МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)**

**Институт информационных технологий, математики и механики**

Направление подготовки: «Фундаментальная информатика и информационные технологии»

**ОТЧЕТ**

по лабораторной работе

**ДИНАМИЧЕСКАЯ СТРУКТУРА ХРАНЕНИЯ ДАННЫХ «ОЧЕРЕДЬ». РАБОТА ОЧЕРЕДИ НА КОЛЬЦЕВОМ БУФЕРЕ.**

**Выполнил:** студент группы

382006-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Н. Грачёв

Подпись

**Проверил:** к.ф.-м.н., доц.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.А. Баркалов

Подпись

Нижний Новгород  
2021

Оглавление

[1. Введение 2](#_Toc91447521)

[2. Постановка задачи 2](#_Toc91447522)

[3. Описание алгоритмов 2](#_Toc91447523)

[Описание алгоритма работы динамической структуры данных «Очередь». 2](#_Toc91447524)

[В каком виде хранить элементы в очереди 3](#_Toc91447525)

[Во время создания очереди 3](#_Toc91447526)

[Проверка на пустоту 4](#_Toc91447527)

[Проверка на переполнение 4](#_Toc91447528)

[Добавить элемент в очередь 4](#_Toc91447529)

[Извлечь элемент из очереди 4](#_Toc91447530)

[Очистка очереди 4](#_Toc91447531)

[Очередь на кольцевом буфере 5](#_Toc91447532)

[4. Описание программы 6](#_Toc91447533)

[Класс «TQueue» 6](#_Toc91447534)

[Тестирование класса при помощи Google Test 7](#_Toc91447535)

[Графический интерфейс – визуализация работы очереди на кольцевом буфере. 9](#_Toc91447536)

[5. Результаты 10](#_Toc91447537)

[6. Заключение 10](#_Toc91447538)

[7. Литература 10](#_Toc91447539)

[8. Приложение 11](#_Toc91447540)

[Queue/TQueue.h 11](#_Toc91447541)

[GoogleTest/QueueTest.cpp 13](#_Toc91447542)

[QueueForm/QForm.h 15](#_Toc91447543)

# Введение

Программист во время разработки программ часто сталкивается с ситуацией, когда необходимо извлекать элементы из некоторого множества в порядке их поступления в множество. Для решения данной задачи отлично подойдет динамическая структура хранения данных «Очередь». Для решения задач на С++ можно воспользоваться готовой очередью из STL (библиотекой стандартных шаблонов). Однако, в учебных целях имеет смысл написать собственный класс, чтобы увидеть во всех подробностях работу очереди.

# Постановка задачи

Задача: Необходимо разработать программу, которая будет содержать в себе файл с классом, реализующий основной функционал динамической структуры хранения данных «Очередь», соответствующие тесты класса, а также программу с пользовательским интерфейсом, которая будет наглядно отображать работу очереди на кольцевом буфере.

# Описание алгоритмов

## Описание алгоритма работы динамической структуры данных «Очередь».

Очередь работает по принципу FIFO (первым зашел – первым вышел). Таким образом, извлечь первый элемент из очереди мы сможем за одну операцию, а вот для того, чтобы добраться до последнего, мы должны извлечь все элементы из очереди.

Очередь должна обладать следующими основными функциями:

* Добавление элемента в конец очереди,
* Извлечение элемента из начала очереди.

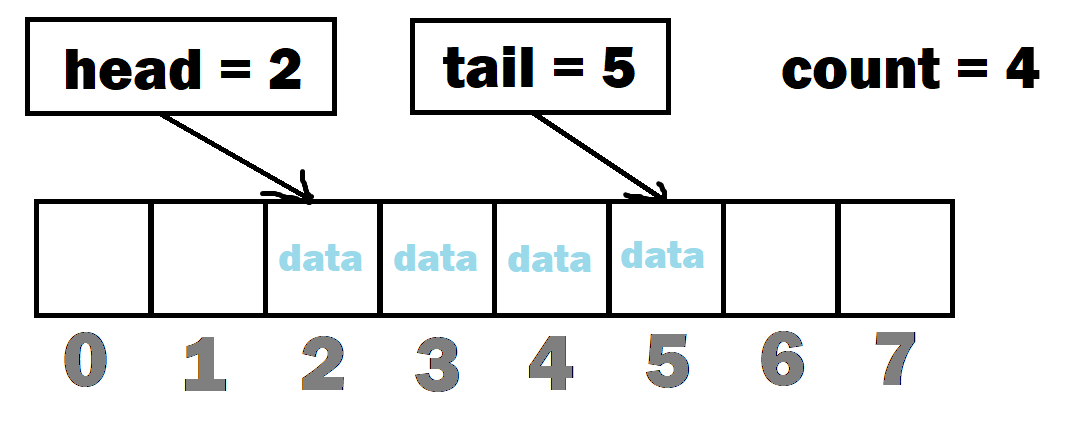
А также вспомогательными методами:

* Проверка на пустоту,
* Проверка на переполнение,
* Очистка очереди.

### В каком виде хранить элементы в очереди

Для хранения элементов в очереди будем использовать массив, под который будет динамически выделяться память на указанное количество элементов при создании очереди. Также понадобятся 4 переменных, первые две – head и tail - будут хранить индексы «головы» и «хвоста» очереди – индексы первого и последнего элемента в массиве, третья - count – количество элементов в очереди, четвертая – maxSize – максимальный размер очереди.

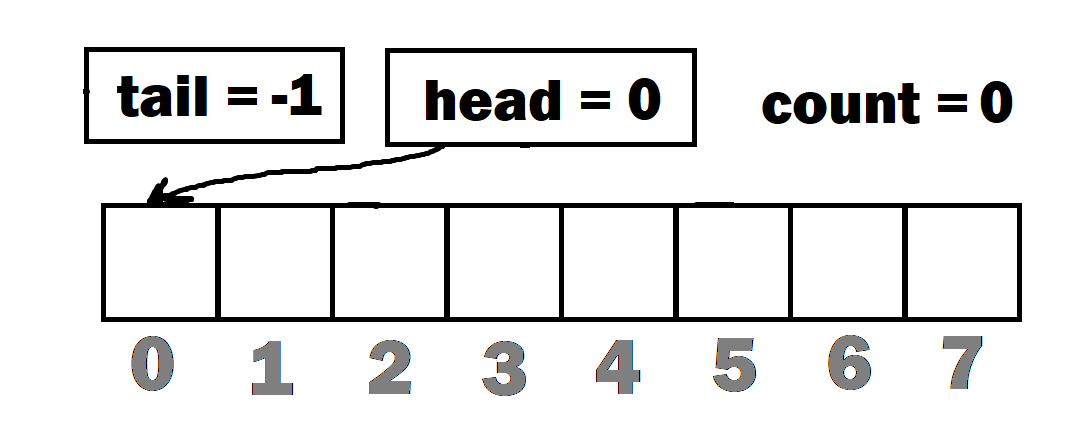
Так выглядит очередь на каком-то этапе своей работы:



### Во время создания очереди

Как только создается очередь, она является пустой, поэтому необходимо при создании переменным count и head присвоить 0, tail присвоить -1, выделить память под массив на maxSize элементов (maxSize указывается при создании очереди).

