МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Криворізький національний університет

Кафедра моделювання та програмного забезпечення

ЛАБОРАТОРНА РОБОТА №1

З дисципліни «Бази даних»

Тема: «Розробка системи управління нереляційними базами даних.

Функціональний підхід»

Виконав студент групи ІПЗ-21-2

Губарєв Р.В.

Перевірив викладач

Білашенко С.В.

Кривий Ріг

2023

1. **Основні відомості про структури даних та основні алгоритми обробки масивів, списків, черг, стеку.**

**Масив** - впорядкований набір фіксованої кількості однотипних елементів, що зберігаються в послідовно розташованих комірках оперативної пам'яті, мають порядковий номер і спільне ім'я, що надає користувач.

Типові алгоритми опрацювання масивів:

* Знаходження суми чи добутку елементів масиву
* Знаходження мінімального та максимального елемента масиву
* Знаходження елемента масиву з певною властивістю
* Впорядкування елементів масиву

**Список** - це структура даних, в якій елементи лінійно впорядковані, але порядок визначається не номерами елементів, а вказівниками, які входять до складу елементів списку та вказують на наступний елемент (в однозв’язних списках) або на наступний та попередній елементи (в двозв’язних списках).

Типові алгоритми опрацювання списків:

* Визначення позиції елемента в списку
* Видалення та додавання елемента

**Черга** – це динамічна структура даних, що працює за принципом «перший прийшов – перший пішов». У черги є голова та хвіст. Елемент, що додається до черги, знаходиться в її хвості. Елемент, що видаляється з черги, знаходиться в її хвості.

Типові алгоритми опрацювання черг:

* Видалення та додавання елемента

**Стек –** це динамічна структура даних, що працює за принципом «останнім прийшов – першим вийшов».

Типові алгоритми опрацювання черг:

* Видалення та додавання елемента

1. **Основні відомості про принципи організації баз даних: призначення, структури, методи керування.**

Правильно спроектована БД повинна задовольняти наступним вимогам:

Мінімальна надмірність. Несуперечливість. Цілісність даних. Незалежність даних. Можливість ведення (додавання і видалення) та актуалізації (коригування, модифікації) даних. Безпека і таємність. Висока продуктивність. Мінімальні витрати. Дотримання стандартів.

1. **Загальна інформація про файлову систему, файли, типи даних та типи файлів**

**Файлова система** — спосіб організації даних, який використовується операційною системою для збереження інформації у вигляді файлів на носіях інформації. Також цим поняттям позначають сукупність файлів та директорій, які розміщуються на логічному або фізичному пристрої.

**Файл** — інформаційний [об'єкт](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82), що містить дані або програми і розміщується на поіменованій ділянці носія даних, сутність, елемент, що дозволяє отримати [доступ](https://uk.wikipedia.org/w/index.php?title=%D0%94%D0%BE%D1%81%D1%82%D1%83%D0%BF&action=edit&redlink=1) до певного [ресурсу](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D1%81%D1%83%D1%80%D1%81) [обчислювальної системи](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D1%8E%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0" \o "Обчислювальна система) і має такі [ознаки](https://uk.wikipedia.org/wiki/%D0%9E%D0%B7%D0%BD%D0%B0%D0%BA%D0%B0" \o "Ознака):

* фіксована назва;
* певну логічну будову (структуру) і відповідні йому операції читання/запису

**Тип даних** — характеристика, яку явно чи неявно надано об'єкту (змінній, функції, полю запису, константі, масиву тощо).

**Формат файлу** (або **тип файлу**) — це усталений стандарт запису інформації у файлі даного типу. Спосіб кодування інформації або даних залежить від застосованої комп'ютерної програми. Часто формат файлу визначається його розширенням.

1. **Основні відомості про програмні засоби роботи з потоками введення-виведення в мовах програмування**

У мові C++ дії, що пов’язані з операціями введення і виве­дення, виконуються за допомогою функцій бібліотек. Функції ведення і виведення бібліотек мови дозволяють читати дані з файлів та пристроїв і писати дані у файли і на пристрої.

Бібліотека мови C++ підтримує три рівня введення-виведення даних:

* введення-виведення потоку;
* введення-виведення нижнього рівня;
* введення-виведення для консолі і порту.

