|  |  |  |
| --- | --- | --- |
| Защищено:  Гапанюк Ю.Е.    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. |  | Демонстрация:  Гапанюк Ю.Е.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. |

**Отчет по лабораторной работе № 3 по курсу**

**Базовые компоненты интернет-технологий**

**ГУИМЦ**

#### Тема работы: " Работа с коллекциями "

24

(количество листов)

Вариант № **4**

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-52Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Чиварзин А.Е. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. |

Москва, МГТУ - 2020

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ**

[1. Описание задания 3](#_Toc50574201)

[2. Диаграмма классов 3](#_Toc50574202)

[3. Текст программы 3](#_Toc50574203)

[4. Результаты выполнения программы 8](#_Toc50574204)

# Описание задания

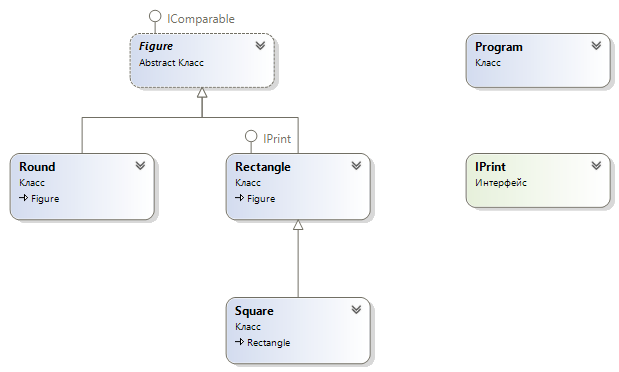
Разработать программу, реализующую работу с коллекциями.

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Создать объекты классов «Прямоугольник», «Квадрат», «Круг».
3. Для реализации возможности сортировки геометрических фигур для класса «Геометрическая фигура» добавить реализацию интерфейса IComparable. Сортировка производится по площади фигуры.
4. Создать коллекцию класса ArrayList. Сохранить объекты в коллекцию. Отсортировать коллекцию. Вывести в цикле содержимое коллекции.
5. Создать коллекцию класса List<Figure>. Сохранить объекты в коллекцию. Отсортировать коллекцию. Вывести в цикле содержимое коллекции.
6. Модифицировать класс разреженной матрицы (проект SparseMatrix) для работы с тремя измерениями – x,y,z. Вывод элементов в методе ToString() осуществлять в том виде, который Вы считаете наиболее удобным. Разработать пример использования разреженной матрицы для геометрических фигур.
7. Реализовать класс «SimpleStack» на основе односвязного списка. Класс SimpleStack наследуется от класса SimpleList (проект SimpleListProject). Необходимо добавить в класс методы:
   1. public void Push(T element) – добавление в стек;
   2. public T Pop() – чтение с удалением из стека.
8. Пример работы класса SimpleStack реализовать на основе геометрических фигур.

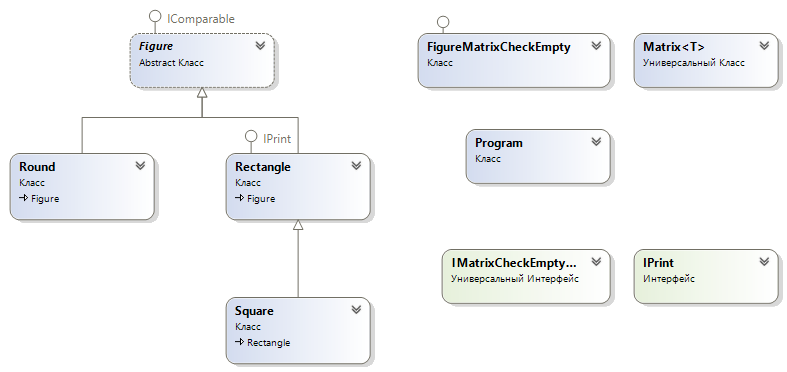
# Диаграмма классов

Поскольку решение состоит из независимых проектов – диаграмма классов будет приведена для каждого проекта.

