**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего**

**профессионального образования**

**«Казанский национальный исследовательский технический университет**

**им. А.Н. Туполева-КАИ»**

Институт компьютерных технологий и защиты информации

Кафедра Прикладной математики и Информатики имени Ю.В. Кожевникова

Лабораторная работа №7

по дисциплине «Объектно-ориентированное программирование»

**Выполнил**

студент группы 4210

Нигамадянов Фанис Магефурович

Казань 2021

**Лабораторная работа №7. Создание и использование контейнерных классов.**

Описание класса-контейнера на базе массива:

1. Поле-итератор

int iterator = 0;

1. Булево поле для итератора

bool for\_iterator = false;

1. Поля для итератора при работе с пользовательским массивом

static int user\_iterator = 0;

1. Поле, хранящее номер варианта

static int num\_of\_variant = 27;

1. Поле, которое пригодится, когда будет изменяться переменная выше

int old\_num\_variant = num\_of\_variant;

1. Значение количества фигур, которое ввел пользователь

int num = Form1.num\_of\_shapes;

1. Создан ли массив

static bool massive\_is\_created = false;

1. Создан ли пользовательский массив

static bool user\_massive\_is\_created = false;

1. Объект класса Random

Random my\_rnd = new Random();

1. Массив, хранящий массивы, хранящие фигуры

private Abstract\_properties[][] massive\_of\_massives = new Abstract\_properties[15][];

1. Создание случайного массива

public Massive\_container()

{

if (massive\_is\_created)

{

return;

}

massive\_of\_massives[0] = new Abstract\_properties[num\_of\_variant];

int my\_rand = my\_rnd.Next(1, 50);

for (int i = 0; i < num\_of\_variant; i++)

{

if (my\_rand >= 1 && my\_rand < 11)

{

massive\_of\_massives[0][i] = new circle();

}

else if (my\_rand >= 11 && my\_rand < 21)

{

massive\_of\_massives[0][i] = new ellipse();

}

else if (my\_rand >= 21 && my\_rand < 31)

{

massive\_of\_massives[0][i] = new rectangle();

}

else if (my\_rand >= 31 && my\_rand < 41)

{

massive\_of\_massives[0][i] = new square();

}

else if (my\_rand >= 41 && my\_rand < 51)

{

massive\_of\_massives[0][i] = new rhomb();

}

my\_rand = my\_rnd.Next(1, 50);

Thread.Sleep(10);

}

massive\_is\_created = true;

System.Windows.Forms.MessageBox.Show("Массив создан!");

}

1. Создание пользовательского массива

public Massive\_container(int n)

{

if (massive\_is\_created)

{

return;

}

massive\_of\_massives[0] = new Abstract\_properties[n];

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

{

massive\_of\_massives[0][i] = null;

}

massive\_is\_created = true;

user\_massive\_is\_created = true;

}// КОНСТРУКТОР ДЛЯ СОЗДАНИЯ ПУСТОГО МАССИВА

1. Показ фигур

public void iterator\_for\_show(Graphics gr)

{

if (!massive\_is\_created)

{

System.Windows.Forms.MessageBox.Show("Массив еще не создан!");

return;

}

else

{

if (!for\_iterator)

{

for (int i = 0; i < massive\_of\_massives[iterator].Length; i++)

{

if (massive\_of\_massives[iterator][i] is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is circle circ\_type)

{

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is square square\_type)

{

square\_type.square\_is\_visible = true;

square\_type.Show(gr, Color.Aquamarine);

}

}

}

else

{

for (int i = 0; i < massive\_of\_massives[iterator+1].Length; i++)

{

if (massive\_of\_massives[iterator + 1][i] is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator + 1][i] is circle circ\_type)

{

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator + 1][i] is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator + 1][i] is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator + 1][i] is square square\_type)

{

square\_type.square\_is\_visible = true;

square\_type.Show(gr, Color.Aquamarine);

}

}

}

}

} // ПОКАЗ ФИГУР

1. Удаление фигур с холста

public void iterator\_for\_delete(Graphics gr)

{

if (!massive\_is\_created)

{

System.Windows.Forms.MessageBox.Show("Массив еще не создан!");

return;

}

gr.Clear(Color.White);

for (int i = 0; i < massive\_of\_massives[iterator].Length; i++)

{

if (massive\_of\_massives[iterator][i] is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = false;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is circle circ\_type)

{

circ\_type.circle\_is\_visible = false;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = false;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = false;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is square square\_type)

{

square\_type.square\_is\_visible = false;

square\_type.Show(gr, Color.Aquamarine);

}

}

} // УДАЛЕНИЕ

1. Метод для перемещения

public void iterator\_for\_move(int x, int y, Graphics gr)

{

if (!massive\_is\_created)

{

System.Windows.Forms.MessageBox.Show("Массив еще не создан!");

return;

}

gr.Clear(Color.White);

if (!for\_iterator)

{

for (int i = 0; i < massive\_of\_massives[iterator].Length; i++)

{

if (massive\_of\_massives[iterator][i] is ellipse ell\_type)

{

ell\_type.Move(x, y);

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is circle circ\_type)

{

circ\_type.Move(x, y);

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rectangle rect\_type)

{

rect\_type.Move(x, y);

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rhomb rh\_type)

{

rh\_type.Move(x, y);

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is square square\_type)

{

square\_type.Move(x, y);

square\_type.square\_is\_visible = true;

square\_type.Show(gr, Color.Aquamarine);

}

}

}

else

{

for (int i = 0; i < massive\_of\_massives[iterator+1].Length; i++)

{

if (massive\_of\_massives[iterator+1][i] is ellipse ell\_type)

{

ell\_type.Move(x, y);

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator+1][i] is circle circ\_type)

{

circ\_type.Move(x, y);

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator+1][i] is rectangle rect\_type)

{

rect\_type.Move(x, y);

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator+1][i] is rhomb rh\_type)

{

rh\_type.Move(x, y);

