Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 5 (лабораторні № 6, 8, 9)**

***з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: «Файли. Системи числення. Бінарні Файли. Символи та Рядкові Змінні. Текстові Файли. Стандартна бібліотека. Створення й використання бібліотек.»

***Виконав:***

студент групи **ШІ-13**

**Шабанов Микита Миколайович**

# **Тема роботи:**

Робота з текстовими та бінарними файлами, символами та рядковими символами; знайомство з системами числення та стандартною бібліотекою.

# **Мета роботи:**

# Оволодіння роботою з файлами у С++, включаючи зчитування та запис файлів.

# Розуміння основних систем числення та їх використання у програмуванні.

# Навчання роботи з бінарними файлами для зберігання та опрацювання даних.

# Розгляд концепцій символів та рядкових змінних, їх операцій та використання у програмуванні.

# Вивчення обробки текстових файлів та роботи зі стрічками у С++.

# Освоєння стандартної бібліотеки С++, що включає функції для оптимізації обробки файлів та рядків.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Текстові та бінарні файли.
* Тема №2: Символи та рядкові змінні.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Текстові та бінарні файли.
  + Джерела Інформації
    - Відео <https://youtu.be/Cz4fl-TUjVk> .
  + Що опрацьовано:
    - Як записувати інформацію в файли та зчитувати її з них
  + Статус: Ознайомлений
  + Початок опрацювання теми: 07.12.2023
  + Завершення опрацювання теми: 09.12.2023
* Тема №2: Символи та рядкові змінні.
  + - Відео <https://youtu.be/zo6B_eJmeLo>
  + Що опрацьовано:
    - Як реалізовані рядкові змінні в С++, як ними користуватися, корисні функції пов’язані зі string.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 05.12.2023
  + Завершення опрацювання теми: 07.12.2023

# **Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №1 **VNS Lab 6 - Task 1-N**

* Варіант завдання – 6
* Деталі завдання: Перетворити рядок так, щоб всі букви в ньому були відсортовані за
* зростанням.

Завдання №2 **VNS Lab 8 - Task 1-N**

* Варіант завдання – 6
* Деталі завдання: Сформувати двійковий файл із елементів, заданої у варіанті структури, роздрукувати його вмістиме, виконати знищення й додавання елементів у відповідності зі своїм варіантом, використовуючи для пошуку елементів що знищуються чи додаються, функцію. Формування, друк, додавання й знищення елементів оформити у вигляді функцій. Передбачити повідомлення про

помилки при відкритті файлу й виконанні операцій вводу/виводу.

Структура "Школяр":

- прізвище, ім'я, по батькові;

- клас;

- номер телефону;

- оцінки по предметах (математика, фізика, українська мова, література).

Знищити всі елементи, у яких є 2 хоча б з одного предмету, додати елемент у

початок файлу.

Завдання №3 **VNS Lab 9 - Task 1-N**

* Варіант завдання – 6
* Деталі завдання:
* Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього інформацію.
* Скопіювати з файлу F1 у файл F2 рядки, починаючи з N до K.
* Підрахувати кількість приголосних букв у файлі F2.

Завдання №4 **Algotester Lab 4v3 (варіант 1)**

* Варіант завдання – 3
* Деталі завдання: Вам дано масив, який складається з N додатніх цілих чисел.

Ваше завдання - розділити його на три частини, по остачі від ділення на 3, по зростанню остачі (тобто спочатку йдуть числа, у яких остача 0, далі числа з остачею 1 і тоді нарешті числа з остачею 2). Далі необхідно ті елементи, остача від ділення на 3 яких парна посортувати по зростанню, а ті, у яких остача 1 - по спаданню. Після цього видаліть усі дублікати з масиву. Виведіть результуючий масив.

Завдання №5 **Algotester Lab 6v3**

* Варіант завдання – 3
* Деталі завдання: У Клінта в черговий раз виключилось світло і йому немає чим зайнятися. Так як навіть це не заставить його подивитися збережені відео про програмування на ютубі - він вирішив придумати свою гру на основі судоку. Гра виглядає так: Є поле розміром N×N, в якому частина клітинок заповнена цифрами, а частина клітинок пусті (позначаються нулем). Також у нього є Q пар координат X та Y. Завданням гри є написати до кожної координати скільки чисел туди можна вписати (якщо вона пуста) і які це числа (обов’язково в посортовані по зростанню!). В клітинку можна вписати лише ті числа, які не зустрічаються в рядку та стовбці, які перетинаються у цій клітинці. Під час гри поле не міняється! Також необовязково, щоб це було валідне судоку! Якщо є клітинка, в яку не можна вписати ніяку цифру - виведіть 0. Також допускаються рядки та стовпці, в яких цифра записана кілька разів.

