**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №3

**«**Работа с коллекциями»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-35б |  | доцент каф. ИУ5 |
| Кожиев Т.К. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

# Постановка задачи:

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. 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:

• public void Push(T element) – добавление в стек;

• public T Pop() – чтение с удалением из стека.

8. Пример работы класса SimpleStack реализовать на основе геометрических фигур.

Текст программы:

Тесты программы:

//Figure.cs

using System;  
using System.Collections.Generic;  
  
namespace lab3  
{  
interface Iprint  
{  
void print();  
}  
  
abstract class figure : IComparable  
{  
public string name { get; protected set; }  
protected double S;  
  
public figure(string name)  
{  
[this.name](https://vk.com/away.php?to=http%3A%2F%2Fthis.name&cc_key=) = name;  
}  
public figure() { }  
  
public abstract double area();  
  
  
public int CompareTo(object other)  
{  
if (other == null) return 1;  
figure otherFig = other as figure;  
if (otherFig != null)  
{  
return this.S.CompareTo(otherFig.S);  
}  
else  
{  
throw new Exception("Невозможно определить объект");  
}  
  
}  
}  
  
class rectangle : figure, Iprint  
{  
public double wide { get; protected set; }  
public double high { get; protected set; }  
// private double S;  
public rectangle(string name, double wide, double high) : base(name)  
{  
this.wide = wide;  
this.high = high;  
}  
  
public override double area()  
{  
S = this.high \* this.wide;  
return S;  
//Console.WriteLine(S);  
}  
  
public override string ToString()  
{  
this.area();  
return $"Площадь {name} со сторонами {wide} {high} равна {S}";  
}  
  
public virtual void print()  
{  
Console.WriteLine(this.ToString());  
}  
  
}  
  
class square : rectangle  
{  
public int side { get; private set; }  
// private double S;  
public square(string name, int side) : base(name, side, side)  
{  
this.side = side;  
}  
  
public override double area()  
{  
S = side \* side;  
return S;  
}  
public override string ToString()  
{  
this.area();  
return $"Площадь {name} со стороной {side} равна {S}";  
}  
public override void print()  
{  
// this.area();  
Console.WriteLine(this.ToString());  
}  
}  
class circle : figure, Iprint  
{  
public int radius { get; private set; }  
// private double S;  
public circle(string name, int radius) : base(name)  
{  
this.radius = radius;  
}  
public override double area()  
{  
double pi = 3.14;  
S = pi \* radius \* radius;  
return Math.Round(S);  
}  
  
public override string ToString()  
{  
this.area();  
return $"Площадь {name} с радиусом {radius} и площадью {S}";  
}  
public void print()  
{  
Console.WriteLine(this.ToString());  
}  
  
}  
}

Program.cs

using System;  
using System.Collections.Generic;  
  
namespace lab3  
{  
class Program  
{  
static void Main(string[] args)  
{  
rectangle ret = new rectangle("Прямоугольника", 5, 7);  
square sq = new square("Квадрат", 5);  
circle cir = new circle("Круга", 8);  
  
  
List<figure> arrayList = new List<figure>();  
  
  
arrayList.Add(ret);  
arrayList.Add(sq);  
arrayList.Add(cir);  
  
for (int i = 0; i < arrayList.Count; i++)  
{  
Console.WriteLine(arrayList[i]);  
}  
  
Console.WriteLine("--------------------------------------------------------");  
  
arrayList.Sort();  
  
for (int i = 0; i < arrayList.Count; i++)  
{  
Console.WriteLine(arrayList[i]);  
}  
  
Console.WriteLine("--------------------------------------------------------");  
  
// трехмерный массив  
// : сделать нормальный вывод  
Matrix<figure> matrixXYZ = new Matrix<figure>(3, 3, 3, sq);  
matrixXYZ[0, 0, 0] = new rectangle("Rectangle", 5, 7);  
matrixXYZ[1, 1, 1] = new circle("Circle", 6);  
matrixXYZ[2, 2, 2] = new square("Square", 5);  
matrixXYZ.print();  
  
  
Console.WriteLine("-----------------------------------------------");  
  
SimpleList<figure> list = new SimpleList<figure>();  
list.Add(ret);  
list.Add(cir);  
  
SimpleStack<figure> stack = new SimpleStack<figure>();  
stack.push(ret);  
stack.push(cir);  
stack.push(sq);  
  
foreach (var item in stack)  
{  
Console.WriteLine(item);  
}  
  
stack.pop();  
Console.WriteLine("------------------------------------------------");  
foreach (var item in stack)  
{  
Console.WriteLine(item);  
}  
}  
}  
}