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator+1][i] is square square\_type)

{

square\_type.Move(x, y);

square\_type.square\_is\_visible = true;

square\_type.Show(gr, Color.Aquamarine);

}

}

}

} // ПЕРЕМЕЩЕНИЕ

1. Метод для расширения массива и добавления в него новых элементов

public void expansion\_of\_massive(Graphics gr)

{

if (!massive\_is\_created)

{

System.Windows.Forms.MessageBox.Show("Массив еще не создан!");

return;

}

if (user\_massive\_is\_created)

{

if (user\_iterator < Form1.num\_of\_shapes)

{

int my\_rand\_1 = my\_rnd.Next(1, 50);

if (my\_rand\_1 >= 1 && my\_rand\_1 < 11)

{

massive\_of\_massives[iterator][user\_iterator] = new circle();

}

else if (my\_rand\_1 >= 11 && my\_rand\_1 < 21)

{

massive\_of\_massives[iterator][user\_iterator] = new ellipse();

}

else if (my\_rand\_1 >= 21 && my\_rand\_1 < 31)

{

massive\_of\_massives[iterator][user\_iterator] = new rectangle();

}

else if (my\_rand\_1 >= 31 && my\_rand\_1 < 41)

{

massive\_of\_massives[iterator][user\_iterator] = new square();

}

else if (my\_rand\_1 >= 41 && my\_rand\_1 < 51)

{

massive\_of\_massives[iterator][user\_iterator] = new rhomb();

}

massive\_of\_massives[iterator][user\_iterator].Show(gr, Color.Aquamarine);

user\_iterator++;

}

else

{

if (user\_iterator == num)

{

System.Windows.Forms.MessageBox.Show("Массив переполнен! Добавил 10%");

if (num\*0.1 < 1)

{

num++;

}

else

{

num = (int)(num + num \* 0.1);

}

if (for\_iterator)

iterator++;

for\_iterator = true;

massive\_of\_massives[iterator + 1] = new Abstract\_properties[num];

for (int i = 0; i < massive\_of\_massives[iterator].Length; i++)

{

massive\_of\_massives[iterator + 1][i] = massive\_of\_massives[iterator][i];

}

}

int my\_rand = my\_rnd.Next(1, 50);

Thread.Sleep(100);

if (my\_rand >= 1 && my\_rand < 11)

{

massive\_of\_massives[iterator + 1][user\_iterator-1] = new circle();

}

else if (my\_rand >= 11 && my\_rand < 21)

{

massive\_of\_massives[iterator + 1][user\_iterator-1] = new ellipse();

}

else if (my\_rand >= 21 && my\_rand < 31)

{

massive\_of\_massives[iterator + 1][user\_iterator-1] = new rectangle();

}

else if (my\_rand >= 31 && my\_rand < 41)

{

massive\_of\_massives[iterator + 1][user\_iterator-1] = new square();

}

else if (my\_rand >= 41 && my\_rand < 51)

{

massive\_of\_massives[iterator + 1][user\_iterator-1] = new rhomb();

}

massive\_of\_massives[iterator + 1][user\_iterator-1].Show(gr, Color.Aquamarine);

user\_iterator++;

}

}

else

{

if (old\_num\_variant == num\_of\_variant)

{

System.Windows.Forms.MessageBox.Show("Массив переполнен! Добавил 10%");

if (num\_of\_variant \* 0.1 < 1)

{

num\_of\_variant++;

}

else

{

num\_of\_variant = (int)(num\_of\_variant + num\_of\_variant \* 0.1);

}

if (for\_iterator)

iterator++;

for\_iterator = true;

massive\_of\_massives[iterator + 1] = new Abstract\_properties[num\_of\_variant];

for (int i = 0; i < old\_num\_variant; i++)

{

massive\_of\_massives[iterator + 1][i] = massive\_of\_massives[iterator][i];

}

}

int my\_rand = my\_rnd.Next(1, 50);

Thread.Sleep(100);

if (my\_rand >= 1 && my\_rand < 11)

{

massive\_of\_massives[iterator + 1][old\_num\_variant] = new circle();

}

else if (my\_rand >= 11 && my\_rand < 21)

{

massive\_of\_massives[iterator + 1][old\_num\_variant] = new ellipse();

}

else if (my\_rand >= 21 && my\_rand < 31)

{

massive\_of\_massives[iterator + 1][old\_num\_variant] = new rectangle();

}

else if (my\_rand >= 31 && my\_rand < 41)

{

massive\_of\_massives[iterator + 1][old\_num\_variant] = new square();

}

else if (my\_rand >= 41 && my\_rand < 51)

{

massive\_of\_massives[iterator + 1][old\_num\_variant] = new rhomb();

}

massive\_of\_massives[iterator + 1][old\_num\_variant].Show(gr, Color.Aquamarine);

old\_num\_variant++;

}

}

1. Уничтожение массива

public void collapsing\_massive(Graphics gc)

{

try

{

gc.Clear(Color.White);

for (int i = 0; i < 15; i++)

{

for (int j = 0; j < 40; i++)

{

massive\_of\_massives[i][j] = null;

}

}

}

catch

{

}

iterator = 0;

for\_iterator = false;

user\_iterator = 0;

massive\_is\_created = false;

user\_massive\_is\_created = false;

System.Windows.Forms.MessageBox.Show("Массив уничтожен!");

}

}

Для объекта класса-контейнера на основе односвязного списка:

public List\_objects(Abstract\_properties data)

{

Data = data;

}

public Abstract\_properties Data { get; set; }

public List\_objects Next { get; set; }

Для реализации класса-контейнера на основе односвязного списка:

1. Конструктор для создания списка по умолчанию

public LinkedList()

{

for (int i = 0; i < Form1.variant; i++)

{

int my\_rand = my\_rnd.Next(1, 50);

List\_objects my\_node;

if (my\_rand >= 1 && my\_rand < 11)

{

my\_node = new List\_objects(new circle());

}

else if (my\_rand >= 11 && my\_rand < 21)

