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

2. **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.

3. 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, що містить дані або програми і розміщується на поіменованій ділянці носія даних, сутність, елемент, що дозволяє отримати доступ до певного ресурсу обчислювальної системи і має такі ознаки:

- 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.

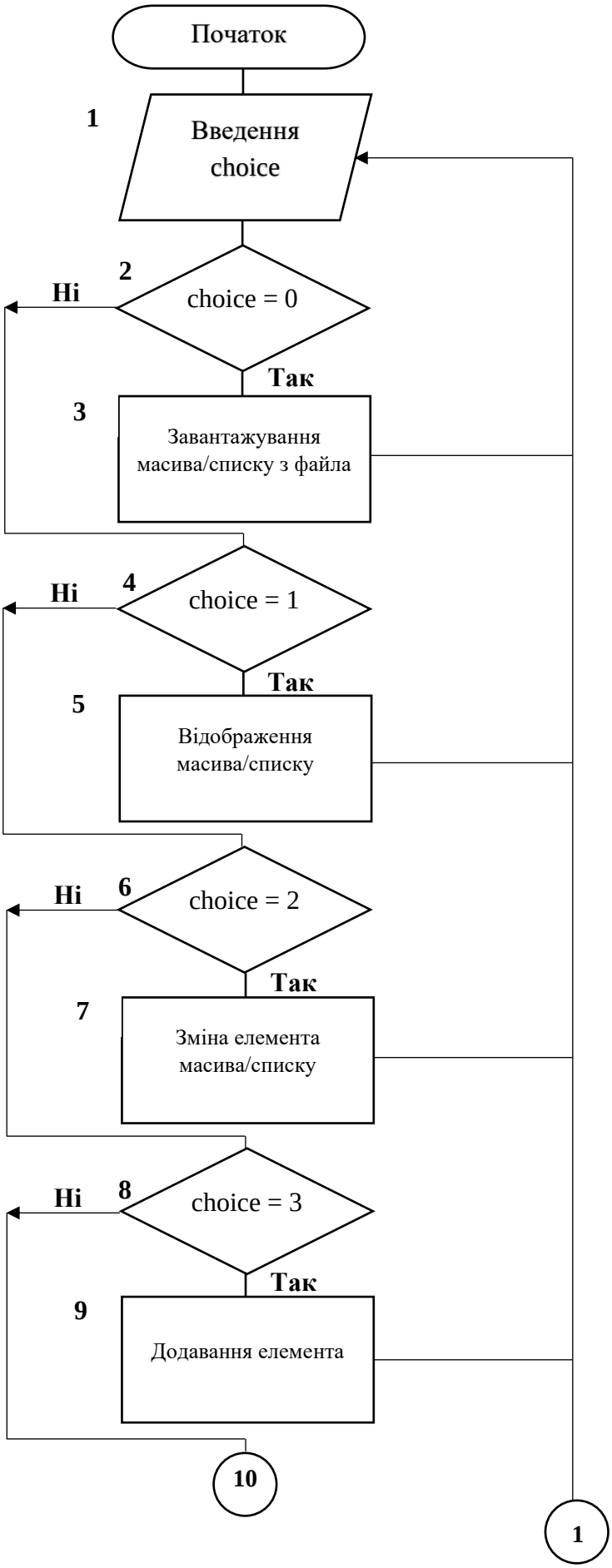
4. Основні відомості про програмні засоби роботи з потоками введення-виведення в мовах програмування

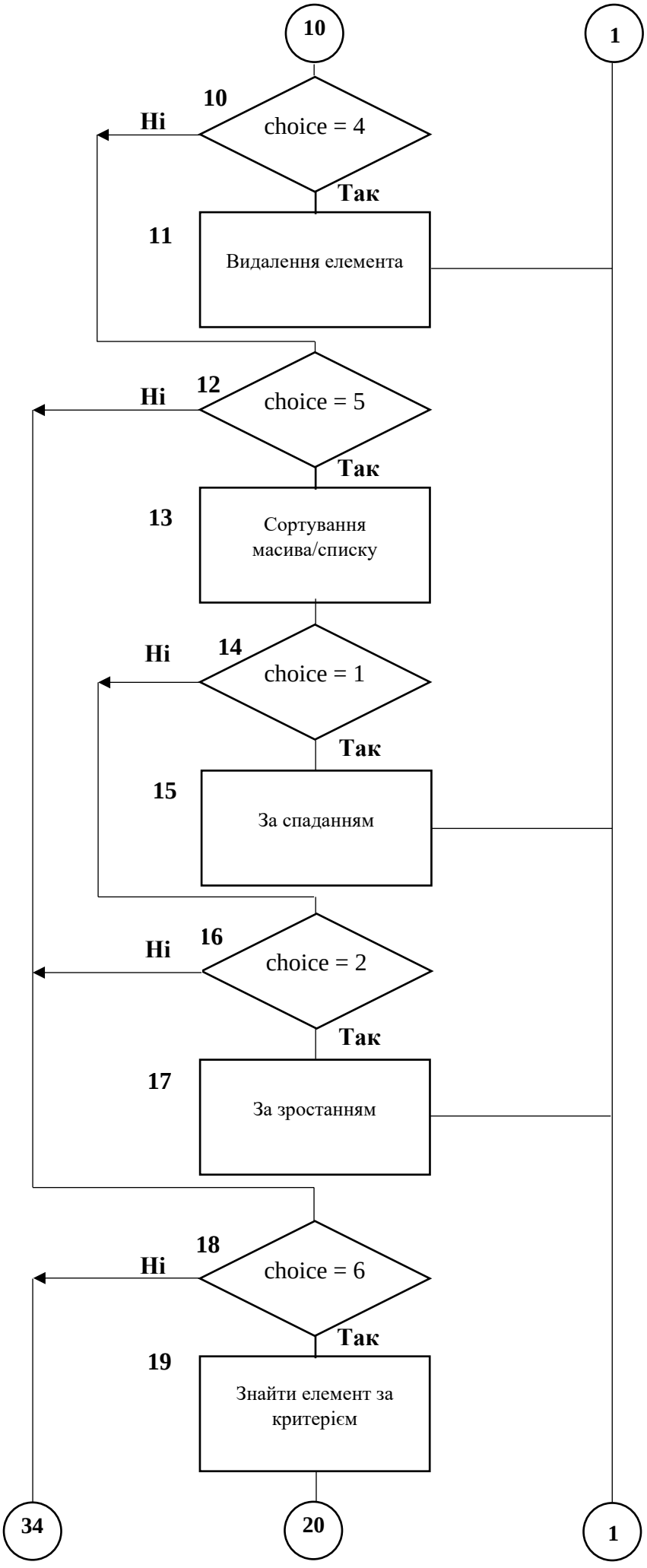
У мові C++ дії, що пов’язані з операціями введення і виведення, виконуються за допомогою функцій бібліотек. Функції ведення і виведення бібліотек мови дозволяють читати дані з файлів та пристроїв і писати дані у файли і на пристрої.
Бібліотека мови C++ підтримує три рівня введення-виведення даних:

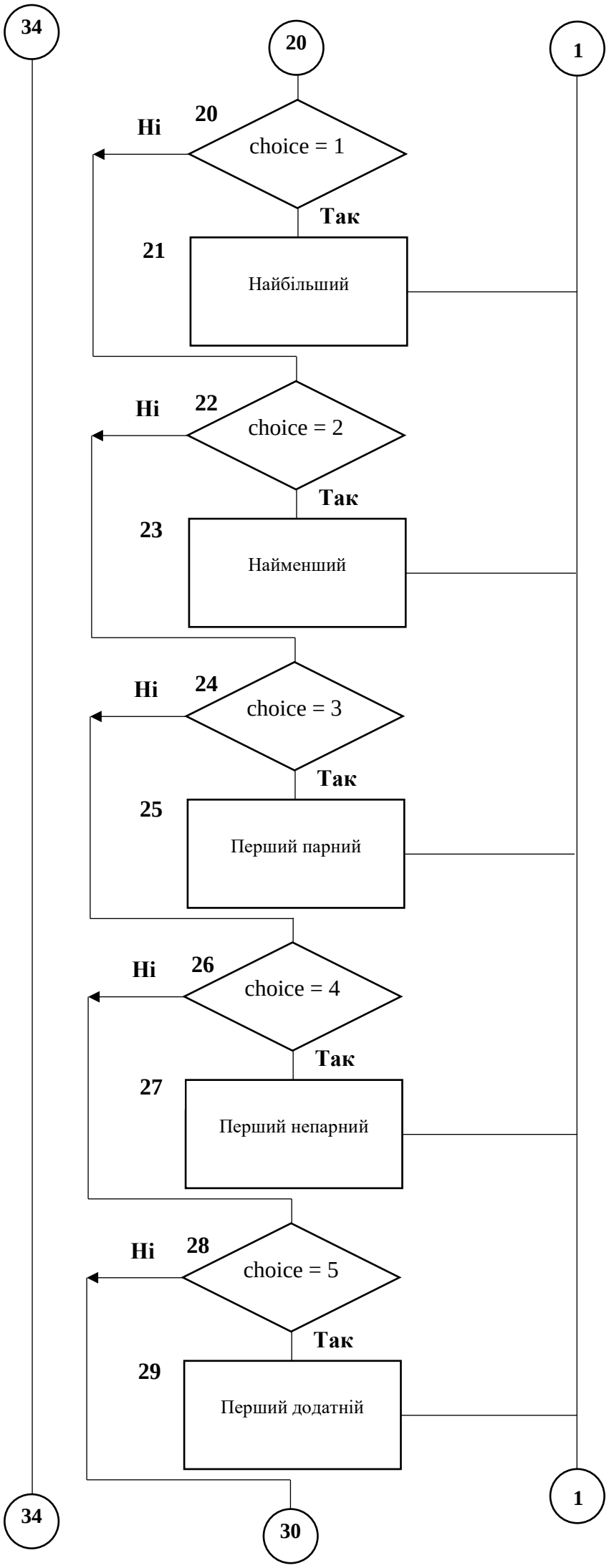
- введення-виведення потоку;
- введення-виведення нижнього рівня;
- введення-виведення для консолі і порту.

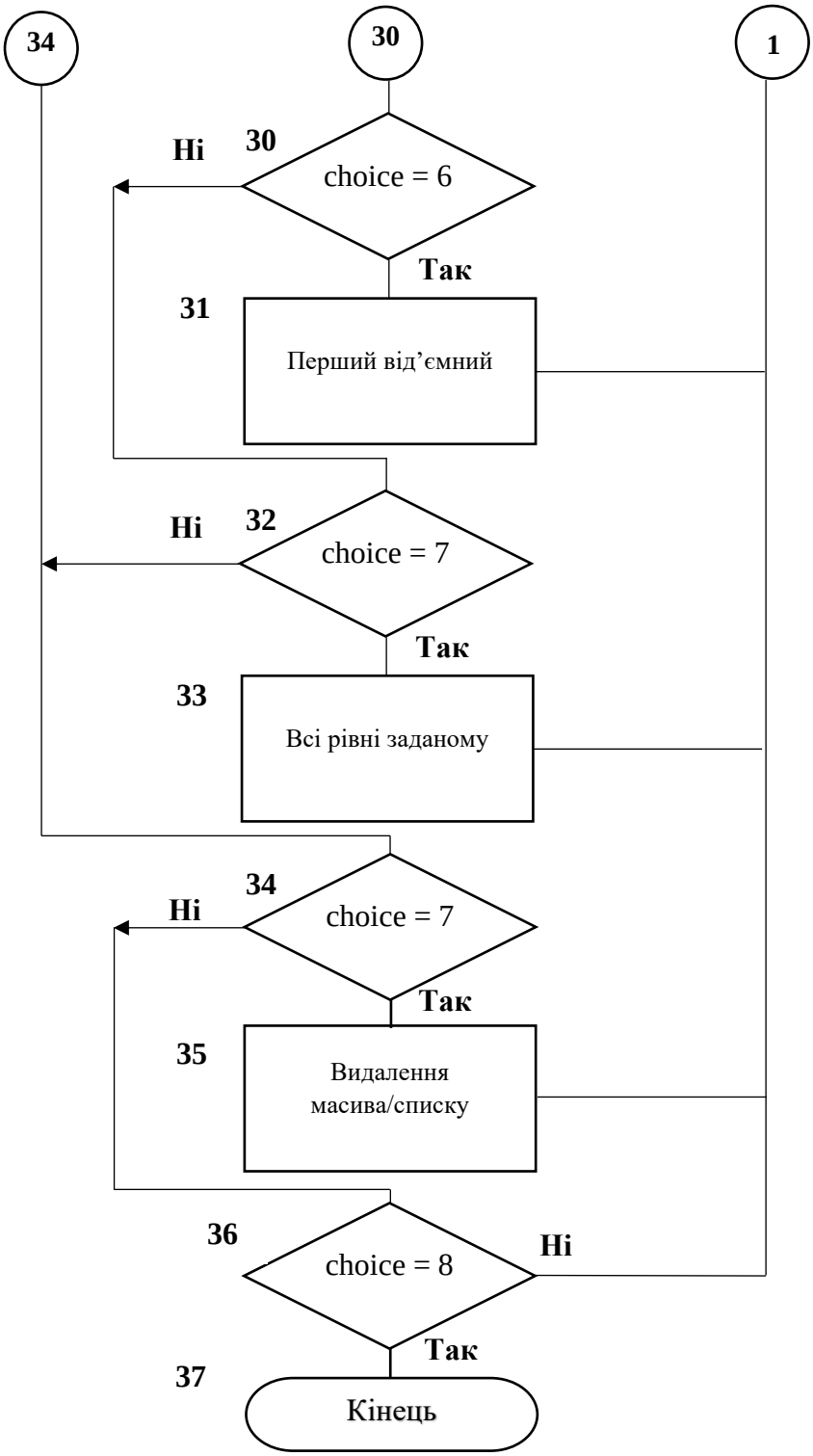
5. Блок-схеми алгоритмів роботи функцій і програм

