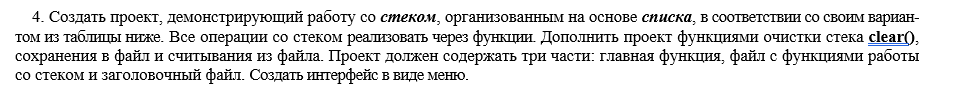
**Варыянт 3**





Stack.cpp:

#include <iostream>

#include "stackFunctions.h"

#include <Windows.h>

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

Stack\* myStack = nullptr;

int choice;

do {

std::cout << "----------------------\n";

std::cout << "1. Дадаць элемент\n";

std::cout << "2. Выдаліць першы дадатны элемент\n";

std::cout << "3. Ачысціць стэк\n";

std::cout << "4. Захаваць стэк у файл\n";

std::cout << "5. Прачытаць стэк з файла\n";

std::cout << "6. Вывесці стэк\n";

std::cout << "0. Выхад\n";

std::cout << "----------------------\n";

std::cout << "Абярыце дзеянне: ";

std::cin >> choice;

switch (choice) {

case 1:

// Даданне элемента ў стэк

int value;

std::cout << "Увядзіце значэнне: ";

std::cin >> value;

push(value, myStack);

break;

case 2:

// Выдаленне першага станоўчага элемента

{

int removed = removeFirstPositive(myStack);

if (removed == -1) {

std::cout << "Стэк не ўтрымлівае станоўчых элементаў\n";

}

else {

std::cout << "Першы станоўчы элемент выдалены: " << removed << "\n";

}

}

break;

case 3:

// Ачыстка стэка

clear(myStack);

std::cout << "Стэк ачышчаны\n";

break;

case 4:

// Захаванне стэка ў файл

saveToFile(myStack, "stack.dat");

break;

case 5:

// Счытванне стэка з файла

readFromFile(myStack, "stack.dat");

break;

case 6:

// Выснова стэка

display(myStack);

break;

case 0:

// Завяршэнне праграмы

clear(myStack); // Ачыстка памяці перад выхадам

std::cout << "Праграма завершана\n";

break;

default:

std::cout << "Няправільны выбар\n";

}

} while (choice!= 0);

return 0;

}

stackFunctions.h:

#pragma once

#include <iostream>

#include <fstream>

struct Stack {

int data;

Stack\* next;

};

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

int pop(Stack\*& myStk);

void clear(Stack\*& myStk);

void saveToFile(Stack\*& myStk, const char\* fileName);

void readFromFile(Stack\*& myStk, const char\* fileName);

void display(Stack\* myStk);

int removeFirstPositive(Stack\*& myStk);

stackFunctions.cpp:

#include "stackFunctions.h"

// Функцыя для дадання новага элемента ў пачатак стэка

void push(int x, Stack\*& myStk) {

// Стварэнне новага вузла і запаўненне яго дадзенымі

Stack\* e = new Stack;

e->data = x;

e->next = myStk; // Усталяваны паказальнік на наступны элемент (перададзены ў функцыю вузел стэка)

myStk = e; // Усталяваны паказальнік на новы элемент у пачатку стэка

}

// Функцыя для выдалення (здабывання) элемента са стэка

int pop(Stack\*& myStk) {

if (myStk == nullptr) {

std::cout << "Стэк пусты!" << std::endl;

return -1;

}

else {

int x = myStk->data;

Stack\* temp = myStk;

myStk = myStk->next; // Перанос вяршыні на наступны элемент

delete temp; // Вызваленне памяці для выдаленага элемента

return x;

}

}

// Функцыя для ачысткі стэка

void clear(Stack\*& myStk) {

while (myStk) {

pop(myStk); // Для кожнага элемента выклікае функцыю, якая выдаліць яго

}

}

// Функцыя для захавання стэка ў файл

void saveToFile(Stack\*& myStk, const char\* fileName) {

std::ofstream file(fileName);

if (!file.is\_open()) {

std::cout << "Памылка адкрыцця файла" << std::endl;

return;

