Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

**Тема:** Лабораторная работа № 6 по ООП - Стек

Семестр: 2

Выполнил студент ИВТ-22-2б:

Солодов Александр Андреевич

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

1. Определить класс-контейнер.
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.
5. Написать тестирующую программу, иллюстрирующую выполнение операций.

*Вариант 12*

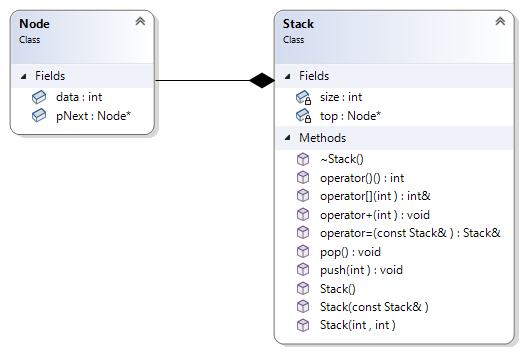
Класс-контейнер СТЕК с ключевыми значениями типа int. Реализовать операции:

[] – доступа по индексу;

() – определение размера списка;

+ число – добавляет константу ко всем элементам вектора;

**UML-диаграмма**

****

**Программное решение**

Stack.h

#pragma once

#include <iostream>

using namespace std;

class Node

{

public:

int data; //данные

Node\* pNext; //указатель на следующий узел

};

class Stack

{

private:

int size; //размер стека

Node\* top; //указатель на вершину стека

public:

Stack();

Stack(int, int);

Stack(const Stack&);

~Stack();

void push(int); //функция для добавления элемента в стек

void pop(); //функция для удаления элемента из стека

//перегрузки операторов

Stack& operator = (const Stack&);

int& operator [] (int);

int operator () ();

void operator + (int);

friend istream& operator >> (istream&, Stack&);

friend ostream& operator << (ostream&, const Stack&);

};

Stack.cpp

#include "Stack.h"

//конструктор по умолчанию

Stack::Stack()

{

size = 0;

top = nullptr;

}

//конструктор с параметрами

Stack::Stack(int s, int d)

{

size = s;

top = nullptr;

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

Node\* addNode = new Node;

addNode->data = d;

addNode->pNext = top;

top = addNode;

}

}

//конструктор копирования

Stack::Stack(const Stack& s)

{

size = s.size;

if (s.top == nullptr) {

top = nullptr;

return;

}

top = new Node;

top->data = s.top->data;

Node\* current = s.top->pNext;

Node\* newCurrent = top;

while (current != nullptr) {

Node\* addNode = new Node;

addNode->data = current->data;

newCurrent->pNext = addNode;

newCurrent = addNode;

current = current->pNext;

}

newCurrent->pNext = nullptr;

}

//деструктор

Stack::~Stack() {

while (size > 0) {

pop();

}

}

//функция для добавления элемента в стек

void Stack::push(int d)

{

Node\* addNode = new Node;

addNode->data = d;

addNode->pNext = top;

top = addNode;

size++;

}

//функция для удаления элемента из стека

void Stack::pop()

{

if (top == nullptr) {

cout << "Стек пуст!" << endl;

return;

}

else {

Node\* nodeDelete = top;

top = top->pNext;

delete nodeDelete;

size--;

}

}

//перегрузка оператора присваивания

Stack& Stack::operator = (const Stack& s)

{

if (this == &s) {

return \*this;

}

while (top != nullptr) {

Node\* temp = top;

top = top->pNext;

delete temp;

size--;

}

Node\* otherCurrent = s.top;

while (otherCurrent != nullptr) {

push(otherCurrent->data);

otherCurrent = otherCurrent->pNext;

}

return \*this;

}

//перегрузка оператора присваивания

int& Stack::operator [] (int index)

{

if (index >= 0 && index < size) {

Node\* current = top;

int newSize = size - 1;

while (newSize != index) {

current = current->pNext;

newSize--;

}

return current->data;

}

}

//перегрузка круглых скобок

int Stack::operator () ()

{

return size;

}

//перегрузка оператора сложения

void Stack::operator + (int add)

{

Node\* current = this->top;

while (current != nullptr) {

current->data = current->data + add;

current = current->pNext;

}

}

//перегрузка операции ввода

istream& operator >> (istream& in, Stack& s)

{

int size;

cout << "Введите размер стека: ";

in >> size;

cout << "Введите элементы стека через Enter: ";

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

int data;

in >> data;

s.push(data);

}

return in;

}

//перегрузка операции вывода

ostream& operator << (ostream& out, const Stack& s) {

Node\* current = s.top;

while (current != nullptr) {

out << current->data << " ";

current = current->pNext;

}

return out;

}

main.cpp

/\* Вариант 12.

Класс- контейнер СТЕК с ключевыми значениями типа int. Реализовать операции:

[] – доступа по индексу;

() – определение размера стека;

+ число – добавляет константу ко всем элементам стека;

\*/

#include "Stack.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "RU");

Stack s;

cin >> s;

cout << "Cтек: " << s << endl;

cout << "Элемент стека с индексом 2: " << s[2] << endl;

s.pop();

cout << "Cтек после удаления элемента: " << s << endl;

s.push(12);

s.push(73);

cout << "Cтек после добавления элементов: " << s << endl;

s + 5;

cout << "Cтек после увеличения элементов: " << s << endl;

cout << "Размер стека: " << s() << endl;

return 0;

}

**Результат работы программы**