{

my\_node = new List\_objects(new ellipse());

}

else if (my\_rand >= 21 && my\_rand < 31)

{

my\_node = new List\_objects(new rectangle());

}

else if (my\_rand >= 31 && my\_rand < 41)

{

my\_node = new List\_objects(new square());

}

else

my\_node = new List\_objects(new rhomb());

my\_rand = my\_rnd.Next(1, 50);

Thread.Sleep(10);

if (i == 0)

{

my\_node.Next = head;

head = my\_node;

if (count == 0)

tail = head;

count++;

}

else

{

if (head == null)

head = my\_node;

else

tail.Next = my\_node;

tail = my\_node;

count++;

}

}

}

1. Конструктор, создающий список по заданным параметрам

public LinkedList(int n)

{

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

{

int my\_rand = my\_rnd.Next(1, 50);

List\_objects my\_node;

if (my\_rand >= 1 && my\_rand < 11)

{

my\_node = new List\_objects(new circle());

}

else if (my\_rand >= 11 && my\_rand < 21)

{

my\_node = new List\_objects(new ellipse());

}

else if (my\_rand >= 21 && my\_rand < 31)

{

my\_node = new List\_objects(new rectangle());

}

else if (my\_rand >= 31 && my\_rand < 41)

{

my\_node = new List\_objects(new square());

}

else

my\_node = new List\_objects(new rhomb());

my\_rand = my\_rnd.Next(1, 50);

Thread.Sleep(10);

if (i == 0)

{

my\_node.Next = head;

head = my\_node;

if (count == 0)

tail = head;

count++;

}

else

{

if (head == null)

head = my\_node;

else

tail.Next = my\_node;

tail = my\_node;

count++;

}

}

}

1. Метод, для прорисовки фигур

public void iterator\_for\_show(Graphics gr)

{

List\_objects current = head;

while (current != null)

{

while (current != null)

{

if (current.Data is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is circle circ\_type)

{

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is square square\_type)

{

square\_type.square\_is\_visible = true;

square\_type.Show(gr, Color.Aquamarine);

}

current = current.Next;

}

current = current.Next;

}

}

1. Метод для очистки фигур

public void iterator\_for\_delete(Graphics gr)

{

List\_objects current = head;

gr.Clear(Color.White);

while (current != null)

{

if (current.Data is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = false;

}

else if (current.Data is circle circ\_type)

{

circ\_type.circle\_is\_visible = false;

}

else if (current.Data is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = false;

}

else if (current.Data is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = false;

}

else if (current.Data is square square\_type)

{

square\_type.square\_is\_visible = false;

}

current = current.Next;

}

}

1. Метод для перемещения фигур

public void iterator\_for\_move(int x, int y,Graphics gr)

{

List\_objects current = head;

gr.Clear(Color.White);

while (current != null)

{

if (current.Data is ellipse ell\_type)

{

ell\_type.Move(x, y);

ell\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is circle circ\_type)

{

circ\_type.Move(x, y);

circ\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is rectangle rect\_type)

{

rect\_type.Move(x, y);

rect\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is rhomb rh\_type)

{

rh\_type.Move(x, y);

rh\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is square square\_type)

{

square\_type.Move(x, y);

square\_type.Show(gr, Color.Aquamarine);

}

current = current.Next;

}

}

1. Метод для увеличения списка

public void expansion\_of\_list(Graphics gr)

{

int my\_rand = my\_rnd.Next(1, 50);

List\_objects added\_element;

if (my\_rand >= 1 && my\_rand < 11)

{

added\_element = new List\_objects(new circle());

}

else if (my\_rand >= 11 && my\_rand < 21)

{

added\_element = new List\_objects(new ellipse());

}

else if (my\_rand >= 21 && my\_rand < 31)

{

added\_element = new List\_objects(new rectangle());

}

else if (my\_rand >= 31 && my\_rand < 41)

{

added\_element = new List\_objects(new square());

}

else

added\_element = new List\_objects(new rhomb());

if (head == null)

head = added\_element;

else

tail.Next = added\_element;

tail = added\_element;

added\_element.Data.Show(gr, Color.Aquamarine);

count++;

1. Метод для добавления элемента в список

} public void Add(Abstract\_properties data)

{

List\_objects node = new List\_objects(data);

if (head == null)

head = node;

else

tail.Next = node;

tail = node;

count++;

1. Метод для удаления элемента списка

public bool Remove(Abstract\_properties data)

{

List\_objects current = head;

List\_objects previous = null;

while (current != null)

{

if (current.Data.Equals(data))

{

// Если узел в середине или в конце

if (previous != null)

{

previous.Next = current.Next;

if (current.Next == null)

tail = previous;

}

else

{

head = head.Next;

if (head == null)

tail = null;

}

count--;

return true;

}

previous = current;

current = current.Next;

}

return false;

}

1. Метод для очистки списка

public void Clear()

{

head = null;

tail = null;

count = 0;

}

1. Метод для добавления первого элемента

public void AppendFirst(Abstract\_properties data)

{

List\_objects node = new List\_objects(data);

node.Next = head;

head = node;

if (count == 0)

tail = head;

count++;

}

1. Методы для перечисления элементов списка

IEnumerator<Abstract\_properties> IEnumerable<Abstract\_properties>.GetEnumerator()

{

List\_objects current = head;

while (current != null)

{

yield return current.Data;

current = current.Next;

}

}

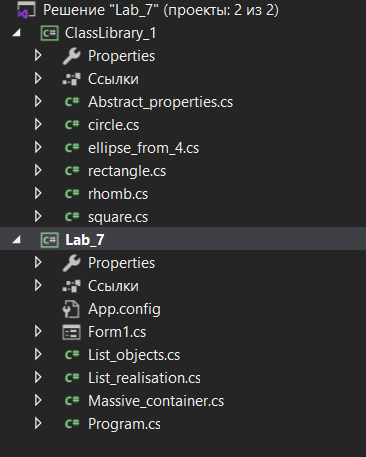
IEnumerator IEnumerable.GetEnumerator()