}

Stack\* temp = myStk;

while (temp) {

// Запіс дадзеных стэка ў файл праз прабел

file << temp->data << " ";

temp = temp->next;

}

file.close();

std::cout << "Стэк запісаны ў файл" << fileName << std::endl;

}

// Функцыя для счытвання стэка з файла

void readFromFile(Stack\*& myStk, const char\* fileName) {

// Стварэнне часовага стэка для прачытаных дадзеных

Stack\* tempStack = nullptr;

std::ifstream file(fileName);

if (!file.is\_open()) {

std::cout << "Памылка адкрыцця файла" << std::endl;

return;

}

int value;

// Счытванне дадзеных з файла і даданне іх у часовы стэк

while (file >> value) {

push(value, tempStack);

}

file.close();

// Даданне прачытаных дадзеных у асноўны стэк у зваротным парадку (што захавае першапачатковы парадак лікаў у файле)

while (tempStack) {

push(pop(tempStack), myStk);

}

std::cout << "Стэк лічаны з файла" << fileName << std::endl;

}

// Функцыя для вывада змесціва стэка

void display(Stack\* myStk) {

if (myStk == nullptr) {

std::cout << "Стэк пусты!" << std::endl;

}

else {

std::cout << "Стэк: ";

while (myStk) {

std::cout << myStk->data << " "; // Вывад значэння бягучага элемента

myStk = myStk->next; // Пераход да наступнага элемента

}

std::cout << std::endl;

}

}

// Функцыя для выдалення першага дадатнага элемента са стэка

int removeFirstPositive(Stack\*& myStk) {

// Ініцыялізацыя паказальнікаў для праходу па стэку

Stack\* current = myStk;

Stack\* prev = nullptr;

// Зменная для захоўвання значэння выдалянага элемента (па змаўчанні -1, калі не знойдзены)

int num = -1;

// Перабор элементаў стэка

while (current != nullptr) {

// Праверка, ці з'яўляецца бягучы элемент дадатны

if (current->data > 0) {

// Выдаленне першага дадатнага элемента

// Калі бягучы элемент - першы ў стэку

if (prev == nullptr) {

myStk = current->next; // Перамяшчэнне паказальніка на пачатак стэка на наступны элемент

}

else {

prev->next = current->next; // Пропуск бягучага элемента пры звязванні папярэдняга і наступнага

}

num = current->data; // Захаванне значэння выдалянага элемента

delete current; // Вызваленне памяці элемента, які выдаляецца.

return num; // Вяртанне значэння выдаленага элемента

}

// Пераход да наступнага элемента

prev = current;

current = current->next;

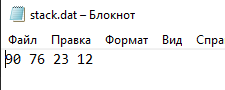
}

// Калі дадатны элемент не знойдзены, вяртаецца значэнне па змаўчанні (-1)

return num;

}

Спачатку



Потым

|  |  |
| --- | --- |
|  |  |

**Варыянт 2**

****

Розніца толькі ў адной функцыі, таму буду тэсціраваць толькі яе

Stack.cpp:

#include <iostream>

#include "stackFunctions.h"

#include <Windows.h>

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

Stack\* myStack = nullptr;

int choice;

do {

std::cout << "----------------------\n";

std::cout << "1. Дадаць элемент\n";

std::cout << "2. Выдаліць першы адмоўны элемент\n";

std::cout << "3. Ачысціць стэк\n";

std::cout << "4. Захаваць стэк у файл\n";

std::cout << "5. Прачытаць стэк з файла\n";

std::cout << "6. Вывесці стэк\n";

std::cout << "0. Выхад\n";

std::cout << "----------------------\n";

std::cout << "Абярыце дзеянне: ";

std::cin >> choice;

switch (choice) {

case 1:

// Даданне элемента ў стэк

int value;

std::cout << "Увядзіце значэнне: ";

std::cin >> value;

push(value, myStack);

break;

case 2:

// Выдаленне першага станоўчага элемента

{

int removed = removeFirstNegative(myStack);

if (removed == -1) {

std::cout << "Стэк не ўтрымлівае адмоўных элементаў\n";

}

else {

std::cout << "Першы адмоўны элемент выдалены: " << removed << "\n";

}

}

break;

case 3:

// Ачыстка стэка

clear(myStack);

std::cout << "Стэк ачышчаны\n";

break;

case 4:

// Захаванне стэка ў файл

saveToFile(myStack, "stack.dat");

break;

case 5:

// Счытванне стэка з файла

readFromFile(myStack, "stack.dat");

break;

case 6:

// Выснова стэка

display(myStack);

break;

case 0:

// Завяршэнне праграмы

clear(myStack); // Ачыстка памяці перад выхадам

std::cout << "Праграма завершана\n";

break;

default:

std::cout << "Няправільны выбар\n";

}

} while (choice!= 0);

return 0;

}

stackFunctions.h:

#pragma once

#include <iostream>

#include <fstream>

struct Stack {

int data;

Stack\* next;

};

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

int pop(Stack\*& myStk);

void clear(Stack\*& myStk);

void saveToFile(Stack\*& myStk, const char\* fileName);

void readFromFile(Stack\*& myStk, const char\* fileName);

void display(Stack\* myStk);

int removeFirstNegative(Stack\*& myStk);

stackFunctions.cpp:

#include "stackFunctions.h"

// Функцыя для дадання новага элемента ў пачатак стэка

void push(int x, Stack\*& myStk) {

// Стварэнне новага вузла і запаўненне яго дадзенымі

Stack\* e = new Stack;

e->data = x;

e->next = myStk; // Усталяваны паказальнік на наступны элемент (перададзены ў функцыю вузел стэка)

myStk = e; // Усталяваны паказальнік на новы элемент у пачатку стэка

}

// Функцыя для выдалення (здабывання) элемента са стэка

int pop(Stack\*& myStk) {

if (myStk == nullptr) {

std::cout << "Стэк пусты!" << std::endl;

return -1;

}

else {

int x = myStk->data;

Stack\* temp = myStk;

myStk = myStk->next; // Перанос вяршыні на наступны элемент

delete temp; // Вызваленне памяці для выдаленага элемента

return x;

}

}

// Функцыя для ачысткі стэка

void clear(Stack\*& myStk) {

while (myStk) {

pop(myStk); // Для кожнага элемента выклікае функцыю, якая выдаліць яго

}

}

// Функцыя для захавання стэка ў файл

void saveToFile(Stack\*& myStk, const char\* fileName) {

std::ofstream file(fileName);

if (!file.is\_open()) {

std::cout << "Памылка адкрыцця файла" << std::endl;

return;

}

Stack\* temp = myStk;

while (temp) {

// Запіс дадзеных стэка ў файл праз прабел

file << temp->data << " ";

temp = temp->next;

}

file.close();

std::cout << "Стэк запісаны ў файл" << fileName << std::endl;

}

// Функцыя для счытвання стэка з файла

void readFromFile(Stack\*& myStk, const char\* fileName) {

// Стварэнне часовага стэка для прачытаных дадзеных

Stack\* tempStack = nullptr;

std::ifstream file(fileName);

if (!file.is\_open()) {

std::cout << "Памылка адкрыцця файла" << std::endl;

return;

}

int value;

// Счытванне дадзеных з файла і даданне іх у часовы стэк

while (file >> value) {

push(value, tempStack);

}

file.close();

// Даданне прачытаных дадзеных у асноўны стэк у зваротным парадку (што захавае першапачатковы парадак лікаў у файле)

while (tempStack) {

push(pop(tempStack), myStk);

}

std::cout << "Стэк лічаны з файла" << fileName << std::endl;

}

// Функцыя для вывада змесціва стэка

void display(Stack\* myStk) {

if (myStk == nullptr) {

std::cout << "Стэк пусты!" << std::endl;

}

else {

std::cout << "Стэк: ";

while (myStk) {

std::cout << myStk->data << " "; // Вывад значэння бягучага элемента

myStk = myStk->next; // Пераход да наступнага элемента

}

std::cout << std::endl;

}

}

// Функцыя для выдалення першага дадатнага элемента са стэка

int removeFirstNegative(Stack\*& myStk) {

// Ініцыялізацыя паказальнікаў для праходу па стэку

Stack\* current = myStk;

Stack\* prev = nullptr;

// Зменная для захоўвання значэння выдалянага элемента (па змаўчанні 1, калі не знойдзены)

int num = 1;

// Перабор элементаў стэка

while (current != nullptr) {

// Праверка, ці з'яўляецца бягучы элемент адмоўным

if (current->data < 0) {

// Выдаленне першага адмоўнага элемента

// Калі бягучы элемент - першы ў стэку

if (prev == nullptr) {

myStk = current->next; // Перамяшчэнне паказальніка на пачатак стэка на наступны элемент

}

else {

prev->next = current->next; // Пропуск бягучага элемента пры звязванні папярэдняга і наступнага

}

num = current->data; // Захаванне значэння выдалянага элемента

delete current; // Вызваленне памяці элемента, які выдаляецца.

return num; // Вяртанне значэння выдаленага элемента

}

// Пераход да наступнага элемента

prev = current;

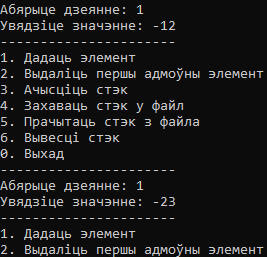
current = current->next;

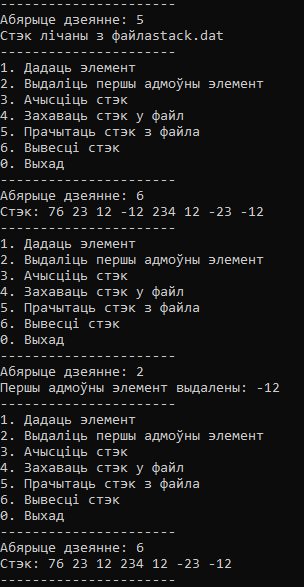
}

// Калі дадатны элемент не знойдзены, вяртаецца значэнне па змаўчанні (-1)

return num;

}





Варыянт 1



Stack.cpp:

#include <iostream>

#include "stackFunctions.h"

#include <Windows.h>

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

Stack\* myStack = nullptr;

int choice;

do {

std::cout << "----------------------\n";

std::cout << "1. Дадаць элемент\n";

std::cout << "2. Разбіць стэк на два, з дадатных і адмоўных лікаў\n";

std::cout << "3. Ачысціць стэк\n";

std::cout << "4. Захаваць стэк у файл\n";

std::cout << "5. Прачытаць стэк з файла\n";

std::cout << "6. Вывесці стэк\n";

std::cout << "0. Выхад\n";

std::cout << "----------------------\n";

std::cout << "Абярыце дзеянне: ";

std::cin >> choice;

switch (choice) {

case 1:

// Даданне элемента ў стэк

int value;

std::cout << "Увядзіце значэнне: ";

std::cin >> value;

push(value, myStack);

break;

case 2:

{

Stack\* posStack = nullptr, \* negStack = nullptr; // Ствараем стэкі для дадатных і адмоўных лікаў

display(myStack);

splitStack(myStack, posStack, negStack);

std::cout << "Дадатны "; display(posStack);

std::cout<< "Адмоўны "; display(negStack);

}

break;

case 3:

// Ачыстка стэка

clear(myStack);

std::cout << "Стэк ачышчаны\n";

break;

case 4:

// Захаванне стэка ў файл

saveToFile(myStack, "stack.dat");

break;

case 5:

// Счытванне стэка з файла

readFromFile(myStack, "stack.dat");

break;

case 6:

// Выснова стэка

display(myStack);

break;

case 0:

// Завяршэнне праграмы

clear(myStack); // Ачыстка памяці перад выхадам

std::cout << "Праграма завершана\n";

break;

default:

std::cout << "Няправільны выбар\n";

}

} while (choice!= 0);

return 0;

}

stackFunctions.h:

#pragma once

#include <iostream>

#include <fstream>

struct Stack {

int data;

Stack\* next;

};

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

int pop(Stack\*& myStk);

void clear(Stack\*& myStk);

void saveToFile(Stack\*& myStk, const char\* fileName);

void readFromFile(Stack\*& myStk, const char\* fileName);

void display(Stack\* myStk);

void splitStack(Stack\*& myStk, Stack\*& pisitiveStack, Stack\*& negativeStack);

stackFunctions.cpp:

#include "stackFunctions.h"

// Функцыя для дадання новага элемента ў пачатак стэка

void push(int x, Stack\*& myStk) {

// Стварэнне новага вузла і запаўненне яго дадзенымі

Stack\* e = new Stack;

e->data = x;

e->next = myStk; // Усталяваны паказальнік на наступны элемент (перададзены ў функцыю вузел стэка)

myStk = e; // Усталяваны паказальнік на новы элемент у пачатку стэка

}

// Функцыя для выдалення (здабывання) элемента са стэка

int pop(Stack\*& myStk) {

if (myStk == nullptr) {

std::cout << "Стэк пусты!" << std::endl;

return -1;

}

else {

int x = myStk->data;

Stack\* temp = myStk;

myStk = myStk->next; // Перанос вяршыні на наступны элемент

delete temp; // Вызваленне памяці для выдаленага элемента

return x;

}

}

// Функцыя для ачысткі стэка

void clear(Stack\*& myStk) {

while (myStk) {

pop(myStk); // Для кожнага элемента выклікае функцыю, якая выдаліць яго

}

}

// Функцыя для захавання стэка ў файл

void saveToFile(Stack\*& myStk, const char\* fileName) {

std::ofstream file(fileName);

if (!file.is\_open()) {

std::cout << "Памылка адкрыцця файла" << std::endl;

return;

}

Stack\* temp = myStk;

while (temp) {

// Запіс дадзеных стэка ў файл праз прабел

file << temp->data << " ";

temp = temp->next;

}

file.close();

std::cout << "Стэк запісаны ў файл" << fileName << std::endl;

}

// Функцыя для счытвання стэка з файла

void readFromFile(Stack\*& myStk, const char\* fileName) {

// Стварэнне часовага стэка для прачытаных дадзеных

Stack\* tempStack = nullptr;

std::ifstream file(fileName);

if (!file.is\_open()) {

std::cout << "Памылка адкрыцця файла" << std::endl;

return;

}

int value;

// Счытванне дадзеных з файла і даданне іх у часовы стэк

while (file >> value) {

push(value, tempStack);

}

file.close();

// Даданне прачытаных дадзеных у асноўны стэк у зваротным парадку (што захавае першапачатковы парадак лікаў у файле)

while (tempStack) {

push(pop(tempStack), myStk);

}

std::cout << "Стэк лічаны з файла" << fileName << std::endl;

}

// Функцыя для вывада змесціва стэка

void display(Stack\* myStk) {

if (myStk == nullptr) {

std::cout << "Стэк пусты!" << std::endl;

}

else {

std::cout << "Стэк: ";

while (myStk) {

std::cout << myStk->data << " "; // Вывад значэння бягучага элемента

myStk = myStk->next; // Пераход да наступнага элемента

}

std::cout << std::endl;

}

}

// Функцыя для рэверсу стэка

void reverseStack(Stack\*& myStack) {

// Ініцыялізацыя часовага стэка

Stack\* tmp = nullptr;

// Пераварочваем стэк

while (myStack) {

// Вымаем элемент з зыходнага стэка і дадаем яго ў часовы стэк

push(pop(myStack), tmp);

}

// Абнаўляем паказальнік на пачатак зыходнага стэка, зараз ён паказвае на перавернуты стэк

myStack = tmp;

}

// Функцыя для разбіцця стэка на два

void splitStack(Stack\*& myStk, Stack\*& positiveStack, Stack\*& negativeStack) {

// Ініцыялізацыя паказальніка для праходу па стэку

Stack\* current = myStk;

// Перабор элементаў стэка

while (current != nullptr) {

// Праверка, ці з'яўляецца бягучы элемент адмоўным

if (current->data < 0) {

push(pop(current), negativeStack); // ЗахоЎваем у стэк дадатных лікаў, выклткаючы push, а значэнне атрымліваем з дапамогай pop

}// Праверка, ці з'яўляецца бягучы элемент дадатным

else if (current->data > 0) {

push(pop(current), positiveStack);

}

else {

pop(current); // Калі 0, то проста выдаляем яго

}

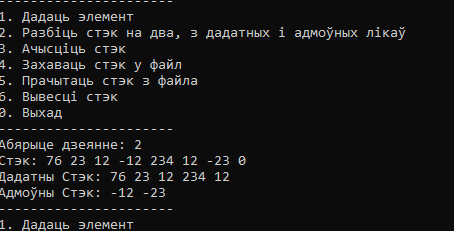
}

// Рэверсуем стэкі. не абавязкова, але проста для прыгажосці, каб захаваць парадак лікаў у зыходным стэку

reverseStack(positiveStack);

reverseStack(negativeStack);

}



**Варыянт 7**



Stack.cpp:

#include <iostream>

#include "stackFunctions.h"

#include <Windows.h>

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

Stack\* myStack = nullptr;

int choice;

do {

std::cout << "----------------------\n";

std::cout << "1. Дадаць элемент\n";

std::cout << "2. Выдаліць першы элемент, большы 100\n";

std::cout << "3. Ачысціць стэк\n";

std::cout << "4. Захаваць стэк у файл\n";

std::cout << "5. Прачытаць стэк з файла\n";

std::cout << "6. Вывесці стэк\n";

std::cout << "0. Выхад\n";

std::cout << "----------------------\n";

std::cout << "Абярыце дзеянне: ";

std::cin >> choice;

switch (choice) {

case 1:

// Даданне элемента ў стэк

int value;

std::cout << "Увядзіце значэнне: ";

std::cin >> value;

push(value, myStack);

break;

case 2:

{

display(myStack);

deleteFirstOccurrence(myStack, 100);

display(myStack);

}

break;

case 3:

// Ачыстка стэка

clear(myStack);

std::cout << "Стэк ачышчаны\n";

break;

case 4:

// Захаванне стэка ў файл

saveToFile(myStack, "stack.dat");

break;

case 5:

// Счытванне стэка з файла

readFromFile(myStack, "stack.dat");

break;

case 6:

// Выснова стэка

display(myStack);

break;

case 0:

// Завяршэнне праграмы

clear(myStack); // Ачыстка памяці перад выхадам

std::cout << "Праграма завершана\n";

break;

default:

std::cout << "Няправільны выбар\n";

}

} while (choice!= 0);

return 0;

}

stackFunctions.h:

#pragma once

#include <iostream>

#include <fstream>

struct Stack {

int data;

Stack\* next;

};

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

int pop(Stack\*& myStk);

void clear(Stack\*& myStk);

void saveToFile(Stack\*& myStk, const char\* fileName);

void readFromFile(Stack\*& myStk, const char\* fileName);

void display(Stack\* myStk);

void deleteFirstOccurrence(Stack\*& myStk, int num);

stackFunctions.cpp:

#include "stackFunctions.h"

// Функцыя для дадання новага элемента ў пачатак стэка

void push(int x, Stack\*& myStk) {

// Стварэнне новага вузла і запаўненне яго дадзенымі

Stack\* e = new Stack;

e->data = x;

e->next = myStk; // Усталяваны паказальнік на наступны элемент (перададзены ў функцыю вузел стэка)

myStk = e; // Усталяваны паказальнік на новы элемент у пачатку стэка

}

// Функцыя для выдалення (здабывання) элемента са стэка

int pop(Stack\*& myStk) {

if (myStk == nullptr) {

std::cout << "Стэк пусты!" << std::endl;

return -1;

