Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им.И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление:710400 «Программная инженерия»

ОТЧЕТ

По дисциплине: «Алгоритмы и структуры данных»

Лабораторная работа №4

Тема: «Работа со стеком»

Выполнил: студент группы

ПИ(б)-2-19 Улан уулу Нурдин

Проверила: Валеева А. А.

Бишкек – 2020

**Практическое задание**

1. **Перечислите основные операции, применяемые при работе со стеками. К каким позициям в стеке они могут применяться?**

* Инициализация стека
* Удаление стека
* Доступ к элементам
* Удаление элемента
* Добавление элемента
* Нахождение количества элементов.

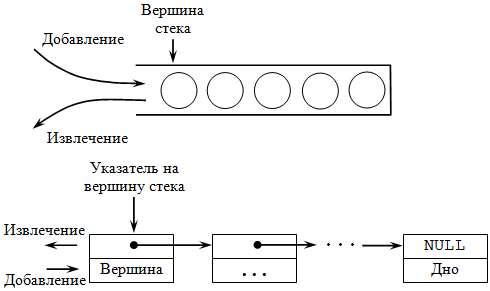
1. **Какой метод доступа применим к элементам стека? В чем его особенности?**

К элементам стека применим только последовательный доступ. Мы не можем обращаться к любым элементам, а только к последнему добавленному.

1. **Перечислите задачи, для решения которых применяются стеки.**

Программный вид стека используется для обхода таких структур данных как деревья или графы. Так же, для решения математических и логических выражений, в которой операнды расположены перед знаками операций. Для отслеживания точек возврата из подпрограмм используется стек вызовов.

1. **Изобразите структуру звена динамического стека.**



1. **Как определить наличие или отсутствие элементов в стеке?**

Если верхний элемент(top) имеет значение NULL, то список пуст, либо можно использовать переменную для подсчёта количества элементов.

1. **Как определить количество элементов в стеке?**

Да, пройдя циклом через стек, и поставив цикл.

while(Стек !=0)

счетчик++;

1. **Перечислите основные отличия стека от очереди.**

Основные различия между стеком и очередью заключаются в том, что в стеке используется метод LIFO (последний пришел первым вышел) для доступа и добавления элементов данных, тогда как в очереди используется метод FIFO (первый пришел первым вышел) для доступа и добавления элементов данных.

**II. Выполнить упражнения**

**Задание**

**1. Постановка задачи**

1. Разработать программу для работы со стеком –реализовать функции:

* создания,
* добавления,
* удаления элемента,
* поиска элемента по ключу,
* поиск позиции по заданному значению,
* нахождения количества элементов в стеке, очистка списка.

1. Разработать программу с функцией поиска максимального значения элемента в стеке.

**2. Описание входных и выходных данных**

**Stack\* \_Top -** переменная типа Stack;

**\_count –** переменная для хранения кол-во элементов в стеке;

**void inition(Stack\*& top) –** Функция для инициализации первого элемента;

**void push(Stack\*& top) –** Функция для добавления нового элемента;

**void output(Stack\* top) –** Функция для вывода элементов в консоль;

**void pop(Stack\*& top) –** Функция для удаления элемента;

**void searchIndex(Stack\* top) –** Функция для поиска элемента по индексу элемента;

**void searchValue(Stack\* top) –** Функция для поиска элемента по значению;

**void searchMax(Stack\* top) –** Функция для поиска максимального элемента;

**void StCount() –** Функция для вывода кол-во элементов в стеке;

**void delSt(Stack\* top) –** Функция для удаления стека;

int main()



void inition(Stack\*& top) void push(Stack\*& top)

void output(Stack\* top) void pop(Stack\*& top)

void searchIndex(Stack\* top)



void searchValue(Stack\* top)

void searchMax(Stack\* top) StCount()



delSt(Stack\* top)



**3.Текст программы**

#include <iostream>

#include <ctime>

#include <string>

#include <fstream>

#include <iomanip>

#include "Windows.h"

using namespace std;

struct Stack {

int \_item;

Stack\* \_next;

};

int x = 0, i = 0;

int \_count = 0;

void inition(Stack\*& top);

void push(Stack\*& top);

void output(Stack\* top);

void pop(Stack\*& top);

void searchIndex(Stack\* top);

void searchValue(Stack\* top);

void searchMax(Stack\* top);

void StCount();

void delSt(Stack\* top);

