Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа №7

По дисциплине «Основы алгоритмизации и программирования»

На тему «**Полустатические структуры данных: стеки**»

Выполнил:

Студент 1 курса 10 группы

Мандрик Алексей Иванович

Преподаватель: асс. Андронова М.В.

Минск, 2024

Создать проект, демонстрирующий работу со ***стеком***, организованным на основе ***списка***,в соответствии со своим вариантом из таблицы ниже. Все операции со стеком реализовать через функции. Дополнить проект функциями очистки стека **clear()**, сохранения в файл и считывания из файла. Проект должен содержать три части: главная функция, файл с функциями работы со стеком и заголовочный файл. Создать интерфейс в виде меню.

**Вариант 8**



**Код программы:**

stdafx.h:

#pragma once

struct stack // Структура

{

int digit;

stack\* next;

};

// Прототипы

void clear(stack\*& myStk);

void pop(stack\*& myStk);

void push(int x, stack\*& myStk);

void toFile(stack\*& myStk);

void fromFile(stack\*& myStk);

void sort(stack\* myStk, stack\*& stack1, stack\*& stack2);

Функции

#include <iostream>

#include <fstream>

#include "stdafx.h"

using namespace std;

void push(int x, stack\*& myStk) //Добавление элемента х в стек

{

stack\* e = new stack; //выделение памяти для нового элемента

e->digit = x; //запись элемента x в поле digit

e->next = myStk; //перенос вершины на следующий элемент

myStk = e; //сдвиг вершины на позицию вперед

}

void pop(stack\*& myStk) //Извлечение элемента из стека

{

if (myStk == NULL)

{

cout << "Стек пуст!" << endl;

return; //если стек пуст - возврат (-1)

}

while (myStk != NULL) {

cout << myStk->digit << " ";

myStk = myStk->next;

}

cout << endl;

}

void toFile(stack\*& myStk) //Запись в файл

{

stack\* e = myStk;

stack buf;

ofstream frm("mstack.txt"); // Открытие

if (frm.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

while (e)

{

buf = \*e;

frm << buf.digit << endl;

e = e->next;

}

frm.close(); // Закртыие

cout << "Стек записан в файл mstack.txt\n";

}

void fromFile(stack\*& myStk) //Считывание из файла

{

stack buf, \* p = nullptr, \* e = nullptr;

ifstream frm("mstack.txt");

if (frm.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

frm.seekg(0);

frm.read((char\*)&buf, sizeof(stack));

while (!frm.eof())

{

push(buf.digit, e);

frm.read((char\*)&buf, sizeof(stack));

}

frm.close();

while (e != NULL)

{

push(buf.digit, p);

myStk = p;

}

cout << "\nСтек считан из файла mstack.txt\n\n";

}

void clear(stack\*& myStk1) // Oчисткa

{

myStk1->next = NULL;

myStk1 = NULL;

cout << "Стек очищен!" << endl;

}

void sort(stack\* myStk, stack\*& stack1, stack\*& stack2) // Сортировка больше меньше 50 с записью в разные стеки

{

stack\* current = myStk;

while (current != nullptr) {

if (current->digit > 50)

{

push(current->digit, stack1);

// Помещаем в stack1, если значение больше 50

}

else

{

push(current->digit, stack2);

// Помещаем в stack2, если значение меньше или равно 50

}

current = current->next;

}

// Вывод содержимого stack1 и stack2

cout << "stack1:" << endl;

if (stack1 == NULL)

{

cout << "Стек пуст!" << endl;

}

while (stack1 != NULL) {

cout << stack1->digit << " ";

stack1 = stack1->next;

}

cout << endl;

cout << "stack2:" << endl;

if (stack1 == NULL)

{

cout << "Стек пуст!" << endl;

}

while (stack2 != NULL) {

cout << stack2->digit << " ";

stack2 = stack2->next;

}

cout << endl;