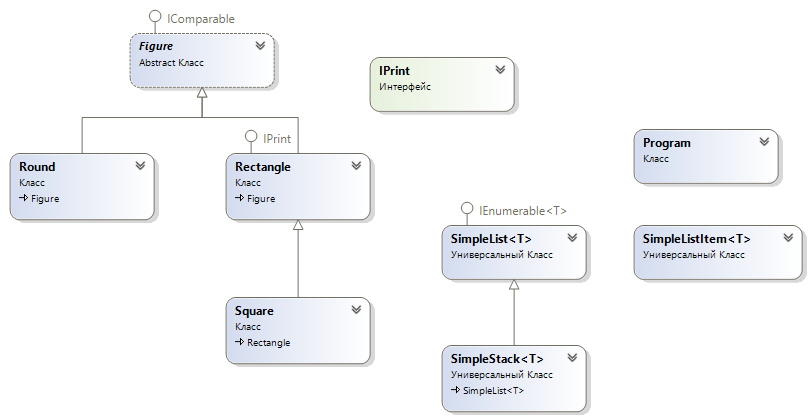
Проект **LAB\_3**



Проект **SparseMatrix**



Проект **SimpleListProject**



# Текст программы

**LAB\_3🡪Program.cs**

using System;

using System.Collections;

using System.Collections.Generic;

namespace LAB\_3

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Чиварзин А. Е. ИУ5Ц-52Б");

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Проект LAB\_3 [1/3]");

Console.ResetColor();

ArrayList al = new ArrayList();

al.Add(new Rectangle(1.0, 2.0));

al.Add(new Square(100.0));

al.Add(new Round(1.0));

for (int i = 0; i < al.Count; i++)

{

Console.Write("[");

Console.Write(i);

Console.WriteLine("] ------------------");

Console.WriteLine(al[i]);

}

al.Sort();

Console.WriteLine(">>>>>> СОРТИРОВКА <<<<<");

for (int i = 0; i < al.Count; i++)

{

Console.Write("[");

Console.Write(i);

Console.WriteLine("] ------------------");

Console.WriteLine(al[i]);

}

Console.WriteLine("### LIST ###");

List<Figure> l = new List<Figure>();

l.Add(new Rectangle(1.0, 2.0));

l.Add(new Square(100.0));

l.Add(new Round(1.0));

for (int i = 0; i < l.Count; i++)

{

Console.Write("[");

Console.Write(i);

Console.WriteLine("] ------------------");

Console.WriteLine(l[i]);

}

l.Sort();

Console.WriteLine(">>>>>> СОРТИРОВКА <<<<<");

for (int i = 0; i < l.Count; i++)

{

Console.Write("[");

Console.Write(i);

Console.WriteLine("] ------------------");

Console.WriteLine(l[i]);

}

}

}

}

**LAB\_3🡪Figures.cs**

using System;

namespace LAB\_3

{

public interface IPrint

{

void Print();

}

public abstract class Figure : IComparable

{

public int CompareTo(object obj)

{

if (obj == null) return 1;

Figure f = obj as Figure;

double a = ploshad();

double b = f.ploshad();

return a.CompareTo(b);

}

public abstract double ploshad();

}

public class Rectangle : Figure, IPrint //Прямоугольник

{

protected double a, b;

public Rectangle(double A, double B)

{

a = A;

b = B;

}

public override double ploshad()

{

return a \* b;

}

public override string ToString()

{

string result = "";

result += "Длинна = ";

result += a.ToString() + " \n";

result += "Ширина = ";

result += b.ToString() + " \n";

result += "Площадь = ";

result += ploshad().ToString() + " \n";

return result;

}

public void Print()

{

Console.WriteLine(ToString());

}

}

public class Square : Rectangle //Квадрат

{

public Square(double A) : base(A, A)

{

}

public override string ToString()

{

string result = "";

result += "Сторона = ";

result += a.ToString() + " \n";

result += "Площадь = ";

result += ploshad().ToString() + " \n";

return result;

}

public new void Print() //new убирает предупреждение о скрытии

{

Console.WriteLine(ToString());

}

}

public class Round : Figure //Круг

{

private double r;

public Round(double R)

{

r = R;

}

public override double ploshad()

{

return Math.PI \* r \* r;

}

public override string ToString()

{

string result = "";

result += "Радиус = ";

result += r.ToString() + " \n";

result += "Площадь = ";

result += ploshad().ToString() + " \n";

return result;

}

public void Print()

{

Console.WriteLine(ToString());

}

}

}

**SpareMatrix🡪Program.cs**

using LAB\_3;

using System;

namespace SparseMatrix

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Чиварзин А. Е. ИУ5Ц-52Б");

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Проект SparseMatrix [2/3]");

Console.ResetColor();