Только что созданная очередь выглядит так:



### Проверка на пустоту

Очередь является пустой, если количество элементов, находящихся в ней равно 0.

### Проверка на переполнение

Очередь переполнена когда количество элементов, находящихся в ней равно значению maxSize.

### Добавить элемент в очередь

Перед добавлением необходимо сделать проверку на переполнение очереди. В переполненную очередь мы не можем добавить элемент.

Чтобы добавить элемент со значением value в очередь, мы должны сначала увеличить на единицу tail, а затем элементу массива с индексом tail присвоить значение value. После этого необходимо увеличить count на 1.

### Извлечь элемент из очереди

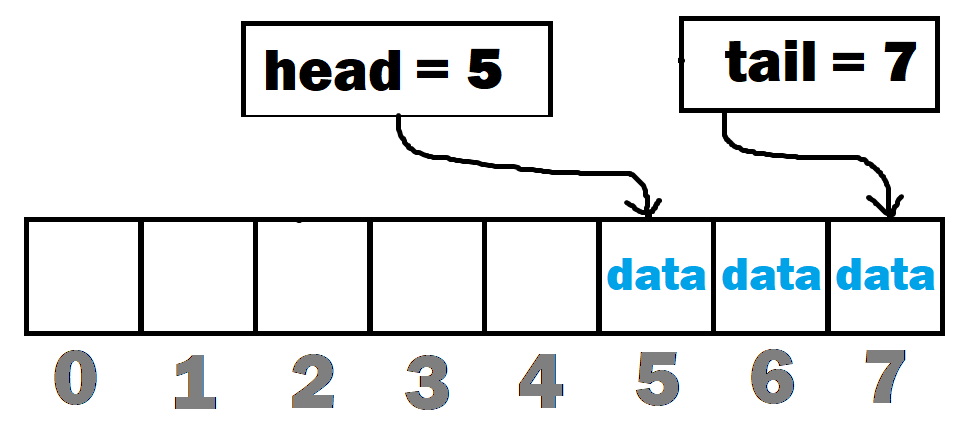
Перед извлечением необходимо сделать проверку на пустоту очереди. Из пустой очереди никакой элемент извлечь нельзя.

Чтобы извлечь элемент из очереди необходимо взять элемент по индексу head и вернуть его как результат. После этого head увеличить на единицу, а count уменьшить на единицу.

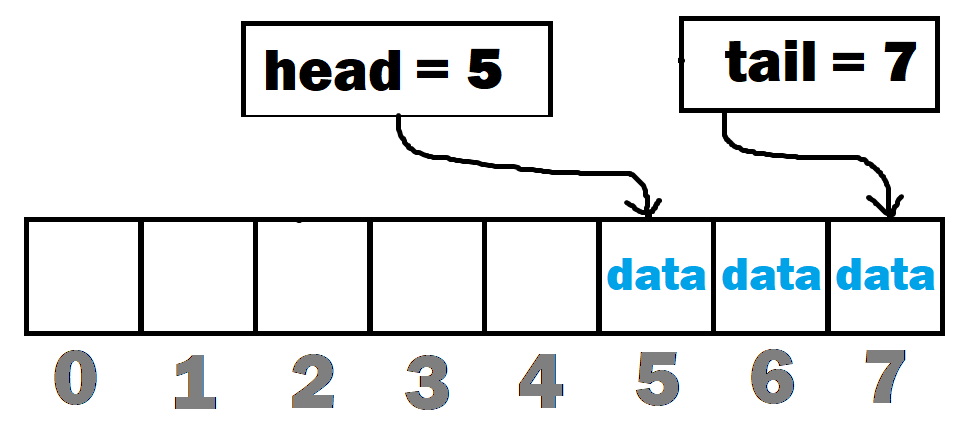
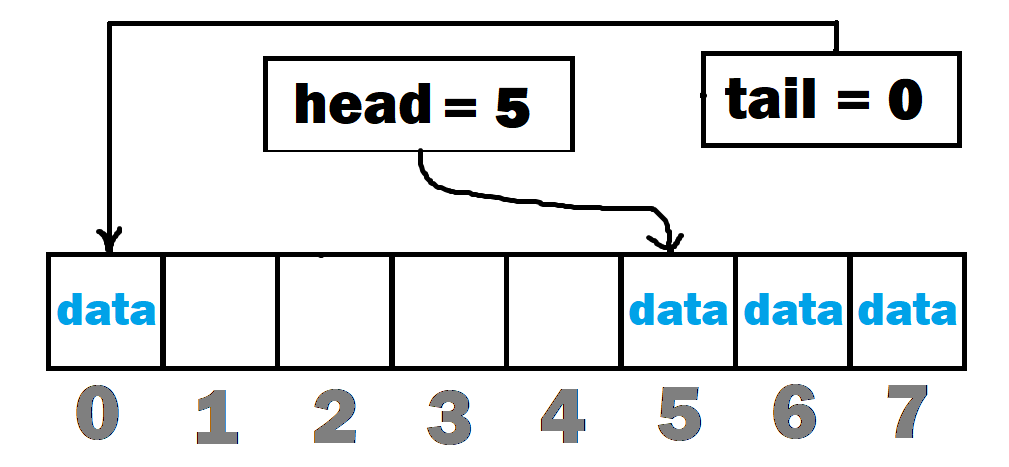
### Очистка очереди

