МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**Кафедра ''САПР''**

Отчет о выполненной лабораторной работе № 3

по дисциплине: “Объектно-ориентированное программирование”

Выполнили ст.гр. 20ВА1:

Кулаков Д.А.

Принял:

Гудков А.А.

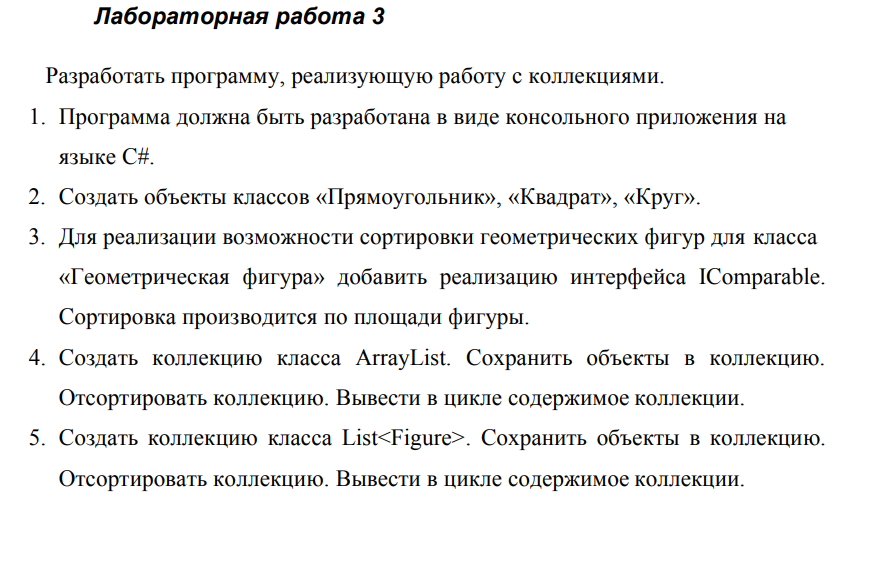
Подмарькова Е.М.

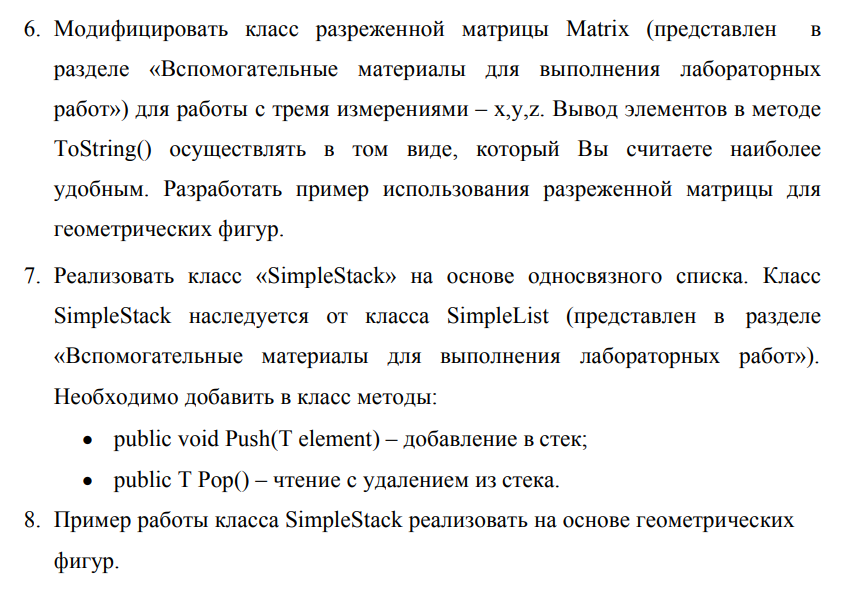
**Цель работы:**

Разработать программу, реализующую работу с коллекциями, ознакомиться с коллекциями на языке C#.

**Ход работы:**

Задание:





Код программы:

using System.Collections;

using System.Text;

using System.Collections.Generic;

namespace laba2

{

class Program

{

static void Main(string[] args)

{

Pryamougolnik p = new Pryamougolnik(1, 3.14);

p.Sq();

p.Print();

Console.WriteLine(" ");

Kvadrat k = new Kvadrat(13.05);

k.Sq();

k.Print();

Console.WriteLine(" ");

Krug kr = new Krug(6);

kr.Sq();

kr.Print();

List(p, k, kr);

List2(p, k, kr);

Console.WriteLine(" ");

