

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук (проектування і програмування інтелектуальних
систем та пристроїв)



ЗВІТ

Про виконання лабораторної роботи №3
«Динамічні об'єкти та способи їх використання.
Черги»

Виконав:
Студент групи ПП-12
Венгрин Владислав

Прийняв:
доцент кафедри САП
Кулешник Я. Ф.

Мета роботи: ознайомитись із особливостями застосування динамічних об'єктів складної структури, а саме з чергами та операціями, які виконуються над елементами такого типу об'єктів. Набути практичних навичок програмування з використанням динамічних об'єктів на прикладі черг.

Варіант 3

Хід виконання лабораторної роботи:

Завдання 1.

Створити чергу. Розробити метод для перевірки, чи є черга порожньою. Елементи черги – змінні типу int.

Код програми:

```
#include <iostream>
```

```
template <typename T>
class Node {
public:
    T data;
    Node<T>* next;

    Node(T val) : data(val), next(nullptr) { };
};

template <typename T>
class Queue {
private:
    Node<T>* Front;
    Node<T>* Rare;
    int size;

public:
    Queue() : Front(nullptr), Rare(nullptr), size(0) { };

    bool isEmpty() const {
        return Front == nullptr;
    };

    int getSize() const {
        return size;
    }

    void enqueue(int val) {
        Node<T>* newNode = new Node<T>(val);
        if (isEmpty()) {
            Front = Rare = newNode;
        }
    }
};
```

```

        else {
            Rare->next = newNode;
            Rare = newNode;
        }
        size++;
    }

void dequeue() {
    if (isEmpty()) {
        std::cout << "Queue is empty." << std::endl;
    }
    else {
        Node<T>* temp = Front;
        Front = Front->next;
        delete temp;
        size--;
    }
}

T peek() const {
    if (isEmpty()) {
        std::cout << "Queue is empty. First element is undefined." << std::endl;
        return T();
    }
    else {
        return Front->data;
    }
}

void display() {
    Node<T>* temp = Front;

    while (temp != nullptr) {
        std::cout << temp->data << " ";
        temp = temp->next;
    }

    std::cout << std::endl;
}

void checkType() const {
    if (!isEmpty()) {
        std::cout << "The type of the elements in this queue is: " << typeid(Front-
>data).name() << std::endl;
    }
}

```

```

        else {
            std::cout << "Queue is empty, no type to check." << std::endl;
        }
    }
};

template<typename T>
void isEmptyForUser(Queue<T>& Q1) {
    if (Q1.isEmpty()) {
        std::cout << "Queue is empty. Add new items using enqueue() method." <<
std::endl;
    }
    else {
        std::cout << "Queue is not empty. View of the queue:" << std::endl;;
        Q1.display();
    }
}

int main() {
    Queue<int> Q1;
    Q1.enqueue(10);
    Q1.enqueue(20);
    Q1.enqueue(30);

    std::cout << "\nCheck if queue is empty..." << std::endl;
    isEmptyForUser(Q1);

    std::cout << "\nCheck the type of elements in queue..." << std::endl;
    Q1.checkType();

    Q1.dequeue();
    Q1.dequeue();
    Q1.dequeue();

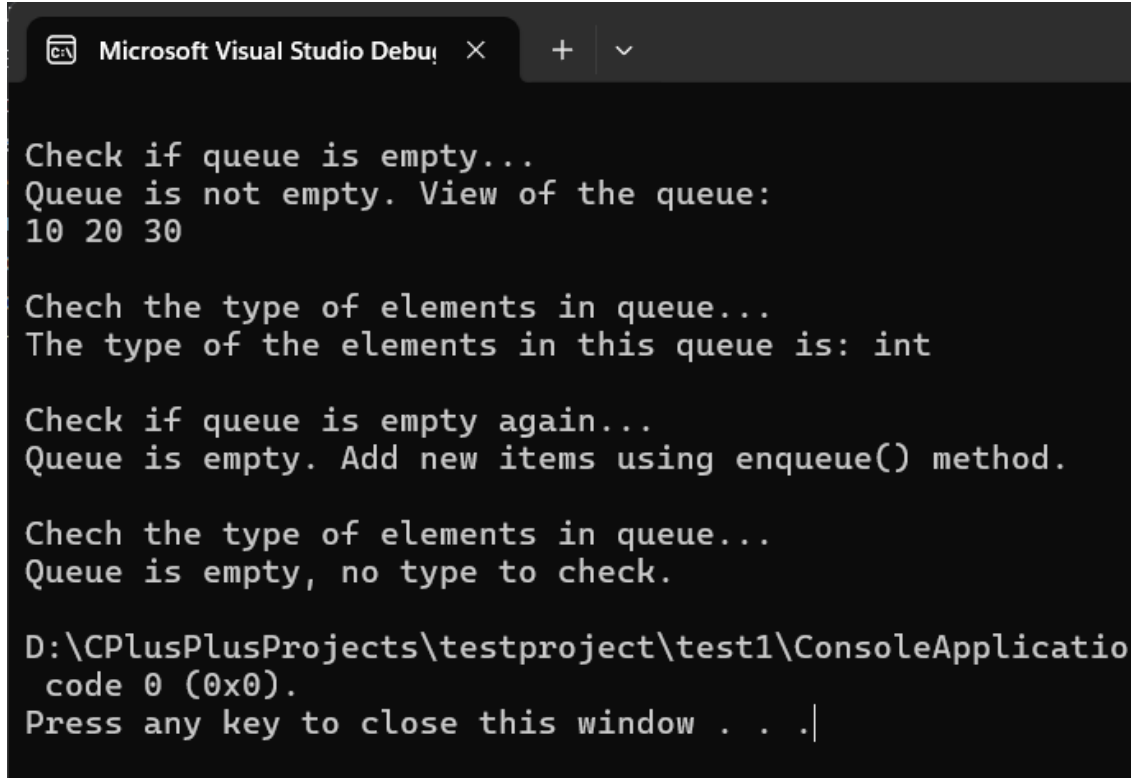
    std::cout << "\nCheck if queue is empty again..." << std::endl;
    isEmptyForUser(Q1);

    std::cout << "\nCheck the type of elements in queue..." << std::endl;
    Q1.checkType();

    return 0;
};