Завдання №6 **Class Practice Work**

* Деталі завдання: Реалізувати функцію створення файла і запису в нього даних: запис текстової стрічки у файл із заданим ім’ям; копіювання вмісту файла у інший файл.

Завдання №7 **Self-Practice Work 1**

* Варіант завдання: **Зуби**
* Деталі завдання: Мале Бісеня любить гострити зуби. А Зла Тітонька любить до нього підходити і питатися: «Що, зуби гостриш?». Бісеняті таке не дуже подобається, тому воно придумало робити таке.

У Малого Бісеняти є n зубів. Кожен зуб має коефіцієнт загостреності. Також існує межа загостреності k. Якщо коефіцієнт загостреності певного зуба є більшим чи рівним межі загостреності, то такий зуб вважається загостреним.

Мале Бісеня хоче наступного разу, коли Зла Тітонька його щось запитає, показати їй якнайбільше загострених зубів, що розташовані поспіль.

Допоможіть Малому Бісеняті дізнатися, скільки найбільше зубів воно зможе показати.

## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №1 **VNS Lab 6 - Task 1-N**

* Планований час на реалізацію – 2 години

Програма №2 **VNS Lab 8 - Task 1-N**

* Планований час на реалізацію – 3 години

Програма №3 **VNS Lab 9 - Task 1-N**

* Планований час на реалізацію – 3 години

Програма №4 **Algotester Lab 4v3**

* Планований час на реалізацію – 1 година

Програма №5 **Algotester Lab 6v3**

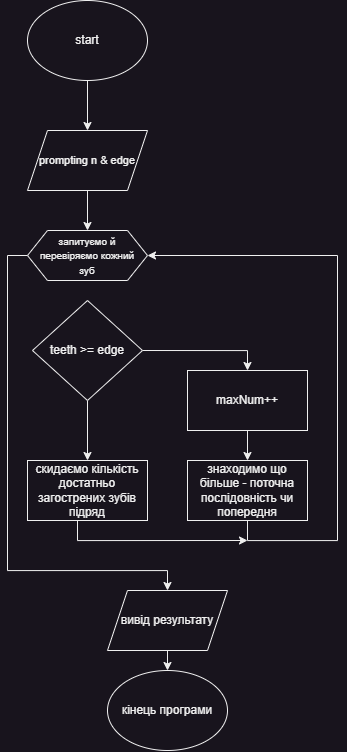
* Планований час на реалізацію – 1 година

Програма №6 **Class Practice Work**

* Планований час на реалізацію 1 година

Програма №7 **Self Practice Work**

* Планований час на реалізацію 20 хвилин
* Flowchart:



*Рисунок 0: блок-схема до програми 1*

## **3. Конфігурація середовища до виконання завдань:**

Незмінна з першого епіку

## **4. Код програм з посиланням на зовнішні ресурси:**

Завдання №1 **VNS Lab 6- Task 1-N**

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

int main()

{

string input;

getline(cin, input);

if(input.size() == 0) {

cout << "wrong input>" << endl;

return 0;

}

sort(input.begin(), input.end());

for(int i = 0; i < input.size(); i++)

cout << input[i];

return 0;

}

Завдання №2 **VNS Lab 8 - Task 1-N**

#include <fstream>

#include <string>

#include <vector>

#include <iostream>

using namespace std;

struct Scoolar {

string lastName, name, surname,

form, telNum;

string

//оцінки з

math,

physic,

ukrLang,

literature;

};

void createFile()

{

ofstream fout("scoolerInfo.dat", ios::binary);

if (fout.is\_open()) {

fout.close();

cout << "file was created>" << endl;

} else

cout << "err of creating file>" << endl;

cout << endl;

}

//зчитування записаних данних про студентів

Scoolar readfile(int index)

{

ifstream fin("scoolerInfo.dat");

for (int i = 0; i < index \* 9; ++i)

{

string line;

getline(fin, line);

}

Scoolar output;

for(int j = 0; j < 9; j++)

{ //дазнаємось про конкретного студента

string line;

getline(fin, line);

switch (j)