{

return ((IEnumerable)this).GetEnumerator();

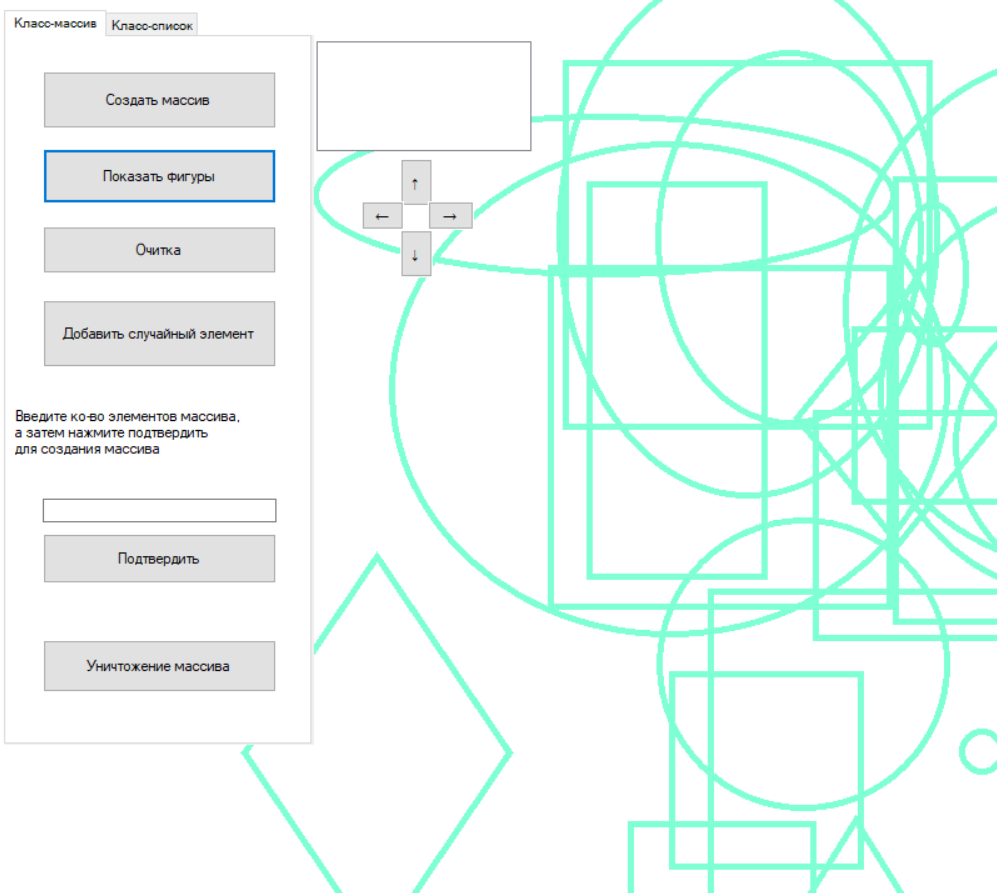
}

Структура программы:



Руководство пользования:

Имеются две вкладки для создания либо массива фигур, либо списка фигур. Можно создать случайны список/массив, а можно в TextBox ввести число фигур и нажать подтвердить для создания соответствующего списка/массива. Перемещение, очистка, показ фигур работают точно так же, как в предыдущих лабораторных работах.



Листинг программы:

*Класс-контейнер на основе массива:*

class Massive\_container

{

int iterator = 0;

bool for\_iterator = false;

static int user\_iterator = 0;

static int num\_of\_variant = 27;

int old\_num\_variant = num\_of\_variant;

int num = Form1.num\_of\_shapes;

static bool massive\_is\_created = false;

static bool user\_massive\_is\_created = false;

Random my\_rnd = new Random();

private Abstract\_properties[][] massive\_of\_massives = new Abstract\_properties[15][];

public Massive\_container()

{

if (massive\_is\_created)

{

return;

}

massive\_of\_massives[0] = new Abstract\_properties[num\_of\_variant];

int my\_rand = my\_rnd.Next(1, 50);

for (int i = 0; i < num\_of\_variant; i++)

{

if (my\_rand >= 1 && my\_rand < 11)

{

massive\_of\_massives[0][i] = new circle();

}

else if (my\_rand >= 11 && my\_rand < 21)

{

massive\_of\_massives[0][i] = new ellipse();

}

else if (my\_rand >= 21 && my\_rand < 31)

{

massive\_of\_massives[0][i] = new rectangle();

}

else if (my\_rand >= 31 && my\_rand < 41)

{

massive\_of\_massives[0][i] = new square();

}

else if (my\_rand >= 41 && my\_rand < 51)

{

massive\_of\_massives[0][i] = new rhomb();

}

my\_rand = my\_rnd.Next(1, 50);

Thread.Sleep(10);

}

massive\_is\_created = true;

System.Windows.Forms.MessageBox.Show("Массив создан!");

}// СОЗДАНИЕ МАССИВА СО СЛУЧАЙНЫМИ ФИГУРАМИ

public Massive\_container(int n)

{

if (massive\_is\_created)

{

return;

}

massive\_of\_massives[0] = new Abstract\_properties[n];

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

{

massive\_of\_massives[0][i] = null;

}

massive\_is\_created = true;

user\_massive\_is\_created = true;

}// КОНСТРУКТОР ДЛЯ СОЗДАНИЯ ПУСТОГО МАССИВА

public void iterator\_for\_show(Graphics gr)

{

if (!massive\_is\_created)

{

System.Windows.Forms.MessageBox.Show("Массив еще не создан!");

return;

}

else

{

if (!for\_iterator)

{

for (int i = 0; i < massive\_of\_massives[iterator].Length; i++)

{

if (massive\_of\_massives[iterator][i] is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is circle circ\_type)

{

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is square square\_type)

{

square\_type.square\_is\_visible = true;

square\_type.Show(gr, Color.Aquamarine);