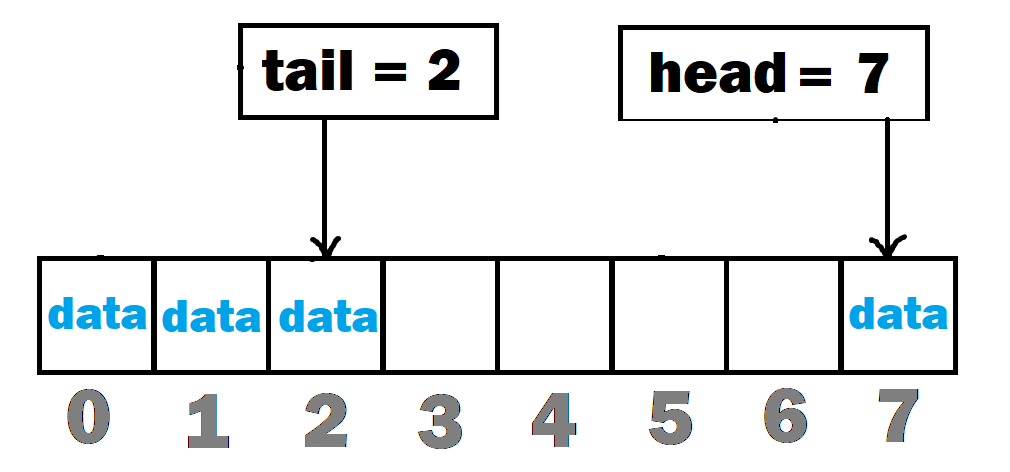
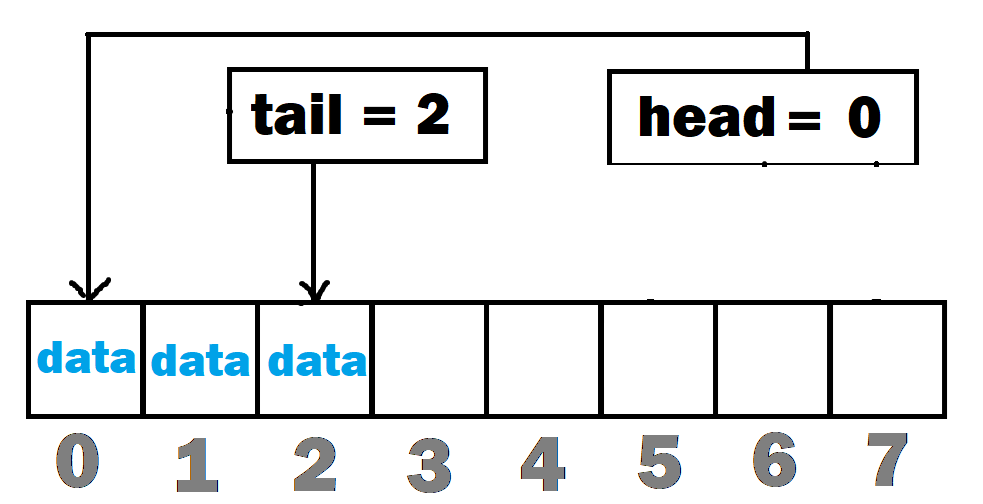
Необходимо извлекать из очереди элементы до тех пор, пока она не станет пустой.

### Очередь на кольцевом буфере

Если будем использовать описанный выше метод при добавлении/удалении элементов из очереди, то рано или поздно получим следующую картину:

При таком раскладе мы уже не сможем добавить элемент в массив, хотя у нас бо́льшая часть массива свободна. В таком случае логично «закольцевать» массив – то есть при добавлении нового элемента, если tail достиг конца массива переместить tail в начало. Аналогично поступаем и с head – если при извлечении переменная head имела индекс последнего элемента, то ей будет присвоен индекс первого элемента массива.

**Добавление:**

**Извлечение:**

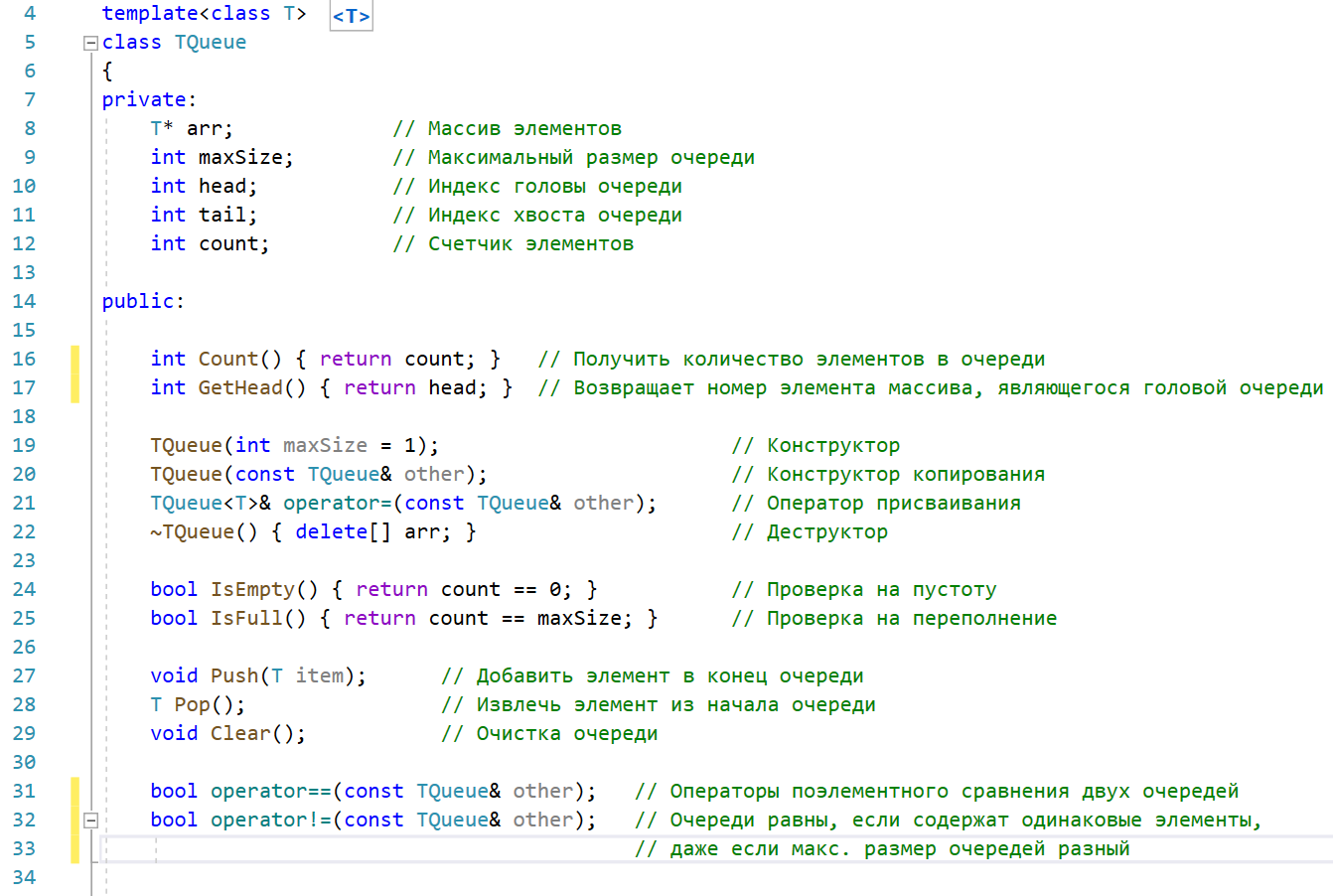
# Описание программы

Программа состоит из трех частей:

* Файл с классом «TQueue»,
* Тестирование работы класса при помощи Google Tests,
* Графический интерфейс программы, реализованный на Windows Forms – визуализация работы очереди на кольцевом буфере.