1. **Блок-схеми алгоритмів роботи функцій і програм**

**Масив / Список**

choice = 3

choice = 2

choice = 1

choice = 0

Введення

choice

Початок

**1**

**2**

**Ні**

**3**

Завантажування масива/списку з файла

**Так**

**4**

**Ні**

**5**

**Так**

Відображення масива/списку

**6**

**Ні**

**Так**

**7**

Зміна елемента масива/списку

**Ні**

**8**

**9**

Додавання елемента

**Так**

**19**

**18**

**17**

**16**

**15**

**14**

**13**

**12**

**10**

**11**

**Так**

**Ні**

choice = 6

Знайти елемент за критерієм

**Ні**

**Так**

За зростанням

choice = 2

**Ні**

**Так**

choice = 5

Сортування масива/списку

**Так**

**Ні**

choice = 4

Видалення елемента

**Так**

choice = 1

За спаданням

**Ні**

**20**

**Так**

choice = 1

Найбільший

**Ні**

**29**

**28**

**27**

**26**

**25**

**24**

**23**

**22**

**21**

**Так**

choice = 4

Перший непарний

**Ні**

**Так**

choice = 5

Перший додатній

**Ні**

**Так**

choice = 3

Перший парний

**Ні**

**Так**

choice = 2

Найменший

**Ні**

**Ні**

Кінець

**Так**

choice = 7

Всі рівні заданому

**Ні**

**Так**

**Ні**

choice = 7

Видалення масива/списку

**33**

**32**

**31**

**30**

**Так**

choice = 6

Перший від’ємний

**Ні**

**34**

**35**

**Так**

choice = 8

**36**

**37**

**Черга / Стек**

choice = 3

choice = 2

choice = 1

choice = 0

Введення

choice

Початок

**1**

**2**

**Ні**

**3**

Завантажування черги/стеку з файла

**Так**

**4**

**Ні**

**5**

**Так**

Відображення черги/стека

**6**

**Ні**

**Так**

**7**

Зміна елемента черги/стека

**Ні**

**8**

**9**

Додавання елемента

**Так**

**19**

**17**

**16**

**15**

**14**

**13**

**12**

**10**

**11**

**Ні**

**Так**

choice = 5

**Так**

**Ні**

choice = 4

Видалення елемента

**18**

**Так**

choice = 3

Перший парний

**Ні**

**Так**

choice = 2

Найменший

**Ні**

**Так**

choice = 1

Найбільший

**Ні**

Знайти елемент за критерієм

**Так**

**Ні**

choice = 7

Видалення масива/списку

**Так**

choice = 7

Всі рівні заданому

**Ні**

**Так**

choice = 6

Перший від’ємний

**Ні**

**29**

**28**

**27**

**26**

**25**

**24**

**23**

**22**

**21**

**20**

**Так**

choice = 5

Перший додатній

**Ні**

**Так**

choice = 4

Перший непарний

**Ні**

**31**

**30**

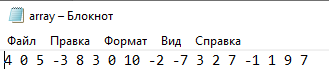
**Так**

choice = 8

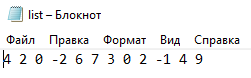
Кінець

1. **Вміст файлів вихідних даних**

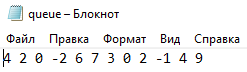
**Масив**

****

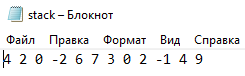
**Список**

****

**Черга**

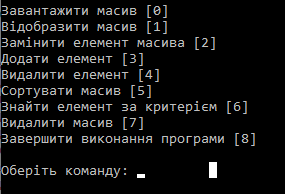
****

**Стек**

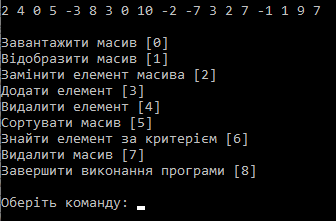
****

1. **Скріншот екрану програми з результатом роботи програми**

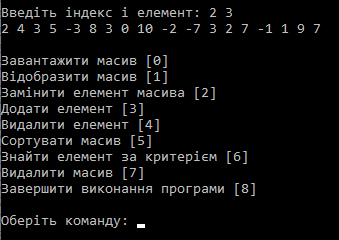
**Масив**

****

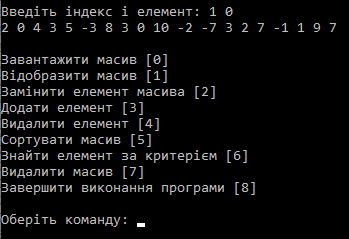
Завантаження з файлу і відображення масива



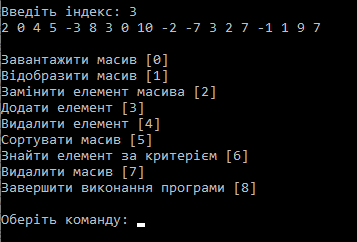
Заміна елемента



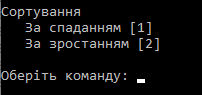
Додавання елемента



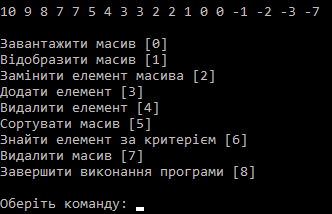
Видалення елемента



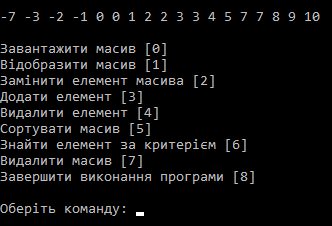
Сортування масива



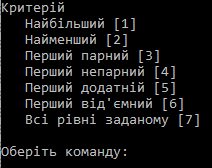
За спаданням

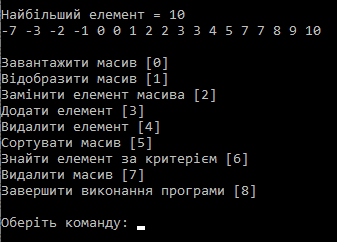


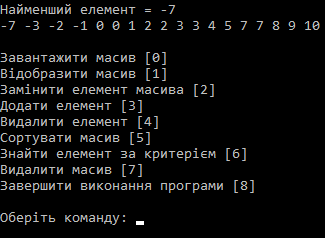
За зростанням

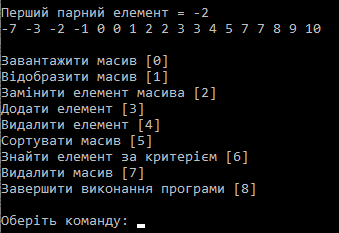


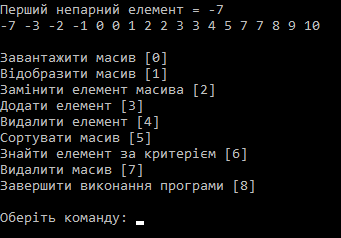
Знаходження елемента за критерієм

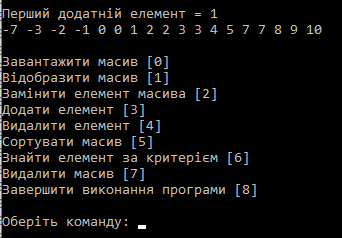


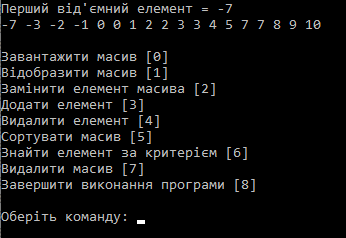


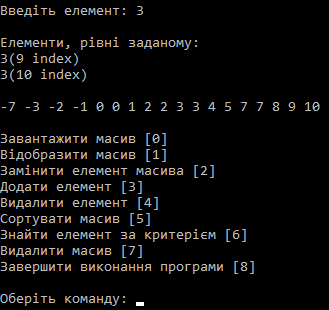




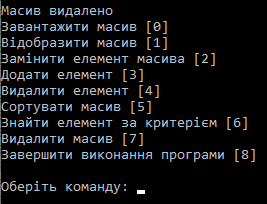






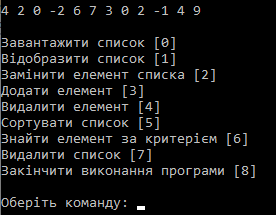


Видалення масива

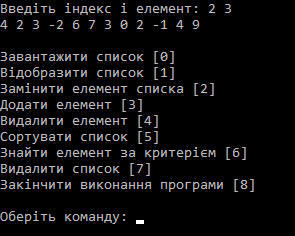


**Список**

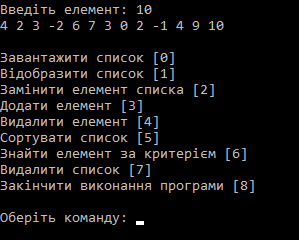
Завантаження з файлу та відображення списку



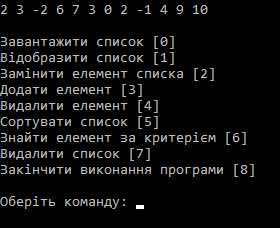
Заміна елемента



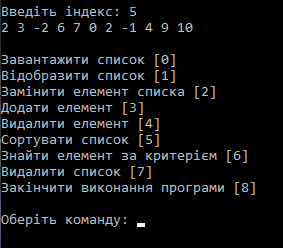
Додавання елемента



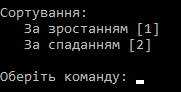
Видалення першого елемента



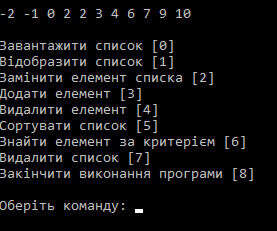
Видалення елемента за індексом



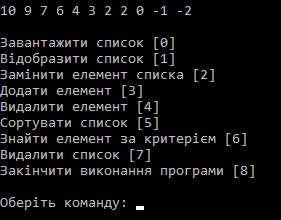
Сортування списку



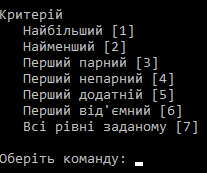
За зростанням

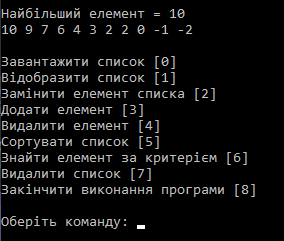


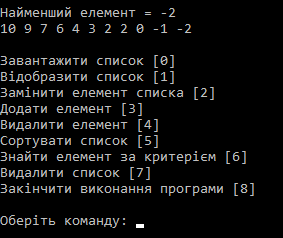
За спаданням

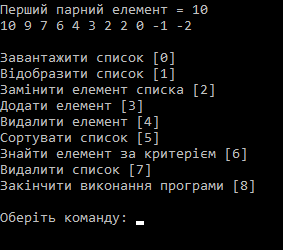


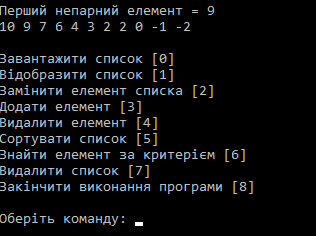
Знаходження елемента за критерієм

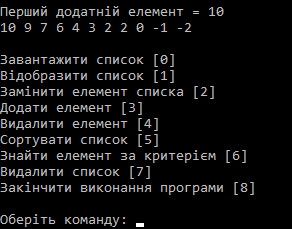


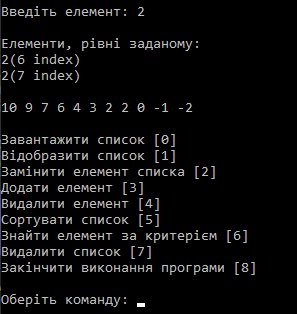






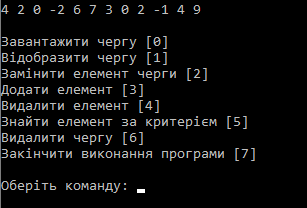




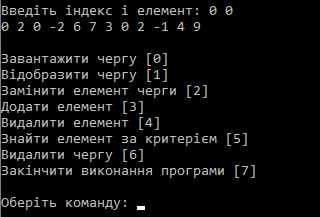


**Черга**

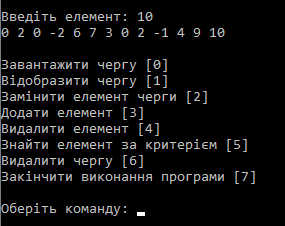
Завантаження та відображення черги



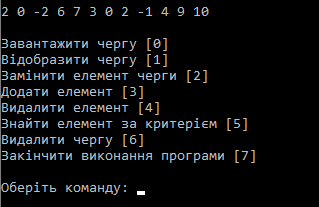
Заміна елемента черги



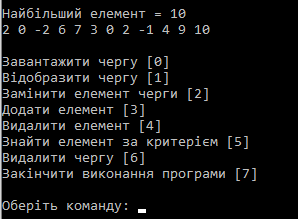
Додавання елемента

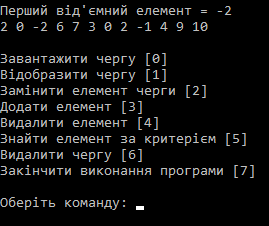


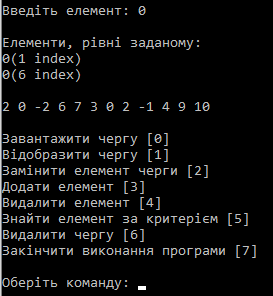
Видалення елемента



Знаходження елемента за критерієм

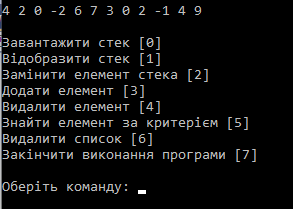




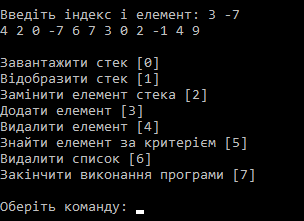


**Стек**

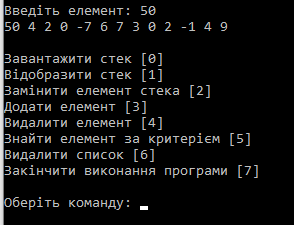
Завантаження та відображення стека



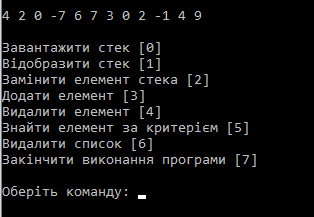
Заміна елемента стека



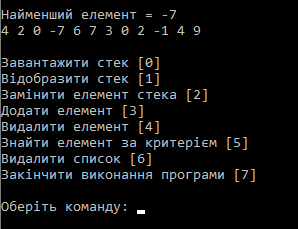
Додавання елемента

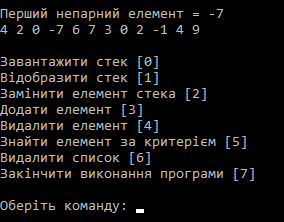


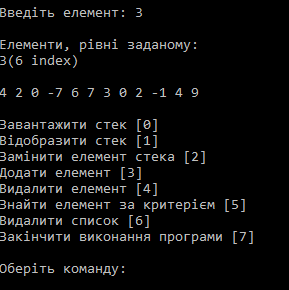
Видалення елемента



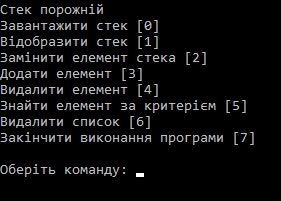
Знаходження елемента за критерієм







Видалення стека



1. **Текст вихідних кодів програм**

|  |
| --- |
| **Масив**  #include <iostream>  #include <iomanip>  #include <fstream>  #include <string>  #include "Windows.h"  using namespace std;  /// <summary>  /// Клас роботи з масивом  /// </summary>  class Array {  int\* arr;  int Size;  public:  double Min;  double Max;  ifstream readFile;  ofstream writeFile;  string filePath = "array.txt";  /// <summary>  /// Метод для отримання розміру масива в файлі  /// </summary>  void GetSize() {  Size = 0;  readFile.open(filePath);  if (!readFile.is\_open())  {  cout << "Помилка відкриття файлу!" << endl;  }  else  {  int x;  while (readFile >> x)  {  Size++;  }  readFile.close();  }  }  /// <summary>  /// Метод для отримання масива з файла  /// </summary>  void GetArray() {  GetSize();  readFile.open(filePath);  if (!readFile.is\_open())  {  cout << "Помилка відкриття файлу!" << endl;  }  else  {  arr = new int[Size];  for (int i = 0; i < Size; i++)  {  readFile >> arr[i];  }  }  }  /// <summary>  /// Метод для виведення масива в консоль  /// </summary>  void Show() {  for (int i = 0; i < Size; i++)  {  cout << arr[i] << " ";  }  cout << endl << endl;  }  /// <summary>  /// Метод зміни елемента за індексом  /// </summary>  /// <param name="element"> Новий елемент </param>  /// <param name="index"> Індекс </param>  void ElementChange(int index, int element) {  writeFile.open(filePath);  if (!writeFile.is\_open())  {  cout << "Помилка відкриття файлу" << endl;  }  else  {  arr[index] = element;  for (int i = 0; i < Size; i++)  {  writeFile << arr[i] << " ";  }  }  writeFile.close();  }  /// <summary>  /// Метод отримання мінімального елемента  /// </summary>  /// <returns> Повертає мінімальний елемент </returns>  double GetMin() {  Min = arr[0];  for (int i = 0; i < Size; i++)  {  if (arr[i] < Min)  Min = arr[i];  else  continue;  }  return Min;  }  /// <summary>  /// Метод отримання максимального елемента  /// </summary>  /// <returns> Повертає максимальний елемент </returns>  double GetMax() {  Max = arr[0];  for (int i = 0; i < Size; i++)  {  if (arr[i] > Max)  Max = arr[i];  else  continue;  }  return Max;  }  /// <summary>  /// Метод для додавання елементу в масив  /// </summary>  /// <param name="value"> Значення елемента </param>  /// <param name="index"> Індекс </param>  void ElementAdd(int index, int value) {  writeFile.open(filePath);  if (!writeFile.is\_open())  {  cout << "Помилка відкриття файлу" << endl;  }  else  {  if (index >= Size + 1)  {  cout << "Неправильно введений індекс\n";  writeFile.close();  return;  }  Size++;  int\* temp = new int[Size];  temp[index] = value;  int j = 0;  for (int i = 0; i < Size; i++)  {  if (i != index)  {  temp[i] = arr[j];  j++;  }  else  continue;  }  arr = new int[Size];  arr = temp;  for (int i = 0; i < Size; i++)  {  writeFile << arr[i] << " ";  }  }  writeFile.close();  }  /// <summary>  /// Метод для видалення елемента  /// </summary>  /// <param name="index"> Індекс елемента </param>  void ElementDelete(int index) {  writeFile.open(filePath);  if (!writeFile.is\_open())  {  cout << "Помилка відкриття файлу" << endl;  }  else  {  if (index >= Size)  {  cout << "Неправильно введений індекс\n";  writeFile.close();  return;  }  int\* temp = new int[Size];  temp = arr;  Size--;  arr = new int[Size];  int j = 0;  for (int i = 0; i < Size + 1; i++)  {  if (i != index)  {  arr[j] = temp[i];  j++;  }  else  continue;  }  for (int i = 0; i < Size; i++)  {  writeFile << arr[i] << " ";  }  }  writeFile.close();  }  /// <summary>  /// Метод для сортування масиву  /// </summary>  /// <param name="check">  /// true - сортування за спаданням  /// false - сортування за зростанням  /// </param>  void Sort(bool check) {  writeFile.open(filePath);  if (!writeFile.is\_open())  {  cout << "Помилка відкриття файлу" << endl;  }  else  {  if (check == true)  {  for (int i = 0; i < Size; i++)  {  for (int j = i; j < Size; j++)  {  if (arr[i] < arr[j])  {  int temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }  }  }  }  else  {  for (int i = 0; i < Size; i++)  {  for (int j = i; j < Size; j++)  {  if (arr[i] > arr[j])  {  int temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }  }  }  }  for (int i = 0; i < Size; i++)  {  writeFile << arr[i] << " ";  }  }  writeFile.close();  }  /// <summary>  /// Метод для пошуку першого парного