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Stack\* \_Top = NULL;

cout << "Инициализируйте первый элемент:";

inition(\_Top);

system("cls");

bool f = true;

int choice;

while (f) {

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

cout << "| Главное Меню: |" << endl;

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

cout << "| 1.Добавление элемента |" << endl;

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

cout << "| 2.Вывод элементов |" << endl;

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

cout << "| 3.Удаление элемента |" << endl;

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

cout << "| 4.Поиска элемента по индексу |" << endl;

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

cout << "| 5.Поиск позиции по значению |" << endl;

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

cout << "| 6.Поиск максимального |" << endl;

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

cout << "| 7.Количество элементов |" << endl;

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

cout << "| 8.Очистка элементов |" << endl;

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

cout << "| 9.Выход |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

switch (choice) {

case 1: push(\_Top); break;

case 2: output(\_Top); break;

case 3: pop(\_Top); break;

case 4: searchIndex(\_Top); break;

case 5: searchValue(\_Top); break;

case 6: searchMax(\_Top); break;

case 7: StCount(); break;

case 8: delSt(\_Top); \_Top = 0; break;

case 9: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

return 0;

}

void inition(Stack\*& top) {

\_count++;

Stack\* \_stack = new Stack;

cin >> x;

\_stack->\_item = x;

\_stack->\_next = top;

top = \_stack;

x = 0;

}

void push(Stack\*& top) {

\_count++;

Stack\* \_stack = new Stack;

cout << "Введите новый элемент:";

cin >> x;

\_stack->\_item = x;

\_stack->\_next = top;

top = \_stack;

x = 0;

}

void output(Stack\* top) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl;

}

else {

Stack\* \_stack;

i = 1;

while (top != NULL) {

x = top->\_item;

\_stack = top;

top = top->\_next;

cout << "st[" << i << "] = " << x << endl;

i++;

}

x = 0;

}

}

void pop(Stack\*& top) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl;

}

else {

Stack\* \_stack = top;

cout << "Удален последний элемент = " << top->\_item << endl;

top = top->\_next;

delete \_stack;

\_count--;

}

}

void searchIndex(Stack\* top) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl;

}

else {

int s;

Stack\* \_stack;

cout << "Введите индекс искоемого элемента:";

cin >> x;

system("cls");

if (x > \_count || x <= 0) {

cout << "Такого индекса нету!!!" << endl;

}

else {

i = 1;

while (top != NULL) {

s = top->\_item;

\_stack = top;

top = top->\_next;

if (i == x) {

cout << "st[" << i << "] = " << x << endl;

}

i++;

}

}

x = 0;

}

}

void searchValue(Stack\* top) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl;

}

else {

int s, k = 0;

Stack\* \_stack;

cout << "Введите искоемый элемент:";

cin >> x;

system("cls");

i = 1;

while (top != NULL) {

s = top->\_item;

\_stack = top;

top = top->\_next;

if (s == x) {

k++;

cout << "st[" << i << "] = " << x << endl;

}

i++;

}

if (k == 0) {

cout << "Введённый элемент не найден!" << endl;

}

x = 0;

}

}

void searchMax(Stack\* top) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl;

}

else {

Stack\* top2 = top;

int s, max = top->\_item;

i = 1;

max = top->\_item;

while (top != NULL) {

s = top->\_item;

if (s > max) {

max = s;

}

top = top->\_next;

i++;

}

cout << "Максимальный элемент равен " << endl;

i = 1;

while (top2 != NULL) {

if (max == top2->\_item) {

cout << "st[" << i << "] = " << max << endl;

}

top2 = top2->\_next;

i++;

}

}

}

void StCount() {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl;

}

else {

cout << "Нынешнее кол-во элементов в Стеке равно:" << \_count << endl;

}

}

void delSt(Stack\* top) {

if (\_count <= 0) {

cout << "Стек пуст!!!" << endl;

}

else {

Stack\* \_stack;

while (top != NULL) {

\_stack = top;

top = top->\_next;

delete \_stack;

}

cout << "Стек удален!!!" << endl;

\_count = 0;

}

}