## Класс «TQueue»

Класс TQueue находится в файле «Queue/TQueue.h» и в нем имеются следующие поля и методы:

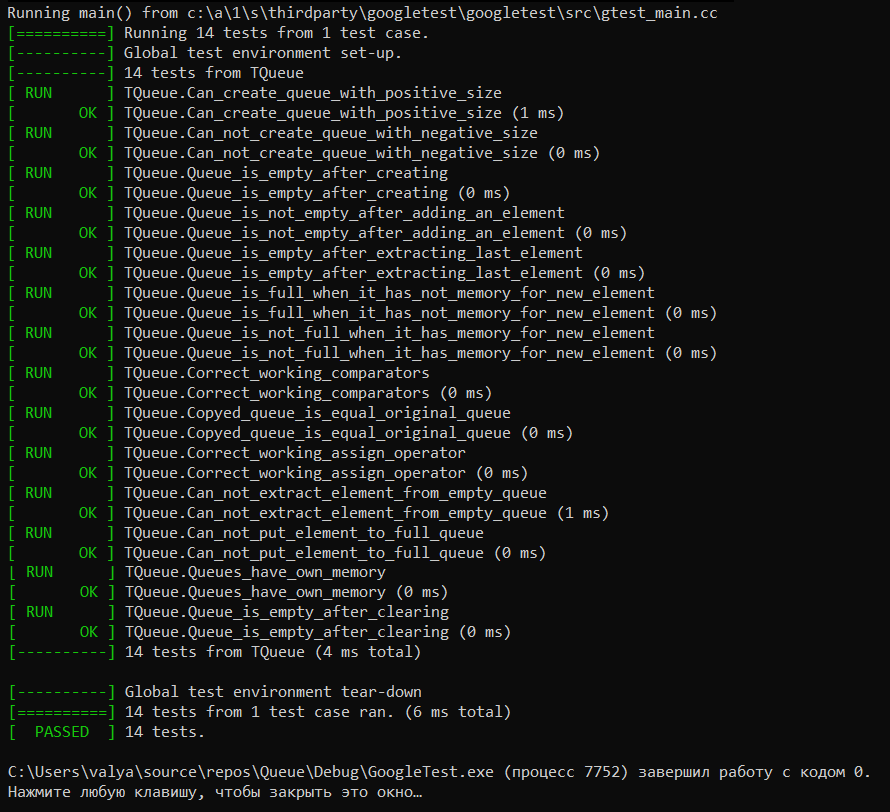


## Тестирование класса при помощи Google Test

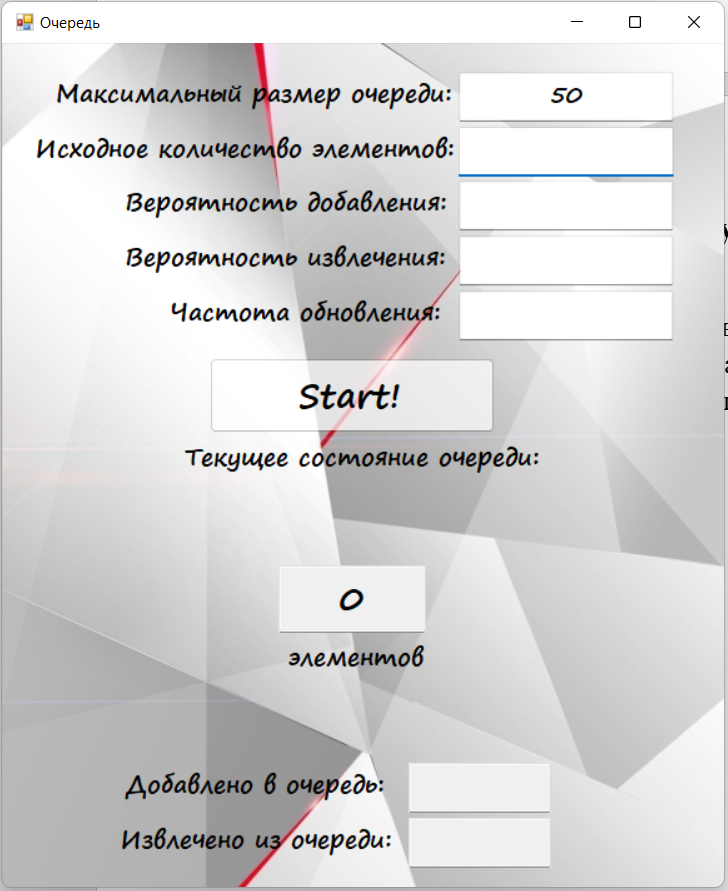
Тесты находятся в файле «GoogleTest/QueueTest.cpp». Были написаны следующие тесты:

* Можем создать очереди с положительным размером,
* Не можем создать очередь с отрицательным размером,
* Очередь пуста после создания,
* Очередь не пуста после добавления элемента,
* Очередь становится пустой после извлечения единственного элемента,
* Очередь переполнена, когда весь массив заполнен,
* Очередь не переполнена, когда у нее имеется еще место под новый элемент,
* Проверка корректности работы операторов сравнения на равенство,
* Скопированная очередь эквивалентна исходной,
* Проверка корректности работы оператора присваивания,
* Не можем извлечь элемент из пустой очереди,
* Не можем добавить элемент в переполненную очередь,
* Каждая очередь имеет собственную независимую память,
* Очередь становится пустой после очистки.

