ФЕДЕРАЛЬНОЕ АГЕНСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ Императора Александра I»

Кафедра «Информационные и вычислительные системы»

Дисциплина «Программирование С++»

**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6**

ВАРИАНТ 19

|  |  |
| --- | --- |
| Выполнил студент  Факультет: АИТ  Группа: ИВБ-211 | Шефнер А. |
| Проверил: | Проузин О.В. |

**Санкт-Петербург**

**2023**

Оценочный лист результатов ЛР № 6

Ф.И.О. студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шефнер Альберт\_\_\_\_\_

Группа \_\_\_\_\_\_\_\_\_\_\_\_\_ИВБ-211\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№**  **п/п** | **Материалы необходимые для оценки знаний, умений**  **и навыков** | **Показатель**  **оценивания** | **Критерии**  **Оценивания** | **Шкала оценивания** | **Оценка** |
| 1 | Лабораторная работа№ | Соответствие методике выполнения | Соответствует | 7 |  |
| Не соответствует | 0 |
| Срок выполнения | Выполнена в срок | 2 |  |
| Выполнена с опозданием на 2 недели | 0 |
| оформление | Соответствует требованиям | 1  0 |  |
| Не соответствует |  |
|  | **ИТОГО количество баллов** |  |  | 10 |  |

Доцент кафедры

«Информационные и вычислительные

системы» Проурзин О.В. «\_\_» \_\_\_\_\_\_\_\_\_\_2023 г.

**Цели работы:**

* Использовать шаблоны классов.

**Задание**

1. Создать шаблонный класс согласно варианту индивидуального задания (см. файл «Варианты\_шаблонных \_классов.docx»).
2. Задать параметры шаблонного класса по умолчанию.
3. Создать явную специализацию для некоторого типа (например, \*char).
4. В главной функции **main** продемонстрировать не менее трех инстанцирований шаблонного класса с различными типами данных (например, int, char\*, float, double), а также по умолчанию. Проверить работоспособность методов и операторов для всех специализаций этого шаблонного класса.

**Требования:**

1. Написать и отладить программу на языке С++, которая должна состоять из двух файлов:

* файла шаблонного класса (спецификация и реализация могут находится в одном файле, однако методы должны быть определены за пределами спецификации шаблона класса);
* основного файла приложения, содержащего главную функцию приложения main(), в которой происходит инстанцирование шаблонного класса.

2. Составить отчет по лабораторной работе.

**Вариант 9**

Разработать шаблонный класс CQueue для реализации очереди. Для хранения элементов должен быть использован одномерный статический массив. Количество элементов очереди, как и их тип, должны быть параметрами шаблона.

Обязательно наличие следующих методов:

* конструктор с параметрами;
* деструктор;
* метод для добавления элемента в очередь;
* метод извлечения элемента из очереди.

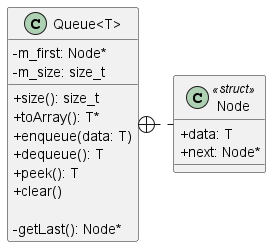
Предусмотреть операторную функцию для перегрузки оператора сравнения.

**Используемые средства**

В качестве интегрированной среды разработки использовалась JetBrains CLion.

Для работы в консоли с потоками ввода-вывода использовалась стандартная библиотека <iostream>.

**UML-диаграмма программы:**

****

**Код программы**

**main.cpp**

#include <iostream>

#include "Queue.h"

#include "vector"

int main() {

system("color F0");

std::vector<int> vec;

Queue<int> queue;

queue.enqueue(4);

std::cout << "Enqueued 4.\n";

queue.enqueue(8);

std::cout << "Enqueued 8.\n";

queue.enqueue(20);

std::cout << "Enqueued 20.\n\n";

std::cout << "Trying peek: " << queue.peek() << "\n\n";

std::cout << "Using toArray(): ";

auto arr = queue.toArray();

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

std::cout << arr[i] << " ";

}

std::cout << "\n\n";

while (queue.size() > 0) {

std::cout <<"dequeued " << queue.dequeue() << '\n';

}

std::cout << "\nEnqueuing 6, 2 and 4." << '\n';

queue.enqueue(6);

queue.enqueue(2);

queue.enqueue(4);

queue.clear();

std::cout << "Size after clear: " << queue.size() << std::endl << std::endl;

system("pause");

return 0;

}

**Queue.h**

#ifndef CPP\_LAB\_6\_QUEUE\_H

#define CPP\_LAB\_6\_QUEUE\_H

#define size\_t unsigned long long int

template <class T = void\*>

class Queue {

private: struct Node {

T data;

Node\* next = nullptr;

};

public:

Queue() = default;

~Queue() { clear(); }

size\_t size();

T\* toArray();

void enqueue(T data);

T dequeue();

T peek();

void clear();

private:

Node\* m\_first = nullptr;

size\_t m\_size = 0;

Node \*getLast();

};

template<class T>

size\_t Queue<T>::size() { return m\_size; }

template<class T>

T \*Queue<T>::toArray() {

T\* array = new T[m\_size];

Node\* current = m\_first;

int index = 0;

while (current != nullptr) {

array[index] = current->data;

index++;

current = current->next;

}

return array;

}

template<class T>

void Queue<T>::enqueue(T data) {

Node\* enqueued = new Node;

enqueued->data = data;

Node\* last = getLast();

(last == nullptr ? m\_first : last->next) = enqueued;

m\_size++;

}

template<class T>

T Queue<T>::dequeue() {

Node\* next = m\_first->next;

T data = m\_first->data;

delete m\_first;

m\_first = next;

m\_size--;

return data;

}

template<class T>

T Queue<T>::peek() {

return m\_first->data;

}

template<class T>

void Queue<T>::clear() {

Node\* current = m\_first;

while (current != nullptr) {

Node\* tmp = current->next;

delete current;

current = tmp;

}

m\_first = nullptr;

m\_size = 0;

}

template<class T>

Queue<T>::Node \*Queue<T>::getLast() {

Node\* current = m\_first;

if(m\_first == nullptr) return nullptr;

while (current->next != nullptr) {

current = current->next;

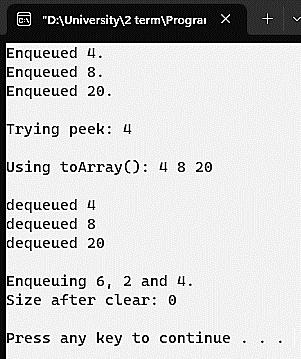
}

return current;

}

#endif //CPP\_LAB\_6\_QUEUE\_H

**Тестовые примеры**

****