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

«Программирование в системах информационной безопасности»

на тему

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

Вариант №19

Выполнил:

студент группы БСУ1801

Плотников П.С

Проверил:

старший преподаватель кафедры ИБ

Барков В. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2021

# **Цель**

Овладеть навыками создания базовых классов с виртуальными и чисто виртуальными функциями, а также производных классов с переопределением указанных функций

# **Задание**

Для динамической структуры данных, разработанной в предыдущей лабораторной работе (стек, очередь или дек) создать абстрактный класс, выделив в него необходимые операции.

|  |  |  |
| --- | --- | --- |
| Стек | Очередь | Дек |
| Stack | Queue | Deque |
| 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 №19 | Стек на основе двунаправленного списка | StackBasedOnBiddirectionalLinkedList |

# **4. Выполнение**

Выполнение задания:

Листинг 1 – Исходный код Stack.cpp

#include <stdio.h>

#include "Stack.h"

#include "Wrap.h"

using namespace std;

template <typename T>

Stack<T>::Stack(int \_size)

{

size = \_size;

pos = 0;

arr = new Wrap<T>[size];

}

template <typename T>

Stack<T>::Stack(const Stack& other)

{

size = other.size;

pos = 0;

arr = new Wrap<T>[size];

for (int i = 0; i < size; i++)

{

arr[i] = other.arr[i];

}

}

template <typename T>

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

{

arr = other.arr;

size = other.size;

other.arr = nullptr;

}

template <typename T>

Stack<T>::~Stack()

{

delete[] arr;

}

template <typename T>

void Stack<T>::Push(T item)

{

if (pos == 0)

{

arr[pos] = Wrap<T>(item);

pos++;

}

else

{

arr[pos] = Wrap<T>(item, &arr[pos - 1], NULL);

arr[pos - 1].setNext(&arr[pos]);

pos++;

}

}

template <typename T>

Wrap<T>\* Stack<T>::Pop(int \_pos)

{

for (int i = \_pos; i < size - 1; i++)

{

Wrap<T> tmp(arr[i + 1].getData());

tmp.setPrev(arr[i].getPrev());

tmp.setNext(arr[i].getNext());

arr[i] = tmp;

}

return &arr[\_pos];

}

template <typename T>

Wrap<T>\* Stack<T>::Get(int \_pos)

{

return &arr[\_pos];

}

template <typename T>

int Stack<T>::getSize() { return size; }

template <typename T>

void Stack<T>::printStack()

{

std::cout << "Stack size: " << size << std::endl;

std::cout << "Used: " << pos << std::endl;

std::cout << std::endl;

for (int i = 0; i < pos; i++)

{

std::cout << "Data: " << arr[i].getData() << std::endl;

std::cout << "Current: " << &arr[i] << std::endl;

std::cout << "Previous: " << arr[i].getPrev() << std::endl;

std::cout << "Next: " << arr[i].getNext() << std::endl;

}

}

Листинг 2 – Исходный код файла Stack.h

#pragma once

#include "Wrap.h"

using namespace std;

template <typename T>

class Stack

{

private:

int size, pos;

Wrap<T>\* arr;

public:

Stack(int \_size);

Stack(const Stack& other);

Stack(Stack&& other);

~Stack();

void Push(T item);

Wrap<T>\* Pop(int \_pos);

Wrap<T>\* Get(int \_pos);

int getSize();

void printStack();

};

Листинг 3 – Исходный код файла Wrap.cpp

#include <stdio.h>

#include "Wrap.h"

using namespace std;

template <typename T>

Wrap<T>::Wrap(T \_data, Wrap<T>\* \_prev, Wrap<T>\* \_next) : data(\_data), prev(\_prev), next(\_next) {}

template <typename T>

Wrap<T>::Wrap(T \_data) : data(\_data), prev(NULL), next(NULL) {}

template <typename T>

Wrap<T>::Wrap() : data(NULL), prev(NULL), next(NULL) {}

template <typename T>

T Wrap<T>::getData() { return data; }

template <typename T>

Wrap<T>\* Wrap<T>::getPrev() { return prev; }

template <typename T>

Wrap<T>\* Wrap<T>::getNext() { return next; }

template <typename T>

void Wrap<T>::setPrev(Wrap\* \_prev) { prev = \_prev; }

template <typename T>

void Wrap<T>::setNext(Wrap\* \_next) { next = \_next; }

Листинг 4 – Исходный код файла Wrap.h

#pragma once

using namespace std;

template <typename T>

class Wrap

{

private:

T data;

Wrap\* prev, \* next;

public:

Wrap(T \_data, Wrap\* \_prev, Wrap\* \_next);

Wrap(T \_data);

Wrap();

T getData();

Wrap\* getPrev();

Wrap\* getNext();

void setPrev(Wrap\* \_prev);

void setNext(Wrap\* \_next);

};

Листинг 5 – Исходный код файла main.cpp

#include <iostream>

#include "Stack.h"

#include "Wrap.h"

#include "Stack.cpp"

#include "Wrap.cpp"

using namespace std;

int main()

{

int size = 5;

Stack<int> s(size);

for (int i = 0; i < size; i++)

{

s.Push(i);

}

s.printStack();

cout << endl;

cout << "POPPED--------------------------------" << endl;

cout << endl;

s.Pop(0);

s.printStack();

system("pause");

}

**Результаты выполнения заданий:**

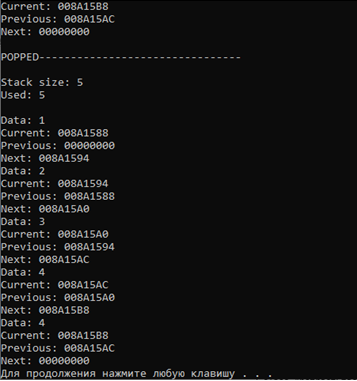


Рисунок 1 – Выполнение задания

[**https://github.com/Pavel-Plotnikov-BSU1801/Laboratornai-12-Plotnikov-P.S.-BSU1801**](https://github.com/Pavel-Plotnikov-BSU1801/Laboratornai-12-Plotnikov-P.S.-BSU1801)

# **Вывод**

В этой лабораторной работе я получил навыки создания базовых классов с виртуальными и чисто виртуальными функциями, а также производных классов с переопределением указанных функций.