Как видим, все тесты успешно пройдены:



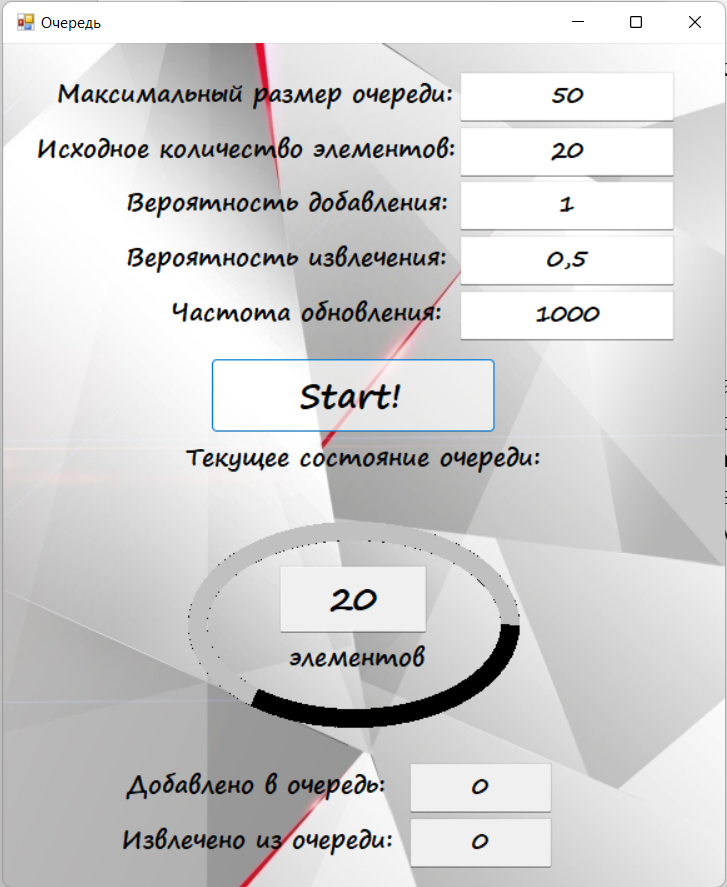
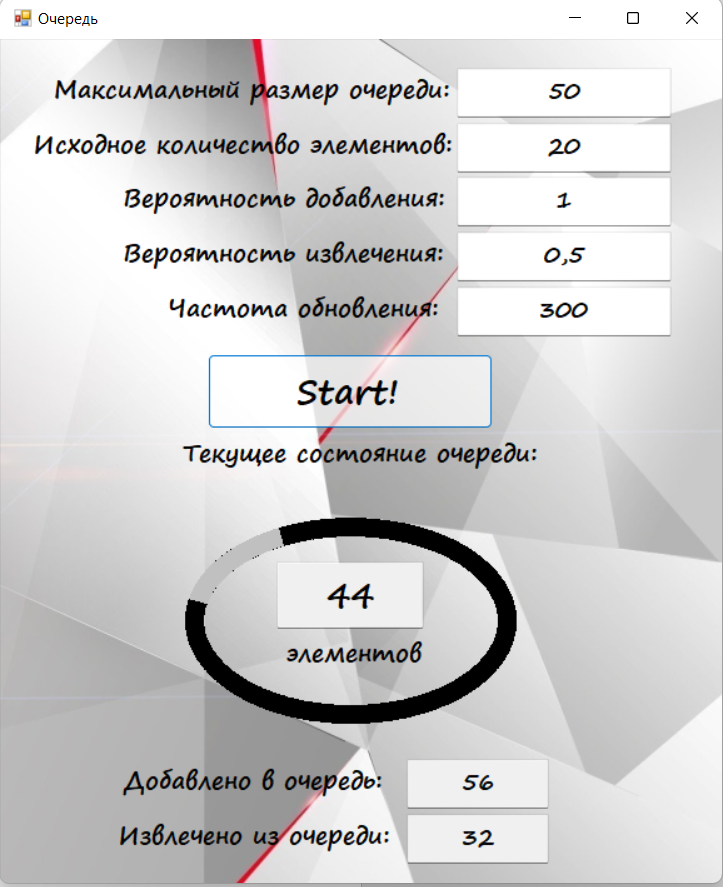
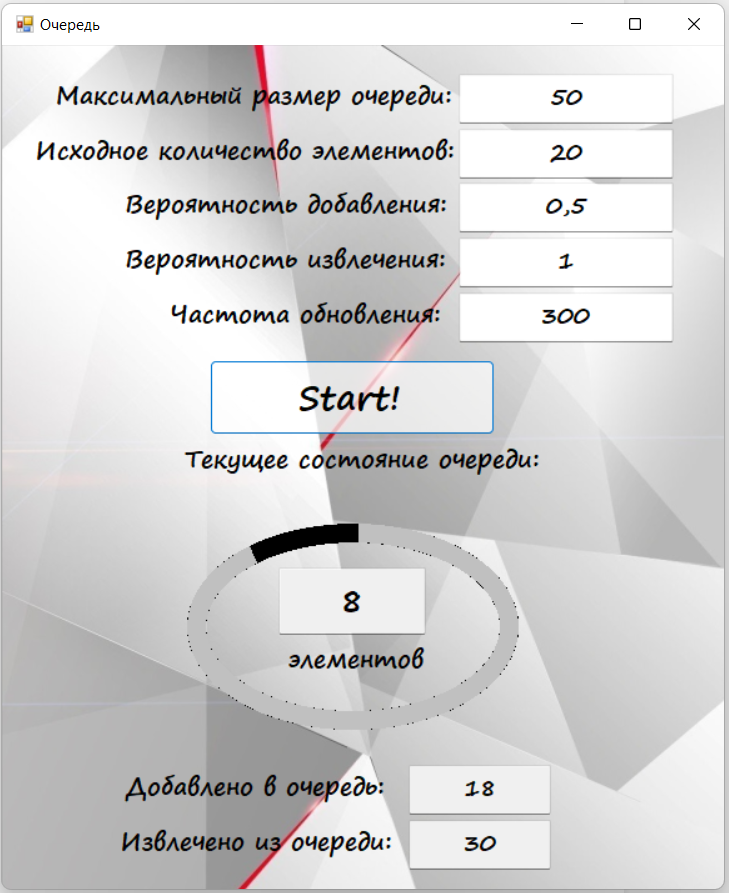
## Графический интерфейс – визуализация работы очереди на кольцевом буфере.

Массив в очереди можно представить в виде кольца, с некоторой закрашенной частью – этой частью будут занятые ячейки очереди. Отследить как происходит работа очереди поможет следующая программа:

В начале пользователь должен указать исходные параметры:

* Размер очереди
* Исходное количество элементов
* Вероятность добавления элемента
* Вероятность извлечения элемента
* Частоту обновления (в мс)

После нажатия кнопки «Start!» начнется работа программы: при добавлении элемента в очередь заполненная часть эллипса будет увеличиваться, при извлечении – уменьшаться.



Элементы извлекаются чаще чем добавляются

Элементы добавляются чаще чем извлекаются

Исходное состояние очереди

# Результаты

Был разработан класс, который реализует работу динамической структуры данных «очередь», используя закольцованный массив определенного размера. Были успешно пройдены тесты основных функций очереди. Также была написана программа с пользовательским интерфейсом, демонстрирующая работу очереди на кольцевом буфере.