}

Main

#include <iostream>

#include "stdafx.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

int choice;

int x;

stack\* myStk = new stack; //выделение памяти для стека

stack\* stack1 = new stack;

stack\* stack2 = new stack;

myStk = NULL; //инициализация первого элемента

stack1 = NULL;

stack2 = NULL;

// Меню

for (;;)

{

cout << "Выберите команду:" << endl;

cout << "1 - Добавление элемента в стек" << endl;

cout << "2 - Извлечение элемента из стека" << endl;

cout << "3 - Запись в файл" << endl;

cout << "4 - Чтение из файла" << endl;

cout << "5 - Сортировка" << endl;

cout << "6 - Очистка стека" << endl;

cin >> choice;

switch (choice)

{

case 1: // Заполнение стека

cout << "Введите количество переменных: " << endl;

int n;

cin >> n;

cout << "Введите n чисел типа int: " << endl;

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

cin >> x;

push(x, myStk);

}

break;

case 2: // Вывод стека

pop(myStk);

break;

case 3: // Запись стека в файл

toFile(myStk);

break;

case 4: // Чтение данных из файла

fromFile(myStk);

break;

case 5: // Разделение на два стека <>50

sort(myStk, stack1, stack2);

cout << "Стеки успешно разделены!" << endl;

break;

case 6: // Очистка стека

clear(myStk);

break;

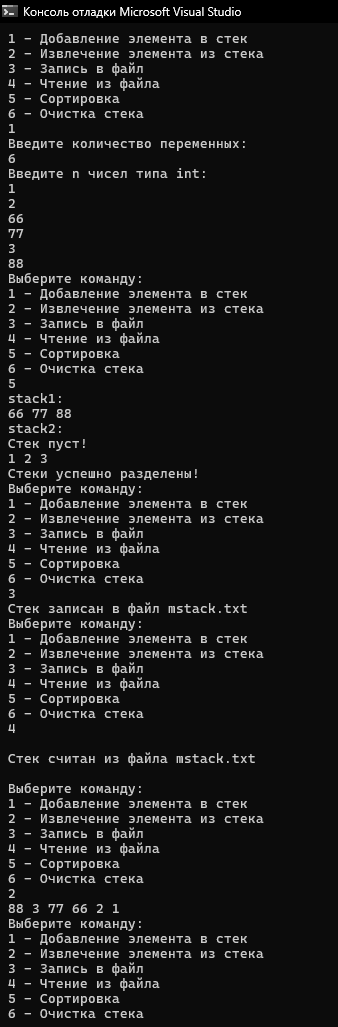
}

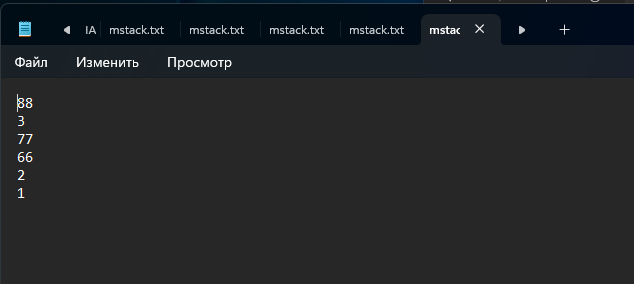
}

return 0;

}

**Результат выполнения:**





**Дополнительные задания:**

**Вариант 2**

|  |
| --- |
| Разработать функцию, которая удаляет из стека первый отрицательный элемент, если такой есть. |
| Разработать функцию, которая удаляет первый положительный элемент, если такой есть. |

**Код программы:**

#include <iostream>

using namespace std;

struct stack

{

int digit;

stack\* next;

};

void push(stack\*& p, int value); // Функция для добавления элемента в стек

void show(stack\* p); // Функция для отображения содержимого стека

void pop(stack\*& p); // Функция для удаления элемента из стека

void clear(stack\*& p); // Функция для очистки стека

int main()