або непарного елементу  /// </summary>  /// <param name="check">  /// true - пошук першого парного  /// false - пошук першого непарного  /// </param>  /// <returns> Повертає результат пошуку </returns>  int EvenOrOddCheck(bool check) {  int result;  if (check == true)  {  for (int i = 0; i < Size; i++)  {  if (arr[i] % 2 == 0) {  result = arr[i];  break;  }  else  continue;  }  }  else  {  for (int i = 0; i < Size; i++)  {  if (arr[i] % 2 != 0) {  result = arr[i];  break;  }  else  continue;  }  }  return result;  }  /// <summary>  /// Метод для пошуку першого додатнього або від'ємного елементу  /// </summary>  /// <param name="check">  /// true - пошук першого додатнього  /// false - пошук першого від'ємного  /// </param>  /// <returns> Повертає результат пошуку </returns>  int PositiveCheck(bool check) {  int result;  if (check == true)  {  for (int i = 0; i < Size; i++)  {  if (arr[i] > 0) {  result = arr[i];  break;  }  else  continue;  }  }  else  {  for (int i = 0; i < Size; i++)  {  if (arr[i] < 0) {  result = arr[i];  break;  }  else  continue;  }  }  return result;  }  /// <summary>  /// Метод для пошуку всіх елементів, рівних заданому  /// </summary>  /// <param name="element"> Заданий елемент </param>  void Search(int element) {  for (int i = 0; i < Size; i++)  {  if (element == arr[i])  {  cout << arr[i] << "(" << i << " index)\n";  }  }  cout << endl;  }  /// <summary>  /// Метод для видалення масиву  /// </summary>  void ArrayDelete() {  delete[] arr;  Size = 0;  cout << "Масив видалено\n";  }  /// <summary>  /// Метод для закриття файлу  /// </summary>  void Close() {  readFile.close();  }  };  /// <summary>  /// Функція, що викликає меню  /// </summary>  /// <returns> Повертає значення, введене користувачем </returns>  int Menu() {  int check;  cout << "Завантажити масив [0]\n";  cout << "Відобразити масив [1]\n";  cout << "Замінити елемент масива [2]\n";  cout << "Додати елемент [3]\n";  cout << "Видалити елемент [4]\n";  cout << "Сортувати масив [5]\n";  cout << "Знайти елемент за критерієм [6]\n";  cout << "Видалити масив [7]\n";  cout << "Завершити виконання програми [8]\n";  cout << "\nОберіть команду: ";  cin >> check;  if (check == 5)  {  system("cls");  cout << "Сортування\n";  cout << " За спаданням [1]\n";  cout << " За зростанням [2]\n";  cout << "\nОберіть команду: ";  cin >> check;  check += 50;  }  if (check == 6)  {  system("cls");  cout << "Критерій\n";  cout << " Найбільший [1]\n";  cout << " Найменший [2]\n";  cout << " Перший парний [3]\n";  cout << " Перший непарний [4]\n";  cout << " Перший додатній [5]\n";  cout << " Перший від'ємний [6]\n";  cout << " Всі рівні заданому [7]\n";  cout << "\nОберіть команду: ";  cin >> check;  check += 60;  }  return check;  }  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  Array arr;  int choice;  int index, element;  for (; ; )  {  choice = Menu();  switch (choice)  {  case 0:  system("cls");  arr.GetArray();  break;  case 1:  system("cls");  arr.Show();  break;  case 2:  system("cls");  cout << "Введіть індекс і елемент: ";  cin >> index >> element;  arr.ElementChange(index, element);  arr.Show();  break;  case 3:  system("cls");  cout << "Введіть індекс і елемент: ";  cin >> index >> element;  arr.ElementAdd(index, element);  arr.Show();  break;  case 4:  system("cls");  cout << "Введіть індекс: ";  cin >> index;  arr.ElementDelete(index);  arr.Show();  break;  case 51:  system("cls");  arr.Sort(true);  arr.Show();  break;  case 52:  system("cls");  arr.Sort(false);  arr.Show();  break;  case 61:  system("cls");  cout << "Найбільший елемент = " << arr.GetMax() << "\n";  arr.Show();  break;  case 62:  system("cls");  cout << "Найменший елемент = " << arr.GetMin() << "\n";  arr.Show();  break;  case 63:  system("cls");  cout << "Перший парний елемент = " << arr.EvenOrOddCheck(true) << "\n";  arr.Show();  break;  case 64:  system("cls");  cout << "Перший непарний елемент = " << arr.EvenOrOddCheck(false) << "\n";  arr.Show();  break;  case 65:  system("cls");  cout << "Перший додатній елемент = " << arr.PositiveCheck(true) << "\n";  arr.Show();  break;  case 66:  system("cls");  cout << "Перший від'ємний елемент = " << arr.PositiveCheck(false) << "\n";  arr.Show();  break;  case 67:  system("cls");  cout << "Введіть елемент: ";  cin >> element;  cout << "\nЕлементи, рівні заданому:\n";  arr.Search(element);  arr.Show();  break;  case 7:  system("cls");  arr.ArrayDelete();  break;  case 8:  system("cls");  arr.Close();  system("pause");  break;  }  }  } |
| **Список**  #include <iostream>  #include <iomanip>  #include <fstream>  #include <Windows.h>  using namespace std;  /// <summary>  /// Структура, що представляє вузол списку  /// </summary>  /// <typeparam name="T"></typeparam>  template <class T>  struct Node {  T data;  Node\* next;  };  /// <summary>  /// Клас, що представляє однозв'язний список  /// </summary>  /// <typeparam name="T"></typeparam>  template <class T>  class List {  private:  Node<T>\* begin;  Node<T>\* end;  int count;  T MAX;  T MIN;  ifstream readFile;  ofstream writeFile;  string filePath = "list.txt";  /// <summary>  /// Метод для пошуку елемента за індексом  /// </summary>  /// <param name="index"> Індекс </param>  /// <returns> Повертає елемент </returns>  Node<T>\* Move(int index) {  if (count > 0)  {  Node<T>\* t = begin;  for (int i = 0; i < index; i++)  {  t = t->next;  }  return t;  }  return nullptr;  }  /// <summary>  /// Метод для отримання розміру списка в файлі  /// </summary>  int GetSize() {  int Size = 0;  readFile.open(filePath);  if (!readFile.is\_open())  {  cout << "Помилка відкриття файлу!" << endl;  }  else  {  int x;  while (readFile >> x)  {  Size++;  }  readFile.close();  }  return Size;  }  public:  /// <summary>  /// Конструктор  /// </summary>  List() {  begin = end = nullptr;  count = 0;  }  /// <summary>  /// Метод для додавання елемента  /// </summary>  /// <param name="\_data"> Елемент </param>  void Add(T \_data) {  Node<T>\* node = new Node<T>;  node->data = \_data;  node->next = nullptr;  if (begin == nullptr)  begin = end = node;  else  {  end->next = node;  end = node;  }  count++;  }  /// <summary>  /// Метод для зміни елемента за індексом  /// </summary>  /// <param name="index"> Індекс </param>  /// <returns> Повертає елемент </returns>  T& operator[](int index)  {  if ((index < 0) || (index > count - 1))  {  throw out\_of\_range("Неправильний індекс.");  }  Node<T>\* t = Move(index);  return t->data;  }  /// <summary>  /// Метод для видалення елемента за індексом  /// </summary>  /// <param name="index"> Індекс </param>  void Delete(int index) {  if (count == 0)  return;  if ((index < 0) || (index >= count))  return;  if (index == 0)  {  Node<T>\* t = begin;  begin = begin->next;  delete t;  }  else  {  Node<T>\* t = Move(index - 1);  Node<T>\* t2 = t->next;  t->next = t2->next;  delete t2;  }  count--;  }  /// <summary>  /// Видалення першого елемента  /// </summary>  void Delete() {  Delete(0);  }  /// <summary>  /// Метод для очищення списку  /// </summary>  void Clear() {  while (begin != nullptr)  {  Delete();  }  }  /// <summary>  /// Деструктор  /// </summary>  ~List() {  Clear();  }  /// <summary>  /// Виведення списку  /// </summary>  void Print() {  if (count == 0)  {  cout << "Список порожній" << endl << endl;  return;  }  Node<T>\* t = begin;  while (t != nullptr)  {  cout << t->data << " ";  t = t->next;  }  cout << endl << endl;  }  /// <summary>  /// Заповнення списка з файлу  /// </summary>  void AddFromTheFile() {  int Size = GetSize();  readFile.open(filePath);  if (!readFile.is\_open())  {  cout << "Помилка відкриття файлу!" << endl;  }  else  {  int temp;  for (int i = 0; i < Size; i++)  {  readFile >> temp;  Add(temp);  }  }  }  /// <summary>  /// Сортування  /// </summary>  /// <param name="check">  /// true - сортування за спаданням  /// false - сортування за зростанням  /// </param>  void Sort(bool check) {  Node<T>\* ptrN = begin;  while (ptrN->next != NULL)  {  Node<T>\* ptr = begin;  while (ptr->next != NULL)  {  if (check == true)  {  if (ptr->data > ptr->next->data)  {  T temp = ptr->data;  ptr->data = ptr->next->data;  ptr->next->data = temp;  }  }  else  {  if (ptr->data < ptr->next->data)  {  T temp = ptr->data;  ptr->data = ptr->next->data;  ptr->next->data = temp;  }  }  ptr = ptr->next;  }  ptrN = ptrN->next;  }  }  /// <summary>  /// Знаходження максимальнго елемента  /// </summary>  /// <returns> Повертає максимальний елемент </returns>  T GetMax() {  Node<T>\* ptr = begin;  MAX = ptr->data;  int i = 0;  while (i != count)  {  if (ptr->data > MAX)  {  MAX = ptr->data;  }  ptr = ptr->next;  i++;  }  return MAX;  }  /// <summary>  /// Знаходження мінімального елемента  /// </summary>  /// <returns> Повертає мінімальний елемент </returns>  T GetMin() {  Node<T>\* ptr = begin;  MIN = ptr->data;  int i = 0;  while (i != count)  {  if (ptr->data < MIN)  {  MIN = ptr->data;  }  ptr = ptr->next;  i++;  }  return MIN;  }  /// <summary>  /// Метод для пошуку першого парного або непарного елементу  /// </summary>  /// <param name="check">  /// true - пошук першого парного  /// false - пошук першого непарного  /// </param>  /// <returns> Повертає результат пошуку </returns>  int EvenOrOddCheck(bool check) {  Node<T>\* ptr = begin;  T result;  if (check == true)  {  int i = 0;  while (i != count)  {  if (ptr->data%2 == 0)  {  result = ptr->data;  break;  }  else {  ptr = ptr->next;  i++;  continue;  }  }  }  else  {  int i = 0;  while (i != count)  {  if (ptr->data % 2 != 0)  {  result = ptr->data;  break;  }  else {  ptr = ptr->next;  i++;  continue;  }  }  }  return result;  }  /// <summary>  /// Метод для пошуку першого додатнього або від'ємного елементу  /// </summary>  /// <param name="check">  /// true - пошук першого додатнього  /// false - пошук першого від'ємного  /// </param>  /// <returns> Повертає результат пошуку </returns>  int PositiveCheck(bool check) {  Node<T>\* ptr = begin;  T result;  if (check == true)  {  int i = 0;  while (i != count)  {  if (ptr->data > 0)  {  result = ptr->data;  break;  }  else {  ptr = ptr->next;  i++;  continue;  }  }  }  else  {  int i = 0;  while (i != count)  {  if (ptr->data < 0)  {  result = ptr->data;  break;  }  else {  ptr = ptr->next;  i++;  continue;  }  }  }  return result;  }  /// <summary>  /// Метод для пошуку всіх елементів, рівних заданому  /// </summary>  /// <param name="element"> Заданий елемент </param>  void Search(int element) {  Node<T>\* ptr = begin;  int i = 0;  while (i != count)  {  if (ptr->data == element)  {  cout << ptr->data << "(" << i << " index)\n";  }  ptr = ptr->next;  i++;  }  cout << endl;  }  /// <summary>  /// Запис списка в файл  /// </summary>  void WriteToTheFile() {  writeFile.open(filePath);  if (!writeFile.is\_open())  {  cout << "Помилка відкриття файлу" << endl;  }  else  {  Node<T>\* ptr = begin;  int i = 0;  while (i != count)  {  writeFile << ptr->data << " ";  ptr = ptr->next;  i++;  }  }  writeFile.close();  }  /// <summary>  /// Метод для закриття файлу  /// </summary>  void Close() {  readFile.close();  }  };  /// <summary>  /// Функція, що викликає меню  /// </summary>  /// <returns> Повертає значення, введене користувачем </returns>  int Menu() {  int check;  cout << "Завантажити список [0]\n";  cout << "Відобразити список [1]\n";  cout << "Замінити елемент списка [2]\n";  cout << "Додати елемент [3]\n";  cout << "Видалити елемент [4]\n";  cout << "Сортувати список [5]\n";  cout << "Знайти елемент за критерієм [6]\n";  cout << "Видалити список [7]\n";  cout << "Закінчити виконання програми [8]\n";  cout << "\nОберіть команду: ";  cin >> check;  if (check == 4)  {  system("cls");  cout << "Видалення:\n";  cout << " Першого елемента [1]\n";  cout << " За індексом [2]\n";  cout << "\nОберіть команду: ";  cin >> check;  check += 40;  }  if (check == 5)  {  system("cls");  cout << "Сортування:\n";  cout << " За зростанням [1]\n";  cout << " За спаданням [2]\n";  cout << "\nОберіть команду: ";  cin >> check;  check += 50;  }  if (check == 6)  {  system("cls");  cout << "Критерій\n";  cout << " Найбільший [1]\n";  cout << " Найменший [2]\n";  cout << " Перший парний [3]\n";  cout << " Перший непарний [4]\n";  cout << " Перший додатній [5]\n";  cout << " Перший від'ємний [6]\n";  cout << " Всі рівні заданому [7]\n";  cout << "\nОберіть команду: ";  cin >> check;  check += 60;  }  return check;  }  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  List<int> L;  int choice;  int index, element;  for (; ; )  {  choice = Menu();  switch (choice)  {  case 0:  system("cls");  L.AddFromTheFile();  break;  case 1:  system("cls");  L.Print();  break;  case 2:  system("cls");  cout << "Введіть індекс і елемент: ";  cin >> index >> element;  L[index] = element;  L.Print();  L.WriteToTheFile();  break;  case 3:  system("cls");    cout << "Введіть елемент: ";  cin >> element;  L.Add(element);  L.Print();  L.WriteToTheFile();  break;  case 41:  system("cls");    L.Delete();  L.Print();  L.WriteToTheFile();  break;  case 42:  system("cls");  cout << "Введіть індекс: ";  cin >> index;  L.Delete(index);  L.Print();  L.WriteToTheFile();  break;  case 51:  system("cls");  L.Sort(true);  L.Print();  L.WriteToTheFile();  break;  case 52:  system("cls");  L.Sort(false);  L.Print();  L.WriteToTheFile();  break;  case 61:  system("cls");  cout << "Найбільший елемент = " << L.GetMax() << endl;  L.Print();  L.WriteToTheFile();  break;  case 62:  system("cls");  cout << "Найменший елемент = " << L.GetMin() << endl;  L.Print();  L.WriteToTheFile();  break;  case 63:  system("cls");  cout << "Перший парний елемент = " << L.EvenOrOddCheck(true) << endl;  L.Print();  L.WriteToTheFile();  break;  case 64:  system("cls");  cout << "Перший непарний елемент = " << L.EvenOrOddCheck(false) << endl;  L.Print();  L.WriteToTheFile();  break;  case 65:  system("cls");  cout << "Перший додатній елемент = " << L.PositiveCheck(true) << endl;  L.Print();  L.WriteToTheFile();  break;  case 66:  system("cls");  cout << "Перший від'ємний елемент = " << L.PositiveCheck(false) << endl;  L.Print();  L.WriteToTheFile();  break;  case 67:  system("cls");  cout << "Введіть елемент: ";  cin >> element;  cout << "\nЕлементи, рівні заданому:\n";  L.Search(element);  L.Print();  L.WriteToTheFile();  break;  case 7:  system("cls");  L.Clear();  L.WriteToTheFile();  break;  case 8:  system("cls");  L.Close();  system("pause");  break;  }  }  } |
| **Черга**  #include <iostream>  #include <iomanip>  #include <fstream>  #include <Windows.h>  using namespace std;  /// <summary>  /// Клас роботи з чергою  /// </summary>  /// <typeparam name="T"></typeparam>  template <typename T>  class Queue {  private:  T\* p;  int count;  ifstream readFile;  ofstream writeFile;  string filePath = "queue.txt";  public:  /// <summary>  /// Метод для отримання розміру черги в файлі  /// </summary>  int GetSize() {  int Size = 0;  readFile.open(filePath);  if (!readFile.is\_open())  {  cout << "Помилка відкриття файлу!" << endl;  }  else  {  int x;  while (readFile >> x)  {  Size++;  }  readFile.close();  }  return Size;  }  /// <summary>  /// Метод для отримання черги з файла  /// </summary>  void AddQueueFromTheFile() {  int Size = GetSize();  readFile.open(filePath);  if (!readFile.is\_open())  {  cout << "Помилка відкриття файлу!" << endl;  }  else  {  int temp;  for (int i = 0; i < Size; i++)  {  readFile >> temp;  Push(temp);  }  }  }  /// <summary>  /// Конструктор  /// </summary>  Queue() {  count = 0;  }  /// <summary>  /// Додавання елемента  /// </summary>  /// <param name="item"> Елемент </param>  void Push(T item) {  T\* p2;  p2 = p;  p = new T[count + 1];    for (int i = 0; i < count; i++)  p[i] = p2[i];  p[count] = item;  count++;  if (count > 1)  delete[] p2;  }  /// <summary>  /// Видалення першого елемента  /// </summary>  T Pop() {  if (count == 0)  return 0;  T\* p2;  p2 = new T[count - 1];  count--;  for (int i = 0; i < count; i++)  p2[i] = p[i + 1];  if (count > 0)  delete[] p;  p = p2;  }  /// <summary>  /// Декструктор  /// </summary>  ~Queue() {  if (count > 0)  delete[] p;  }  /// <summary>  /// Видалення черги  /// </summary>  void Clear() {  if (count > 0)  {  delete[] p;  count = 0;  }  }  /// <summary>  /// Виведення черги  /// </summary>  void Print() {  if (count == 0)  cout << "Черга порожня";  for (int i = 0; i < count; i++)  cout << p[i] << " ";  cout << endl << endl;  }  /// <summary>  /// Запис черги в файл  /// </summary>  void WriteToTheFile() {  writeFile.open(filePath);  if (!writeFile.is\_open())  {  cout << "Помилка відкриття файлу" << endl;  }  else  {  for (int i = 0; i < count; i++)  {  writeFile << p[i] << " ";  }  }  writeFile.close();  }  /// <summary>  /// Метод зміни елемента за індексом  /// </summary>  /// <param name="element"> Новий елемент </param>  /// <param name="index"> Індекс </param>  void ElementChange(int index, int element) {  p[index] = element;  }  /// <summary>  /// Метод отримання мінімального елемента  /// </summary>  /// <returns> Повертає мінімальний елемент </returns>  double GetMin() {  int Min = p[0];  for (int i = 0; i < count; i++)  {  if (p[i] < Min)  Min = p[i];  else  continue;  }  return Min;  }  /// <summary>  /// Метод отримання максимального елемента  /// </summary>  /// <returns> Повертає максимальний елемент </returns>  double GetMax() {  int Max = p[0];  for (int i = 0; i < count; i++)  {  if (p[i] > Max)  Max = p[i];  else  continue;  }  return Max;  }  /// <summary>  /// Метод для пошуку першого парного або непарного елементу  /// </summary>  /// <param name="check">  /// true - пошук першого парного  /// false - пошук першого непарного  /// </param>  /// <returns> Повертає результат пошуку </returns>  int EvenOrOddCheck(bool check) {  int result;  if (check == true)  {  for (int i = 0; i < count; i++)  {  if (p[i] % 2 == 0) {  result = p[i];  break;  }  else  continue;  }  }  else  {  for (int i = 0; i < count; i++)  {  if (p[i] % 