{

case 0:

output.lastName = line;

break;

case 1:

output.name = line;

break;

case 2:

output.surname = line;

break;

case 3:

output.form = line;

break;

case 4:

output.telNum = line;

break;

case 5:

output.math = line;

break;

case 6:

output.physic = line;

break;

case 7:

output.ukrLang = line;

break;

case 8:

output.literature = line;

break;

default:

break;

}

}

return output;

}

void writeToFile(Scoolar scoolar, int studentsCount)

{

/\* ifstream fin("scoolerInfo.dat", ios::binary);

if(fin.is\_open())

{

//пропускаємо вже існуючих студентів, записуючи нового в кінець

for(int i = 0; i < studentsCount \* 9; ++i)

{

string line;

getline(fin, line);

}

} \*/

ofstream fout("scoolerInfo.dat", ios::binary | ios::app);

if (fout.is\_open())

{

fout << scoolar.lastName << endl;

fout << scoolar.name << endl;

fout << scoolar.surname << endl;

fout << scoolar.form << endl;

fout << scoolar.telNum << endl;

fout << scoolar.math << endl;

fout << scoolar.physic << endl;

fout << scoolar.ukrLang << endl;

fout << scoolar.literature << endl;

cout << "Student information saved in file!" << endl;

} else

cout << "Error writing to file!" << endl;

fout.close();

}

Scoolar addNewScooler(int studentsCount)

{

Scoolar input;

// string whatToWrite[] = {"Last name> ", "Name> ", "Surname> "};

cout << "Last name> "; cin >> input.lastName;

cout << "Name> "; cin >> input.name;

cout << "Surname> "; cin >> input.surname;

cout << "Form> "; cin >> input.form;

cout << "Tel. number> "; cin >> input.telNum;

cout << "math grades> "; cin >> input.math;

cout << "physic grades> "; cin >> input.physic;

cout << "ukr lang grades> "; cin >> input.ukrLang;

cout << "literature grades> "; cin >> input.literature;

cout << "student created!> ";

// students.push\_back(input);

writeToFile(input, studentsCount);

return input;

}

void ShowInfoAbout(int student, vector<Scoolar> allStudents)

{

Scoolar sclr = allStudents[student - 1];

cout << "#<INFO>#" << endl;

cout << "Student: " << sclr.lastName << " " << sclr.name << " " << sclr.surname << endl;

cout << "Study in: " << sclr.form << "th grade" << endl;

cout << "Tel. number: " << sclr.telNum << endl;

cout << "subject grades:" << endl

<< " Math: " << sclr.math << endl

<< " Physic: " << sclr.physic << endl

<< " Ukr. lang: " << sclr.ukrLang << endl

<< " Literature: " << sclr.literature << endl;

cout << "-----------------" << endl

<< "To show info about student - put serial number, to exit - put 0> ";

}

void ShowAllStudents(vector<Scoolar> allStudents, int length)

{

cout << "#<STUDENTS>#" << endl;

for(int i = 0; i < length; i++)

{

cout << i+1 << ") " << allStudents[i].lastName << endl;

}

cout << "^^^^^^^^^^^^^^" << endl;

cout << "To see more information about a student, write his serial number; to exit enter 0" << endl << "> ";

int point;

while (true)

{

cin >> point;

if(point > length)

cout << "incorect student serial number; Try aganin> ";

else if(point > 0)

ShowInfoAbout(point, allStudents);

else break;

}

}

int howManyStudents()

{

ifstream fin("scoolerInfo.dat", ios::binary);

int totalLines = 0;

string line;

while(getline(fin, line))

totalLines++;

return totalLines / 9;

}

void deleteStudent(vector<Scoolar>& students, int index) {

if (index < 0 || index >= students.size()) {

cerr << "Invalid index>" << endl;

return;

}

ifstream fin("scoolerInfo.dat");

ofstream fout("temp.dat");

if (!fin.is\_open() || !fout.is\_open()) {

cerr << "Error opening files." << endl;

return;

}

string line;

int curLine = 0;

while (getline(fin, line)) {

curLine++;

if (curLine >= (index-1) \* 9 && curLine <= index \* 9)

continue;

fout << line << endl;

}

fin.close();

fout.close();

remove("scoolerInfo.dat");

rename("temp.dat", "scoolerInfo.dat");

cout << "Student deleted successfully." << endl;

}

int main()

{

int studentsCount = howManyStudents();

vector<Scoolar> students;

