Об’єктно орієнтоване програмування

**Лабораторна робота №3**

# Черга

# Теоретичні відомості

Черга – структура даних з типом доступу до елементів «перший прийшов – першим вийшов» (FIFO, First In – First Out). Додавання елемента можливо лише в кінець черги, а витяг – тільки з початоку черги, при цьому обраний елемент з черги видаляється. У різних бібліотеках методи додавання і вилучення елементів в чергу можуть називатися по-різному. Часто для додавання використовують назву методів push або enqueue, а для вилучення – pop або dequeue (рис. 1).

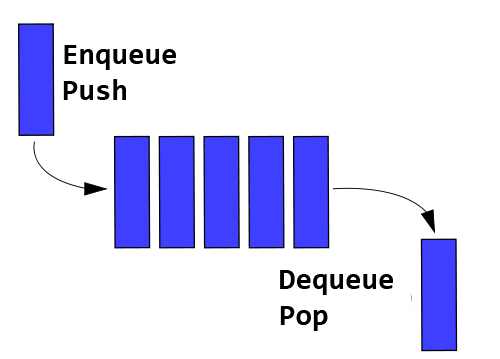


Рисунок 3.1 – Графічне зображення черги

**Приклад.** Написати програму, яка реалізує чергу для зберігання цілих чисел. Користувач може поміщати і отримувати дані з черги. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан.

Лістинг 3.1 – Файл queue.h

#ifndef QUEUE\_H\_INCLUDED

#define QUEUE\_H\_INCLUDED

#include <QTableWidget>

class **Queue**

{

int \*q; // Черга

int qEnd; // Поточний розмір черги

int maxQLength; // Максимальний розмір черги

QTableWidget \*qtable; // QTableWidget для відображення

public:

**Queue**(int MaxLength);// Конструктор

~**Queue**(); // Деструктор

bool **push**(int elem); // Дадавання елементу в чергу

bool **pop**(int & elem);// Вилучення елементу з черги

bool **isEmpty**(); // Черга порожня?

bool **isFull**(); // Черга заповнена?

void **setQTable**(QTableWidget \*qtable);// Для встановлення посилання на QTableWidget

};

#endif // QUEUE\_H\_INCLUDED

Лістинг 3.2 – Файл queue.cpp

#include "queue.h"

Queue::**Queue**(int MaxLength)

{

qEnd = 0;

maxQLength = MaxLength;

q = new int[maxQLength];

qtable = nullptr;

}

Queue::~**Queue**()

{

delete []q;

}

bool Queue::**isEmpty**()

{

return qEnd == 0;

}

bool Queue::**isFull**()

{

return qEnd == maxQLength;

}

bool Queue::**push**(int elem)

{

if(isFull())

return false;

q[qEnd] = elem; // Записуємо новий елемент

qEnd++;

if(qtable)

{

qtable->insertRow(0);

qtable->setItem(0, 0, new QTableWidgetItem(QString::number(elem)));

}

return true;

}

bool Queue::**pop**(int &elem)

{

if(isEmpty())

return false;

elem = q[0]; // Записуємо в еlem вилучаємий елемент

for(int i = 0; i < qEnd - 1; i++) // Зсув елементів, що залишилися

q[i] = q[i + 1];

qEnd--; // Зменшуємо довжину черги

if(qtable)

qtable->removeRow(qEnd);

return true;

}

void Queue::**setQTable**(QTableWidget\* qtable)

{

this->qtable = qtable;

}

Лістинг 3.3 – Файл mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "queue.h"

#include <QMessageBox>

Queue \*q;

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

// Наші налаштування

ui->tableWidget->setColumnCount(1);

ui->tableWidget->setHorizontalHeaderItem(0, new QTableWidgetItem("Дані"));

q = new Queue(10);

q->setQTable(ui->tableWidget);

}

MainWindow::~***MainWindow***()

{

delete ui;

delete q;

}

void MainWindow::**on\_pushButton\_clicked**()

{

int elem = ui->lineEdit->text().toInt();

if(!q->push(elem))

QMessageBox::information(this, "Черга", "Переповнення");

}

void MainWindow::**on\_pushButton\_2\_clicked**()