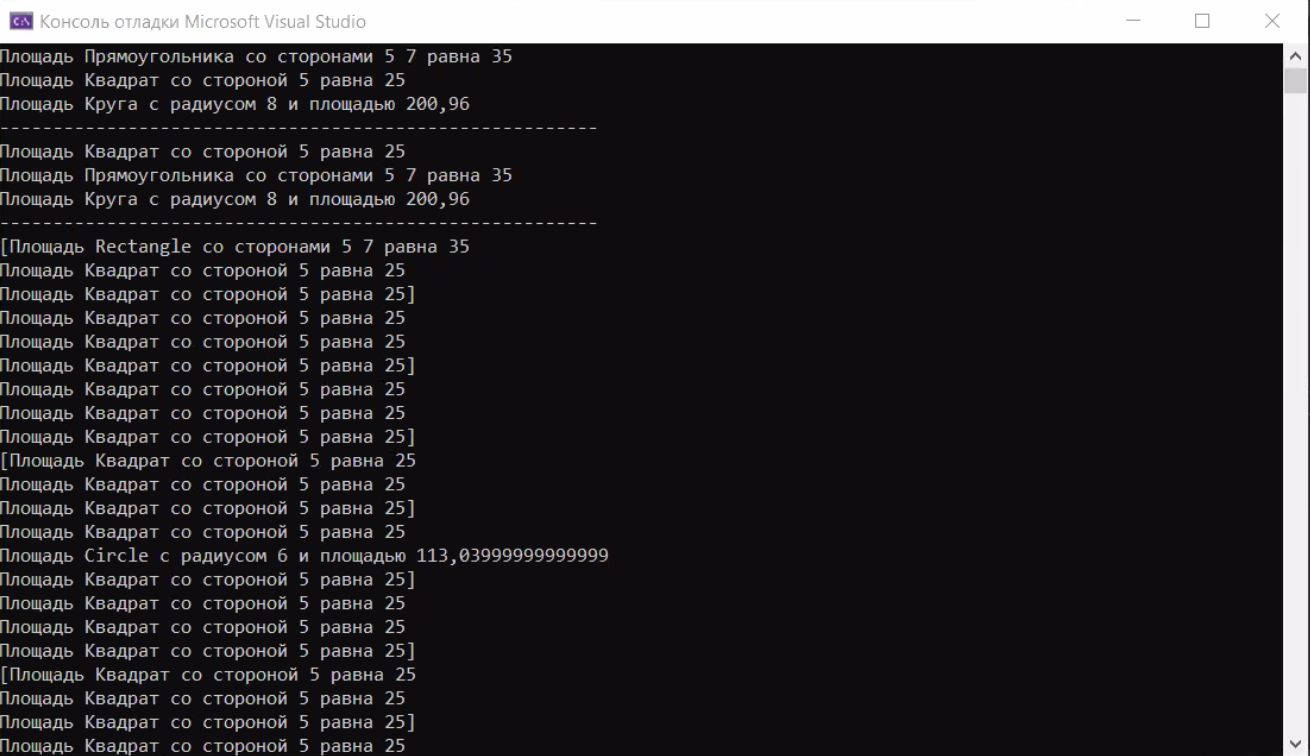
SimpleStack.cs

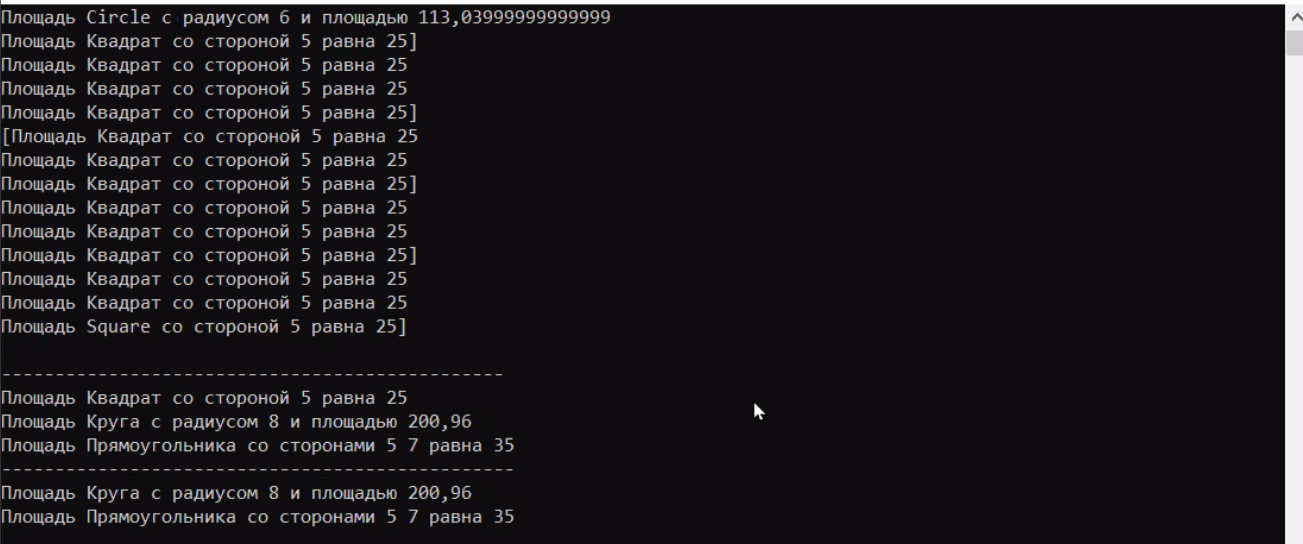
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace lab3  
{  
/// Элемент списка  
public class SimpleListItem<T>  
{  
public T data { get; set; } /// Данные  
  
