Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1 «Основные конструкции языка Python»

Выполнил:
студент группы ИУ5-32Б
Сайфутдинов Р.И.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

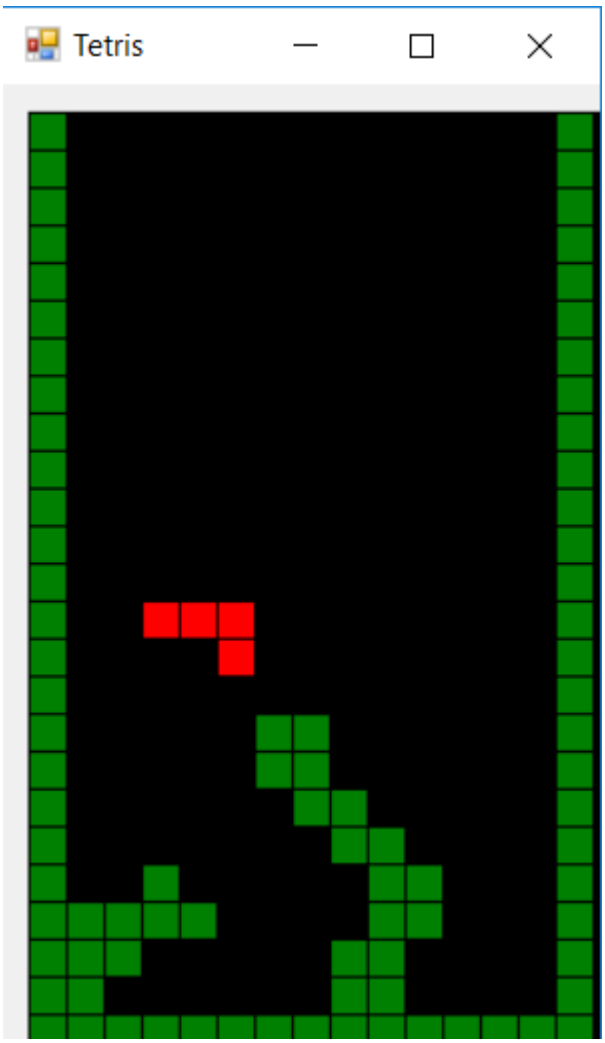
Описание задания

1. Выберите язык программирования (который Вы ранее не изучали) и (1) напишите по нему реферат с примерами кода или (2) реализуйте на нем небольшой проект (с детальным текстовым описанием).
2. Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка или содержать решение какой-либо задачи на этом языке.
3. Необходимо установить на свой компьютер компилятор (интерпретатор, транспилятор) этого языка и произвольную среду разработки.
4. В случае написания реферата необходимо разработать и откомпилировать примеры кода (или модифицировать стандартные примеры).
5. В случае создания проекта необходимо детально комментировать код.
6. При написании реферата (создании проекта) необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.
7. Приветствуется написание черновика статьи по результатам выполнения ДЗ.
8. Черновик статьи может быть подготовлен группой студентов, которые исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

```
using System;
using System.Linq;
using System.Drawing;
using System.Windows.Forms;
namespace LittleTetris{
    public partial class Form1 : Form{
        public const int width = 15, height = 25, k = 15;
        public int[,] shape = new int[2, 4];
        public int[,] field = new int[width, height];
        public Bitmap bitfield = new Bitmap(k * (width + 1) + 1, k * (height + 3)
+ 1);
        public Graphics gr;
        public Form1(){
            InitializeComponent();
            gr = Graphics.FromImage(bitfield);
            for (int i = 0; i < width; i++)
                field[i, height - 1] = 1;
            for (int i = 0; i < height; i++) {
                field[0, i] = 1;
                field[width - 1, i] = 1;
            }
            SetShape();
        }
        public void FillField(){
            gr.Clear(Color.Black);
            for (int i = 0; i < width; i++)
                for (int j = 0; j < height; j++)
                    if (field[i, j] == 1){
                        gr.FillRectangle(Brushes.Green, i * k, j * k, k, k);
                        gr.DrawRectangle(Pens.Black, i * k, j * k, k, k);
                    }
            for (int i = 0; i < 4; i++){
```

```
                gr.FillRectangle(Brushes.Red, shape[1, i] * k, shape[0, i] * k,
k, k);
                gr.DrawRectangle(Pens.Black, shape[1, i] * k, shape[0, i] * k, k,
k);
            }
            FieldPictureBox.Image = bitfield;
        }
        private void TickTimer_Tick(object sender, System.EventArgs e){
            if (field[8, 3] == 1)
                Environment.Exit(0);
            for (int i = 0; i < 4; i++)
                shape[0, i]++;
            for (int i = height - 2; i > 2; i--){
                var cross = (from t in Enumerable.Range(0,