# Заключение

Таким образом, мы увидели, как работает динамическая структура хранения данных «очередь» определенного размера на кольцевом буфере. В дальнейшем, от ограничения очереди в заранее заданный размер можно будет избавиться, если для хранения элементов вместо массива использовать списковую структуру.

# Литература

1. Microsoft [электронный ресурс]: // Документация по языку C++

URL: <https://docs.microsoft.com/ru-ru/cpp/cpp/>

1. Р. Лафоре Объектно-ориентированное программирование в C++ – Санкт-Петербург: “Питер”, 2018
2. Канал Simple Code, «Очередь, как структура данных»  
   <https://www.youtube.com/watch?v=fmHyFTji-Lc>
3. https://ru.wikipedia.org/wiki/Очередь\_(программирование)

***Ссылка на GitHub на репозиторий с проектом:***<https://github.com/TechnoWolf96/Lab-Queue_QueueVisualiser_ProcessorStation>

# Приложение

## Queue/TQueue.h

#pragma once

#include <iostream>

template<class T>

class TQueue

{

private:

T\* arr; // Массив элементов

int maxSize; // Максимальный размер очереди

int head; // Индекс головы очереди

int tail; // Индекс хвоста очереди

int count; // Счетчик элементов

public:

int Count() { return count; } // Получить количество элементов в очереди

int GetHead() { return head; } // Возвращает номер элемента массива, являющегося головой очереди

TQueue(int maxSize = 1); // Конструктор

TQueue(const TQueue& other); // Конструктор копирования

TQueue<T>& operator=(const TQueue& other); // Оператор присваивания

~TQueue() { delete[] arr; } // Деструктор

bool IsEmpty() { return count == 0; } // Проверка на пустоту

bool IsFull() { return count == maxSize; } // Проверка на переполнение

void Push(T item); // Добавить элемент в конец очереди

T Pop(); // Извлечь элемент из начала очереди

void Clear(); // Очистка очереди

bool operator==(const TQueue& other); // Операторы поэлементного сравнения двух очередей

bool operator!=(const TQueue& other); // Очереди равны, если содержат одинаковые элементы,

// даже если макс. размер очередей разный

friend std::ostream& operator<< (std::ostream& os, const TQueue& queue)

{

os << '[';

for (int i = 0; i < queue.count-1; i++)

{

int index = (queue.head + i) % queue.maxSize;

os << queue.arr[index] << ", ";

}

if (queue.count != 0)

os << queue.arr[(queue.head + queue.count - 1) % queue.maxSize] << ']';

else os << ']';

return os;

}

};

template<class T>

inline TQueue<T>::TQueue(int maxSize)

{

this->maxSize = maxSize;

arr = new T[maxSize];

head = 0;

tail = -1;

count = 0;

}

template<class T>

inline TQueue<T>::TQueue(const TQueue& other)

{

maxSize = other.maxSize;

arr = new T[maxSize];

head = other.head;

tail = other.tail;

count = other.count;

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

{

int index = (head + i) % maxSize;

arr[index] = other.arr[index];

}

}

template<class T>

inline TQueue<T>& TQueue<T>::operator=(const TQueue& other)

{

if (maxSize != other.maxSize)

{

delete[] arr;

maxSize = other.maxSize;

arr = new T[maxSize];

}

head = other.head;

tail = other.tail;

count = other.count;

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

{

int index = (head + i) % maxSize;

arr[index] = other.arr[index];

}

return \*this;

}

template<class T>

inline void TQueue<T>::Push(T item)

{

if (IsFull()) throw "Queue is full";

tail = (tail + 1) % maxSize;

arr[tail] = item;

count++;

}

template<class T>

inline T TQueue<T>::Pop()

{

if (IsEmpty()) throw "Queue is empty";

T temp = arr[head];

head = (head + 1) % maxSize;

count--;

return temp;

}

template<class T>

inline void TQueue<T>::Clear()

{

head = 0;

tail = -1;

count = 0;

}

template<class T>

inline bool TQueue<T>::operator==(const TQueue& other)

{

if (count != other.count) return false;

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

{

int index1 = (head + i) % maxSize;

int index2 = (other.head + i) % other.maxSize;

if (arr[index1] != other.arr[index2]) return false;

}

return true;

}

template<class T>

inline bool TQueue<T>::operator!=(const TQueue& other)

{

return !operator==(other);

}

## GoogleTest/QueueTest.cpp

#include "pch.h"

#include "../Queue/TQueue.h"

TEST(TQueue, Can\_create\_queue\_with\_positive\_size)

{

EXPECT\_NO\_THROW(TQueue<int> a(1));

}

TEST(TQueue, Can\_not\_create\_queue\_with\_negative\_size)

{

EXPECT\_ANY\_THROW(TQueue<int> a(-1));

}

TEST(TQueue, Queue\_is\_empty\_after\_creating)

{

TQueue<int> a(3);

EXPECT\_TRUE(a.IsEmpty());

}

TEST(TQueue, Queue\_is\_not\_empty\_after\_adding\_an\_element)

{

TQueue<int> a(3);

a.Push(0);

EXPECT\_TRUE(!a.IsEmpty());

}

TEST(TQueue, Queue\_is\_empty\_after\_extracting\_last\_element)

{

TQueue<int> a(3);

a.Push(0);

a.Pop();

EXPECT\_TRUE(a.IsEmpty());

}

TEST(TQueue, Queue\_is\_full\_when\_it\_has\_not\_memory\_for\_new\_element)

{

TQueue<int> a(3);

a.Push(0), a.Push(0), a.Push(0);

EXPECT\_TRUE(a.IsFull());

}

TEST(TQueue, Queue\_is\_not\_full\_when\_it\_has\_memory\_for\_new\_element)

{

TQueue<int> a(4);

a.Push(0), a.Push(0), a.Push(0);

EXPECT\_TRUE(!a.IsFull());

}

TEST(TQueue, Correct\_working\_comparators)

{

TQueue<int> a(4);

a.Push(0), a.Push(1), a.Push(2);

a.Pop(), a.Pop(), a.Push(3);

TQueue<int> b(5);

b.Push(2), b.Push(3);

EXPECT\_TRUE(b == a);

EXPECT\_FALSE(b != a);

}

TEST(TQueue, Copyed\_queue\_is\_equal\_original\_queue)

{

TQueue<int> a(4);

a.Push(0), a.Push(1), a.Push(2);

TQueue<int> b(a);

EXPECT\_TRUE(b == a);

}

TEST(TQueue, Correct\_working\_assign\_operator)