Масив / Список

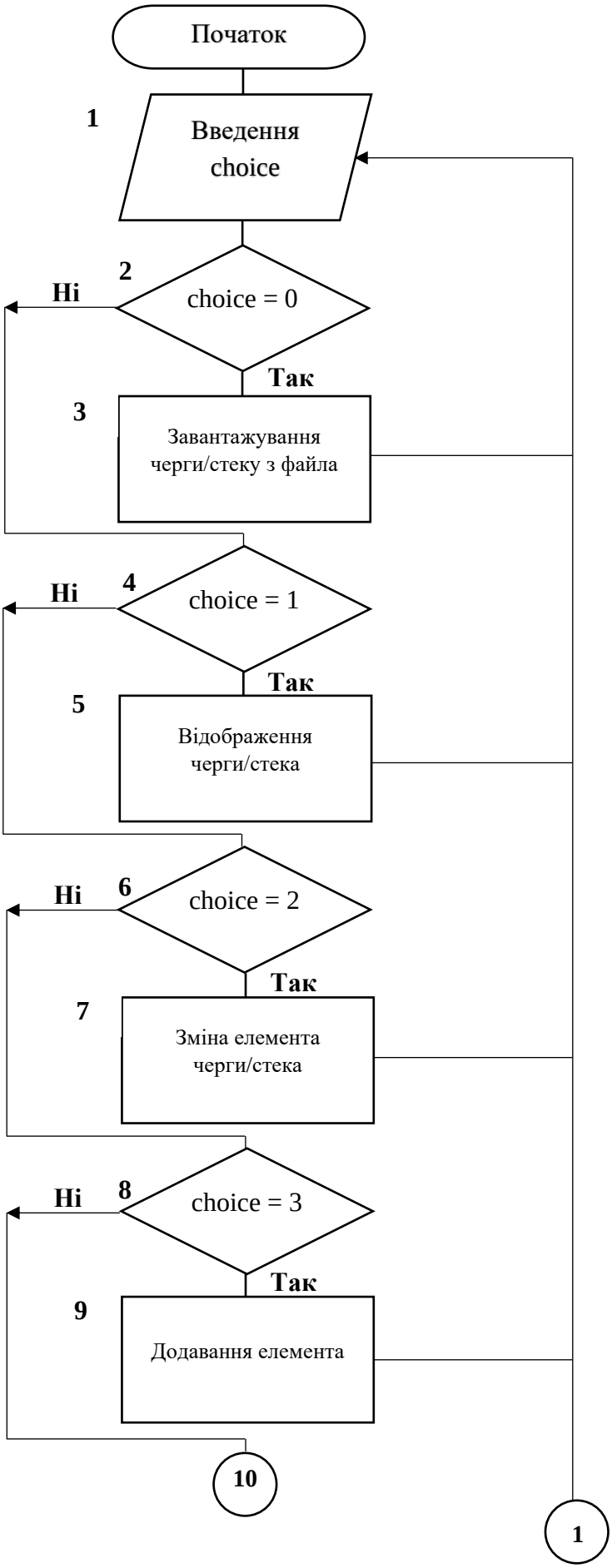

Початок
1
Введення
choice
2
choice = 0
Ні
Так
3
Завантажування
масива/списку з файла
4
choice = 1
Ні
Так
5
Відображення
масива/списку
6
choice = 2
Ні
Так
7
Зміна елемента
масива/списку
8
choice = 3
Ні
Так
9
Додавання елемента
10
1

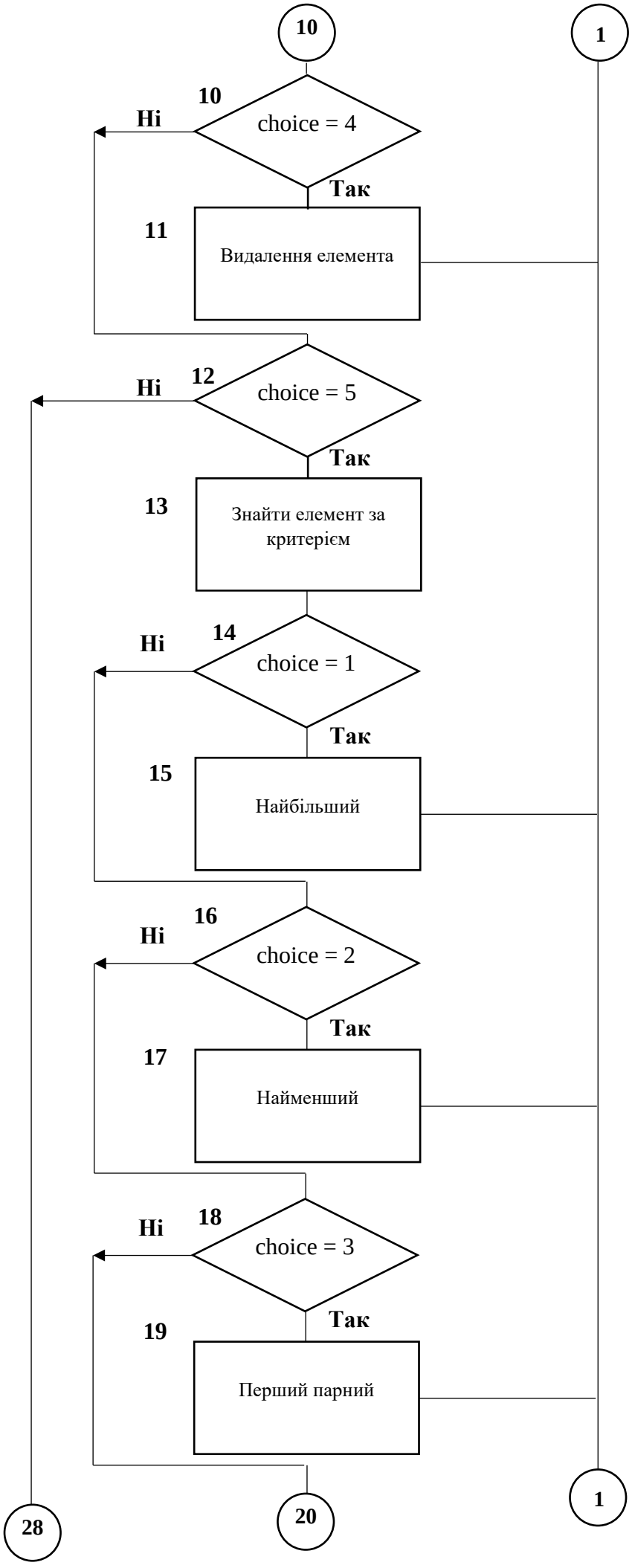

10
1
10
Ні
choice = 4
Так
11
Видалення елемента
12
Ні
choice = 5
Так
13
Сортування масива/списку
14
Ні
choice = 1
Так
15
За спаданням
16
Ні
choice = 2
Так
17
За зростанням
18
Ні
choice = 6
Так
19
Знайти елемент за критерієм
34
20
1