field.GetLength(0)).Select(j => field[j, i]).ToArray() where t == 1 select
t).Count();
                if (cross == width)
                    for (int k = i; k > 1; k--)
                        for (int l = 1; l < width - 1; l++)
                            field[l, k] = field[l, k - 1];}
            if (FindMistake()){
                for (int i = 0; i < 4; i++)
                    field[shape[1, i], --shape[0, i]]++;
                SetShape();}
            FillField();
        }
        private void Form1_KeyDown(object sender, KeyEventArgs e){
           switch (e.KeyCode){
            case Keys.A:
                for (int i = 0; i < 4; i++)
                    shape[1, i]--;
                if (FindMistake())
                    for (int i = 0; i < 4; i++)
                            shape[1, i]++;
                break;
            case Keys.D:
                for (int i = 0; i < 4; i++)
                        shape[1, i]++;
                if (FindMistake())
                        for (int i = 0; i < 4; i++)
                            shape[1, i]--;
                break;
            case Keys.W:
                var shapeT = new int[2, 4];
                Array.Copy(shape, shapeT, shape.Length);
                int maxx = 0, maxy = 0;
                for (int i = 0; i < 4; i++){
                    if (shape[0, i] > maxy)
                            maxy = shape[0, i];
                    if (shape[1, i] > maxx)
                            maxx = shape[1, i];
```

```
                }
                for (int i = 0; i < 4; i++) {
                    int temp = shape[0, i];
                    shape[0, i] = maxy - (maxx - shape[1, i]) - 1;
                    shape[1, i] = maxx - (3 - (maxy - temp)) + 1;
                }
                if (FindMistake())
                    Array.Copy(shapeT, shape, shape.Length);
                break;
            }
        }
        public void SetShape(){
            Random x = new Random(DateTime.Now.Millisecond);
            switch (x.Next(7)){
                case 0: shape = new int[,] { { 2, 3, 4, 5 }, { 8, 8, 8, 8 } };
break;
                case 1: shape = new int[,] { { 2, 3, 2, 3 }, { 8, 8, 9, 9 } };
break;
                case 2: shape = new int[,] { { 2, 3, 4, 4 }, { 8, 8, 8, 9 } };
break;
                case 3: shape = new int[,] { { 2, 3, 4, 4 }, { 8, 8, 8, 7 } };
break;
                case 4: shape = new int[,] { { 3, 3, 4, 4 }, { 7, 8, 8, 9 } };
break;
                case 5: shape = new int[,] { { 3, 3, 4, 4 }, { 9, 8, 8, 7 } };
break;
                case 6: shape = new int[,] { { 3, 4, 4, 4 }, { 8, 7, 8, 9 } };
break;
            }
        }
        public bool FindMistake(){
            for (int i = 0; i < 4; i++)
                if (shape[1, i] >= width || shape[0, i] >= height ||
                    shape[1, i] <= 0 || shape[0, i] <= 0 ||
                    field[shape[1, i], shape[0, i]] == 1)
                        return true;
            return false;
        }
    }
}
```


Tetris