Matrix<Figura> matrix = new Matrix<Figura>(3, 3, 3, null);

matrix[0, 0, 0] = new Pryamougolnik(2, 3);

matrix[0, 0, 1] = new Kvadrat(8);

matrix[0, 1, 0] = new Krug(45);

matrix[2, 2, 0] = new Krug(23);

matrix[1, 0, 1] = new Kvadrat(22);

matrix[1, 1, 1] = new Pryamougolnik(33, 1);

Console.Write(matrix.ToString());

Console.WriteLine(" ");

SimpleList<Figura> list1 = new SimpleList<Figura>();

list1.Add(k);

list1.Add(p);

list1.Add(kr);

foreach (var x in list1) Console.WriteLine(x);

list1.Sort();

Console.WriteLine("\n Сортировка списка");

foreach (var x in list1) Console.WriteLine(x);

Console.WriteLine(" ");

}

static void List(Pryamougolnik p, Kvadrat k, Krug kr)

{

Console.WriteLine(" ");

ArrayList list = new ArrayList() { p.Sq(), k.Sq(), kr.Sq() };

list.Sort();

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

{

Console.WriteLine(list[j]);

}

}

static void List2(Pryamougolnik p, Kvadrat k, Krug kr)

{

Console.WriteLine(" ");

List<double> list = new List<double>() { p.Sq(), k.Sq(), kr.Sq() };

list.Sort();

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

{

Console.WriteLine((double)list[j]);

}

}

}

interface IPrint

{

void Print();

}

public class Matrix<T>

{

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

int maxX;

int maxY;

int maxZ;

T nullElement;

public Matrix(int px, int py, int pz, T nullElementParam)

{

this.maxX = px;

this.maxY = py;

this.maxZ = pz;

this.nullElement = nullElementParam;

}

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

{

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 ");

}

public override string ToString()

{

StringBuilder sb = new StringBuilder();

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

{

sb.Append("[\n");

for (int x = 0; x < maxX; x++)

{

sb.Append("\t[");

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

{

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

if (z > 0)

sb.Append("\t");

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

sb.Append(this[x, y, z].ToString());

else

sb.Append("none");

}

sb.Append("]\n");

}

sb.Append("]\n");

}

return sb.ToString();

}

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

{

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

}

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

{

get

{

ChekBounds(x, y, z);

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

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

return this.\_matrix[key];

else

return this.nullElement;

}

set

{

ChekBounds(x, y, z);

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

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

}

}

}

public class SimpleListItem<T>

{

public T data

{

get;

set;

}

public SimpleListItem<T> next

{

get;

set;

}

public SimpleListItem(T param)

{

data = param;

}

}

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

{

protected SimpleListItem<T> first = null;

protected SimpleListItem<T> last = null;

protected int \_count;

public int Count

{

get { return \_count; }

protected set { \_count = value; }

}

public void Add(T element)

{

SimpleListItem<T> newItem = new SimpleListItem<T>(element);

Count++;

if (last == null)

{

first = newItem;

last = newItem;

}

else

{

last.next = newItem;

last = newItem;

}

}

public SimpleListItem<T> GetItem(int number)

{

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

{

throw new Exception("Выход за границу {number}");

}

SimpleListItem<T> current = first;

int i = 0;

while (i < number)

{

current = current.next;

i++;

}

return current;

}

public T Get(int number)

{

return GetItem(number).data;

}

public IEnumerator<T> GetEnumerator()

{

SimpleListItem<T> current = first;

while (current != null)

{

yield return current.data;

current = current.next;

}

}

System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

private void Swap(int i, int j)

{

SimpleListItem<T> ci = GetItem(i);

SimpleListItem<T> cj = GetItem(j);

T temp = ci.data;

ci.data = cj.data;

cj.data = temp;

}

protected 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);

}

}

public void Sort()

{

Sort(0, this.Count - 1);

}

}

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

{

public SimpleStack() : base() { }

public void Add(T element) { }

public void GetItem(int number) { }

public void Get(int number) { }

public void Sort() { }

public void Push(T element)

{

SimpleListItem<T> item = new SimpleListItem<T>(element);

Count++;

if (first == null)

{

first = item;

last = item;

}

else

{

item.next = first;

first = item;

}

}

public T Pop()

{

T tmp = first.data;

if (first.next != null)

first = first.next;

else

first = null;

return tmp;

}

}

abstract class Figura : IComparable

{

virtual public double Sq()

{

return 0;

}

public int CompareTo(object obj)

{

if (obj == null) return 1;

Figura otherFigura = obj as Figura;

if (otherFigura != null)

return this.Sq().CompareTo(otherFigura.Sq());

else

throw new ArgumentException("Object is not a Temperature");

}

}

class Pryamougolnik : Figura, IPrint

{

double S;

public double shirina

{

get;

set;

}

public double visota

{

get;

set;

}

public Pryamougolnik(double Shirina, double Visota)