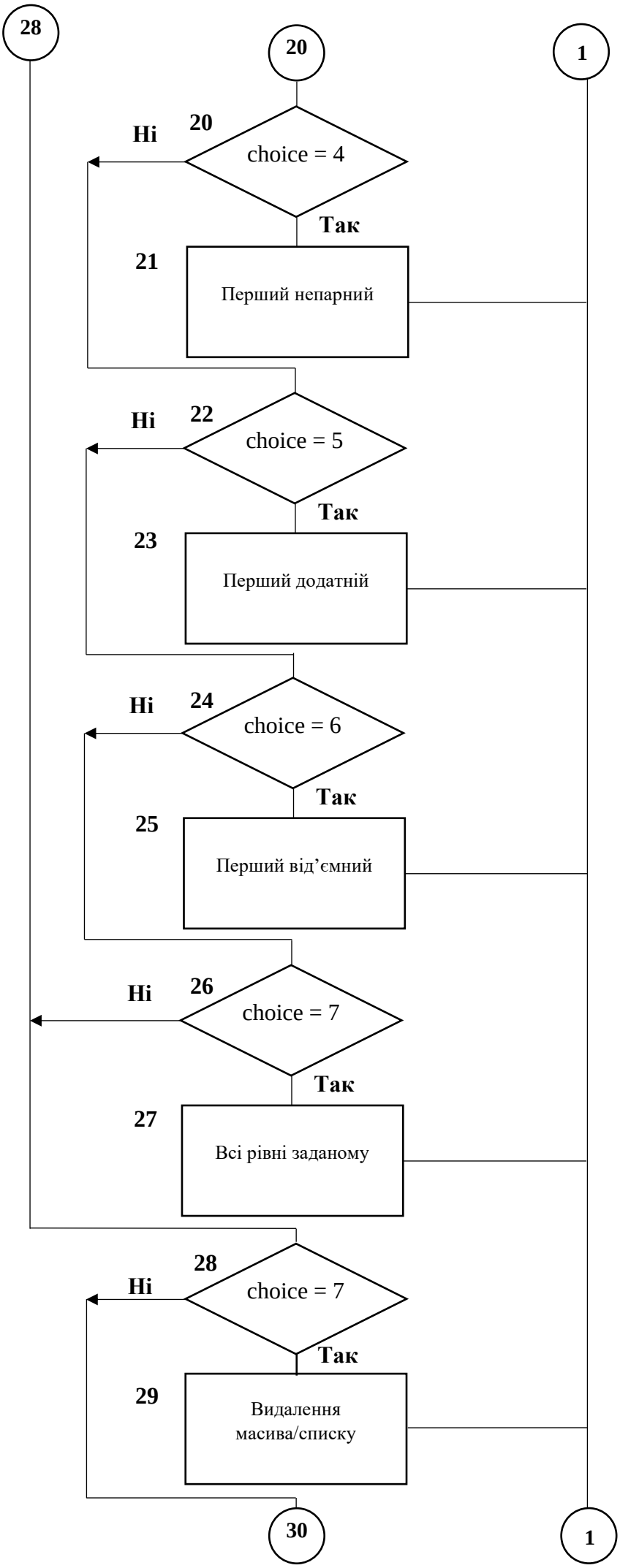

34
20
1
20
Ні
choice = 1
Так
21
Найбільший
Ні
22
choice = 2
Так
23
Найменший
Ні
24
choice = 3
Так
25
Перший парний
Ні
26
choice = 4
Так
27
Перший непарний
Ні
28
choice = 5
Так
29
Перший додатній
1
34
30

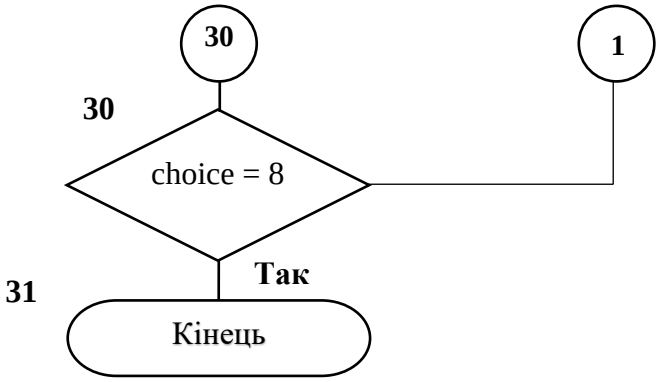

34
30
1
Ні
30
choice = 6
Так
31
Перший від’ємний
Ні
32
choice = 7
Так
33
Всі рівні заданому
34
Ні
choice = 7
Так
35
Видалення масива/списку
36
choice = 8
Ні
Так
37
Кінець

Черга / Стек


Початок
1
Введення
choice
2
choice = 0
Ні
Так
3
Завантажування
черги/стеку з файла
4
choice = 1
Ні
Так
5
Відображення
черги/стека
6
choice = 2
Ні
Так
7
Зміна елемента
черги/стека
8
choice = 3
Ні
Так
9
Додавання елемента
10
1


10
1
10
Ні
choice = 4
Так
11
Видалення елемента
12
Ні
choice = 5
Так
13
Знайти елемент за критерієм
14
Ні
choice = 1
Так
15
Найбільший
16
Ні
choice = 2
Так
17
Найменший
18
Ні
choice = 3
Так
19
Перший парний
1
28
20


28
20
1
Ні
20
choice = 4
Так
21
Перший непарний
Ні
22
choice = 5
Так
23
Перший додатній
Ні
24
choice = 6
Так
25
Перший від’ємний
Ні
26
choice = 7
Так
27
Всі рівні заданому
Ні
28
choice = 7
Так
29
Видалення масива/списку
30
1