}

}

}

else

{

for (int i = 0; i < massive\_of\_massives[iterator+1].Length; i++)

{

if (massive\_of\_massives[iterator + 1][i] is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator + 1][i] is circle circ\_type)

{

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator + 1][i] is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator + 1][i] is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator + 1][i] is square square\_type)

{

square\_type.square\_is\_visible = true;

square\_type.Show(gr, Color.Aquamarine);

}

}

}

}

} // ПОКАЗ ФИГУР

public void iterator\_for\_delete(Graphics gr)

{

if (!massive\_is\_created)

{

System.Windows.Forms.MessageBox.Show("Массив еще не создан!");

return;

}

gr.Clear(Color.White);

for (int i = 0; i < massive\_of\_massives[iterator].Length; i++)

{

if (massive\_of\_massives[iterator][i] is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = false;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is circle circ\_type)

{

circ\_type.circle\_is\_visible = false;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = false;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = false;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is square square\_type)

{

square\_type.square\_is\_visible = false;

square\_type.Show(gr, Color.Aquamarine);

}

}

} // УДАЛЕНИЕ

public void iterator\_for\_move(int x, int y, Graphics gr)

{

if (!massive\_is\_created)

{

System.Windows.Forms.MessageBox.Show("Массив еще не создан!");

return;

}

gr.Clear(Color.White);

if (!for\_iterator)

{

for (int i = 0; i < massive\_of\_massives[iterator].Length; i++)

{

if (massive\_of\_massives[iterator][i] is ellipse ell\_type)

{

ell\_type.Move(x, y);

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is circle circ\_type)

{

circ\_type.Move(x, y);

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rectangle rect\_type)

{

rect\_type.Move(x, y);

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is rhomb rh\_type)

{

rh\_type.Move(x, y);

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator][i] is square square\_type)

{

square\_type.Move(x, y);

square\_type.square\_is\_visible = true;

square\_type.Show(gr, Color.Aquamarine);

}

}

}

else

{

for (int i = 0; i < massive\_of\_massives[iterator+1].Length; i++)

{

if (massive\_of\_massives[iterator+1][i] is ellipse ell\_type)

{

ell\_type.Move(x, y);

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator+1][i] is circle circ\_type)

{

circ\_type.Move(x, y);

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator+1][i] is rectangle rect\_type)

{

rect\_type.Move(x, y);

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator+1][i] is rhomb rh\_type)

{

rh\_type.Move(x, y);

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (massive\_of\_massives[iterator+1][i] is square square\_type)

{

square\_type.Move(x, y);

square\_type.square\_is\_visible = true;

square\_type.Show(gr, Color.Aquamarine);

}

}

}

} // ПЕРЕМЕЩЕНИЕ

public void expansion\_of\_massive(Graphics gr)

{

if (!massive\_is\_created)

{

System.Windows.Forms.MessageBox.Show("Массив еще не создан!");

return;

}

if (user\_massive\_is\_created)

{

if (user\_iterator < Form1.num\_of\_shapes)

{

int my\_rand\_1 = my\_rnd.Next(1, 50);

if (my\_rand\_1 >= 1 && my\_rand\_1 < 11)

{

massive\_of\_massives[iterator][user\_iterator] = new circle();

}

else if (my\_rand\_1 >= 11 && my\_rand\_1 < 21)

{

massive\_of\_massives[iterator][user\_iterator] = new ellipse();

}

else if (my\_rand\_1 >= 21 && my\_rand\_1 < 31)

{

massive\_of\_massives[iterator][user\_iterator] = new rectangle();

}

else if (my\_rand\_1 >= 31 && my\_rand\_1 < 41)

{

massive\_of\_massives[iterator][user\_iterator] = new square();

}

else if (my\_rand\_1 >= 41 && my\_rand\_1 < 51)

{

massive\_of\_massives[iterator][user\_iterator] = new rhomb();

}

massive\_of\_massives[iterator][user\_iterator].Show(gr, Color.Aquamarine);

user\_iterator++;

}

else

{

if (user\_iterator == num)

{

System.Windows.Forms.MessageBox.Show("Массив переполнен! Добавил 10%");

if (num\*0.1 < 1)

{

num++;

}

else

{

num = (int)(num + num \* 0.1);

}

if (for\_iterator)

iterator++;

for\_iterator = true;

massive\_of\_massives[iterator + 1] = new Abstract\_properties[num];

for (int i = 0; i < massive\_of\_massives[iterator].Length; i++)

{

massive\_of\_massives[iterator + 1][i] = massive\_of\_massives[iterator][i];

}

}

int my\_rand = my\_rnd.Next(1, 50);

Thread.Sleep(100);

if (my\_rand >= 1 && my\_rand < 11)

{

massive\_of\_massives[iterator + 1][user\_iterator-1] = new circle();

}

else if (my\_rand >= 11 && my\_rand < 21)

{

massive\_of\_massives[iterator + 1][user\_iterator-1] = new ellipse();

}

else if (my\_rand >= 21 && my\_rand < 31)

{

massive\_of\_massives[iterator + 1][user\_iterator-1] = new rectangle();

}

else if (my\_rand >= 31 && my\_rand < 41)

{

massive\_of\_massives[iterator + 1][user\_iterator-1] = new square();

}

else if (my\_rand >= 41 && my\_rand < 51)

{

massive\_of\_massives[iterator + 1][user\_iterator-1] = new rhomb();

}

massive\_of\_massives[iterator + 1][user\_iterator-1].Show(gr, Color.Aquamarine);

user\_iterator++;

}

}

else

{

if (old\_num\_variant == num\_of\_variant)

{

System.Windows.Forms.MessageBox.Show("Массив переполнен! Добавил 10%");

if (num\_of\_variant \* 0.1 < 1)

{

num\_of\_variant++;

}

else

{

num\_of\_variant = (int)(num\_of\_variant + num\_of\_variant \* 0.1);

