МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Списки. Стеки. Очереди.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к отчету по лабораторной работе №1

по дисциплине

Алгоритмы и структуры данных

РУКОВОДИТЕЛЬ:

Капранов Сергей Николаевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

СТУДЕНТ:

Семехин Василий Дмитриевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

19-ИВТ-3

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020

**Задача 19**

Реализовать функции работы со стеком (удаление, вставка).

**Реализация кода на языке С++**

**Шаблонный класс Stack**

/\*Assert - это специальная конструкция, позволяющая проверять предположения о значениях произвольных данных в произвольном месте программы.

Эта конструкция может автоматически сигнализировать при обнаружении некорректных данных,

что обычно приводит к аварийному завершению программы с указанием места обнаружения некорректных данных.\*/

template <typename T>

class Stack

{

private:

T\* stackPtr; //Указатель на стек

int size; //Максимальное количество элементов в стеке

int top; //Номер текущего элемента стека

public:

Stack(int = 10); //По умолчанию размер стека равен 10 элементам

Stack(const Stack<T>&); //Конструктор копирования

~Stack(); //Деструктор

void push(const T&); //Поместить элемент в вершину стека

T pop(); //Удалить элемент из вершины

//стека и вернуть его

void printStack(); //Вывод стека на экран

int getStackSize() const; //Получить размер стека

T\* getPtr() const; //Получить указатель на стек

int getTop() const; //Получить номер текущего элемента в стеке

};

//Конструктор Стека

template <typename T>

Stack<T>::Stack(int maxSize) : size(maxSize) //Инициализация константы

{

stackPtr = new T[size]; //Выделить память под стек

top = 0; //Инициализируем текущий элемент

} //Нулем

//Конструктор копирования

template <typename T>

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

size(otherStack.getStackSize()) //Инициализация константы

{

stackPtr = new T[size];

top = otherStack.getTop(); //Выделить память под новый стек

for (int ix = 0; ix < top; ix++) {

stackPtr[ix] = otherStack.getPtr()[ix];

}

}

//Функция деструктора Стека

template <typename T>

Stack<T>::~Stack()

{

delete[] stackPtr; //Удаляем стек

}

//Функция добавления элемента в стек

template <typename T>

void Stack<T>::push(const T& value)

{

//Проверяем размер стека

assert(top < size); //Номер текущего элемента должен быть

stackPtr[top++] = value; //Меньше размера стека

} //Помещаем элемент в стек

//Функция удаления элемента из стека

template <typename T>

T Stack<T>::pop()

{

//Проверяем размер стека

assert(top > 0); //Номер текущего элемента должен быть больше 0

stackPtr[--top]; //Удаляем элемент из стека

}

//Вывод стека на экран

template <typename T>

void Stack<T>::printStack()

{

for (int ix = top - 1; ix >= 0; ix--)

cout << "|" << setw(4) << stackPtr[ix] << endl;

}

//Вернуть размер стека

template <typename T>

int Stack<T>::getStackSize() const

{

return size;

}

//Вернуть указатель на стек (для конструктора копирования)

template <typename T>

T\* Stack<T>::getPtr() const

{

return stackPtr;

}

//Вернуть размер стека

template <typename T>

int Stack<T>::getTop() const

{

return top;

}

**Основной модуль main**

int main()

{

cout << "Hello, this program implements the stack" << endl << "Please, enter number of element`s (less then 15): " << endl;

int numb; //numb - количество элементов в стеке

//Данный цикл реализует проверку ввода

//Переменной numb

//В случае ввода букв или других

//Недопустимых символов

//Выведется сообщение об ошибке и ввод

//Повторится

while (!(cin >> numb) || (cin.peek() != '\n') || numb <= 0 || numb > 15)

{

cin.clear();

while (cin.get() != '\n');

cout << "Error, enter correct value!" << endl;

}

Stack<char> stackSymbol(numb);

cout << "Now enter " << numb << " stack element without negative.\nIf you enter more the extra elements will be discarded. " << endl << "Do it with the enter key or in a srting: " << endl;

int ct = 0; //Счетчик

char ch; //Переменная, в которую будет

//введены элемены стека

while (ct++ < numb)

{

cin >> ch;

stackSymbol.push(ch); //Помещаем элементы в стек

}

cout << endl;

stackSymbol.printStack(); //Печать стека

cout << "\n\nУдалим элемент из стека\n";

stackSymbol.pop();

stackSymbol.printStack(); //Печать стека

Stack<char> newStack(stackSymbol);

cout << "\n\nСработал конструктор копирования!\n";

newStack.printStack();

return 0;

}

**Скриншоты работы программы**