6. Вміст файлів вихідних даних

Масив

array – Блокнот

Файл Правка Формат Вид Справка

```
4 0 5 -3 8 3 0 10 -2 -7 3 2 7 -1 1 9 7
```

Список

list – Блокнот

Файл Правка Формат Вид Справка

```
4 2 0 -2 6 7 3 0 2 -1 4 9
```

Черга

queue – Блокнот

Файл Правка Формат Вид Справка

```
4 2 0 -2 6 7 3 0 2 -1 4 9
```

Стек

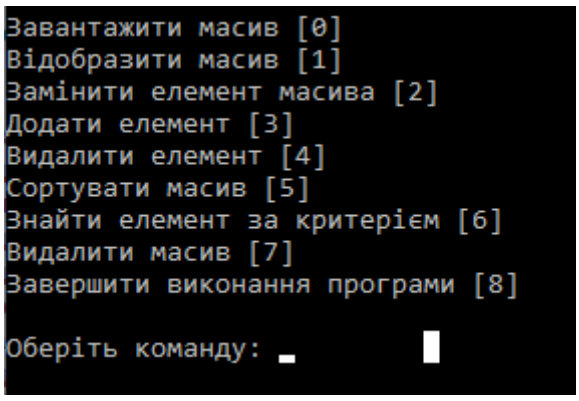
stack – Блокнот

Файл Правка Формат Вид Справка

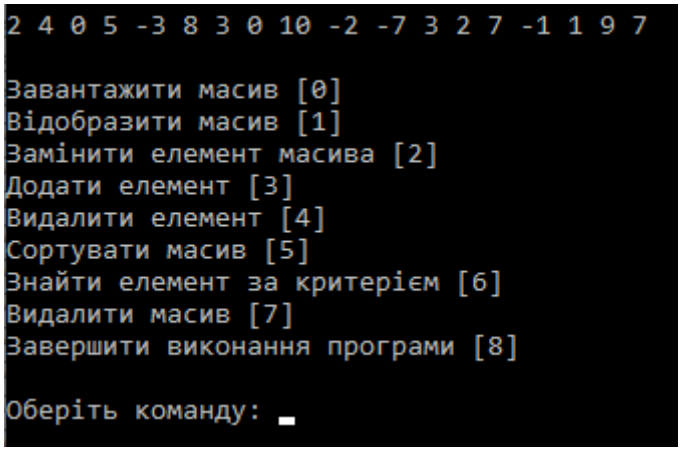
```
4 2 0 -2 6 7 3 0 2 -1 4 9
```

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

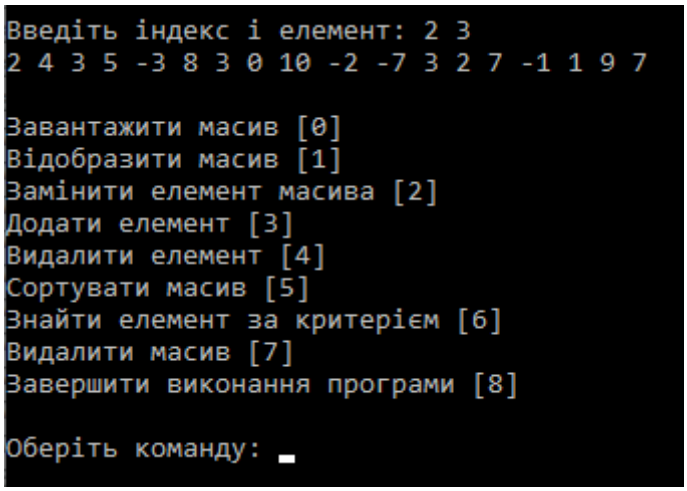
Масив



Завантаження з файлу і відображення масива



Заміна елемента



Додавання елемента

```
Введіть індекс і елемент: 1 0
2 0 4 3 5 -3 8 3 0 10 -2 -7 3 2 7 -1 1 9 7

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду: _
```

Видалення елемента

```
Введіть індекс: 3
2 0 4 5 -3 8 3 0 10 -2 -7 3 2 7 -1 1 9 7

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду: _
```

Сортування масива

```
Сортування
   За спаданням [1]
   За зростанням [2]

Оберіть команду: _
```

За спаданням

```
10 9 8 7 7 5 4 3 3 2 2 1 0 0 -1 -2 -3 -7

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду: _
```

За зростанням

```
-7 -3 -2 -1 0 0 1 2 2 3 3 4 5 7 7 8 9 10

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду:
```

Знаходження елемента за критерієм

```
Критерій
   Найбільший [1]
   Найменший [2]
   Перший парний [3]
   Перший непарний [4]
   Перший додатній [5]
   Перший від'ємний [6]
   Всі рівні заданому [7]

Оберіть команду:
```

```
Найбільший елемент = 10
-7 -3 -2 -1 0 0 1 2 2 3 3 4 5 7 7 8 9 10

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду:
```

```
Найменший елемент = -7
-7 -3 -2 -1 0 0 1 2 2 3 3 4 5 7 7 8 9 10

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду:
```

```
Перший парний елемент = -2
-7 -3 -2 -1 0 0 1 2 2 3 3 4 5 7 7 8 9 10

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду:
```

```
Перший непарний елемент = -7
-7 -3 -2 -1 0 0 1 2 2 3 3 4 5 7 7 8 9 10

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду:
```

```
Перший додатній елемент = 1
-7 -3 -2 -1 0 0 1 2 2 3 3 4 5 7 7 8 9 10

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду:
```

```
Перший від'ємний елемент = -7
-7 -3 -2 -1 0 0 1 2 2 3 3 4 5 7 7 8 9 10

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду:
```

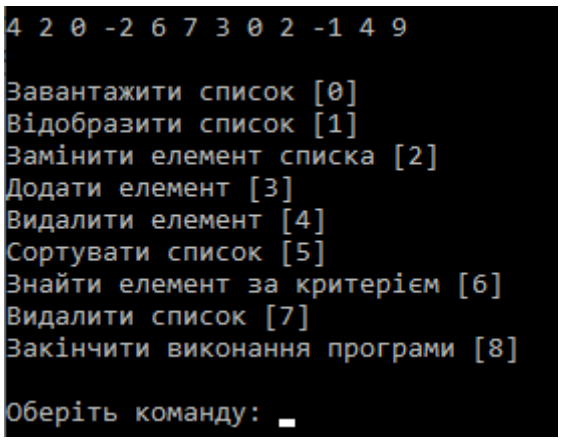
```
Введіть елемент: 3

Елементи, рівні заданому:
3(9 index)
3(10 index)

-7 -3 -2 -1 0 0 1 2 2 3 3 4 5 7 7 8 9 10

Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду:
```