//якщо ми ще не додавали студентів в базу, то створюємо новий файл

if(studentsCount == 0)

createFile();

else

{

//а якщо додавали, то считуємо всю відому інформацію та записуємо її

for (int i = 0; i < studentsCount; i++)

students.push\_back(readfile(i));

}

//це менюшка

while (true)

{

int point;

cout << " #<MENU># " << endl

<< "0. Out" << endl

<< "1. Add new scooler" << endl

<< "2. See all students" << endl

<< "3. Delete studenet" << endl

<<endl;

cin >> point;

switch (point)

{

case 0:

return 0;

break;

case 1:

students.push\_back(addNewScooler(studentsCount));

studentsCount++;

break;

case 2:

ShowAllStudents(students, studentsCount);

break;

case 3:

cout << "enter index> ";

int point;

cin >> point;

if(point > studentsCount)

cout << "Error serial num>" << endl;

else if(point > 0)

{

students.erase(students.begin() + point-1);

deleteStudent(students, point);

studentsCount--;

}

break;

default:

break;

}

}

return 0;

}

Завдання №3 **VNS Lab 9 - Task 1-N**

#include <fstream>

#include <iostream>

using namespace std;

ofstream CopyFile(int start, int end)

{

ifstream fin ("F1.txt");

ofstream fout ("F2.txt");

if(!fin.is\_open())

{

cout << "error whily try to open file> " << endl;

return fout;

}

string line;

for(int i = 0; getline(fin, line); i++)

{

if(i < start)

continue;

else if(i > end)

continue;

fout << line << endl;

}

return fout;

}

bool isConsonant(char chr)

{

chr = tolower(chr);

if(chr == 'b' || chr == 'c' || chr == 'd' || chr == 'f' ||

chr == 'g' || chr == 'h' || chr == 'j' || chr == 'k' ||

chr == 'l' || chr == 'm' || chr == 'n' || chr == 'p' ||

chr == 'r' || chr == 's' || chr == 't' || chr == 'v' ||

chr == 'w' || chr == 'x' || chr == 'y' || chr == 'z' ||

chr == 'q')

{

return true;

}

else return false;

}

int CalculateConsonantLetters(string fileName)

{

ifstream fin(fileName);

string line;

int result = 0;

for (int i = 0; getline(fin, line); i++)

{

for (int j = 0; j < line.size(); j++)

{

if(isConsonant(line[j]))

result++;

}

}

return result;

}

int main()

{

int n, k;

cin >> n >> k;

ofstream textFile = CopyFile(n, k);

int result = CalculateConsonantLetters("F2.txt");

cout << "Кількість приголосних у файлі F2: " << result << endl;

return 0;

}

Завдання №4 **Algotester Lab 4v3**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

bool CheckIfNumberOccurs(vector<int> arr, int num)

{

for (int i = 0; i < arr.size(); i++)

{

if(arr[i] == num)

return true;

}

return false;

}

vector<int> sortedNumbers(vector<int> numbers)

{

vector<int> noRemainder, rem1, rem2;

for (int num : numbers)

{

if (num % 3 == 0)

noRemainder.push\_back(num);

else if (num % 3 == 1)

rem1.push\_back(num);

else

rem2.push\_back(num);

}

sort(noRemainder.begin(), noRemainder.end());

sort(rem2.begin(), rem2.end());

sort(rem1.begin(), rem1.end(), greater<int>());

vector<int> temp, result;

temp.insert(temp.end(), noRemainder.begin(), noRemainder.end());

temp.insert(temp.end(), rem1.begin(), rem1.end());

temp.insert(temp.end(), rem2.begin(), rem2.end());

for(int i = 0; i < temp.size(); i++)

{

if(!CheckIfNumberOccurs(result, temp[i]))

result.push\_back(temp[i]);

else continue;

}

return result;

}

int main()

{

//input

int n;

cin >> n;

vector<int> numbers(n);

for (int i=0; i<n; ++i)

cin >> numbers[i];

vector<int> result = sortedNumbers(numbers);

cout << result.size() << endl;

for (int i=0; i < result.size(); ++i)

cout << result[i] << " ";

return 0;

}

Завдання №5 **Algotester Lab 6v3**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main()

{

//input

int length;

cin >> length;

string field[length];

for (int i = 0; i < length; i++)

cin >> field[i];

int q, iteration = 0;

cin >> q;

int prompting[q][2];

for (int i = 0; i < q; i++)

{

for (int j = 0; j < 2; j++)

cin >> prompting[i][j];

}

for (int iteration = 0; iteration < q; iteration++)

{

int cX = prompting[iteration][0], cY = prompting[iteration][1];

cX--; cY--;

if (field[cX][cY] != '0')

cout << "1" << endl << field[cX][cY] << endl << endl;

else

{

vector<int> nums;

for (int i = 0; i < length; i++)

nums.push\_back(field[i][cY] - '0');

for (int i = 0; i < length; i++)

nums.push\_back(field[cX][i] - '0');

int result = 0;

for (int i = 1; i <= length; i++)

{

auto search = find(nums.begin(), nums.end(), i);

if (search == nums.end())

result++;

}

cout << result << endl;

for (int i = 1; i <= length; i++)

{

auto search = find(nums.begin(), nums.end(), i);

if (search == nums.end())

cout << i << " ";

}

cout << endl << endl;

}

}

}

Завдання №6 **Class Practice Work**

#include <fstream>

#include <iostream>

#include <string>

#include <vector>

using namespace std;

bool InputWaiter(string errorText)

{

char empty;

cout << errorText

<< "enter anything to continue> ";

cin >> empty;

return false;

}

bool CreateNewFile(string fileName, vector<string> &createdFiles)

{

for (int i = 0; i < createdFiles.size(); i++)

{

//перевірка на те чи існує вже такий файл

if(fileName == createdFiles[i] || (fileName + ".txt") == createdFiles[i])

return InputWaiter("this file already exist...");

}

string fileType; int typeLength = 0;

for (int i = fileName.size() - 1; i >= 0; i--)

{ //перевірка на допустиму назву

if

( fileName[i] == '\\' || fileName[i] == '/' ||

fileName[i] == ':' || fileName[i] == '\*' ||

fileName[i] == '?' || fileName[i] == '"' ||

fileName[i] == '<' || fileName[i] == '>' ||

fileName[i] == '|' )

{

return InputWaiter("Wrong file name. Try again...");

}

//дізнаємось який тип файлу хоче створити користувач

typeLength++;

if(fileName[i] == '.' && fileType.size() == 0)

{

for(int j = 0; j < typeLength; j++)

fileType += fileName[i + j];

}

}

//якщо користувач не вказав тип створеного файлу,

//то по дефотлу будемо створювати з типом txt

if(fileType.size() == 0)

fileName += ".txt";

//безпосереднє створення файлу, все ради чого це було. Ура!

ofstream fout(fileName);

createdFiles.push\_back(fileName); //додаємо створений файл до загального списку

//зберігаємо назву файлу

ofstream wrtieNewFileToList("storageFile.dat", ios::app);

wrtieNewFileToList << fileName << endl;

return true;

}

void DeleteFile(int index, vector<string> &allFileNames)

{

index--;

if(index < 0 || index > allFileNames.size())

{

InputWaiter("wrong index... ");

return;

}

vector<string> temp;

ifstream fin("storageFile.dat");

ofstream tempStorage("tempStorage.dat");

for(int i = 0; i < allFileNames.size(); i++)

{

string line;

getline(fin, line);

if(i == index)

{

//видалення самого файлу

remove(allFileNames[i].c\_str());

continue;

}

//змінюємо вектор з назвами

temp.push\_back(allFileNames[i]);

//змінюємо файл з назвами

tempStorage << line << endl;

}

allFileNames = temp;

fin.close();

tempStorage.close();

remove("storageFile.dat");

rename("tempStorage.dat", "storageFile.dat");

}

void SeeAllFiles(vector<string> &createdFiles, bool withDeleting = false)

{

if(withDeleting)

{

cout << "Select which file you want to delete (0 to exit): " << endl;

} else

{

cout << "All created files: " << endl;

}

for (int i = 0; i < createdFiles.size(); i++)

cout << i+1 << ") " << createdFiles[i] << endl;

cout << "^^^^^^^^^" << endl;

if(withDeleting)

{

cout << "input> ";

int point;

cin >> point;

if(point >= 1 && point <= createdFiles.size())

DeleteFile(point, createdFiles);

else if (point != 0)

InputWaiter("incorect index");

} else

{

cout << "enter anything> ";

char empty;

cin >> empty;

}

}

bool WriteToFile(string fileName, string content)