Matrix<Figure> m = new Matrix<Figure>(3, 3, 3, new FigureMatrixCheckEmpty());

m[0, 0, 0] = new Rectangle(1.0, 1.0);

m[1, 1, 1] = new Round(3.14);

m[2, 2, 2] = new Square(10);

Console.WriteLine("Матрица");

Console.WriteLine(m);

for (int i = 0; i < 3; i++)

Console.WriteLine(m[i, 0, 0]);

}

}

class FigureMatrixCheckEmpty : IMatrixCheckEmpty<Figure>

{

/// <summary>

/// В качестве пустого элемента возвращается null

/// </summary>

public Figure getEmptyElement()

{

return null;

}

/// <summary>

/// Проверка что переданный параметр равен null

/// </summary>

public bool checkEmptyElement(Figure element)

{

bool Result = false; if (element == null)

{

Result = true;

}

return Result;

}

}

}

**SareMatrx🡪Figures.cs**

using System;

/\*

\* Копия файла из проекта LAB\_3

\*/

namespace LAB\_3

{

public interface IPrint

{

void Print();

}

public abstract class Figure : IComparable

{

public int CompareTo(object obj)

{

if (obj == null) return 1;

Figure f = obj as Figure;

double a = ploshad();

double b = f.ploshad();

return a.CompareTo(b);

}

public abstract double ploshad();

}

public class Rectangle : Figure, IPrint //Прямоугольник

{

protected double a, b;

public Rectangle(double A, double B)

{

a = A;

b = B;

}

public override double ploshad()

{

return a \* b;

}

public override string ToString()

{

string result = "";

result += "Длинна = ";

result += a.ToString() + " \n";

result += "Ширина = ";

result += b.ToString() + " \n";

result += "Площадь = ";

result += ploshad().ToString() + " \n";

return result;

}

public void Print()

{

Console.WriteLine(ToString());

}

}

public class Square : Rectangle //Квадрат

{

public Square(double A) : base(A, A)

{

}

public override string ToString()

{

string result = "";

result += "Сторона = ";

result += a.ToString() + " \n";

result += "Площадь = ";

result += ploshad().ToString() + " \n";

return result;

}

public new void Print() //new убирает предупреждение о скрытии

{

Console.WriteLine(ToString());

}

}

public class Round : Figure //Круг

{

private double r;

public Round(double R)

{

r = R;

}

public override double ploshad()

{

return Math.PI \* r \* r;

}

public override string ToString()

{

string result = "";

result += "Радиус = ";

result += r.ToString() + " \n";

result += "Площадь = ";

result += ploshad().ToString() + " \n";

return result;

}

public void Print()

{

Console.WriteLine(ToString());

}

}

}

**SpareMatrix🡪iMatrixCheckEmpty.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace SparseMatrix

{

/// <summary>

/// Проверка пустого элемента матрицы

/// </summary>

public interface IMatrixCheckEmpty<T>

{

/// <summary>

/// Возвращает пустой элемент

/// </summary>

T getEmptyElement();

/// <summary>

/// Проверка что элемент является пустым

/// </summary>

bool checkEmptyElement(T element);

}

}

**SpareMatrix🡪Matrix.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace SparseMatrix

{

public class Matrix<T>

{

/// <summary>

/// Словарь для хранения значений

/// </summary>

Dictionary<string, T> \_matrix = new Dictionary<string, T>();

/// <summary>

/// Количество элементов по горизонтали (максимальное количество столбцов)

/// </summary>

int maxX;

/// <summary>

/// Количество элементов по вертикали (максимальное количество строк)

/// </summary>

int maxY;

/// <summary>

/// Количество элементов по глубине (максимальное количество слоёв)

/// </summary>

int maxZ;

/// <summary>

/// Реализация интерфейса для проверки пустого элемента

/// </summary>

IMatrixCheckEmpty<T> сheckEmpty;

/// <summary>

/// Конструктор

/// </summary>

public Matrix(int px, int py, int pz, IMatrixCheckEmpty<T> сheckEmptyParam)

{

this.maxX = px;

this.maxY = py;

this.maxZ = pz;

this.сheckEmpty = сheckEmptyParam;

}

/// <summary>

/// Индексатор для доступа к данных

/// </summary>

public T this[int x, int y, int z]

{

set

{

CheckBounds(x, y, z);

string key = DictKey(x, y, z);

this.\_matrix.Add(key, value);

}

get

{

CheckBounds(x, y, z);

string key = DictKey(x, y, z);

if (this.\_matrix.ContainsKey(key))