{

setlocale(LC\_CTYPE, "Russian"); // Установка локали для корректного отображения русского текста

stack\* first = nullptr; // Инициализация указателя на вершину стека как nullptr

int value;

menu:

int choice;

cout << "Сделайте выбор:" << endl;

cout << " 1 - Ввод чисел" << endl;

cout << " 2 - Вывод чисел" << endl;

cout << " 3 - Удалить элемент < 0" << endl;

cout << " 4 - Очистить стек" << endl;

cout << " 5 - Выход" << endl;

cin >> choice; // Ввод выбора пользователя

switch (choice) {

case 1: // Ввод чисел

cout << "Введите количество переменных: " << endl;

int n; cin >> n; // Ввод количества чисел

cout << "Введите " << n << " чисел типа int: " << endl;

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

cin >> value; // Ввод числа

push(first, value); // Добавление числа в стек

}

goto menu; // Переход к меню

break;

case 2: // Вывод чисел

show(first); // Отображение содержимого стека

goto menu; // Переход к меню

break;

case 3: // Удалить элемент < 0

stack \* pp;

pp = first;

while (pp != NULL) {

if ((pp->next)->digit < 0) { // Проверка наличия элемента меньше 0

pop(pp); // Удаление элемента

goto menu; // Переход к меню

}

pp = pp->next; // Переход к следующему элементу

}

cout << "Таких элементов нет! \n"; // Вывод сообщения, если таких элементов нет

goto menu; // Переход к меню

break;

case 4: // Очистить стек

stack \* ppp;

ppp = first;

clear(ppp); // Очистка стека

goto menu; // Переход к меню

break;

default:

return 0; // Завершение программы

break;

}

return 0;

}

// Функция для добавления элемента в стек

void push(stack\*& p, int value)

{

stack\* newP = new stack; // Создание нового элемента стека

if (newP != NULL) // Проверка наличия места для добавления элемента

{

newP->digit = value; // Присваивание значению нового элемента

newP->next = p; // Назначение следующего элемента указателю нового элемента

p = newP; // Перемещение указателя на вершину стека на новый элемент

}

else

cout << "Операция добавления не выполнена" << endl; // Вывод сообщения об ошибке

}

// Функция для отображения содержимого стека

void show(stack\* p) {

if (p->next == NULL)

{

cout << "Список пуст!\n"; // Вывод сообщения о пустом стеке

return;

}

while (p != NULL) {

cout << p->digit << " "; // Вывод элементов стека

p = p->next; // Переход к следующему элементу

}

cout << endl;

}

// Функция для удаления элемента из стека

void pop(stack\*& p) {

p->next = (p->next)->next; // Пропуск следующего элемента

}

// Функция для очистки стека

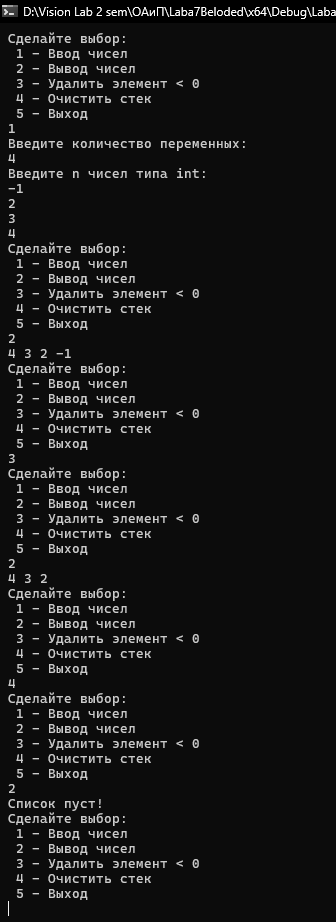
void clear(stack\*& p) {

p->next = NULL; // Установка указателя следующего элемента на NULL

p = NULL; // Установка указателя вершины стека на NULL

}

**Результат выполнения:**

****

**Вариант 3**

****

**Код программы:**

#include <iostream>

using namespace std;

struct stack

{

int digit;

stack\* next;