{

TQueue<int> a(4);

a.Push(0), a.Push(1), a.Push(2);

TQueue<int> b(1);

b = a;

EXPECT\_TRUE(b == a);

}

TEST(TQueue, Can\_not\_extract\_element\_from\_empty\_queue)

{

TQueue<int> a(3);

ASSERT\_ANY\_THROW(a.Pop());

}

TEST(TQueue, Can\_not\_put\_element\_to\_full\_queue)

{

TQueue<int> a(3);

a.Push(0), a.Push(1), a.Push(2);

ASSERT\_ANY\_THROW(a.Push(3));

}

TEST(TQueue, Queues\_have\_own\_memory)

{

TQueue<int> a(3);

a.Push(0), a.Push(1), a.Push(2);

TQueue<int> b(3);

b = a;

b.Pop(); b.Push(3);

ASSERT\_TRUE(a != b);

}

TEST(TQueue, Queue\_is\_empty\_after\_clearing)

{

TQueue<int> a(3);

a.Push(0), a.Push(1), a.Push(2);

a.Clear();

ASSERT\_TRUE(a.IsEmpty());

}

## QueueForm/QForm.h

#pragma once

#include "../Queue/TQueue.h"

namespace QueueForm {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

TQueue<int> queue;

int maxSize;

float probAdd;

float probExtract;

int added;

int extracted;

/// <summary>

/// Сводка для QForm

/// </summary>

public ref class QForm : public System::Windows::Forms::Form

{

Random^ rand;

private: System::Windows::Forms::Timer^ timer1;

private: System::Windows::Forms::PictureBox^ pictureBox1;

private: System::Windows::Forms::Label^ label10;

Graphics^ graphicsForm;

public:

QForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~QForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Label^ label1;

protected:

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::TextBox^ textBox4;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::TextBox^ textBox5;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::TextBox^ textBox6;

private: System::Windows::Forms::TextBox^ textBox7;

private: System::Windows::Forms::Label^ label8;

private: System::Windows::Forms::TextBox^ textBox8;

private: System::Windows::Forms::Label^ label9;

private: System::ComponentModel::IContainer^ components;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->components = (gcnew System::ComponentModel::Container());

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(QForm::typeid));

this->label1 = (gcnew System::Windows::Forms::Label());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->label2 = (gcnew System::Windows::Forms::Label());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->textBox4 = (gcnew System::Windows::Forms::TextBox());

this->label4 = (gcnew System::Windows::Forms::Label());

this->textBox5 = (gcnew System::Windows::Forms::TextBox());

this->label5 = (gcnew System::Windows::Forms::Label());

this->label6 = (gcnew System::Windows::Forms::Label());

this->button1 = (gcnew System::Windows::Forms::Button());

this->label7 = (gcnew System::Windows::Forms::Label());

this->textBox6 = (gcnew System::Windows::Forms::TextBox());

this->textBox7 = (gcnew System::Windows::Forms::TextBox());

this->label8 = (gcnew System::Windows::Forms::Label());

this->textBox8 = (gcnew System::Windows::Forms::TextBox());

this->label9 = (gcnew System::Windows::Forms::Label());

this->timer1 = (gcnew System::Windows::Forms::Timer(this->components));

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

this->label10 = (gcnew System::Windows::Forms::Label());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->BackColor = System::Drawing::Color::Transparent;

this->label1->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(49, 28);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(416, 43);

this->label1->TabIndex = 0;

this->label1->Text = L"Максимальный размер очереди:";

//

// textBox1

//

this->textBox1->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->textBox1->Location = System::Drawing::Point(487, 28);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(228, 48);

this->textBox1->TabIndex = 1;

this->textBox1->Text = L"50";

this->textBox1->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

this->textBox1->TextChanged += gcnew System::EventHandler(this, &QForm::textBox1\_TextChanged);

//

// textBox2

//

this->textBox2->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->textBox2->Location = System::Drawing::Point(487, 82);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(228, 48);

this->textBox2->TabIndex = 3;

this->textBox2->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// label2

//

this->label2->AutoSize = true;

this->label2->BackColor = System::Drawing::Color::Transparent;

this->label2->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label2->Location = System::Drawing::Point(26, 82);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(439, 43);

this->label2->TabIndex = 2;

this->label2->Text = L"Исходное количество элементов:";

//

// textBox3

//

this->textBox3->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->textBox3->Location = System::Drawing::Point(487, 136);

this->textBox3->Name = L"textBox3";

this->textBox3->Size = System::Drawing::Size(228, 48);

this->textBox3->TabIndex = 5;

this->textBox3->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// label3

//

this->label3->AutoSize = true;

this->label3->BackColor = System::Drawing::Color::Transparent;

this->label3->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(123, 136);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(342, 43);

this->label3->TabIndex = 4;

this->label3->Text = L"Вероятность добавления:";

//

// textBox4

//

this->textBox4->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->textBox4->Location = System::Drawing::Point(487, 190);

this->textBox4->Name = L"textBox4";

this->textBox4->Size = System::Drawing::Size(228, 48);

this->textBox4->TabIndex = 7;

this->textBox4->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// label4

//

this->label4->AutoSize = true;

this->label4->BackColor = System::Drawing::Color::Transparent;

this->label4->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label4->Location = System::Drawing::Point(123, 190);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(340, 43);

this->label4->TabIndex = 6;

this->label4->Text = L"Вероятность извлечения:";

//

// textBox5

//

this->textBox5->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->textBox5->Location = System::Drawing::Point(487, 244);

this->textBox5->Name = L"textBox5";

this->textBox5->Size = System::Drawing::Size(228, 48);

this->textBox5->TabIndex = 9;

this->textBox5->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// label5

//

this->label5->AutoSize = true;

this->label5->BackColor = System::Drawing::Color::Transparent;

this->label5->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label5->Location = System::Drawing::Point(171, 244);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(292, 43);

this->label5->TabIndex = 8;

this->label5->Text = L"Частота обновления:";

//

// label6

//

this->label6->AutoSize = true;

this->label6->BackColor = System::Drawing::Color::Transparent;

this->label6->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label6->Location = System::Drawing::Point(186, 387);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(372, 43);

this->label6->TabIndex = 10;

this->label6->Text = L"Текущее состояние очереди:";

//

// button1

//

this->button1->Font = (gcnew System::Drawing::Font(L"Segoe Print", 19.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button1->Location = System::Drawing::Point(221, 310);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(304, 74);

this->button1->TabIndex = 11;

this->button1->Text = L"Start!";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &QForm::button1\_Click);