```

Результати програми:



```
Microsoft Visual Studio Debug Console
Check if queue is empty...
Queue is not empty. View of the queue:
10 20 30

Check the type of elements in queue...
The type of the elements in this queue is: int

Check if queue is empty again...
Queue is empty. Add new items using enqueue() method.

Check the type of elements in queue...
Queue is empty, no type to check.

D:\CPlusPlusProjects\testproject\test1\ConsoleApplication1\code 0 (0x0).
Press any key to close this window . . .|
```

Відповіді на питання для самоконтролю:

- 1. У чому переваги і недоліки організації структур у вигляді черги?**
Переваги: проста реалізація, ефективність при обробці запитів за принципом FIFO, використовується для планування задач, обробки подій, буферизації даних.
Недоліки: обмежений доступ тільки до першого (видалення) і останнього (додавання) елементів, неможливість довільного доступу до середніх елементів, обмеження розміру черги в масивних реалізаціях.
- 2. Для моделювання яких реальних завдань зручно використовувати чергу?**
Черга зручно використовується для моделювання черг в операційних системах (обробка запитів), управління потоками в мережах, буферизації даних, організації системи планування задач, в чергах обробки подій у графічних інтерфейсах користувача, а також для розподілених систем.
- 3. Яке значення зберігає вказівник на чергу?**
Вказівник на чергу зазвичай зберігає адресу першого елемента черги або вказівник на місце, де починається обробка елементів черги. Це дозволяє швидко здійснювати операції додавання та видалення.
- 4. Які існують обмеження на тип інформаційного поля в черзі?**
Тип інформаційного поля черги зазвичай обмежений тим, що всі елементи черги повинні бути однакового типу, наприклад, ціле число, об'єкт певного класу, структура тощо. Це необхідно для забезпечення коректної обробки елементів.
- 5. З якою метою в програмах виконується перевірка на відсутність елементів у черзі? Чи можлива ситуація запису нових елементів у чергу на вже зайняті**

власними елементами ділянки пам'яті?

Перевірка на відсутність елементів у черзі необхідна для запобігання помилкам при спробі видалити елемент з порожньої черги або зчитати елемент, коли черга пуста. У масивних реалізаціях черги переповнення можливе, якщо немає місця для нових елементів, що може призвести до помилки або необхідності розширення пам'яті. У зв'язаному списку така проблема менш критична, але може виникнути вичерпання пам'яті.

6. **З якою метою в програмах виконується видалення черги після закінчення роботи з нею? Як зміниться робота програми, якщо операцію видалення не виконувати?**

Видалення черги після її використання необхідне для звільнення пам'яті, щоб уникнути витоків пам'яті (memory leaks). Якщо цього не зробити, програма може втрачати ресурси, що з часом призведе до значного збільшення використаної пам'яті, особливо в довготривалих процесах. У мовах з автоматичним управлінням пам'яттю, таких як Python, ця проблема менш критична, але в C++ необхідно явно видалити динамічно виділену пам'ять.

Висновок:

Я ознайомився із особливостями застосування динамічних об'єктів складної структури, а саме з чергами та операціями, які виконуються над елементами такого типу об'єктів. Набув практичних навичок програмування з використанням динамічних об'єктів на прикладі черг.