Видалення масива

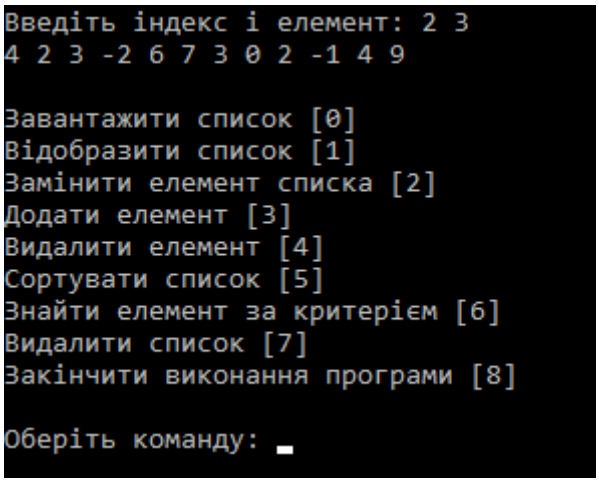
```
Масив видалено
Завантажити масив [0]
Відобразити масив [1]
Замінити елемент масива [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати масив [5]
Знайти елемент за критерієм [6]
Видалити масив [7]
Завершити виконання програми [8]

Оберіть команду:
```

Список

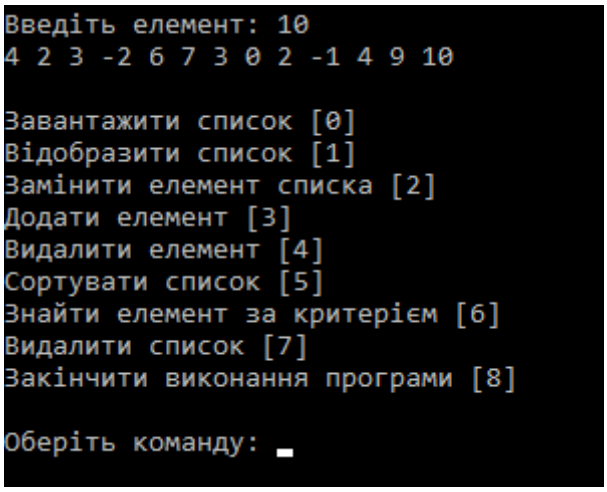
Завантаження з файлу та відображення списку



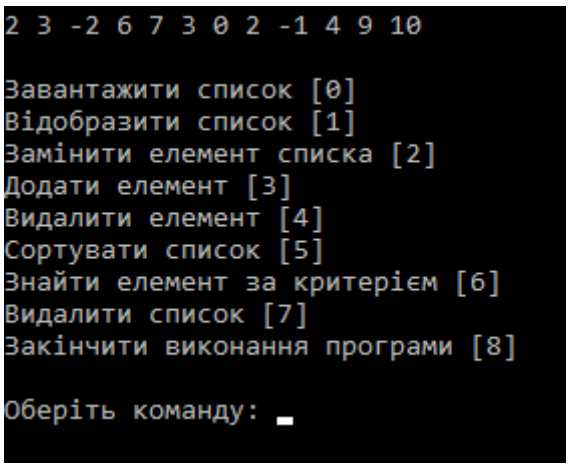
Заміна елемента



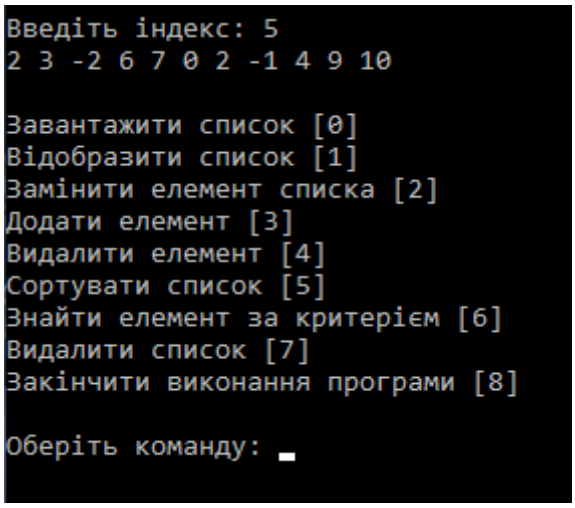
Додавання елемента



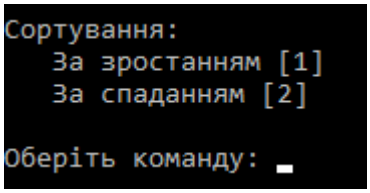
Видалення першого елемента



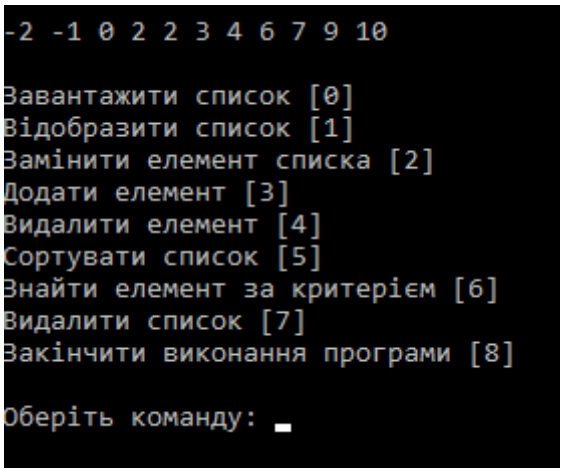
Видалення елемента за індексом



Сортування списку



За зростанням



За спаданням

```
10 9 7 6 4 3 2 2 0 -1 -2

Завантажити список [0]
Відобразити список [1]
Замінити елемент списка [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати список [5]
Знайти елемент за критерієм [6]
Видалити список [7]
Закінчити виконання програми [8]

Оберіть команду:
```

Знаходження елемента за критерієм

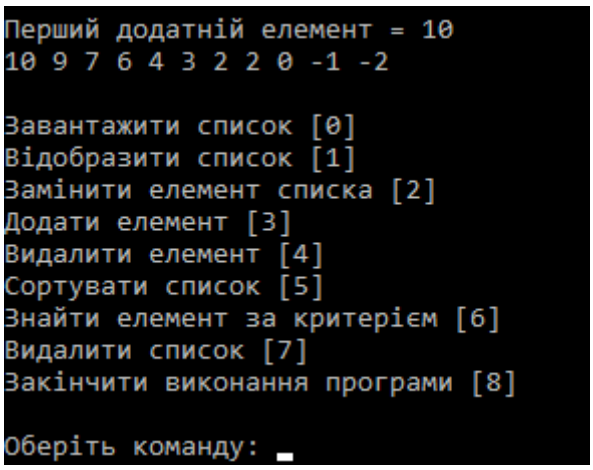
```
Критерій
   Найбільший [1]
   Найменший [2]
   Перший парний [3]
   Перший непарний [4]
   Перший додатній [5]
   Перший від'ємний [6]
   Всі рівні заданому [7]

Оберіть команду:
```

