Учреждение образования

«БЕЛОРУССКИЙГОСУДАРСТВЕННЫЙТЕХНОЛОГИЧЕСКИЙУНИВЕРСИТЕТ»

Н. Н. Пустовалова, Н. В. Пацей

# ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

**в 2-х частях**

**Часть 2**

**Лабораторный практикум**

**Лабораторная работа № 7. Полустатические структуры данных: стеки**

**Лабораторная работа № 7. Полустатические структуры данных: стеки**

|  |  |
| --- | --- |
| **Задание** | **Программа** |
| 1. Изучить реализацию стека на основе *динамического массива*, выполнив программу, приведенную в правой части. |  |
| 2. В программе, приведенной справа, демонстрируется реализация стека на основе *односвязного* *списка*. |  |
| 3. В правой части представлен *проект, состоящий из трех* *частей*: программный модуль с главной функцией, программный модуль с функциями пользователя, осуществляющими операции со стеком, и заголовочный файл.  В программе создается стек из символов аеглийского алфавита на основе *односвязного списка*. |  |

**projectStack.cpp:**

#include <iostream>

#include "myStack.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

int choice;

char x;

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

myStk = NULL; //инициализация первого элемента

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 - Вывод стека с очисткой" << endl;

cout << "6 - Выход" << endl;

cin >> choice;

switch (choice)

{

case 1:

cout << "Введите элемент: " << endl;

cin >> x;

push(x, myStk);

break;

case 2:

x = pop(myStk);

if (x != -1)

{

cout << "Извлеченный элемент: " << x << endl;

}

break;

case 3:

toFile(myStk);

break;

case 4:

fromFile(myStk);

break;

case 5:

cout << "Весь стек: " << endl;

show(myStk);

break;

case 6:

return 0;

break;

}

}

return 0;

}

**myStsack.cpp**

#include <iostream>

#include <fstream>

#include "myStack.h"

using namespace std;

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

{

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

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

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

myStk = e; //сдвиг вершины на позицию вперед

}

char pop(Stack\*& myStk) //Извлечение (удаление) элемента из стека

{

if (myStk == nullptr)

{

cout << "Стек пуст!" << endl;

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

}

else

{

Stack\* e = myStk; //е-переменная для хранения адреса элемента

char x = myStk->data; //запись элемента из поля data в перем. x

if (myStk != nullptr)

{

myStk = myStk->next; //перенос вершины

}

delete e;

return x;

}

}

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

{

Stack\* e = myStk;

Stack buf;

ofstream frm("mStack.dat");

if (frm.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

while (e != nullptr)

{

buf = \*e;

frm.write((char\*)&buf, sizeof(Stack));

e = e->next;

}

frm.close();

cout << "Стек записан в файл mStack.dat\n";

}

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

{

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

ifstream frm("mStack.dat");

if (frm.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

frm.seekg(0);

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

while (!frm.eof())

{

push(buf.data, e);

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

}

frm.close();

while (e != nullptr)

{

buf.data = pop(e);

push(buf.data, p);

myStk = p;

}

cout << "\nСтек считан из файла mStack.dat\n\n";

}

void show(Stack\*& myStk) //Вывод стека с очисткой

{

Stack\* e = myStk;

if (e == nullptr)

{

cout << "Стек пуст!" << endl;

}

while (e != nullptr)

{

cout << pop(e) << " ";

}

cout << endl;

}

**myStack.h**

struct Stack

{

char data; //информационный элемент

Stack\* next; //указатель на следующий элемент

};

void show(Stack\*& myStk);

char pop(Stack\*& myStk);

void push(char x, Stack\*& myStk);

void toFile(Stack\*& myStk);

void fromFile(Stack\*& myStk);

|  |  |
| --- | --- |
| 4. 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.  **Вар.1:**  Разработать функцию, которая по одному стеку строит два новых: **Stack1** из положительных элементов и **Stack2** из отрицательных. |  |

**Главная функция:**

#include <iostream>

#include "function.h"

using namespace std;

void printline()

{

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

}

void showmenu()

{

printline();

cout << "Меню:\n" << "1 - Добавления нового значения в стек\n";

cout << "2 - Удаление элемента из стека\n" << "3 - Запись стека в файл\n";

cout << "4 - Чтение стека из файла\n" << "5 - Очистка стека\n";

cout << "6 - Построить stack1 и stack2\n" << "7 - Вывод стека\n";

cout << "8 - Вывод меню\n" << "0 - Выход из программы\n";

printline();