{

return this.\_matrix[key];

}

else

{

return this.сheckEmpty.getEmptyElement();

}

}

}

/// <summary>

/// Проверка границ

/// </summary>

void CheckBounds(int x, int y, int z)

{

if (x < 0 || x >= this.maxX)

{

throw new ArgumentOutOfRangeException("X",

"x=" + x + " выходит за границы");

}

if (y < 0 || y >= this.maxY)

{

throw new ArgumentOutOfRangeException("Y",

"y=" + y + " выходит за границы");

}

if (z < 0 || z >= this.maxZ)

{

throw new ArgumentOutOfRangeException("Z",

"z=" + z + " выходит за границы");

}

}

/// <summary>

/// Формирование ключа

/// </summary>

string DictKey(int x, int y, int z)

{

return x.ToString() + "\_" + y.ToString() + "\_" + z.ToString();

}

/// <summary>

/// Приведение к строке

/// </summary> /// <returns></returns>

public override string ToString()

{

StringBuilder b = new StringBuilder();

for (int k = 0; k < maxZ; k++)

{

b.Append("=====> Слой № " + k + "<=====\n");

for (int j = 0; j < maxY; j++)

{

b.Append("["); for (int i = 0; i < maxX; i++)

{

//Добавление разделителя-табуляции

if (i > 0)

{

b.Append("\t");

}

//Если текущий элемент не пустой

if (!this.сheckEmpty.checkEmptyElement(this[i, j, k]))

{

//Добавить приведенный к строке текущий элемент

b.Append(this[i, j, k].ToString());

}

else

{

//Иначе добавить признак пустого значения

b.Append(" - ");

}

}

b.Append("]\n");

}

}

return b.ToString();

}

}

}

**SimpleListProject🡪Program.cs**

using LAB\_3;

using System;

namespace SimpleListProject

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!"); Console.WriteLine("Чиварзин А. Е. ИУ5Ц-52Б");

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Проект SimpleListProject [3/3]");

Console.ResetColor();

SimpleStack<Figure> fList = new SimpleStack<Figure>();

fList.push(new Rectangle(1.0, 10.0));

fList.push(new Square(25.0));

fList.push(new Round(35.5));

Console.WriteLine("POP ------");

Console.WriteLine(fList.pop());

Console.WriteLine("POP ------");

Console.WriteLine(fList.pop());

Console.WriteLine("POP ------");

Console.WriteLine(fList.pop());

}

}

}

**SimpleListProject🡪Figures.cs**

using System;

/\*

\* Копия файла из проекта LAB\_3

\*/

namespace LAB\_3

{

public interface IPrint

{

void Print();

}

public abstract class Figure : IComparable

{

public int CompareTo(object obj)

{

if (obj == null) return 1;

Figure f = obj as Figure;

double a = ploshad();

double b = f.ploshad();

return a.CompareTo(b);

}

public abstract double ploshad();

}

public class Rectangle : Figure, IPrint //Прямоугольник

{

protected double a, b;

public Rectangle(double A, double B)

{

a = A;

b = B;

}

public override double ploshad()

{

return a \* b;

}

public override string ToString()

{

string result = "";

result += "Длинна = ";

result += a.ToString() + " \n";

result += "Ширина = ";

result += b.ToString() + " \n";

result += "Площадь = ";

result += ploshad().ToString() + " \n";

return result;

}

public void Print()

{

Console.WriteLine(ToString());

}

}

public class Square : Rectangle //Квадрат

{

public Square(double A) : base(A, A)

{

}

public override string ToString()

{

string result = "";

result += "Сторона = ";

result += a.ToString() + " \n";

result += "Площадь = ";

result += ploshad().ToString() + " \n";

return result;

}

public new void Print() //new убирает предупреждение о скрытии

{

Console.WriteLine(ToString());

}

}

public class Round : Figure //Круг

{

private double r;

public Round(double R)

{

r = R;

}

public override double ploshad()

{

return Math.PI \* r \* r;

}

public override string ToString()

{

string result = "";

result += "Радиус = ";

result += r.ToString() + " \n";

result += "Площадь = ";

result += ploshad().ToString() + " \n";

return result;

}

public void Print()

{

Console.WriteLine(ToString());