public SimpleListItem<T> next { get; set; } /// Следующий элемент  
  
public SimpleListItem(T param) // конструктор  
{  
[this.data](https://vk.com/away.php?to=http%3A%2F%2Fthis.data&cc_key=) = param;  
}  
}  
/// Список  
public class SimpleList<T> : IEnumerable<T>  
where T : IComparable  
{  
/// Первый элемент списка  
protected SimpleListItem<T> first = null;  
/// Последний элемент списка  
protected SimpleListItem<T> last = null;  
/// Количество элементов  
public int Count  
{  
get { return \_count; }  
protected set { \_count = value; }  
}  
int \_count;  
  
  
virtual 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](https://vk.com/away.php?to=http%3A%2F%2Fthis.last.next&cc_key=) = newItem;  
//Просоединенный элемент считается последним  
this.last = newItem;  
}  
}  
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](https://vk.com/away.php?to=http%3A%2F%2Fcurrent.next&cc_key=);  
//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](https://vk.com/away.php?to=http%3A%2F%2Fcurrent.data&cc_key=);  
//Переход к следующему элементу current = [current.next](https://vk.com/away.php?to=http%3A%2F%2Fcurrent.next&cc_key=);  
current = [current.next](https://vk.com/away.php?to=http%3A%2F%2Fcurrent.next&cc_key=);  
}  
}  
//Реализация обощенного IEnumerator<T> требует реализации необобщенного интерфейса  
//Данный метод добавляется автоматически при реализации интерфейса  
System.Collections.IEnumerator  
System.Collections.IEnumerable.GetEnumerator()  
{  
return GetEnumerator();  
}  
/// Cортировка  
public void Sort()  
{  
Sort(0, this.Count - 1);  
}  
  
/// Реализация алгоритма быстрой сортировки  
private void Sort(int low, int high)  
{  
int i = low;  
int j = high;  
T x = Get((low + high) / 2);  
do  
{  
while (Get(i).CompareTo(x) < 0) ++i; while (Get(j).CompareTo(x) > 0) —j; if (i <= j)  
{  
Swap(i, j);  
i++; j--;  
}  
}  
while (i <= j);  
if (low < j) Sort(low, j);  
if (i < high) Sort(i, high);  
}  
  
/// <summary>  
/// Вспомогательный метод для обмена элементов при сортировке /// </summary>  
private void Swap(int i, int j)  
{  
SimpleListItem<T> ci = GetItem(i);  
SimpleListItem<T> cj = GetItem(j);  
T temp = [ci.data](https://vk.com/away.php?to=http%3A%2F%2Fci.data&cc_key=);  
[ci.data](https://vk.com/away.php?to=http%3A%2F%2Fci.data&cc_key=) = [cj.data](https://vk.com/away.php?to=http%3A%2F%2Fcj.data&cc_key=);  
[cj.data](https://vk.com/away.php?to=http%3A%2F%2Fcj.data&cc_key=) = temp;  
}  
}  
  
class SimpleStack<T>:SimpleList<figure>, IEnumerable<T>  
{  
SimpleListItem<T> head;  
public SimpleStack():base()  
{  
  