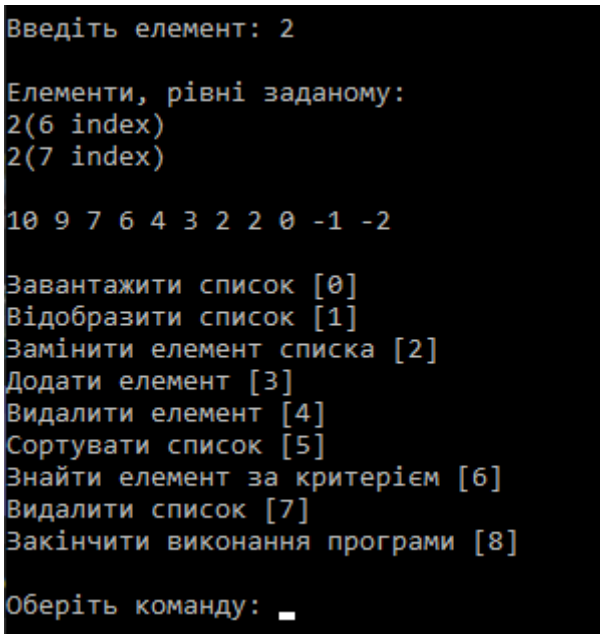
```
Найбільший елемент = 10
10 9 7 6 4 3 2 2 0 -1 -2

Завантажити список [0]
Відобразити список [1]
Замінити елемент списка [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати список [5]
Знайти елемент за критерієм [6]
Видалити список [7]
Закінчити виконання програми [8]

Оберіть команду:
```

```
Найменший елемент = -2
10 9 7 6 4 3 2 2 0 -1 -2

Завантажити список [0]
Відобразити список [1]
Замінити елемент списка [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати список [5]
Знайти елемент за критерієм [6]
Видалити список [7]
Закінчити виконання програми [8]

Оберіть команду:
```

```
Перший парний елемент = 10
10 9 7 6 4 3 2 2 0 -1 -2

Завантажити список [0]
Відобразити список [1]
Замінити елемент списка [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати список [5]
Знайти елемент за критерієм [6]
Видалити список [7]
Закінчити виконання програми [8]

Оберіть команду:
```

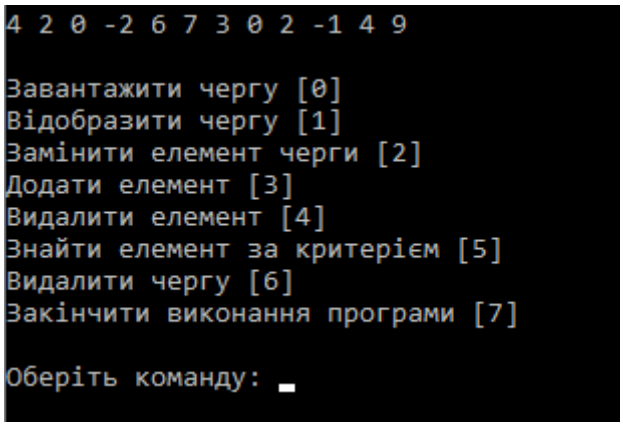
```
Перший непарний елемент = 9
10 9 7 6 4 3 2 2 0 -1 -2

Завантажити список [0]
Відобразити список [1]
Замінити елемент списка [2]
Додати елемент [3]
Видалити елемент [4]
Сортувати список [5]
Знайти елемент за критерієм [6]
Видалити список [7]
Закінчити виконання програми [8]

Оберіть команду:
```





Черга

Завантаження та відображення черги



Заміна елемента черги

```
Введіть індекс і елемент: 0 0
0 2 0 -2 6 7 3 0 2 -1 4 9

Завантажити чергу [0]
Відобразити чергу [1]
Замінити елемент черги [2]
Додати елемент [3]
Видалити елемент [4]
Знайти елемент за критерієм [5]
Видалити чергу [6]
Закінчити виконання програми [7]

Оберіть команду:
```

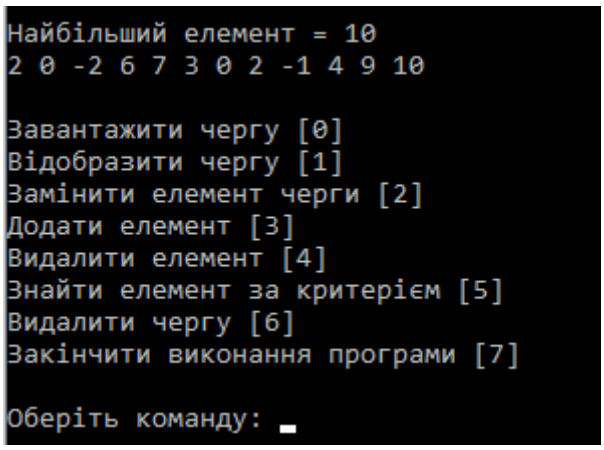
Додавання елемента

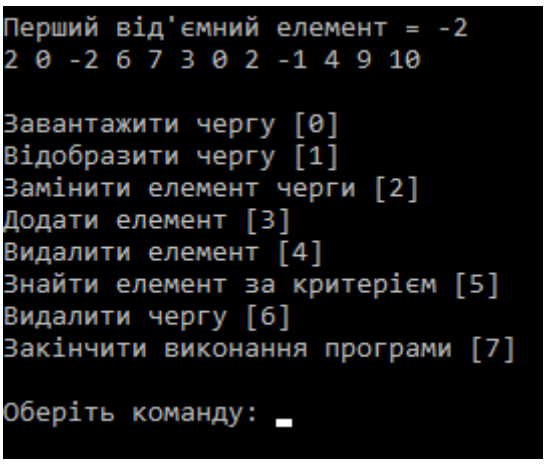
```
Введіть елемент: 10
0 2 0 -2 6 7 3 0 2 -1 4 9 10

Завантажити чергу [0]
Відобразити чергу [1]
Замінити елемент черги [2]
Додати елемент [3]
Видалити елемент [4]
Знайти елемент за критерієм [5]
Видалити чергу [6]
Закінчити виконання програми [7]

Оберіть команду:
```