}

}

}

**SimpleListProject🡪SimpleList.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Text;

namespace SimpleListProject

{

public class SimpleListItem<T>

{

/// <summary>

/// Данные

/// </summary>

public T data { get; set; }

/// <summary>

/// Следующий элемент

/// </summary>

public SimpleListItem<T> next { get; set; }

///конструктор

public SimpleListItem(T param)

{

this.data = param;

}

}

public class SimpleList<T> : IEnumerable<T> where T : IComparable

{

/// <summary>

/// Первый элемент списка

/// </summary>

protected SimpleListItem<T> first = null;

/// <summary>

/// Последний элемент списка

/// </summary>

protected SimpleListItem<T> last = null;

/// <summary>

/// Количество элементов

/// </summary>

public int Count

{

get { return \_count; }

protected set { \_count = value; }

}

int \_count;

/// <summary>

/// Добавление элемента

/// </summary>

public void Add(T element)

{

SimpleListItem<T> newItem =

new SimpleListItem<T>(element);

this.Count++;

//Добавление первого элемента

if (last == null)

{

this.first = newItem; this.last = newItem;

}

//Добавление следующих элементов

else

{

//Присоединение элемента к цепочке

this.last.next = newItem;

//Присоединенный элемент считается последним

this.last = newItem;

}

}

/// <summary>

/// Чтение контейнера с заданным номером

/// </summary>

public SimpleListItem<T> GetItem(int number)

{

if ((number < 0) || (number >= this.Count))

{

//Можно создать собственный класс исключения

throw new Exception("Выход за границу индекса");

}

SimpleListItem<T> current = this.first; int i = 0;

//Пропускаем нужное количество элементов

while (i < number)

{

//Переход к следующему элементу

current = current.next;

//Увеличение счетчика

i++;

}

return current;

}

/// <summary>

/// Чтение элемента с заданным номером

/// </summary>

public T Get(int number)

{

return GetItem(number).data;

}

/// <summary>

/// Для перебора коллекции

/// </summary>

public IEnumerator<T> GetEnumerator()

{

SimpleListItem<T> current = this.first;

//Перебор элементов

while (current != null)

{

//Возврат текущего значения

yield return current.data;

//Переход к следующему элементу

current = current.next;

}

}

/// <summary>

/// Для перебора коллекции

/// </summary>

IEnumerator IEnumerable.GetEnumerator()

{

SimpleListItem<T> current = this.first;

//Перебор элементов

while (current != null)

{

//Возврат текущего значения

yield return current.data;

//Переход к следующему элементу

current = current.next;

}

}

}

}

**SimpleListProject🡪SimpleStack.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace SimpleListProject

{

public class SimpleStack<T> : SimpleList<T> where T : IComparable

{

public SimpleStack () { }

public void push(T t)

{

Add(t);

}

public T pop()

{

T res = last.data;

SimpleListItem<T> newLast = first;

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

newLast = newLast.next;

}

last = newLast;

newLast.next = null;

Count--;

return res;

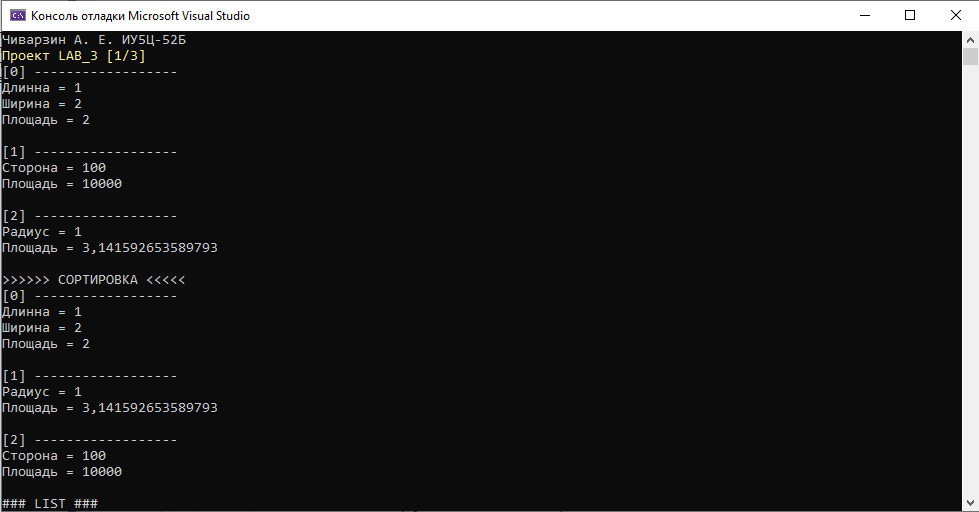
}

}

}

# Результаты выполнения программы

**Ппрект LAB\_3**



Чиварзин А. Е. ИУ5Ц-52Б

Проект LAB\_3 [1/3]