{

int elem;

if(q->pop(*elem*))

ui->lineEdit->setText(QString::number(elem));

else

QMessageBox::information(this, "Черга", "Черга пуста");

}

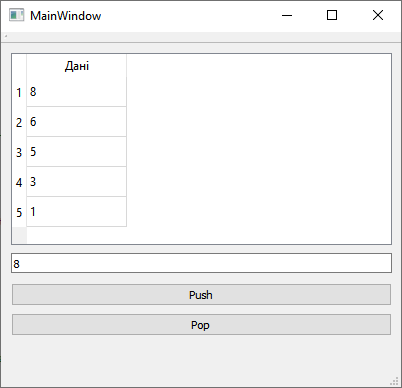


Рисунок 3.2 – Результат роботи програми

**Лабораторна робота №3 (3-й семестр)**

Написати програму, в якій

* 1. Реалізовано клас, що описує чергу для зберігання дійсних чисел;
  2. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан. Користувачеві також виводиться результат виконання п. 3;
  3. У класі реалізований метод для обчислення суми позитивних елементів черги;
  4. Величина заповнення черги візуалізується за допомогою компонента QProgressBar.

Написати програму, в якій

* 1. Реалізовано клас, що описує чергу для зберігання дійсних чисел;
  2. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан. Користувачеві також виводиться результат виконання п. 3;
  3. У класі реалізований метод для обчислення мінімального значення елементів черги;
  4. Величина заповнення черги візуалізується за допомогою компонента QProgressBar.

Написати програму, в якій

* 1. Реалізовано клас, що описує чергу для зберігання дійсних чисел;
  2. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан. Користувачеві також виводиться результат виконання п. 3;
  3. У класі реалізований метод для обчислення максимального значення елементів черги;
  4. Величина заповнення черги візуалізується за допомогою компонента QProgressBar.

Написати програму, в якій

* 1. Реалізовано клас, що описує чергу для зберігання дійсних чисел;
  2. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан. Користувачеві також виводиться результат виконання п. 3;
  3. У класі реалізований метод для обчислення суми елементів черги;
  4. Величина заповнення черги візуалізується за допомогою компонента QProgressBar.

Написати програму, в якій

* 1. Реалізовано клас, що описує чергу для зберігання дійсних чисел;
  2. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан. Користувачеві також виводиться результат виконання п. 3;
  3. У класі реалізований метод для обчислення добутку елементів черги;
  4. Величина заповнення черги візуалізується за допомогою компонента QProgressBar.

Написати програму, в якій

* 1. Реалізовано клас, що описує чергу для зберігання дійсних чисел;
  2. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан. Користувачеві також виводиться результат виконання п. 3;
  3. У класі реалізований метод для обчислення мінімального позитивного значення елементів черги;
  4. Величина заповнення черги візуалізується за допомогою компонента QProgressBar.

Написати програму, в якій

* 1. Реалізовано клас, що описує чергу для зберігання дійсних чисел;
  2. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан. Користувачеві також виводиться результат виконання п. 3;
  3. У класі реалізований метод для обчислення максимального від'ємного значення елементів черги;
  4. Величина заповнення стека візуалізується за допомогою компонента QProgressBar.

Написати програму, в якій

* 1. Реалізовано клас, що описує чергу для зберігання дійсних чисел;
  2. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан. Користувачеві також виводиться результат виконання п. 3;
  3. У класі реалізований метод для обчислення мінімального за модулем значення елементів черги;
  4. Величина заповнення черги візуалізується за допомогою компонента QProgressBar.

Написати програму, в якій

* 1. Реалізовано клас, що описує чергу для зберігання дійсних чисел;
  2. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан. Користувачеві також виводиться результат виконання п. 3;
  3. У класі реалізований метод для обчислення максимального по модулю значення елементів черги;
  4. Величина заповнення стека візуалізується за допомогою компонента QProgressBar.

**Варіант \* (Додаткове завдання)**

* 1. Реалізувати клас, що описує чергу для зберігання дійсних чисел з пріоритетами, пріоритет задається як ціле число;
  2. Вміст черги впорядковується відповідно до пріоритету;
  3. Вміст черги зв'язується з деякими компонентом таким чином, щоб користувач міг бачити її актуальний стан.