}

if (for\_iterator)

iterator++;

for\_iterator = true;

massive\_of\_massives[iterator + 1] = new Abstract\_properties[num\_of\_variant];

for (int i = 0; i < old\_num\_variant; i++)

{

massive\_of\_massives[iterator + 1][i] = massive\_of\_massives[iterator][i];

}

}

int my\_rand = my\_rnd.Next(1, 50);

Thread.Sleep(100);

if (my\_rand >= 1 && my\_rand < 11)

{

massive\_of\_massives[iterator + 1][old\_num\_variant] = new circle();

}

else if (my\_rand >= 11 && my\_rand < 21)

{

massive\_of\_massives[iterator + 1][old\_num\_variant] = new ellipse();

}

else if (my\_rand >= 21 && my\_rand < 31)

{

massive\_of\_massives[iterator + 1][old\_num\_variant] = new rectangle();

}

else if (my\_rand >= 31 && my\_rand < 41)

{

massive\_of\_massives[iterator + 1][old\_num\_variant] = new square();

}

else if (my\_rand >= 41 && my\_rand < 51)

{

massive\_of\_massives[iterator + 1][old\_num\_variant] = new rhomb();

}

massive\_of\_massives[iterator + 1][old\_num\_variant].Show(gr, Color.Aquamarine);

old\_num\_variant++;

}

} public void collapsing\_massive(Graphics gc)

{

try

{

gc.Clear(Color.White);

for (int i = 0; i < 15; i++)

{

for (int j = 0; j < 40; i++)

{

massive\_of\_massives[i][j] = null;

}

}

}

catch

{

}

iterator = 0;

for\_iterator = false;

user\_iterator = 0;

massive\_is\_created = false;

user\_massive\_is\_created = false;

System.Windows.Forms.MessageBox.Show("Массив уничтожен!");

}

}

}

*Класс контейнер на основе односвязного списка:*

public class LinkedList: IEnumerable<Abstract\_properties> // односвязный список

{

Random my\_rnd = new Random();

List\_objects head; // головной/первый элемент

List\_objects tail; // последний/хвостовой элемент

int count; // количество элементов в списке

public LinkedList()

{

for (int i = 0; i < Form1.variant; i++)

{

int my\_rand = my\_rnd.Next(1, 50);

List\_objects my\_node;

if (my\_rand >= 1 && my\_rand < 11)

{

my\_node = new List\_objects(new circle());

}

else if (my\_rand >= 11 && my\_rand < 21)

{

my\_node = new List\_objects(new ellipse());

}

else if (my\_rand >= 21 && my\_rand < 31)

{

my\_node = new List\_objects(new rectangle());

}

else if (my\_rand >= 31 && my\_rand < 41)

{

my\_node = new List\_objects(new square());

}

else

my\_node = new List\_objects(new rhomb());

my\_rand = my\_rnd.Next(1, 50);

Thread.Sleep(10);

if (i == 0)

{

my\_node.Next = head;

head = my\_node;

if (count == 0)

tail = head;

count++;

}

else

{

if (head == null)

head = my\_node;

else

tail.Next = my\_node;

tail = my\_node;

count++;

}

}

}

public LinkedList(int n)

{

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

{

int my\_rand = my\_rnd.Next(1, 50);

List\_objects my\_node;

if (my\_rand >= 1 && my\_rand < 11)

{

my\_node = new List\_objects(new circle());

}

else if (my\_rand >= 11 && my\_rand < 21)

{

my\_node = new List\_objects(new ellipse());

}

else if (my\_rand >= 21 && my\_rand < 31)

{

my\_node = new List\_objects(new rectangle());

}

else if (my\_rand >= 31 && my\_rand < 41)

{

my\_node = new List\_objects(new square());

}

else

my\_node = new List\_objects(new rhomb());

my\_rand = my\_rnd.Next(1, 50);

Thread.Sleep(10);

if (i == 0)

{

my\_node.Next = head;

head = my\_node;

if (count == 0)

tail = head;

count++;

}

else

{

if (head == null)

head = my\_node;

else

tail.Next = my\_node;

tail = my\_node;

count++;

}

}

}

public void iterator\_for\_show(Graphics gr)

{

List\_objects current = head;

while (current != null)

{

while (current != null)

{

if (current.Data is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is circle circ\_type)

{

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is square square\_type)

{

square\_type.square\_is\_visible = true;

square\_type.Show(gr, Color.Aquamarine);

}

current = current.Next;

}

current = current.Next;

}

}

public void iterator\_for\_delete(Graphics gr)

{

List\_objects current = head;

gr.Clear(Color.White);

while (current != null)

{

if (current.Data is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = false;

}

else if (current.Data is circle circ\_type)

{

circ\_type.circle\_is\_visible = false;

}

else if (current.Data is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = false;

}

else if (current.Data is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = false;

}

else if (current.Data is square square\_type)

{

square\_type.square\_is\_visible = false;

}

current = current.Next;

}

}

public void iterator\_for\_move(int x, int y,Graphics gr)

{

List\_objects current = head;

gr.Clear(Color.White);

while (current != null)

{

if (current.Data is ellipse ell\_type)

{

ell\_type.Move(x, y);

ell\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is circle circ\_type)

{

circ\_type.Move(x, y);

circ\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is rectangle rect\_type)

{

rect\_type.Move(x, y);

rect\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is rhomb rh\_type)

{

rh\_type.Move(x, y);

rh\_type.Show(gr, Color.Aquamarine);

}

else if (current.Data is square square\_type)

{

square\_type.Move(x, y);

square\_type.Show(gr, Color.Aquamarine);

}

current = current.Next;

}

}

public void expansion\_of\_list(Graphics gr)

{

if (count == Form1.num\_of\_shapes)

{

System.Windows.Forms.MessageBox.Show("Список заполнен! Теперь вы расширяете список");

}

int my\_rand = my\_rnd.Next(1, 50);

List\_objects added\_element;

if (my\_rand >= 1 && my\_rand < 11)

{

added\_element = new List\_objects(new circle());

