МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»

Кафедра автоматизированных систем обработки информации и управления

Отчет по лабораторной работе № 7

по дисциплине «Технологии программирования»

Тема: «Cписки»

Выполнил: Стычинский С.В., МВИ-121

Проверили: Кузьмина Т. М. и Адаев Р. Б.

Москва 2023

Задание

Задание состоит из двух частей. Варианты частей задания вычисляется следующим образом: Пусть n-номер студента в списке группы. Находим остаток от деления n на 4 и прибавляем единицу (n%4+1) получаем вариант первой части задания. Для определения варианта второй части задания нужно найти остаток от деления n на 5 и прибавить двойку (n%5+2), получим вариант второй части задания.

Мой вариант 6 для типа списка: очередь, и вариант 4 для элементов очереди.

Вариант 4. Класс, содержащий следующие члены.

a.Поле, в котором хранится имя элемента, имя запрашивается у пользователя в момент создания элемента;

b.Метод возвращающий «имя» элемента.

c.Конструктор класса с одним параметром

Вариант 6

Очередь.Первый и последний элементы должны быть закрашены как текущие.

На форме должны быть кнопки, выполняющие следующие действия:

a.Удалять элемент.

b.Вставлять элемент.

В первом задании определяется тип элемента списка. Во втором задании определяется вид списка. Поскольку будет строиться визуальная модель работы со списком, то элементы списка должны иметь фиксированную форму и размеры. Каждый студент сам определяет геометрическую фигуру (многоугольник или эллипс), которая будет изображать элемент списка. Многоугольники должны быть уникальны для каждого студента и нужного размера, поэтому выбор студента контролируется преподавателем. Способ рисования списка зависит от типа его элементов. Если элемент имеет координаты, то элемент рисуется с использованием этих координат, в этом случае, выводится порядковый номер элемента списка. Номер желательно нарисовать внутри фигуры, изображающей элемент. Если элемент не имеет координат, то у него есть «имя». В этом случае все элементы рисуются по горизонтали. Обозначим через xi координату х i-того элемента. Координаты первого элемента выбираются произвольно, а координата х2 вычисляется по формуле х2=х1+с, где с – ширина выбранного студентом многоугольника, х3=х2+с и т.д. Внутри многоугольника выводится «имя» элемента. Если, при задании элемента, нужна дополнительная информация, то на форму помещаются поля ввода. Например, если элемент содержит координаты, то эти координаты задает пользователь, либо используя поля ввода, либо 55 указывая мышкой на нужную точку (точнее, выполняя щелчок левой кнопкой мыши в нужной точке).

Теоретический материал

Список — это абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза. Значения, входящие в список называются элементами списка; если значение встречается несколько раз, каждое вхождение считается отдельным элементом. Список можно считать компьютерной реализацией математического понятия конечной последовательности. В программе, списки могут быть реализованы с помощью массивов; но эта реализация не является оптимальной, поскольку вставка или удаление элемента из середины списка весьма трудоёмкие операции. Чаще всего списки реализуются в виде отдельно расположенных элементов, в которых хранятся не только значения, но и ссылки на соседние элементы. Если список односвязный, то в первом элементе хранится ссылка на второй элемент, во втором - на третий и т.д. Если список односвязный кольцевой, то в последнем элементе хранится ссылка на первый элемент, в не кольцевом списке такой ссылки нет. Если список двусвязный, то в каждом элементе храниться две ссылки: на соседа с большим номером и на соседа меньшим номером. Двусвязные списки так же бывают кольцевыми и не кольцевыми. Во втором случае в первом элементе есть ссылка на второй, а в последнем – на первый. Список является чрезвычайно гибкой структурой т.к. его легко сделать большим или меньшим. Списки можно объединить или разорвать на меньшие списки. Элементы списка доступны для вставки, удлинения и просмотра. В определении списка часто используется понятие текущая позиция или курсор. Как правило, именно элемент, находящийся в текущей позиции, мы можем просмотреть или удалить, вставляется новый элемент, так же в текущую позицию или сразу в следующую за ней. При обращении к элементу по номеру, потребности в текущей позиции нет, но существует достаточно много алгоритмов, работающих со списками, в которых используется это понятие. В данной работе мы будем создавать визуальные модели различных видов списков.

Очередь

Очередь работает по принципу: первый вошёл – первый вышел. Пример очереди – это очередь в магазине, может быть в Интернет магазине. Пространство имен System.Collections содержит класс Queue, пространство имен System.Collections.Generic – обобщенный класс Queue , предназначенные для работы с очередями. Функции, выполняемые с очередями аналогичны функциям, работающим со стеками. Существенное отличие заключается в том, что вставка новых элементов осуществляется в конец очереди, а не в начало.

Queue q=new Queue();//Создание очереди

q.Enqueue(8);

//Добавление числа q.Enqueue(9);

//Добавление числа int i=(int)q.Dequeue();

//Просмотр и удаление первого элемента i=(int)q.Peek();

//Просмотр без удаления первого элемента

Этапы выполнения лабораторной работы

1. Выбрать многоугольник, который будет изображать элемент списка, согласовать свой выбор с преподавателем.

2. Опираясь на вторую часть задания, разработать интерфейс программы.

3. Описать классы «Элемент» и «Список элементов», дав им другие имена. При выполнении этого этапа, желательно использовать приведенные примеры.

4. Создать обработчики событий, позволяющие выполнить работу со списком, определенным второй частью задания.

5. Отладить программу

6. Написать отчет.

