МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
Кафедра систем управління літальних апаратів

Лабораторна робота № 8

з дисципліни «Алгоритмізація та програмування»
на тему «Реалізація алгоритмів сортування та робота з файлами на мові С ++»

ХАІ.301.174.312.8

Виконав студент гр. 312

_____ Діхтяренко М.Г.
(підпис, дата) (П.І.Б.)

Перевірив
_____ к.т.н., доц. Олена ГАВРИЛЕНКО
(підпис, дата) (П.І.Б.)

2024

МЕТА РОБОТИ

Вивчити теоретичний матеріал по алгоритмам обробки масивів на мові С+ +, а також бібліотеки для роботи з файлами і реалізувати оголошення, введення з файлу, обробку і виведення в файл одновимірних і двовимірних масивів на мові С ++ в середовищі Visual Studio.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. За допомогою текстового редактору створити текстовий файл «array_in_n.txt» з елементами вихідного масиву (n - 91). У програмі на С++ перетворити масив відповідно до свого варіанту завдання, ім'я файлу і необхідні змінні ввести з консолі. Вивести результати у файл «array_out_n.txt».

Завдання 2. За допомогою текстового редактору створити текстовий файл «matr_in_n.txt» з елементами вихідного двовимірного масиву (n - номер варіанта). У програмі обробити матрицю відповідно до свого варіанту завдання (Matrix40), ім'я файлу і необхідні змінні ввести з консолі. Дописати результати в той же файл.

Завдання 3. Вивчити метод сортування відповідно до свого варіанту (21), проаналізувати його складність і продемонструвати на прикладі з 7-ми елементів. Реалізувати у вигляді окремої функції алгоритм сортування елементів масиву. Також окремими функціями реалізувати зчитування масиву з текстового файлу і виведення відсортованого масиву в консоль.

Завдання 4. Для багаторазового виконання будь-якого з трьох зазначених вище завдань на вибір розробити алгоритм організації меню в командному вікні. Введення, виведення, обробку масивів реалізувати окремими функціями з параметрами.

ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі Array#91.

Типи даних:

1. **int**:
    - Використовується для зберігання цілих чисел, таких як розмір масиву та елементи масиву.
2. **string**:

- Використовується для зберігання імен файлів ("array_in_91.txt", "array_out_91.txt").

3. **ifstream**:
   - Використовується для зчитування даних з файлу "array_in_91.txt".
4. **ofstream**:
   - Використовується для запису даних у файл "array_out_91.txt".
5. **bool**:
   - Використовується для перевірки наявності файлів та успішного відкриття потоків для роботи з файлами.
6. **void**:
   - Використовується для визначення функцій, які не повертають значень.
7. **const int**:
   - Використовується для оголошення константи "N".

Алгоритм вирішення:

1. Введення вхідних даних:
   - Зчитати змінні вхідного масиву та параметри K та L з файлу "array_in_91.txt".
2. Перевірка вхідних даних:
   - Перевірити, чи існують файли "array_in_91.txt" та "array_out_91.txt".
   - Перевірити правильність значень K та L: K має бути меншим за L.
3. Створення масиву:
   - Створити масив та заповнити його елементами, зчитаними з файлу "array_in_91.txt".
4. Обробка масиву відповідно до варіанту завдання:
   - Створити новий масив, в який потраплять елементи вихідного масиву, крім тих, що знаходяться між позиціями K і L.
   - Записати цей новий масив у файл "array_out_91.txt".
5. Виведення результатів:
   - Вивести вміст масиву до та після обробки для перевірки правильності результатів.’

Лістинг коду вирішення задачі 1, наведено в дод. А (стор. 9).

Екран роботи програми наведено в дод. Б (стор. 17).

Завдання 2.

Вирішення задачі Matrix40.

Типи даних:

1. **int**:
    - Використовується для зберігання цілих чисел, таких як розміри матриці та елементи матриці.
2. **string**:
    - Використовується для зберігання імен файлів ("matr_in_40.txt").
3. **ifstream**:
    - Використовується для зчитування даних з файлу "matr_in_40.txt".
4. **ofstream**:
    - Використовується для запису даних у файл "matr_in_40.txt".
5. **bool**:
    - Використовується для перевірки наявності файлу та успішного відкриття потоків для роботи з файлами.
6. **float**:
    - Використовується для зберігання дробових чисел у форматі з плаваючою комою, якщо це потрібно для обробки даних.
7. **int***:
    - Використовується для створення двовимірного масиву.

Алгоритм вирішення:

1. Введення вхідних даних:
    - Зчитати змінні рядків та стовпців матриці з файлу "matr_in_40.txt".
2. Перевірка вхідних даних:
    - Перевірити, чи існує файл "matr_in_40.txt".
    - Перевірити правильність значень рядків та стовпців: їхні значення мають бути меншими за обмеження (наприклад, 20).
3. Створення та заповнення матриці:
    - Створити двовимірний масив за допомогою динамічного виділення пам'яті з використанням зчитаних розмірів.
    - Заповнити матрицю елементами, зчитаними з файлу "matr_in_40.txt".
4. Обробка матриці відповідно до варіанту завдання:

- Для кожного рядка матриці:
  - Знайти найбільший та найменший елементи.
  - Обміняти їх місцями.

5. Запис результатів у файл:
   - Записати змінений масив у файл "matr_in_40.txt".
6. Виведення результатів:
   - Вивести вміст матриці до та після обробки для перевірки правильності результатів.
7. Очистка пам'яті:
   - Звільнити пам'ять, виділену для двовимірного масиву.

Лістинг коду вирішення задачі 1, Matrix40 наведено в дод. А (стор. 9).
Екран роботи програми наведено в дод. Б (стор. 17).

Завдання 3.

Вирішення задачі Sort21.

Типи даних:

1. **float**:
   - Використовується для зберігання чисел з плаваючою комою, які будуть сортовані.
2. **string**:
   - Використовується для зберігання імен файлів, таких як "sort_in_21.txt".
3. **ifstream**:
   - Потік для читання даних з файлу "sort_in_21.txt".
4. **ofstream**:
   - Потік для запису даних у файл "sort_in_21.txt".
5. **int**:
   - Використовується для зберігання цілих чисел, таких як розмір масиву та індекси.
6. **const int**:
   - Використовується для оголошення констант, таких як розмір масиву або кількість елементів.
7. **void**:

- Використовується для визначення функцій, які не повертають значень.

8. **bool**:

- Використовується для виконання умовних перевірок, наприклад, перевірки наявності файлу.

Алгоритм вирішення:

Введення вхідних даних:

- Зчитати змінні з файлу "sort_in_21.txt", такі як розмір масиву та числа для сортування.

Перевірка вхідних даних:

- Перевірити, чи існує файл "sort_in_21.txt".

Створення та заповнення масиву:

- Створити масив за допомогою динамічного виділення пам'яті з використанням розміру, зчитаного з файлу.
- Заповнити масив елементами, зчитаними з файлу.

Сортування масиву:

- Використати алгоритм сортування, який вибраний для реалізації (наприклад, сортування вставками).
- Викликати функцію сортування, яка приймає масив, його розмір та ім'я файлу з вхідними даними.

Запис результатів у файл:

- Записати відсортований масив у файл "sort_in_21.txt".

Виведення результатів:

- Вивести вміст масиву після сортування для перевірки правильності результатів.

Очистка пам'яті:

- Звільнити пам'ять, виділену для масиву.

Лістинг коду вирішення задачі 3, наведено в дод. А (стор. 9).

Екран роботи програми наведено в дод. Б (стор. 17).

Завдання 4.

Вирішення задачі

Типи даних:

int:

- Використовується для зберігання цілих чисел, таких як вибір користувача для вибору завдання.

string:

- Використовується для зберігання текстових рядків, наприклад, для збереження імені файлу.

void:

- Використовується для визначення функцій, які не повертають значень.

bool:

- Використовується для виконання умовних перевірок, наприклад, для перевірки правильності введеного вибору користувача.

Алгоритм вирішення:

1. **Введення вхідних даних:**
   - Запитати користувача про вибір завдання для виконання.
   - Користувач вводить номер завдання, яке потрібно виконати.
2. **Перевірка введених даних:**
   - Перевірити правильність введеного користувачем номера завдання.
   - Якщо номер не відповідає жодному із завдань, повідомити користувача про помилку та повторити запит на ввід.
3. **Виклик функцій для вибраного завдання:**
   - В залежності від введеного користувачем номера, викликати відповідну функцію для обробки вибраного завдання.
   - Кожна функція повинна виконувати відповідну дію для обраного завдання: обробка масиву, обробка матриці або сортування.
4. **Повторне виконання або вихід:**
   - Після виконання обраного завдання попросити користувача, чи він бажає повторити вибір завдання або вийти з програми.
   - Якщо користувач вибирає повторний запуск, алгоритм повторюється з кроку 1.
   - Якщо користувач вибирає вихід, програма завершує свою роботу.

Лістинг коду вирішення задачі 3, наведено в дод. А (стор. 9).

Екран роботи програми наведено в дод. Б (стор. 17).

ВИСНОВКИ

В ході виконання лабораторної роботи 8 було вивчено теоретичний матеріал по алгоритмам обробки масивів на мові С++, а також бібліотеки для роботи з файлами і реалізувати оголошення, введення з файлу, обробку і виведення в файл одновимірних і двовимірних масивів на мові С ++ в середовищі Visual Studio, а також відпрацьовані навички написання коду.

ДОДАТОК А
Лістинг коду програми

```
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>

#define N 20

using namespace std;

void array91();
void get_nums(int size, int array[N]);
bool checkfiles(string in, string out);
void editNewSize(int*& arr, int n, string output, int K, int M);


void matrix40();
bool checkfile(string in);
void fillArray(string f_in, int**& arr, const int rows, const int cols);
void searchNumbers(int** arr, const int rows, const int columns, string f_out);

void sort21();
void insertionSort(float arr[N], int n, string in);

int main() {

	int choice = 0;

	while (choice != 4) {

		cout << "Choose the task! : "
			"\n1.Array#91"
			"\n2.Matrix#40"
			"\n3.Sort#21"
			"\n4.Exit" << endl;

		cin >> choice;

		switch (choice) {
		case 1: {
			array91();		//Task array91
			break;
		}
		case 2: {
			matrix40();		//Task matrix40
			break;
		}
		case 3: {
```

```
                sort21();         //Task sort21
                break;
            }
            case 4: {
                cout << "Program is end!";
                break;
            }
            default: {
                cout << "Wrong one, try again";
            }
            }
      }
}

//Task Array91

void array91() {

      int n = 0, startPos = 0, size = 0;
      int mas[N];

      get_nums(size, mas);
}


void get_nums(int size, int array[N]) {

      string filename_in = "array_in_91.txt";
      string filename_out = "array_out_91.txt";

      int startPos = 0;

      ifstream f;

      if (checkfiles(filename_in, filename_out)) {

            f.open(filename_in);

            string lenght;

            getline(f, lenght);
            int size = stoi(lenght);

            int* arr = new int(size);

            getline(f, lenght);

            int K = stoi(lenght);

            getline(f, lenght);
```

```
            int L = stoi(lenght);

            cout << "K = " << K << "\n L = " << L;

            if (K > L) {
                  cout << "K cannot be > than L";
                  exit(0);
            }

            string tempor[N];
            getline(f, tempor[0]);

            for (int i = 0; i < size; i++) {
                  int spacePos = tempor[0].find(' ', startPos);
                  std::string numStr = tempor[0].substr(startPos, spacePos -
startPos);
                  arr[i] = stoi(numStr);
                  startPos = spacePos + 1;
            }

            for (int i = 0; i < size; i++) {
                  cout << "arr[" << i << "] = " << arr[i] << endl;
            }

            f.close();

            editNewSize(arr, size, filename_out, K, L);

      }
      else {
            cout << "File not found";
      }
}

bool checkfiles(string in, string out) {

      ifstream f_in;
      ifstream f_out;

      f_in.open(in);
      f_out.open(out);

      if (!f_in.is_open() || !f_out.is_open()) {
            f_in.close();
            f_out.close();
            return 0;
      }
      else if (f_in.is_open() && f_out.is_open()) {
            return 1;
      }
      else {
```

```
			return 0;
		}
}

void editNewSize(int*& arr, int n, string filename_out, int K, int L) {

		int save = n-(L-K);

		int* temp = new int(save);

		for (int i = 0; i < K; i++) {
			temp[i] = arr[i];
		}
		for (int i = L; i < n; i++,K++) {
			temp[K] = arr[i];
		}


		cout << "New Array\n";

		for (int i = 0; i < save; i++) {
			cout << "Arr[" << i << "] = " << temp[i] << endl;
		}

		ofstream outp(filename_out);
		if (!outp)
		{
			cerr << "Cannot open a file!" << endl;
			exit(1);
		}
		else {
			outp << "New Array: " << endl;
			for (int i = 0; i < save; i++) {
				outp << temp[i] << " ";
			}
		}
}

//Task Matrix40

void matrix40() {
		int n = 0, startPos = 0, rows = 0, cols = 0, skip = 0;
		string filename;

		string filename_in = "matr_in_40.txt";


		ifstream f;

		if (checkfile(filename_in)) {
			f.open(filename_in);
```

```
            string size;

            getline(f, size);

            for (int i = 0; i < size.size(); i++) {
                  if (size[i] == ' ') {
                        skip = i;
                  }
            }
            rows = stoi(size);
            if (rows > 20) {
                  cout << "Rows < 20!" << endl;
                  exit(0);
            }
            else {
                  for (int i = 0; i < size.size() - skip; i++) {
                        size[i] = size[skip + i];
                        size[skip + i] = 0;
                  }

                  cols = stoi(size);
                  if (cols > 20) {
                        cout << "Columns < 20!" << endl;
                        exit(0);
                  }
                  else {
                        cout << "ROWS = " << rows << endl << "COLS = " << cols
<< endl;
                        int** arr = new int* [rows];
                        for (int i = 0; i < rows; i++) {
                              arr[i] = new int[cols];
                        }
                        fillArray(filename_in, arr, rows, cols);
                        searchNumbers(arr, rows, cols, filename_in);
                  }
            }
      }
      else {
            cout << "File do not found";
      }
}

void fillArray(string f_in, int**& arr, const int rows, const int cols) {
      string out;
      float num = 0;
      int numRows = 0, numCols = 0;

      ifstream f;

      istringstream iss(out);
```

```
	f.open(f_in);

	std::getline(f, out);

	while (getline(f, out) && numRows < rows) {
		istringstream iss(out);
		numCols = 0;

		while (iss >> num && numCols < cols) {
			arr[numRows][numCols] = num;
			numCols++;
		}
		numRows++;
	}

	for (int i = 0; i < rows; i++) {
		for (int j = 0; j < cols; j++) {
			cout << arr[i][j] << " ";
		}
		cout << "\n";
	}
}

bool checkfile(string in) {

	ifstream f_in;
	f_in.open(in);

	if (f_in.is_open()) {
		return 1;
	}
	else {
		return 0;
	}
}

void searchNumbers(int** arr, const int rows, const int columns, string f_out) {

	for (int i = 0; i < rows; i++) {
		int MAX = 3E+5;
		int MIN = 0;
		int MAXtemp = 0;
		int MINtemp = 0;
		for (int j = 0; j < columns; j++) {
			if (arr[i][j] < MAX) {
				MAX = arr[i][j];
				MAXtemp = j;
			}
			if (arr[i][j] > MIN) {
				MIN = arr[i][j];
				MINtemp = j;
```

```
                }
            }
            swap(arr[i][MAXtemp], arr[i][MINtemp]);
        }

        ofstream outp;
        outp.open(f_out, ios::app);

        if (!outp)
        {
            cerr << "Can't open a file" << std::endl;
            exit(1);
        }
        else {
            outp << "\nEdited array:";
            for (int i = 0; i < rows; i++) {
                for (int j = 0; j < columns; j++) {
                    outp << arr[i][j] << " ";
                }
                outp << "\n";
            }
        }
}

//Task Sort21

void sort21() {
    const int M = 20;

    float arr[M];
    string filename;

    string filename_in = "sort_in_21.txt";

    ifstream f;
    istringstream iss(filename_in);

    int count = 0;

    if (checkfile(filename_in)) {
        f.open(filename_in);

        while (count < M && f >> arr[count]) {
            count++;
        }

        for (int i = 0; i < count; i++) {
            cout << arr[i] << " ";
        }

        insertionSort(arr, count, filename_in);
```

```
	}
	else {
		cout << "File does not found";
	}
}

void insertionSort(float arr[N], int n, string f_in)
{
		for (int i = 1; i < n; ++i) {
			int key = arr[i];
			int j = i - 1;

			while (j >= 0 && arr[j] > key) {
				arr[j + 1] = arr[j];
				j = j - 1;
			}
			arr[j + 1] = key;
		}

	ofstream outp;
	outp.open(f_in, ios::app);

	if (!outp)
	{
		cerr << "Cannot open a file!" << std::endl;
		exit(1);
	}
	else {
		outp << "\n";
		for (int i = 0; i < n; i++) {
			outp << arr[i] << " ";
		}
	}
}

//End
```

