МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ»

(МТУСИ)

Кафедра «Информационная безопасность»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №12

по дисциплине

«Одиночное и множественное наследование. Виртуальные и чисто виртуальные функции. Абстрактные и конкретные классы.»

на тему

«Шаблонные функции и классы»

Вариант 9

Выполнил:

студент группы БСУ1801

Копылов М.А.

Проверил:

старший преподаватель кафедры ИБ

Барков В.В.

Москва, 2021

# Цель работы

Изучить динамические структуры данных, овладеть навыками создания конструкторов копирования, перемещения, деструкторов, перегрузки операций копирования и перемещения.

# Задание

|  |  |  |
| --- | --- | --- |
| int GetSize() const;  void Push(const T &element);  T Pop();  T Peek(); | int GetSize();  void Push(const T &element);  T Pop();  T Peek(); | int GetSize() const;  void PushFront(const T &element);  T PopFront();  void PushBack(const T &element);  T PopBack();  T PeekFront() const;  T PeekBack() const; |

Унаследовать разработанный в предыдущей лабораторной работе класс от созданного в этой работе абстрактного класса.

Разработать ещё одну реализацию динамической структуры данных, указанной в индивидуальном задании. Разработать соответствующий класс, унаследовав его от абстрактного класса и определив все требуемые операции. Предусмотреть конструкторы инициализации, копирования, перемещения, деструктор, функции вставки и удаления элемента, просмотра доступного элемента и функцию, проверяющую наличие элементов.

Перегрузить операции присваивания, перемещения и потокового вывода для вывода содержимого динамической структуры на экран.

Класс разработать в варианте шаблона.

Создать функцию, получающую указатель на базовый класс и демонстрирующую работу

Создать функцию, получающую ссылку на базовый класс и демонстрирующую работу.

**Индивидуальное задание**

**Вариант 9**

|  |  |  |
| --- | --- | --- |
| 9 | Очередь на основе однонаправленного циклического списка | QueueBasedOnUnidirectionalCyclicLinkedList |

# Выполнение

Листинг 1 – программный код файла Header.h

#include <iostream>

#include "AbstractQueue.h"

using namespace std;

class NoElementsException :public exception

{

public:

NoElementsException(const char\* msg) :

exception(msg) {}

};

template <typename T>

struct Node

{

int a;

Node\* next;

};

template <typename T>

class Queue: AbstractQueue<T>

{

public:

Queue();

Queue(T);

Queue<T>& operator=(Queue&&);

int GetSize() const override;

void Push(T) override;

void Pop(T&) override;

bool HasElements() const;

void Peek(T&)override;

private:

Node<T>\* next;

Node<T>\* tail;

int size;

};

Листинг 2 – программный код файла main.cpp

#include "Header.h"

#include <conio.h>

#include<iostream>

template <typename T>

Queue<T>::Queue() {}

template <typename T>

Queue<T>::Queue(T a)

{

next = new Node<T>;

next->prev = nullptr;

next->next = nullptr;

next->a = a;

}

template <typename T>

Queue<T>& Queue<T>::operator=(Queue&& other)

{

if (this == &other)

{

return \*this;

}

delete next;

next = other.next;

other.next = nullptr;

}

template <typename T>

int Queue<T>::GetSize() const

{

if (!HasElements()) { return 0; }

int i = 0;

Node<T>\* a = next;

while (a != nullptr)

{

a = a->next;

++i;

}

return i;

}

template <typename T>

void Queue<T>::Push(const T a)

{

Node<T>\* newnode = new Node<T>;

Node<T>\* checking = next;

if (HasElements())

{

while (checking->next != nullptr)

{

checking = checking->next;

}

checking->next = newnode;

}

else

{

next = newnode;

}

newnode->a = a;

newnode->prev = checking;

newnode->next = nullptr;

}

template <typename T>

void Queue<T>::Pop(T& a)

{

if (!HasElements()) { throw NoElementsException("Deck is Empty!"); };

Node<T>\* tempnode = next->next;

if (next->next != nullptr)

{

next->next->prev = nullptr;

}

a = next->a;

delete next;

next = tempnode;

}

template <typename T>

void Queue<T>::Peek(T& a)

{

if (!HasElements()) { throw NoElementsException("Deck is Empty!"); };

a = next->a;

}

template <typename T>

bool Queue<T>::HasElements() const

{

if (this->next != 0)

{

return true;

}

else

{

return false;

}

}

void menu()

{

cout << "1. Добавить элемент в очередь\n";

cout << "2. Получить элемент из очереди\n";

cout << "3. Посмотреть элемент из очереди\n";

cout << "4. Длина очереди\n";

cout << "5. Выход\n";

};

int main()

{

setlocale(LC\_ALL, "RUS");

Queue<int> a;

int b;

bool flag = true;

while (flag)

{

menu();

switch (\_getch())

{

case '1':

cout << "Добавить элемент: ";

cin >> b;

a.Push(b);

break;

case '2':

try

{

a.Pop(b);

cout << "Полученный элемент " << b << endl;

}

catch (NoElementsException ex)

{

cout << "Очередь пуста\n";

}

break;

case '3':

try

{

a.Peek(b);

cout << "Последний элемент " << b << endl;

}

catch (NoElementsException ex)

{

cout << "Очередь пуста\n";

}

break;

case '4':

b = a.GetSize();

cout << "Длина очереди: " << b << endl;

break;

case '5':

flag = false;

break;

}

}

}

Листинг 3 – AbstractQueue.h

template <typename T>

class AbstractQueue

{

public:

virtual void Push(T) = 0;

virtual void Pop(T&) = 0;

virtual void Peek(T&) = 0;

virtual int GetSize() const = 0;

};

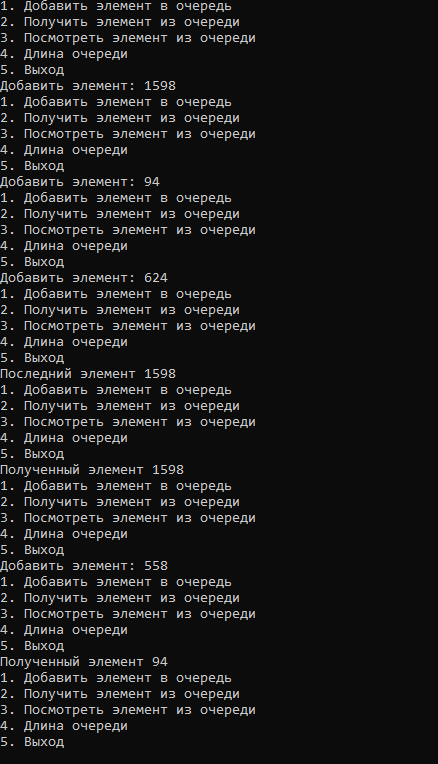


Рисунок 1 – Результат работы с программой.