};

void push(stack\*& p, int value); // Функция для добавления элемента в стек

void show(stack\* p); // Функция для отображения содержимого стека

void pop(stack\*& p); // Функция для удаления элемента из стека

void clear(stack\*& p); // Функция для очистки стека

int main()

{

setlocale(LC\_CTYPE, "Russian"); // Установка локали для корректного отображения русского текста

stack\* first = nullptr; // Инициализация указателя на вершину стека как nullptr

int value;

menu:

int choice;

cout << "Сделайте выбор:" << endl;

cout << " 1 - Ввод чисел" << endl;

cout << " 2 - Вывод чисел" << endl;

cout << " 3 - Удалить элемент < 0" << endl;

cout << " 4 - Очистить стек" << endl;

cout << " 5 - Выход" << endl;

cin >> choice; // Ввод выбора пользователя

switch (choice) {

case 1: // Ввод чисел

cout << "Введите количество переменных: " << endl;

int n; cin >> n; // Ввод количества чисел

cout << "Введите " << n << " чисел типа int: " << endl;

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

cin >> value; // Ввод числа

push(first, value); // Добавление числа в стек

}

goto menu; // Переход к меню

break;

case 2: // Вывод чисел

show(first); // Отображение содержимого стека

goto menu; // Переход к меню

break;

case 3: // Удалить элемент < 0

stack \* pp;

pp = first;

while (pp != NULL) {

if ((pp->next)->digit > 0) { // Проверка наличия элемента меньше 0

pop(pp); // Удаление элемента

goto menu; // Переход к меню

}

pp = pp->next; // Переход к следующему элементу

}

cout << "Таких элементов нет! \n"; // Вывод сообщения, если таких элементов нет

goto menu; // Переход к меню

break;

case 4: // Очистить стек

stack \* ppp;

ppp = first;

clear(ppp); // Очистка стека

goto menu; // Переход к меню

break;

default:

return 0; // Завершение программы

break;

}

return 0;

}

// Функция для добавления элемента в стек

void push(stack\*& p, int value)

{

stack\* newP = new stack; // Создание нового элемента стека

if (newP != NULL) // Проверка наличия места для добавления элемента

{

newP->digit = value; // Присваивание значению нового элемента

newP->next = p; // Назначение следующего элемента указателю нового элемента

p = newP; // Перемещение указателя на вершину стека на новый элемент

}

else

cout << "Операция добавления не выполнена" << endl; // Вывод сообщения об ошибке

}

// Функция для отображения содержимого стека

void show(stack\* p) {

if (p->next == NULL)

{

cout << "Список пуст!\n"; // Вывод сообщения о пустом стеке

return;

}

while (p != NULL) {

cout << p->digit << " "; // Вывод элементов стека

p = p->next; // Переход к следующему элементу

}

cout << endl;