2 != 0) {  result = p[i];  break;  }  else  continue;  }  }  return result;  }  /// <summary>  /// Метод для пошуку першого додатнього або від'ємного елементу  /// </summary>  /// <param name="check">  /// true - пошук першого додатнього  /// false - пошук першого від'ємного  /// </param>  /// <returns> Повертає результат пошуку </returns>  int PositiveCheck(bool check) {  int result;  if (check == true)  {  for (int i = 0; i < count; i++)  {  if (p[i] > 0) {  result = p[i];  break;  }  else  continue;  }  }  else  {  for (int i = 0; i < count; i++)  {  if (p[i] < 0) {  result = p[i];  break;  }  else  continue;  }  }  return result;  }  /// <summary>  /// Метод для пошуку всіх елементів, рівних заданому  /// </summary>  /// <param name="element"> Заданий елемент </param>  void Search(int element) {  for (int i = 0; i < count; i++)  {  if (element == p[i])  {  cout << p[i] << "(" << i << " index)\n";  }  }  cout << endl;  }  /// <summary>  /// Метод для закриття файлу  /// </summary>  void Close() {  readFile.close();  }  };  /// <summary>  /// Функція, що викликає меню  /// </summary>  /// <returns> Повертає значення, введене користувачем </returns>  int Menu() {  int check;  cout << "Завантажити чергу [0]\n";  cout << "Відобразити чергу [1]\n";  cout << "Замінити елемент черги [2]\n";  cout << "Додати елемент [3]\n";  cout << "Видалити елемент [4]\n";  cout << "Знайти елемент за критерієм [5]\n";  cout << "Видалити чергу [6]\n";  cout << "Закінчити виконання програми [7]\n";  cout << "\nОберіть команду: ";  cin >> check;  if (check == 5)  {  system("cls");  cout << "Критерій\n";  cout << " Найбільший [1]\n";  cout << " Найменший [2]\n";  cout << " Перший парний [3]\n";  cout << " Перший непарний [4]\n";  cout << " Перший додатній [5]\n";  cout << " Перший від'ємний [6]\n";  cout << " Всі рівні заданому [7]\n";  cout << "\nОберіть команду: ";  cin >> check;  check += 50;  }  return check;  }  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  Queue<int> q;  int choice;  int index, element;  for (; ; )  {  choice = Menu();  switch (choice)  {  case 0:  system("cls");  q.AddQueueFromTheFile();  break;  case 1:  system("cls");  q.Print();  break;  case 2:  system("cls");  cout << "Введіть індекс і елемент: ";  cin >> index >> element;  q.ElementChange(index, element);  q.Print();  q.WriteToTheFile();  break;  case 3:  system("cls");  cout << "Введіть елемент: ";  cin >> element;  q.Push(element);  q.Print();  q.WriteToTheFile();  break;  case 4:  system("cls");  q.Pop();  q.Print();  q.WriteToTheFile();  break;  case 51:  system("cls");  cout << "Найбільший елемент = " << q.GetMax() << "\n";  q.Print();  break;  case 52:  system("cls");  cout << "Найменший елемент = " << q.GetMin() << "\n";  q.Print();  break;  case 53:  system("cls");  cout << "Перший парний елемент = " << q.EvenOrOddCheck(true) << "\n";  q.Print();  break;  case 54:  system("cls");  cout << "Перший непарний елемент = " << q.EvenOrOddCheck(false) << "\n";  q.Print();  break;  case 55:  system("cls");  cout << "Перший додатній елемент = " << q.PositiveCheck(true) << "\n";  q.Print();  break;  case 56:  system("cls");  cout << "Перший від'ємний елемент = " << q.PositiveCheck(false) << "\n";  q.Print();  break;  case 57:  system("cls");  cout << "Введіть елемент: ";  cin >> element;  cout << "\nЕлементи, рівні заданому:\n";  q.Search(element);  q.Print();  break;  case 6:  system("cls");  q.Clear();  q.WriteToTheFile();  break;  case 7:  system("cls");  q.Close();  system("pause");  break;  }  }  } |
| **Стек**  #include <iostream>  #include <iomanip>  #include <fstream>  #include <Windows.h>  using namespace std;  /// <summary>  /// Структура, що представляє вузол стека  /// </summary>  /// <typeparam name="T"></typeparam>  template <class T>  struct Node {  T data;  Node\* next;  };  /// <summary>  /// Клас, що представляє роботу стека  /// </summary>  /// <typeparam name="T"></typeparam>  template <class T>  class Stack {  private:  Node<T>\* head;  int count;  T MAX;  T MIN;  ifstream readFile;  ofstream writeFile;  string filePath = "stack.txt";  /// <summary>  /// Метод для пошуку елемента за індексом  /// </summary>  /// <param name="index"> Індекс </param>  /// <returns> Повертає елемент </returns>  Node<T>\* Move(int index) {  if (count > 0)  {  Node<T>\* t = head;  for (int i = 0; i < index; i++)  {  t = t->next;  }  return t;  }  return nullptr;  }  /// <summary>  /// Метод для отримання розміру стека в файлі  /// </summary>  int GetSize() {  int Size = 0;  readFile.open(filePath);  if (!readFile.is\_open())  {  cout << "Помилка відкриття файлу!" << endl;  }  else  {  int x;  while (readFile >> x)  {  Size++;  }  readFile.close();  }  return Size;  }  public:  /// <summary>  /// Конструктор  /// </summary>  Stack() {  head = nullptr;  count = 0;  }  /// <summary>  /// Заповнення стека з файлу  /// </summary>  void AddFromTheFile() {  int Size = GetSize();  readFile.open(filePath);  if (!readFile.is\_open())  {  cout << "Помилка відкриття файлу!" << endl;  }  else  {  T\* t = new T[Size];  for (int i = 0; i < Size; i++)  readFile >> t[i];  for (int j = Size-1; j >= 0; j--)  Push(t[j]);  }  }  /// <summary>  /// Метод для додавання елемента  /// </summary>  /// <param name="\_data"> Елемент </param>  void Push(T \_data) {  Node<T>\* node = new Node<T>;  node->data = \_data;  node->next = head;  head = node;  count++;  }  /// <summary>  /// Метод для видалення елемента  /// </summary>  void Pop() {  Node<T>\* node;  T data;  data = head->data;  node = head;  head = head->next;  count--;  delete node;  }  /// <summary>  /// Виведення стека  /// </summary>  void Print() {  if (head == nullptr)  cout << "Стек порожній\n";  else  {  Node<T>\* node;  node = head;  while (node != nullptr)  {  cout << node->data << " ";  node = node->next;  }  cout << endl << endl;  }  }  /// <summary>  /// Очищення стеку  /// </summary>  void Clear() {  Node<T>\* node;  Node<T>\* node2;  node = head;  while (node != nullptr)  {  node2 = node;  node = node->next;  delete node2;  count--;  }  head = nullptr;  }  /// <summary>  /// Метод для зміни елемента за індексом  /// </summary>  /// <param name="index"> Індекс </param>  /// <returns> Повертає елемент </returns>  T& operator[](int index)  {  if ((index < 0) || (index > count - 1))  {  throw out\_of\_range("Неправильний індекс");  }  Node<T>\* t = Move(index);  return t->data;  }  /// <summary>  /// Знаходження максимальнго елемента  /// </summary>  /// <returns> Повертає максимальний елемент </returns>  T GetMax() {  Node<T>\* node = head;  MAX = node->data;  int i = 0;  while (i != count)  {  if (node->data > MAX)  {  MAX = node->data;  }  node = node->next;  i++;  }  return MAX;  }  /// <summary>  /// Знаходження мінімального елемента  /// </summary>  /// <returns> Повертає мінімальний елемент </returns>  T GetMin() {  Node<T>\* node = head;  MIN = node->data;  int i = 0;  while (i != count)  {  if (node->data < MIN)  {  MIN = node->data;  }  node = node->next;  i++;  }  return MIN;  }  /// <summary>  /// Метод для пошуку першого парного або непарного елементу  /// </summary>  /// <param name="check">  /// true - пошук першого парного  /// false - пошук першого непарного  /// </param>  /// <returns> Повертає результат пошуку </returns>  int EvenOrOddCheck(bool check) {  Node<T>\* ptr = head;  T result;  if (check == true)  {  int i = 0;  while (i != count)  {  if (ptr->data % 2 == 0)  {  result = ptr->data;  break;  }  else {  ptr = ptr->next;  i++;  continue;  }  }  }  else  {  int i = 0;  while (i != count)  {  if (ptr->data % 2 != 0)  {  result = ptr->data;  break;  }  else {  ptr = ptr->next;  i++;  continue;  }  }  }  return result;  }  /// <summary>  /// Метод для пошуку першого додатнього або від'ємного елементу  /// </summary>  /// <param name="check">  /// true - пошук першого додатнього  /// false - пошук першого від'ємного  /// </param>  /// <returns> Повертає результат пошуку </returns>  int PositiveCheck(bool check) {  Node<T>\* ptr = head;  T result;  if (check == true)  {  int i = 0;  while (i != count)  {  if (ptr->data > 0)  {  result = ptr->data;  break;  }  else {  ptr = ptr->next;  i++;  continue;  }  }  }  else  {  int i = 0;  while (i != count)  {  if (ptr->data < 0)  {  result = ptr->data;  break;  }  else {  ptr = ptr->next;  i++;  continue;  }  }  }  return result;  }  /// <summary>  /// Метод для пошуку всіх елементів, рівних заданому  /// </summary>  /// <param name="element"> Заданий елемент </param>  void Search(int element) {  Node<T>\* ptr = head;  int i = 0;  while (i != count)  {  if (ptr->data == element)  {  cout << ptr->data << "(" << i << " index)\n";  }  ptr = ptr->next;  i++;  }  cout << endl;  }  /// <summary>  /// Запис списка в файл  /// </summary>  void WriteToTheFile() {  writeFile.open(filePath);  if (!writeFile.is\_open())  {  cout << "Помилка відкриття файлу" << endl;  }  else  {  Node<T>\* ptr = head;  int i = 0;  while (i != count)  {  writeFile << ptr->data << " ";  ptr = ptr->next;  i++;  }  }  writeFile.close();  }  /// <summary>  /// Метод для закриття файлу  /// </summary>  void Close() {  readFile.close();  }  };  /// <summary>  /// Функція, що викликає меню  /// </summary>  /// <returns> Повертає значення, введене користувачем </returns>  int Menu() {  int check;  cout << "Завантажити стек [0]\n";  cout << "Відобразити стек [1]\n";  cout << "Замінити елемент стека [2]\n";  cout << "Додати елемент [3]\n";  cout << "Видалити елемент [4]\n";  cout << "Знайти елемент за критерієм [5]\n";  cout << "Видалити список [6]\n";  cout << "Закінчити виконання програми [7]\n";  cout << "\nОберіть команду: ";  cin >> check;  if (check == 5)  {  system("cls");  cout << "Критерій\n";  cout << " Найбільший [1]\n";  cout << " Найменший [2]\n";  cout << " Перший парний [3]\n";  cout << " Перший непарний [4]\n";  cout << " Перший додатній [5]\n";  cout << " Перший від'ємний [6]\n";  cout << " Всі рівні заданому [7]\n";  cout << "\nОберіть команду: ";  cin >> check;  check += 50;  }  return check;  }  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  Stack<int> St;  int choice;  int index, element;  for (; ; )  {  choice = Menu();  switch (choice)  {  case 0:  system("cls");  St.AddFromTheFile();  break;  case 1:  system("cls");  St.Print();  break;  case 2:  system("cls");  cout << "Введіть індекс і елемент: ";  cin >> index >> element;  St[index] = element;  St.Print();  St.WriteToTheFile();  break;  case 3:  system("cls");  cout << "Введіть елемент: ";  cin >> element;  St.Push(element);  St.Print();  St.WriteToTheFile();  break;  case 4:  system("cls");  St.Pop();  St.Print();  St.WriteToTheFile();  break;  case 51:  system("cls");  cout << "Найбільший елемент = " << St.GetMax() << endl;  St.Print();  break;  case 52:  system("cls");  cout << "Найменший елемент = " << St.GetMin() << endl;  St.Print();  break;  case 53:  system("cls");  cout << "Перший парний елемент = " << St.EvenOrOddCheck(true) << endl;  St.Print();  break;  case 54:  system("cls");  cout << "Перший непарний елемент = " << St.EvenOrOddCheck(false) << endl;  St.Print();  break;  case 55:  system("cls");  cout << "Перший додатній елемент = " << St.PositiveCheck(true) << endl;  St.Print();  break;  case 56:  system("cls");  cout << "Перший від'ємний елемент = " << St.PositiveCheck(false) << endl;  St.Print();  break;  case 57:  system("cls");  cout << "Введіть елемент: ";  cin >> element;  cout << "\nЕлементи, рівні заданому:\n";  St.Search(element);  St.Print();  break;  case 6:  system("cls");    St.Clear();  St.WriteToTheFile();    break;  case 7:  system("cls");  St.Close();  system("pause");  break;  }  }  } |

1. **Висновки**

В цій лабораторній роботі я вивчив такі структури даних, як масив, список, черга, стек, навчився обробляти їх та працювати з ними. Також я навчився працювати з файлами і файловою системою.

Для цієї лабораторної роботи я створив 4 консольні програми для опрацювання масиву, списку, черги та стеку.