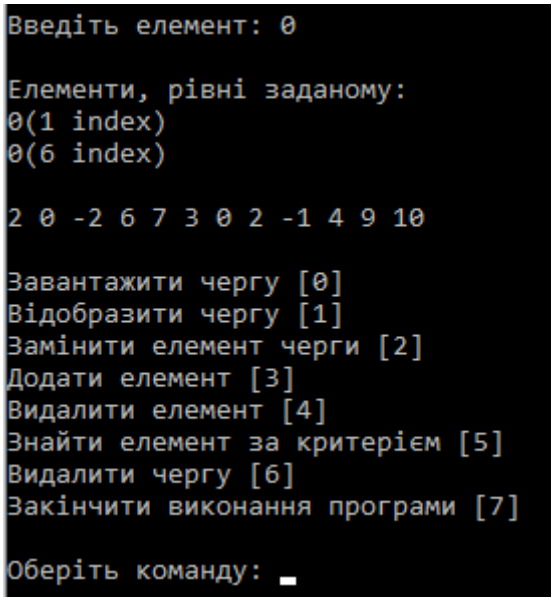
Видалення елемента

```
2 0 -2 6 7 3 0 2 -1 4 9 10

Завантажити чергу [0]
Відобразити чергу [1]
Замінити елемент черги [2]
Додати елемент [3]
Видалити елемент [4]
Знайти елемент за критерієм [5]
Видалити чергу [6]
Закінчити виконання програми [7]

Оберіть команду:
```

Знаходження елемента за критерієм

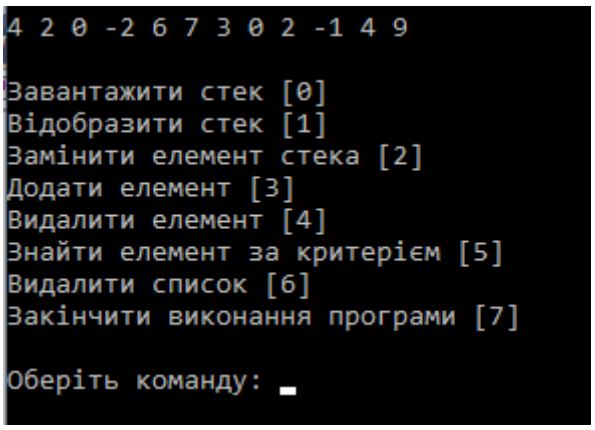




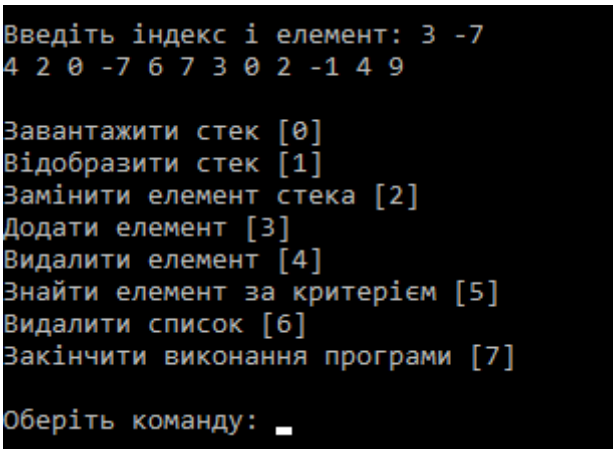


Стек

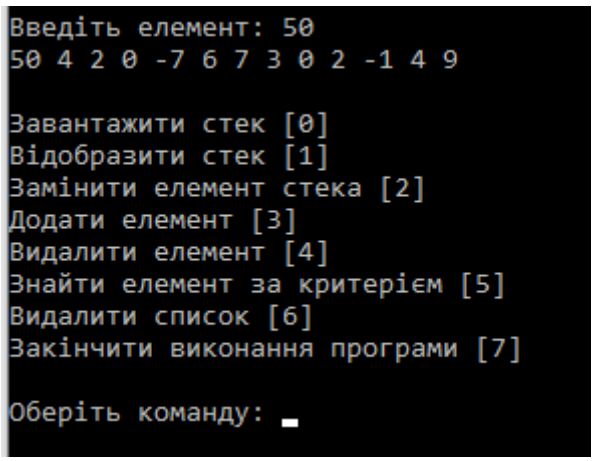
Завантаження та відображення стека



Заміна елемента стека



Додавання елемента



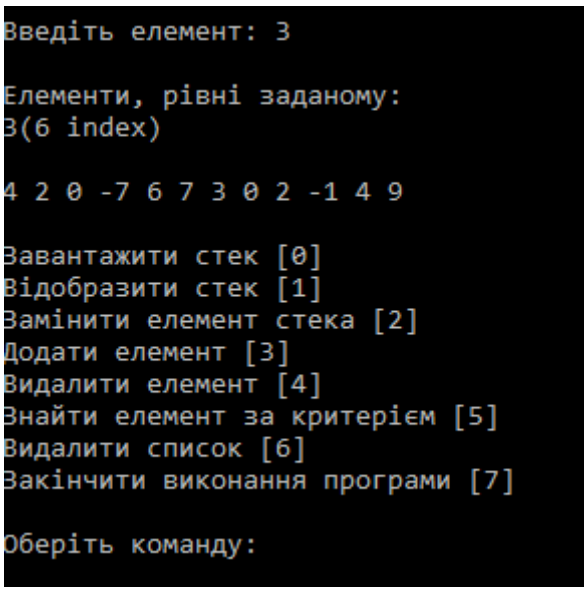
Видалення елемента

```
4 2 0 -7 6 7 3 0 2 -1 4 9

Завантажити стек [0]
Відобразити стек [1]
Замінити елемент стека [2]
Додати елемент [3]
Видалити елемент [4]
Знайти елемент за критерієм [5]
Видалити список [6]
Закінчити виконання програми [7]

Оберіть команду:
```

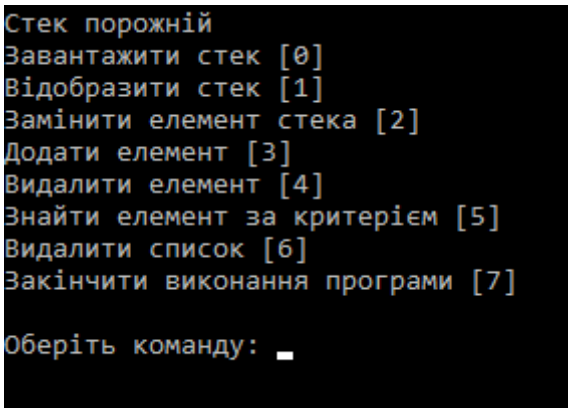
Знаходження елемента за критерієм

```
Найменший елемент = -7
4 2 0 -7 6 7 3 0 2 -1 4 9

Завантажити стек [0]
Відобразити стек [1]
Замінити елемент стека [2]
Додати елемент [3]
Видалити елемент [4]
Знайти елемент за критерієм [5]
Видалити список [6]
Закінчити виконання програми [7]

Оберіть команду:
```

```
Перший непарний елемент = -7
4 2 0 -7 6 7 3 0 2 -1 4 9

Завантажити стек [0]
Відобразити стек [1]
Замінити елемент стека [2]
Додати елемент [3]
Видалити елемент [4]
Знайти елемент за критерієм [5]
Видалити список [6]
Закінчити виконання програми [7]

Оберіть команду:
```



Видалення стека



8. 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;
        }
    }
}
```

9. 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.