Распечатка программы

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics.Contracts;

using System.Drawing;

using System.Linq;

using System.Reflection.Emit;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ProgressBar;

namespace lab7

{

public partial class Form1 : Form

{

Queue myQueue = new Queue(); // 12 - размер накопителя для очереди

Random rnd = new Random(); // для генератора рандомных чисел

Boolean flag = false; //для отрисовки ряда чисел при запуске формы, реализовано в классе onPaint

public Form1()

{

InitializeComponent();

//label1.Text = "Накопитель очереди: " + myQueue.getCapacity().ToString();

label2.Text = "Элементов в очереди: " + myQueue.Size().ToString();

}

protected override void OnPaint(PaintEventArgs e) // специальный класс чтобы элементы сразу рисовались при запуске формы

{

if (!flag)

{

Graphics g = CreateGraphics();

myQueue.Draw(g);

flag = true; // onPaint вызывается постоянно, флаг нужен чтобы оно отрисовалась единажды при запуске

}

}

// Заполнение очереди рандомными значениям

// Невероятно гениальный генератор для заполнения очереди значениями, используется в классе button2\_Click

// Генерация длины от 1 до 2, и заполнение случайными символами посредством конвертации из ASCII

public string Generator()

{

int nameLength = rnd.Next(1, 3); // генерируем случайное число от 1 до 2

string name = "";

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

{

int asciiValue = rnd.Next(97, 123); // генерация случайного числа от 97 до 122

char character = (char)asciiValue; // конвертируем число в ASCII-символ

name += character.ToString(); // добавляем символ к имени

}

return name;

}

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

{

Graphics g = CreateGraphics();

myQueue.Enqueue(new unit(Generator().ToString()));

myQueue.Draw(g);

label2.Text = "Элементов в очереди: " + myQueue.Size().ToString();

}

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

{

Graphics g = CreateGraphics();

if (myQueue.Size() == 0)

{

MessageBox.Show("Ошибка. Очередь пуста.");

return;

}

myQueue.Dequeue();

myQueue.Draw(g);

label2.Text = "Элементов в очереди: " + myQueue.Size().ToString();

}

//Вариант 4. Класс, содержащий следующие члены.

//a.Поле, в котором хранится имя элемента, имя запрашивается у пользователя в момент создания элемента;

//b.Метод возвращающий «имя» элемента.

//c.Конструктор класса с одним параметром

//Вариант 6

//Очередь.Первый и последний элементы должны быть закрашены как текущие.

//На форме должны быть кнопки, выполняющие следующие действия:

//a.Удалять элемент.

//b.Вставлять элемент.

}

public class Queue

{

private Queue<unit> queue;

public Queue()

{

queue = new Queue<unit>();

}

public void Enqueue(unit item)

{

queue.Enqueue(item);

}

public unit Dequeue()

{

return queue.Dequeue();

}

public bool IsEmpty()

{

return queue.Count == 0;

}

public int Size()

{

return queue.Count;

}

public unit Peek()

{

return queue.Peek();

}

public void Draw(Graphics g) //Рисование элементов списка

{

if (queue.Count != 0)

{

g.Clear(Color.White);

g.FillEllipse(new SolidBrush(Color.LightBlue), 21, 50, 20, 20);

g.FillEllipse(new SolidBrush(Color.LightBlue), queue.Count \* 21, 50, 20, 20);

Pen pen = new Pen(Color.Red);

unit[] items = queue.ToArray();

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

{

//MessageBox.Show(i.ToString());

g.DrawString(items[i].getName(), new Font("Arial", 8), new

SolidBrush(Color.Black), (i+1) \* 21 + 3, 52);

g.DrawEllipse(pen, (i+1)\* 21, 50, 20, 20);

}

}

else

{

g.Clear(Color.White);

}

}

}

public class unit // класс элементов очереди

{

private string name;

public unit(string name)

{

this.name = name;

}

public string getName()

{

return name;

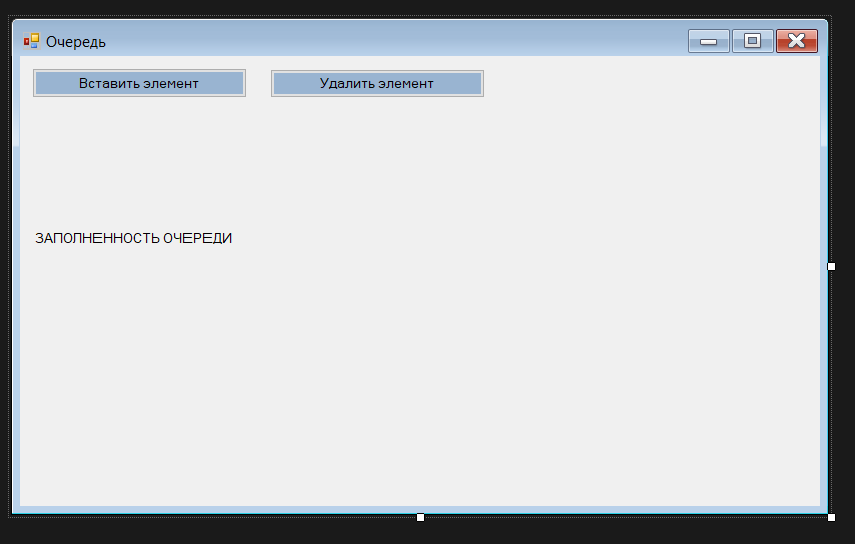
}

}

}

Результат работы программы

Форма:



Результат работы программы:

