЙ

Приложение Б

(обязательное)

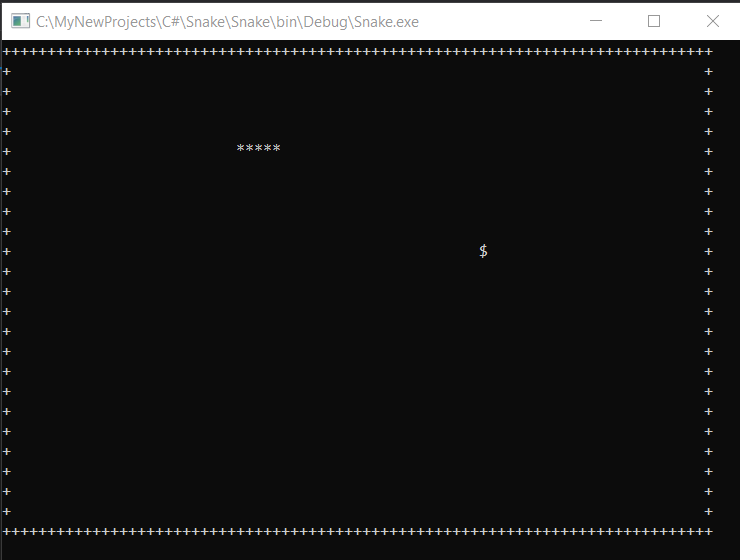


Рисунок 1 – Результат выполнения программы

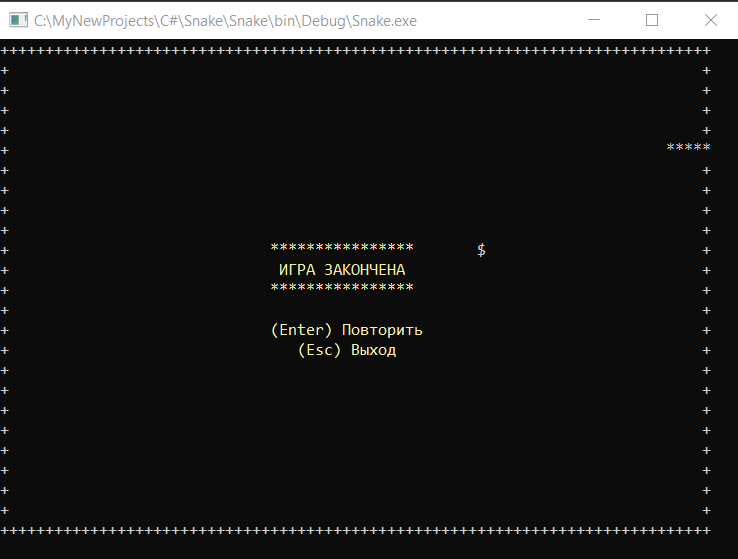


Рисунок 2 – Вывод меню проигрыша