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 18

Выполнил:

студент группы БСУ1801

Нуждин В.В.

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 3**

|  |  |  |
| --- | --- | --- |
| 3 | Стек на основе двунаправленного списка | StackBasedOnBidirectionalLinkedList |

# Выполнение

Листинг 1 – программный код файла Header.h

#include <iostream>

#include "AbstractStack.h"

using namespace std;

class NoElementsException :public exception

{

public:

NoElementsException(const char\* msg) :

exception(msg) {}

};

template <typename T>

struct Node

{

T a;

Node\* prev;

Node\* next;

};

template <typename T>

class Stack: AbstractStack<T>

{

public:

Stack();

Stack(T);

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

int GetSize() const;

void Push(const T);

void Pop(T&);

bool HasElements() const;

void Peek(T&);

private:

Node<T>\* tail;

Node<T>\* next;

int count;

};

Листинг 2 – программный код файла main.cpp

#include "Header.h"

#include <conio.h>

#include<iostream>

using namespace std;

template <typename T>

Stack<T>::Stack() {}

template <typename T>

Stack<T>::Stack(T a)

{

tail = new Node<T>;

tail->prev = nullptr;

tail->next = nullptr;

tail->a = a;

count = 0;

}

template <typename T>

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

{

if (this == &other)

{

return \*this;

}

delete tail;

delete next;

tail = other.tail;

other.tail = nullptr;

}

template <typename T>

int Stack<T>::GetSize() const

{

return count;

}

template <typename T>

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

{

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

Node<T>\* checking = next;

if (HasElements())

{

while (checking->next != nullptr)

{

checking = checking->next;

}

next = newnode;

tail->prev = nullptr;

}

else

{

tail = newnode;

next = newnode;

}

newnode->a = a;

newnode->prev = checking;

newnode->next = nullptr;

count = count + 1;

}

template <typename T>

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

{

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

a = next->a;

if (next != tail)

{

next = next->prev;

next->next = nullptr;

}

else

{

next = tail = nullptr;

}

count = count - 1;

}

template <typename T>

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

{

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

a = next->a;

}

template <typename T>

bool Stack<T>::HasElements() const

{

if (this->tail != 0)

{

return true;

}

else

{

return false;

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

Stack<double> a;

double b;

bool flag = true;

while (flag)

{

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

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

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

cout << "4. Длина стека\n";

cout << "5. Выход\n";

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 << "Стек пуст\n";

}

break;

case '3':

try

{

a.Peek(b);

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

}

catch (NoElementsException ex)

{

cout << "Стек пуст\n";

}

break;

case '4':

b = a.GetSize();

cout << "Длина стека: " << b << endl;

break;

case '5':

flag = false;

break;

}

}

}

Листинг 3 – AbstractStack.h

template <typename T>

class AbstractStack

{

public:

virtual void Push(T) = 0;

virtual void Pop(T&) = 0;

virtual void Peek(T&) = 0;

virtual int GetSize() const = 0;

};

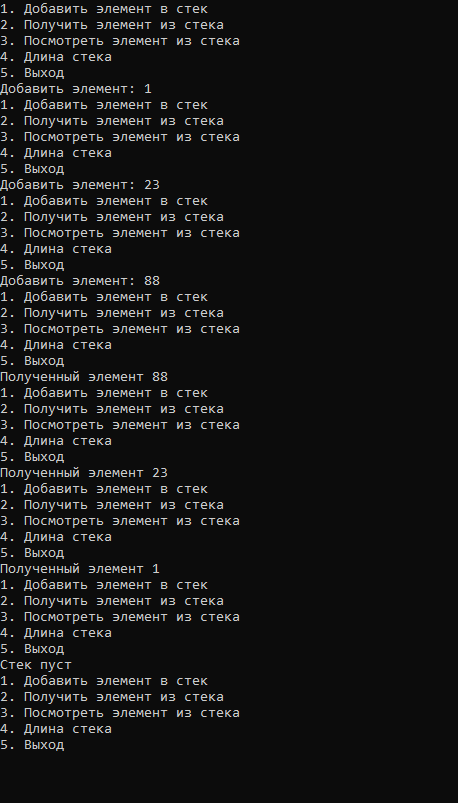


Рисунок 1 – Результат работы с программой.