//

// label7

//

this->label7->AutoSize = true;

this->label7->BackColor = System::Drawing::Color::Transparent;

this->label7->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label7->Location = System::Drawing::Point(297, 584);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(154, 43);

this->label7->TabIndex = 12;

this->label7->Text = L"элементов";

this->label7->Click += gcnew System::EventHandler(this, &QForm::label7\_Click);

//

// textBox6

//

this->textBox6->BackColor = System::Drawing::SystemColors::Menu;

this->textBox6->Font = (gcnew System::Drawing::Font(L"Segoe Print", 19.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox6->Location = System::Drawing::Point(295, 515);

this->textBox6->Name = L"textBox6";

this->textBox6->ReadOnly = true;

this->textBox6->Size = System::Drawing::Size(156, 66);

this->textBox6->TabIndex = 13;

this->textBox6->Text = L"0";

this->textBox6->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// textBox7

//

this->textBox7->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->textBox7->Location = System::Drawing::Point(433, 763);

this->textBox7->Name = L"textBox7";

this->textBox7->ReadOnly = true;

this->textBox7->Size = System::Drawing::Size(150, 48);

this->textBox7->TabIndex = 17;

this->textBox7->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// label8

//

this->label8->AutoSize = true;

this->label8->BackColor = System::Drawing::Color::Transparent;

this->label8->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label8->Location = System::Drawing::Point(117, 763);

this->label8->Name = L"label8";

this->label8->Size = System::Drawing::Size(290, 43);

this->label8->TabIndex = 16;

this->label8->Text = L"Извлечено из очереди:";

//

// textBox8

//

this->textBox8->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->textBox8->Location = System::Drawing::Point(433, 709);

this->textBox8->Name = L"textBox8";

this->textBox8->ReadOnly = true;

this->textBox8->Size = System::Drawing::Size(150, 48);

this->textBox8->TabIndex = 15;

this->textBox8->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// label9

//

this->label9->AutoSize = true;

this->label9->BackColor = System::Drawing::Color::Transparent;

this->label9->Font = (gcnew System::Drawing::Font(L"Segoe Print", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label9->Location = System::Drawing::Point(123, 709);

this->label9->Name = L"label9";

this->label9->Size = System::Drawing::Size(279, 43);

this->label9->TabIndex = 14;

this->label9->Text = L"Добавлено в очередь:";

//

// timer1

//

this->timer1->Tick += gcnew System::EventHandler(this, &QForm::timer1\_Tick);

//

// pictureBox1

//

this->pictureBox1->BackColor = System::Drawing::Color::Transparent;

this->pictureBox1->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox1.BackgroundImage")));

this->pictureBox1->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->pictureBox1->Location = System::Drawing::Point(34, 236);

this->pictureBox1->Name = L"pictureBox1";

this->pictureBox1->Size = System::Drawing::Size(131, 128);

this->pictureBox1->TabIndex = 18;

this->pictureBox1->TabStop = false;

this->pictureBox1->Visible = false;

//

// label10

//

this->label10->AutoSize = true;

this->label10->BackColor = System::Drawing::Color::Transparent;

this->label10->Font = (gcnew System::Drawing::Font(L"Segoe Print", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label10->ForeColor = System::Drawing::Color::DarkRed;

this->label10->Location = System::Drawing::Point(12, 367);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(176, 70);

this->label10->TabIndex = 19;

this->label10->Text = L"Неверно заданы\r\nпараметры!";

this->label10->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

this->label10->Visible = false;

//

// QForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"$this.BackgroundImage")));

this->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->ClientSize = System::Drawing::Size(770, 832);

this->Controls->Add(this->label10);

this->Controls->Add(this->pictureBox1);

this->Controls->Add(this->textBox7);

this->Controls->Add(this->label8);

this->Controls->Add(this->textBox8);

this->Controls->Add(this->label9);

this->Controls->Add(this->textBox6);

this->Controls->Add(this->label7);

this->Controls->Add(this->button1);

this->Controls->Add(this->label6);

this->Controls->Add(this->textBox5);

this->Controls->Add(this->label5);

this->Controls->Add(this->textBox4);

this->Controls->Add(this->label4);

this->Controls->Add(this->textBox3);

this->Controls->Add(this->label3);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->label2);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->label1);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::Fixed3D;

this->Name = L"QForm";

this->Text = L"Очередь";

this->Load += gcnew System::EventHandler(this, &QForm::QForm\_Load);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void label7\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

try

{

timer1->Interval = Convert::ToInt32(textBox5->Text);

rand = gcnew Random();

added = 0;

extracted = 0;

textBox8->Text = added.ToString();

textBox7->Text = extracted.ToString();

// Запоминание введенных данных

maxSize = Convert::ToInt32(textBox1->Text);

int beginCount = Convert::ToInt32(textBox2->Text);

probAdd = Convert::ToDouble(textBox3->Text);

probExtract = Convert::ToDouble(textBox4->Text);

// Создание очереди

queue = TQueue<int>(maxSize);

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

queue.Push(1);

DrawQueue();

pictureBox1->Visible = false;

label10->Visible = false;

//Запуск таймера

timer1->Enabled = true;

}

catch (...)

{

pictureBox1->Visible = true;

label10->Visible = true;

}

}

void DrawQueue()

{

// Вычисление углов

float startAngle = 360 \* ((float)queue.GetHead() / (float)maxSize);

float finishAngle = 360 \* ((float)queue.Count() / (float)maxSize);

// Красим серый эллипс

Pen^ whitePen = gcnew Pen(Color::Silver);

whitePen->Width = 15.0f;

graphicsForm->DrawEllipse(whitePen, 155, 390, 250, 150);

// Красим текущее состояние

Pen^ blackPen = gcnew Pen(Color::Black);

blackPen->Width = 15.0f;

graphicsForm->DrawArc(blackPen, 155.0, 390.0, 250.0, 150.0, startAngle, finishAngle);

textBox6->Text = queue.Count().ToString();

}

private: System::Void timer1\_Tick(System::Object^ sender, System::EventArgs^ e) {

if (probAdd > rand->NextDouble() && !queue.IsFull())

{

queue.Push(1);

added++;

textBox8->Text = added.ToString();

}

if (probExtract > rand->NextDouble() && !queue.IsEmpty())

{

queue.Pop();

extracted++;

textBox7->Text = extracted.ToString();

}

DrawQueue();

}

private: System::Void QForm\_Load(System::Object^ sender, System::EventArgs^ e) {

graphicsForm = this->CreateGraphics();

Pen^ whitePen = gcnew Pen(Color::Silver);

whitePen->Width = 15.0f;

graphicsForm->DrawEllipse(whitePen, 155, 390, 250, 150);

}

private: System::Void textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

};

}