ДОДАТОК Б

Скрін-шот вікна виконання програми

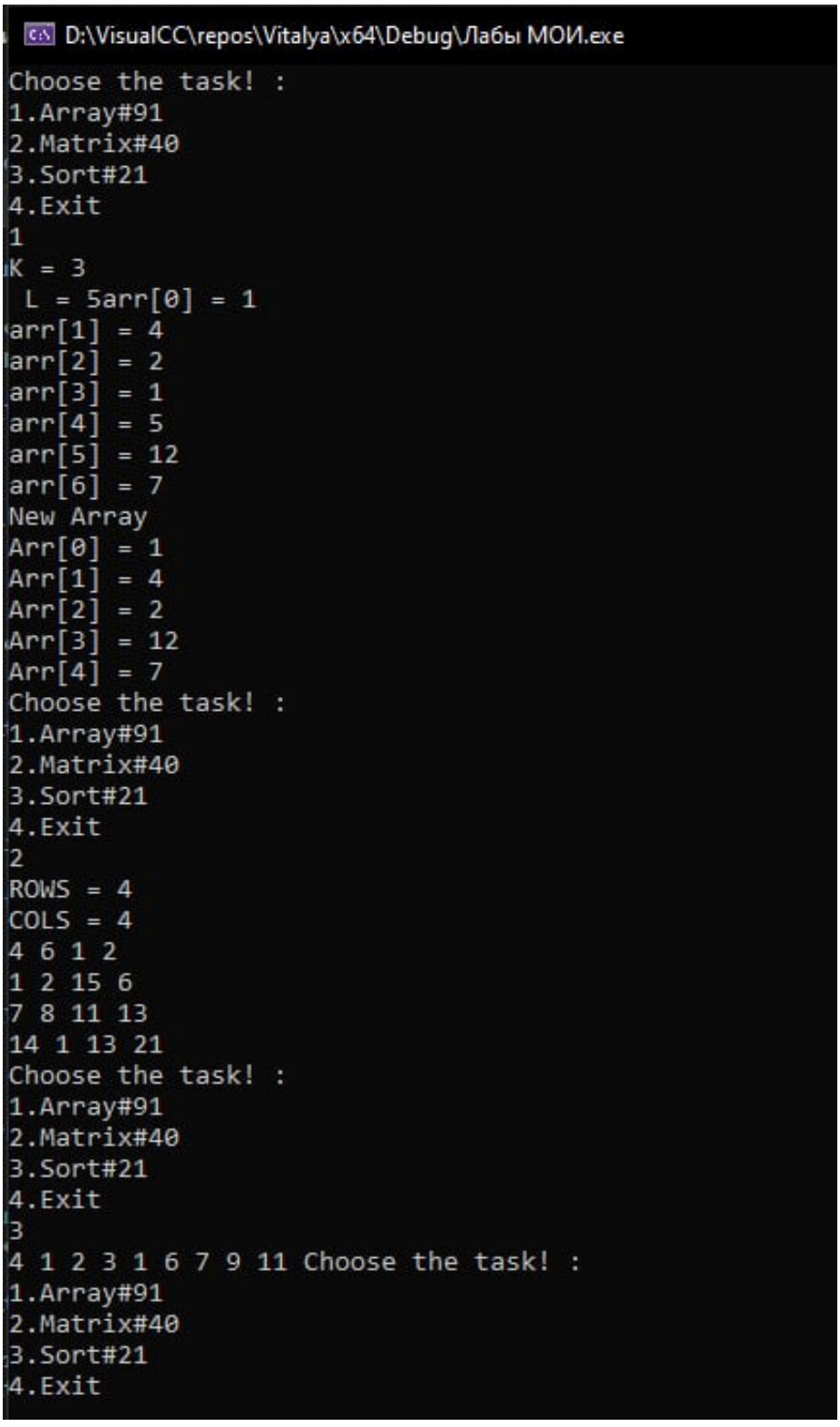

Рисунок Б - Екран виконання програми для вирішення завдання 1,2