}

else {

int x = myStk->data;

Stack\* temp = myStk;

myStk = myStk->next; // Перанос вяршыні на наступны элемент

delete temp; // Вызваленне памяці для выдаленага элемента

return x;

}

}

// Функцыя для ачысткі стэка

void clear(Stack\*& myStk) {

while (myStk) {

pop(myStk); // Для кожнага элемента выклікае функцыю, якая выдаліць яго

}

}

// Функцыя для захавання стэка ў файл

void saveToFile(Stack\*& myStk, const char\* fileName) {

std::ofstream file(fileName);

if (!file.is\_open()) {

std::cout << "Памылка адкрыцця файла" << std::endl;

return;

}

Stack\* temp = myStk;

while (temp) {

// Запіс дадзеных стэка ў файл праз прабел

file << temp->data << " ";

temp = temp->next;

}

file.close();

std::cout << "Стэк запісаны ў файл" << fileName << std::endl;

}

// Функцыя для счытвання стэка з файла

void readFromFile(Stack\*& myStk, const char\* fileName) {

// Стварэнне часовага стэка для прачытаных дадзеных

Stack\* tempStack = nullptr;

std::ifstream file(fileName);

if (!file.is\_open()) {

std::cout << "Памылка адкрыцця файла" << std::endl;

return;

}

int value;

// Счытванне дадзеных з файла і даданне іх у часовы стэк

while (file >> value) {

push(value, tempStack);

}

file.close();

// Даданне прачытаных дадзеных у асноўны стэк у зваротным парадку (што захавае першапачатковы парадак лікаў у файле)

while (tempStack) {

push(pop(tempStack), myStk);

}

std::cout << "Стэк лічаны з файла" << fileName << std::endl;

}

// Функцыя для вывада змесціва стэка

void display(Stack\* myStk) {

if (myStk == nullptr) {

std::cout << "Стэк пусты!" << std::endl;

}

else {

std::cout << "Стэк: ";

while (myStk) {

std::cout << myStk->data << " "; // Вывад значэння бягучага элемента

myStk = myStk->next; // Пераход да наступнага элемента

}

std::cout << std::endl;

}

}

// Функцыя для рэверсу стэка

void reverseStack(Stack\*& myStack) {

// Ініцыялізацыя часовага стэка

Stack\* tmp = nullptr;

// Пераварочваем стэк

while (myStack) {

// Вымаем элемент з зыходнага стэка і дадаем яго ў часовы стэк

push(pop(myStack), tmp);

}

// Абнаўляем паказальнік на пачатак зыходнага стэка, зараз ён паказвае на перавернуты стэк

myStack = tmp;

}

// Функцыя для выдалення першага ўваходжання элемента, большага зададзенага значэння, са стэка

void deleteFirstOccurrence(Stack\*& myStk, int num) {

// Паказальнікі для бягучага элемента, папярэдняга элемента і часовага элемента

Stack\* current = myStk, \* prev = nullptr, \* temp;

// Пошук першага элемента, большага зададзенага значэння

while (current != nullptr) {

if (current->data > num) {

// Праверка, ці з'яўляецца бягучы элемент першым у стэку

if (prev != nullptr) {

// Калі не першы, то карэктуем паказальнікі і выдаляны бягучы элемент

prev->next = current->next;

delete current;

}

else {

// Калі першы, то захоўваем бягучы элемент у часовай зменнай,

// пераходзім да наступнага элемента і выдаляем захаваны элемент

temp = current;

current = current->next;

delete temp;

myStk = current; // Абнаўляем стэк новым пачаткам

}

// Перапыняем цыкл, бо знайшлі і выдалілі першае ўваходжанне

break;

}

// Пераход да наступнага элемента ў стэку

prev = current;

current = current->next;

}

}