}

// Функция для удаления элемента из стека

void pop(stack\*& p) {

p->next = (p->next)->next; // Пропуск следующего элемента

}

// Функция для очистки стека

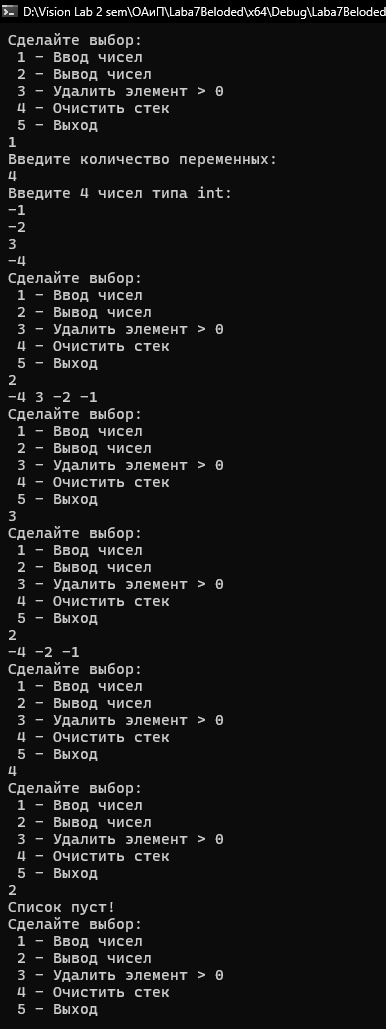
void clear(stack\*& p) {

p->next = NULL; // Установка указателя следующего элемента на NULL

p = NULL; // Установка указателя вершины стека на NULL

}

**Результат выполнения:**



**Вариант 7**

****

**Код программы:**

#include <iostream>

using namespace std;

struct stack

{

int digit;

stack\* next;

};

void push(stack\*& p, int value); // Функция для добавления элемента в стек

void show(stack\* p); // Функция для отображения содержимого стека

void pop(stack\*& p); // Функция для удаления элемента из стека

void clear(stack\*& p); // Функция для очистки стека

int main()

{

setlocale(LC\_CTYPE, "Russian"); // Установка локали для корректного отображения русского текста

stack\* first = nullptr; // Инициализация указателя на вершину стека как nullptr

int value;

menu:

int choice;

cout << "Сделайте выбор:" << endl;

cout << " 1 - Ввод чисел" << endl;

cout << " 2 - Вывод чисел" << endl;

cout << " 3 - Удалить элемент < 0" << endl;

cout << " 4 - Очистить стек" << endl;

cout << " 5 - Выход" << endl;

cin >> choice; // Ввод выбора пользователя

switch (choice) {

case 1: // Ввод чисел

cout << "Введите количество переменных: " << endl;

int n; cin >> n; // Ввод количества чисел

cout << "Введите " << n << " чисел типа int: " << endl;

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

cin >> value; // Ввод числа

push(first, value); // Добавление числа в стек

}

goto menu; // Переход к меню

break;

case 2: // Вывод чисел

show(first); // Отображение содержимого стека

goto menu; // Переход к меню

break;

case 3: // Удалить элемент < 0

stack \* pp;

pp = first;

while (pp != NULL) {

if ((pp->next)->digit > 0) { // Проверка наличия элемента меньше 0

pop(pp); // Удаление элемента

goto menu; // Переход к меню

}

pp = pp->next; // Переход к следующему элементу

}

cout << "Таких элементов нет! \n"; // Вывод сообщения, если таких элементов нет

goto menu; // Переход к меню

break;

case 4: // Очистить стек

stack \* ppp;

ppp = first;

clear(ppp); // Очистка стека

goto menu; // Переход к меню

break;

default:

return 0; // Завершение программы

break;

}

return 0;

}

// Функция для добавления элемента в стек

void push(stack\*& p, int value)

{

stack\* newP = new stack; // Создание нового элемента стека

if (newP != NULL) // Проверка наличия места для добавления элемента

{

newP->digit = value; // Присваивание значению нового элемента

newP->next = p; // Назначение следующего элемента указателю нового элемента

p = newP; // Перемещение указателя на вершину стека на новый элемент

}

else

cout << "Операция добавления не выполнена" << endl; // Вывод сообщения об ошибке

}

// Функция для отображения содержимого стека

void show(stack\* p) {

if (p->next == NULL)

{

cout << "Список пуст!\n"; // Вывод сообщения о пустом стеке

return;

}

while (p != NULL) {

cout << p->digit << " "; // Вывод элементов стека

p = p->next; // Переход к следующему элементу

}

cout << endl;

}

// Функция для удаления элемента из стека

void pop(stack\*& p) {

p->next = (p->next)->next; // Пропуск следующего элемента

}

// Функция для очистки стека

void clear(stack\*& p) {

p->next = NULL; // Установка указателя следующего элемента на NULL

p = NULL; // Установка указателя вершины стека на NULL

}

**Результат выполнения:**