}  
  
public void push(T element)  
{  
SimpleListItem<T> node = new SimpleListItem<T>(element);  
[node.next](https://vk.com/away.php?to=http%3A%2F%2Fnode.next&cc_key=) = head;  
head = node;  
Count++;  
}  
  
public T pop()  
{  
if(Count == 0) throw new Exception("Стек пустой");  
SimpleListItem<T> temp = head;  
head = [head.next](https://vk.com/away.php?to=http%3A%2F%2Fhead.next&cc_key=);  
Count--;  
return [temp.data](https://vk.com/away.php?to=http%3A%2F%2Ftemp.data&cc_key=);  
}  
public IEnumerator<T> GetEnumerator()  
{  
SimpleListItem<T> current = this.head;  
//Перебор элементов  
while (current != null)  
{  
//Возврат текущего значения  
yield return [current.data](https://vk.com/away.php?to=http%3A%2F%2Fcurrent.data&cc_key=);  
//Переход к следующему элементу current = [current.next](https://vk.com/away.php?to=http%3A%2F%2Fcurrent.next&cc_key=);  
current = [current.next](https://vk.com/away.php?to=http%3A%2F%2Fcurrent.next&cc_key=);  
}  
}  
System.Collections.IEnumerator  
System.Collections.IEnumerable.GetEnumerator()  
{  
return GetEnumerator();  
}  
}  
}

SparceMatix.cs

using System;  
using System.Collections.Generic;  
using System.Diagnostics.CodeAnalysis;  
using System.Linq;  
using System.Text;  
  
namespace lab3  
{  
public class Matrix<T>  
{  
  
Dictionary<string, T> \_matrix = new Dictionary<string, T>(); // Словарь для хранения значений  
  
int maxX; // Количество элементов по горизонтали (максимальное количество столбцов)  
int maxY; //Количество элементов по вертикали (максимальное количество строк)  
int maxZ;  
T nullElement; //Пустой элемент, который возвращается если элемент с нужными координатами не был задан  
  
public Matrix(int px, int py, T nullElementParam)  
{  
this.maxX = px;  
this.maxY = py;  
this.nullElement = nullElementParam;  
}  
public Matrix(int px, int py, int pz, T nullElementParam)  
{  
this.maxX = px;  
this.maxY = py;  
this.maxZ = pz;  
this.nullElement = nullElementParam;  
}  
  
  
public T this[int x, int y] // Индексатор для доступа к данных  
{  
get  
{  
CheckBounds(x, y);  
string key = DictKey(x, y);  
if (this.\_matrix.ContainsKey(key))  
{  
return this.\_matrix[key];  
}  
else  
{  
return this.nullElement;  
}  
}  
set  
{  
CheckBounds(x, y);  
string key = DictKey(x, y); this.\_matrix.Add(key, value);  
}  
}  
  
public T this[int x, int y, int z] // Индексатор для доступа к данных XYZ  
{  
get  
{  
CheckBounds(x, y, z);  
string key = DictKey(x, y, z);  
if (this.\_matrix.ContainsKey(key))  
{  
return this.\_matrix[key];  
}  
else  
{  
return this.nullElement;  
}  
}  
set  
{  
CheckBounds(x, y, z);  
string key = DictKey(x, y, z); this.\_matrix.Add(key, value);  
}  
}  
  
  
void CheckBounds(int x, int y) // Проверка границ  
{  
if (x < 0 || x >= this.maxX) throw new Exception("x=" + x + " выходит за границы");  
if (y < 0 || y >= this.maxY) throw new Exception("y=" + y + " выходит за границы");  
}  
  
void CheckBounds(int x, int y, int z) // Проверка границ для трех мерной матрицы  
{  
if (x < 0 || x >= this.maxX) throw new Exception("x=" + x + " выходит за границы");  
if (y < 0 || y >= this.maxY) throw new Exception("y=" + y + " выходит за границы");  
if (z < 0 || z >= this.maxZ) throw new Exception("z=" + z + " выходит за границы");  
}  
  
string DictKey(int x, int y) // Формирование ключа XY  
{  
return x.ToString() + "\_" + y.ToString();  
}  
  
string DictKey(int x, int y, int z) // Формирование ключа XYZ  
{  
return x.ToString() + "\_" + y.ToString() + "\_" + z.ToString();  
}  
  
public override string ToString() // Приведение к строке  
{  
  
StringBuilder b = new StringBuilder();  
  
for (int j = 0; j < maxX; j++)  
{  
b.Append("[");  
for (int i = 0; i < maxY; i++)  
{  
for (int k = 0; k < maxZ; k++)  
{  
if (k > 0) b.Append("\n");  
  
b.Append(this[i, j, k].ToString());  
}  
b.Append("]\n");  
}  
}  
  
  
return b.ToString();  
}  
  
public void print()  
{  
Console.WriteLine(this.ToString());  
}  
  
  
}  
}

Тесты программы





24.12.2020  
Кожиев Т.К.