{

shirina = Shirina;

visota = Visota;

}

protected Pryamougolnik()

{

}

public override string ToString()

{

return shirina.ToString();

return visota.ToString();

return S.ToString();

}

public void Print()

{

ToString();

Console.WriteLine(shirina.ToString());

Console.WriteLine(visota.ToString());

Console.WriteLine(S.ToString());

}

public override double Sq()

{

S = shirina \* visota;

return S;

}

}

class Kvadrat : Pryamougolnik, IPrint

{

double S, storona;

public Kvadrat(double Storona)

{

storona = Storona;

}

public override string ToString()

{

return storona.ToString();

}

public override double Sq()

{

S = storona \* storona;

return S;

}

public void Print()

{

ToString();

Console.WriteLine(storona.ToString());

Console.WriteLine(S.ToString());

}

}

class Krug : Figura, IPrint

{

double S;

public double radius

{

get; set;

}

public Krug(double Radius)

{

radius = Radius;

}

public override string ToString()

{

return radius.ToString();

return S.ToString();

}

public void Print()

{

ToString();

Console.WriteLine(radius.ToString());

Console.WriteLine(S.ToString());

}

public override double Sq()

{

S = 3.14 \* radius \* radius;

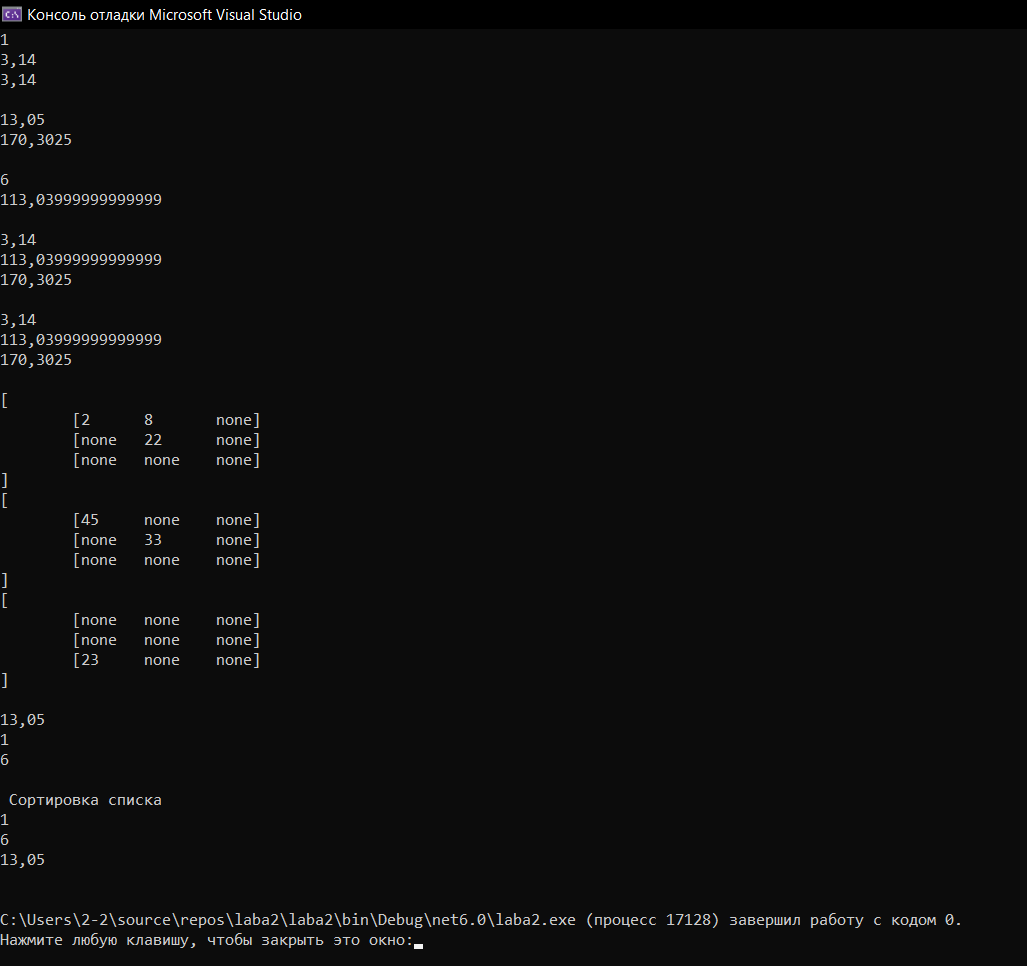
return S;

}

}

}

Результат:



**Вывод:**

Я разработал программу, реализующую работу с коллекциями, ознакомился с коллекциями на языке C#.