}

else if (my\_rand >= 11 && my\_rand < 21)

{

added\_element = new List\_objects(new ellipse());

}

else if (my\_rand >= 21 && my\_rand < 31)

{

added\_element = new List\_objects(new rectangle());

}

else if (my\_rand >= 31 && my\_rand < 41)

{

added\_element = new List\_objects(new square());

}

else

added\_element = new List\_objects(new rhomb());

if (head == null)

head = added\_element;

else

tail.Next = added\_element;

tail = added\_element;

added\_element.Data.Show(gr, Color.Aquamarine);

count++;

}

class List\_objects

{

public List\_objects(Abstract\_properties data)

{

Data = data;

}

public Abstract\_properties Data { get; set; }

public List\_objects Next { get; set; }

}

*Основной класс формы:*

public partial class Form1 : Form // пытаюсь все испортить

{

public Form1()

{

InitializeComponent();

this.WindowState = FormWindowState.Maximized;

this.BackColor = System.Drawing.Color.White;

}

Graphics gc;

bool text\_box\_is\_wrote = false;

bool text\_box\_of\_list\_is\_wrote = false;

public static int num\_of\_shapes;

bool massive\_is\_created = false;

bool list\_is\_created = false;

bool user\_list\_is\_created = false;

int move\_pixels = 10;

public static int variant = 27;

int itr = 0;

Random my\_rnd = new Random();

Massive\_container[] massives = new Massive\_container[2];

LinkedList[] list\_massive = new LinkedList[2];

private void button1\_Click\_1(object sender, EventArgs e)

{

if (massive\_is\_created)

{

MessageBox.Show("Массив уже создан!");

}

massives[0] = new Massive\_container();

massive\_is\_created = true;

}// СОЗДАНИЕ МАССИВА

private void button3\_Click(object sender, EventArgs e)

{

if (text\_box\_is\_wrote)

{

massives[1].expansion\_of\_massive(gc = CreateGraphics());

}

else

massives[0].expansion\_of\_massive(gc = CreateGraphics());

} // РАСШИРЕНИЕ МАССИВА

private void button4\_Click(object sender, EventArgs e)

{

if (text\_box\_is\_wrote)

{

massives[1].iterator\_for\_delete(gc = CreateGraphics());

}

else

massives[0].iterator\_for\_delete(gc = CreateGraphics());

}// УДАЛЕНИЕ ФИГУР

private void button5\_Click(object sender, EventArgs e)

{

if (text\_box\_is\_wrote)

{

massives[1].iterator\_for\_show(gc = CreateGraphics());

}

else

massives[0].iterator\_for\_show(gc = CreateGraphics());

} // ПОКАЗ ФИГУР

private void button6\_Click(object sender, EventArgs e)

{

if (list\_is\_created)

{

gc.Clear(Color.White);

list\_massive[0].iterator\_for\_move(0,-move\_pixels, gc);

return;

}

else if (user\_list\_is\_created)

{

gc.Clear(Color.White);

list\_massive[1].iterator\_for\_move(0, -move\_pixels, gc);

return;

}

if (text\_box\_is\_wrote)

{

massives[1].iterator\_for\_move(0, -move\_pixels, gc);

}

else

massives[0].iterator\_for\_move(0, -move\_pixels, gc);

}// ДВИЖЕНИЕ ВВЕРХ

private void button9\_Click(object sender, EventArgs e)

{

if (list\_is\_created)

{

gc.Clear(Color.White);

list\_massive[0].iterator\_for\_move(0,move\_pixels, gc);

return;

}

else if (user\_list\_is\_created)

{

gc.Clear(Color.White);

list\_massive[1].iterator\_for\_move(0,move\_pixels, gc);

return;

}

if (text\_box\_is\_wrote)

{

massives[1].iterator\_for\_move(0, move\_pixels, gc);

}

else

massives[0].iterator\_for\_move(0, move\_pixels, gc);

}// ДВИЖЕНИЕ ВНИЗ

private void button8\_Click(object sender, EventArgs e)

{

if (list\_is\_created)

{

gc.Clear(Color.White);

list\_massive[0].iterator\_for\_move(move\_pixels, 0, gc);

return;

}

else if (user\_list\_is\_created)

{

gc.Clear(Color.White);

list\_massive[1].iterator\_for\_move(move\_pixels, 0, gc);

return;

}

if (text\_box\_is\_wrote)

{

massives[1].iterator\_for\_move(move\_pixels, 0, gc);

}

else

massives[0].iterator\_for\_move(move\_pixels, 0, gc);

}// ДВИЖЕНИЕ ВПРАВО

private void button7\_Click(object sender, EventArgs e)

{

if (list\_is\_created)

{

gc.Clear(Color.White);

list\_massive[0].iterator\_for\_move(-move\_pixels, 0, gc);

return;

}

else if(user\_list\_is\_created)

{

gc.Clear(Color.White);

list\_massive[1].iterator\_for\_move(-move\_pixels, 0, gc);

return;

}

if (text\_box\_is\_wrote)

{

massives[1].iterator\_for\_move(-move\_pixels, 0, gc);

}

else

massives[0].iterator\_for\_move(-move\_pixels, 0, gc);

}// ДВИЖЕНИЕ ВЛЕВО

private void button2\_Click(object sender, EventArgs e)

{

if(massive\_is\_created)

{

MessageBox.Show("Массив уже создан!");

return;

}

num\_of\_shapes = int.Parse(textBox1.Text);

massives[1] = new Massive\_container(num\_of\_shapes);

text\_box\_is\_wrote = true;

massive\_is\_created = true;

MessageBox.Show("Пустой массив создан!");

} // ПОДТВЕРЖДЕНИЕ ВВОДА

private void button11\_Click(object sender, EventArgs e)

{

if (text\_box\_is\_wrote)

{

massives[1].collapsing\_massive(gc);

}

else

massives[0].collapsing\_massive(gc);

text\_box\_is\_wrote = false;

massive\_is\_created = false;