[0] ------------------

Длинна = 1

Ширина = 2

Площадь = 2

[1] ------------------

Сторона = 100

Площадь = 10000

[2] ------------------

Радиус = 1

Площадь = 3,141592653589793

>>>>>> СОРТИРОВКА <<<<<

[0] ------------------

Длинна = 1

Ширина = 2

Площадь = 2

[1] ------------------

Радиус = 1

Площадь = 3,141592653589793

[2] ------------------

Сторона = 100

Площадь = 10000

### LIST ###

[0] ------------------

Длинна = 1

Ширина = 2

Площадь = 2

[1] ------------------

Сторона = 100

Площадь = 10000

[2] ------------------

Радиус = 1

Площадь = 3,141592653589793

>>>>>> СОРТИРОВКА <<<<<

[0] ------------------

Длинна = 1

Ширина = 2

Площадь = 2

[1] ------------------

Радиус = 1

Площадь = 3,141592653589793

[2] ------------------

Сторона = 100

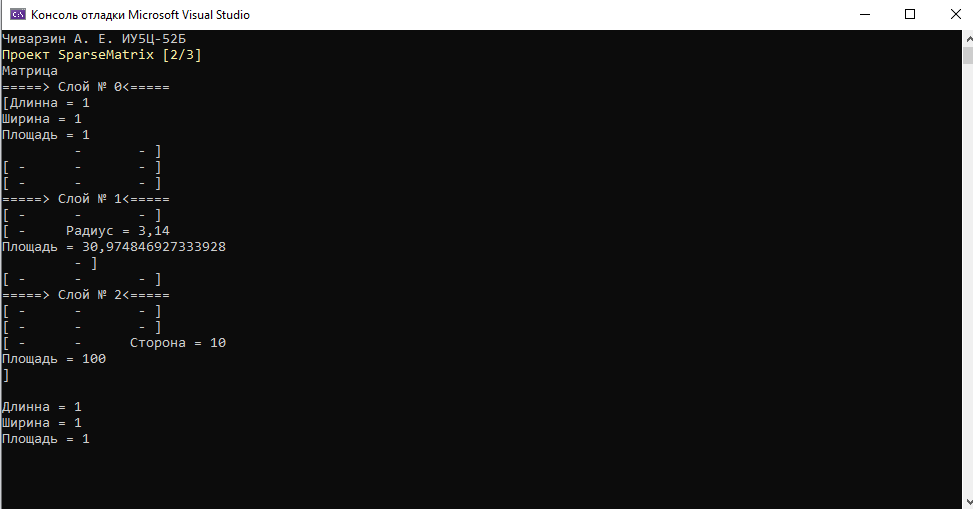
Площадь = 10000

A:\sasha\YandexDisk\YandexDisk\МГТУ\5-й семестр\Основные компоненты интернет-технологий\ЛР\VS\LAB\_3\LAB\_3\bin\Debug\netcoreapp3.1\LAB\_3.exe (процесс 14260) завершил работу с кодом 0.

Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".

Нажмите любую клавишу, чтобы закрыть это окно…

**Проект SpareMatrix**



Чиварзин А. Е. ИУ5Ц-52Б

Проект SparseMatrix [2/3]

Матрица

=====> Слой № 0<=====

[Длинна = 1

Ширина = 1

Площадь = 1

- - ]

[ - - - ]

[ - - - ]

=====> Слой № 1<=====

[ - - - ]

[ - Радиус = 3,14

Площадь = 30,974846927333928

- ]

[ - - - ]

=====> Слой № 2<=====

[ - - - ]

[ - - - ]

[ - - Сторона = 10

Площадь = 100

]

Длинна = 1

Ширина = 1

Площадь = 1

A:\sasha\YandexDisk\YandexDisk\МГТУ\5-й семестр\Основные компоненты интернет-технологий\ЛР\VS\LAB\_3\SparseMatrix\bin\Debug\netcoreapp3.1\SparseMatrix.exe (процесс 3508) завершил работу с кодом 0.

Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".

Нажмите любую клавишу, чтобы закрыть это окно…

**Проект SimpleListProject**