}

int main()

{

int userInput;

stack\* first = new stack;

first = nullptr;

setlocale(LC\_CTYPE, "Russian");

showmenu();

do

{

cin >> userInput;

switch (userInput)

{

case 1:

inputToStack(first);

showList(first);

break;

case 2:

if (!isStackEmpty(first))

{

delFromStack(first);

}

else

{

cout << "Стек пуст\n";

}

break;

case 3:

inputStackToFile(first);

clear(first); //При записи в файл стек записывается в обратном порядке из-за свойства FILO

readFromFile(first); //Для того чтобы это исправить запишем его 2 раза (2-ая инверсия)

remove("data.bin");

inputStackToFile(first);

cout << "Стек успешно записан в файл data.bin\n";

break;

case 4:

readFromFile(first);

cout << "Данные из файла успешно считаны и записаны в стек: \n";

showList(first);

break;

case 5:

clear(first);

cout << "Стек успешно очищен\n";

break;

case 6:

buildNewStack(first);

break;

case 7:

showList(first);

break;

case 8:

showmenu();

break;

case 0:

break;

default:

cout << "Ошибка ввода";

break;

}

cout << "Что дальше?\n";

} while (userInput != 0);

}

**Файл с функциями для работы со стеком:**

#include <iostream>

#include "function.h"

using namespace std;

FILE\* fdata;

void inputToStack(stack\*& first)

{

stack\* newel = new stack;

cout << "Введите новое значение, которое хотите записать в стек: ";

cin >> newel->value;

newel->next = first;

first = newel;

}

void delFromStack(stack\*& first)

{

stack\* previous = first, \* following = first, \* temp;

int value;

cout << "Введите значение элемента, который хотите удалить: ";

cin >> value;

if (first->value == value)

{

temp = first;

first = first->next;

delete(temp);

}

else

{

while (following->value != value && following != nullptr)

{

previous = following;

following = following->next;

}

if (following != nullptr)

{

temp = following;

previous->next = following->next;

delete(temp);

cout << "Число " << value <<" успешно удалено из стека:\n";

showList(first);

}

else

{

cout << "Заданного числа нет в списке\n";

}

}

}

void showList(stack\* first)

{

stack\* continious = first;

if (first != nullptr)

{

while (continious != nullptr)

{

cout << continious->value << "-->";

continious = continious->next;

}

cout << "NULL\n";

}

else

{

cout << "Стек пуст\n";

}

}

bool isStackEmpty(stack\* first)

{

if (first == nullptr)

{

return true;

}

return false;

}

void inputStackToFile(stack\* first)

{

stack\* buf = first;

if (!fopen\_s(&fdata, "data.bin", "ab"))

{

while (buf != nullptr)

{

fwrite(buf, sizeof(buf), 1, fdata);

buf = buf->next;

}

fclose(fdata);

}

else

{

cout << "Ошибка открытия файла\n";

}

}

void readFromFile(stack\*& first)

{

stack\* buf = new stack;

if (!fopen\_s(&fdata, "data.bin", "rb"))

{

while (fread(buf, sizeof(buf), 1, fdata))

{

buf->next = first;

first = buf;

buf = new stack;

}

fclose(fdata);

}

else

{

cout << "Ошибка открытия файла для чтения\n";

}

}

void clear(stack\*& first)

{

while (first != nullptr)

{

stack\* temp = first;

first = first->next;

delete(temp);

}

}

void buildNewStack(stack\* first)

{

stack\* el = new stack, \* stack1 = nullptr, \* stack2 = nullptr;

if (first != nullptr)

{

while (first != nullptr)

{

el->value = first->value;

if (el->value > 0)

{

el->next = stack1;

stack1 = el;

}

else if (el->value < 0)

{

el->next = stack2;

stack2 = el;

}

first = first->next;

el = new stack;

}

cout << "stack1 и stack2 успешно заполнены:\n\n stack1: ";

showList(stack1);

cout << "stack2: ";

showList(stack2);

}

else

{

cout << "Стек пуст\n";

}

}

**Заголовочный файл:**

struct stack

{

int value;

stack\* next;

};

void inputToStack(stack\*& first);

void delFromStack(stack\*& first);

bool isStackEmpty(stack\* first);

void showList(stack\* first);

void inputStackToFile(stack\* first);

void readFromFile(stack\*& first);

void clear(stack\*& first);

void buildNewStack(stack\* first);