}// УНИЧТОЖЕНИЕ МАССИВА

private void tabControl1\_Click(object sender, EventArgs e)

{

if (massive\_is\_created)

{

try { gc.Clear(Color.White); }

catch { };

if (text\_box\_is\_wrote)

{

massives[1].collapsing\_massive(gc);

}

else

massives[0].collapsing\_massive(gc);

text\_box\_is\_wrote = false;

massive\_is\_created = false;

}

else if (list\_is\_created||user\_list\_is\_created)

{

try

{

gc.Clear(Color.White);

}

catch

{

}

try

{

foreach (var item in list\_massive[0])

{

list\_massive[0].Remove(item);

}

foreach (var item in list\_massive[1])

{

list\_massive[1].Remove(item);

}

}

catch

{}

text\_box\_of\_list\_is\_wrote = false;

list\_is\_created = false;

user\_list\_is\_created = false;

itr = 0;

MessageBox.Show("Список уничтожен!");

}

} //НАЖАТИЕ ПО ВКЛАДКЕ

private void button10\_Click(object sender, EventArgs e) // СОЗДАНИЕ СПИСКА

{

if (user\_list\_is\_created || list\_is\_created)

{

MessageBox.Show("Список уже создан!");

return;

}

list\_massive[0] = new LinkedList();

foreach (var item in list\_massive[0])

{

item.Show(gc = CreateGraphics(), Color.Aquamarine);

}

list\_is\_created = true;

}

private void button13\_Click(object sender, EventArgs e)

{

gc.Clear(Color.White);

if (list\_is\_created)

{

foreach (var item in list\_massive[0])

{

if (item is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = false;

ell\_type.Show(gc, Color.Aquamarine);

}

else if (item is circle circ\_type)

{

circ\_type.circle\_is\_visible = false;

circ\_type.Show(gc, Color.Aquamarine);

}

else if (item is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = false;

rect\_type.Show(gc, Color.Aquamarine);

}

else if (item is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = false;

rh\_type.Show(gc, Color.Aquamarine);

}

else if (item is square square\_type)

{

square\_type.square\_is\_visible = false;

square\_type.Show(gc, Color.Aquamarine);

}

}

}

else if(user\_list\_is\_created)

{

foreach (var item in list\_massive[1])

{

if (item is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = false;

ell\_type.Show(gc, Color.Aquamarine);

}

else if (item is circle circ\_type)

{

circ\_type.circle\_is\_visible = false;

circ\_type.Show(gc, Color.Aquamarine);

}

else if (item is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = false;

rect\_type.Show(gc, Color.Aquamarine);

}

else if (item is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = false;

rh\_type.Show(gc, Color.Aquamarine);

}

else if (item is square square\_type)

{

square\_type.square\_is\_visible = false;

square\_type.Show(gc, Color.Aquamarine);

}

}

}

} //ОЧИСТКА ФИГУР В СПИСКЕ

private void button12\_Click(object sender, EventArgs e)

{

if (list\_is\_created)

{

foreach (var item in list\_massive[0])

{

if (item is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gc, Color.Aquamarine);

}

else if (item is circle circ\_type)

{

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gc, Color.Aquamarine);

}

else if (item is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gc, Color.Aquamarine);

}

else if (item is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gc, Color.Aquamarine);

}

else if (item is square square\_type)

{

square\_type.square\_is\_visible = true;

square\_type.Show(gc, Color.Aquamarine);

}

}

}

else if (user\_list\_is\_created)

{

foreach (var item in list\_massive[1])

{

if (item is ellipse ell\_type)

{

ell\_type.ellipse\_is\_visible = true;

ell\_type.Show(gc, Color.Aquamarine);

}

else if (item is circle circ\_type)

{

circ\_type.circle\_is\_visible = true;

circ\_type.Show(gc, Color.Aquamarine);

}

else if (item is rectangle rect\_type)

{

rect\_type.rectangle\_is\_visible = true;

rect\_type.Show(gc, Color.Aquamarine);

}

else if (item is rhomb rh\_type)

{

rh\_type.rhomb\_is\_visible = true;

rh\_type.Show(gc, Color.Aquamarine);

}

else if (item is square square\_type)

{

square\_type.square\_is\_visible = true;

square\_type.Show(gc, Color.Aquamarine);

}

}

}

} // ПОКАЗ ФИГУР В СПИСКЕ

private void button14\_Click(object sender, EventArgs e)

{

if (list\_is\_created || user\_list\_is\_created)

{

MessageBox.Show("Массив уже создан!");

return;

}

num\_of\_shapes = int.Parse(textBox2.Text);

list\_massive[1] = new LinkedList(num\_of\_shapes);

foreach (var item in list\_massive[1])

{

item.Show(gc = CreateGraphics(), Color.Aquamarine);

}

text\_box\_of\_list\_is\_wrote = true;

user\_list\_is\_created = true;

MessageBox.Show("Ваш список создан!");

} // СВОЙ СПИСОК

private void button15\_Click(object sender, EventArgs e)

{

if (list\_is\_created)

{

list\_massive[0].expansion\_of\_list(gc = CreateGraphics());

}

else if (user\_list\_is\_created)

{

list\_massive[1].expansion\_of\_list(gc = CreateGraphics());

}

else

MessageBox.Show("Вы еще не создали список!");

}// РАСШИРЕНИЕ СПИСКА

private void button16\_Click(object sender, EventArgs e)

{

gc.Clear(Color.White);

if (list\_is\_created)

{

foreach (var item in list\_massive[0])

{

list\_massive[0].Remove(item);

}

}

else if (user\_list\_is\_created)

{

foreach (var item in list\_massive[1])

{

list\_massive[1].Remove(item);

}

}

else

{

MessageBox.Show("Список еще не создан!");

}

text\_box\_of\_list\_is\_wrote = false;

list\_is\_created = false;

user\_list\_is\_created = false;

itr = 0;

MessageBox.Show("Список уничтожен!");

}

}