{

//відкриваємо файл щоб можна було записувати з кінця

ofstream fout(fileName, ios::app);

fout << content << endl;

InputWaiter("Success!");

return true;

}

vector<string> AlreadyCreatedFiles(string storageFile = "storageFile.dat")

{

ifstream fin(storageFile);

vector<string> result;

if(fin.is\_open())

{

string line;

while (getline(fin, line))

{

result.push\_back(line);

}

} else

{

ofstream fout("storageFile.dat");

}

return result;

}

int main()

{

vector<string> allFiles = AlreadyCreatedFiles();

// << "0. Stop the program" << endl

while (true)

{

//інтерфейс меню

cout << " #<MENU># " << endl

<< "1. Create New File" << endl

<< "2. See All Created Files" << endl

<< "3. Delete File" << endl;

// cout << "input> ";

int point;

cout << "input> ";

cin >> point;

string fileName;

switch (point)

{

case 1:

cout << "enter file name> ";

cin >> fileName;

CreateNewFile(fileName, allFiles);

break;

case 2:

SeeAllFiles(allFiles);

break;

case 3:

SeeAllFiles(allFiles, true);

break;

default:

break;

}

}

}

Завдання №7 **Self-Practice**

#include <iostream>

int main()

{

int n, edge;

std::cin >> n >> edge;

int maxNum = 0, output = 0;

for(int i = 0; i < n; i++)

{

int teeth;

std::cin >> teeth;

if (teeth >= edge)

{

maxNum++;

output = std::max(maxNum, output);

}

else

maxNum = 0;

}

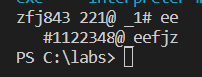
std::cout << output;

return 0;

}

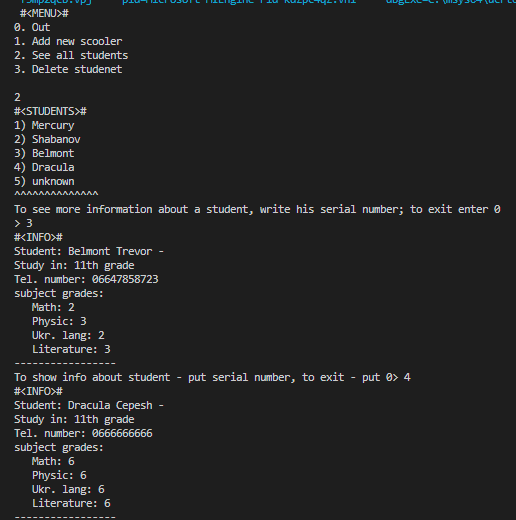
## **5. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №1 **VNS Lab 6 - Task 1-N**

*Рисунок 1: результат виконання програми 1*

Час затрачений на виконання завдання: 20 хвилин

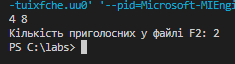
Завдання №2 **VNS Lab 8 - Task 1-N**



*Рисунок 2: результат виконання програми 2*

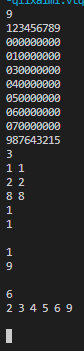
Час затрачений на виконання завдання: 3 години

Завдання №3 **VNS Lab 9 - Task 1-N**

*Рисунок 3: результат виконання програми 3*

Час затрачений на виконання завдання: 1 година

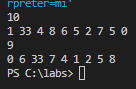
Завдання №4 **Algotester Lab4v3**



*Рисунок 4: результат виконання програми 4*

Час затрачений на виконання завдання: 2 години

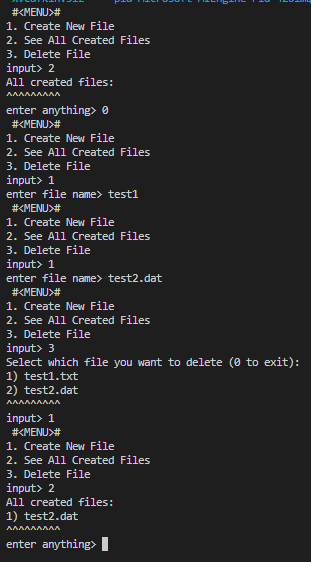
Завдання №5 **Algotester Lab6v3**



*Рисунок 5: результат виконання програми 5*

Час затрачений на виконання завдання: 1 година

Завдання №6 **Class Practice Work**



*Рисунок 6: результат виконання програми 6*

Час затрачений на виконання завдання: 2 години

Завдання №7 **Self-Practice Work Зуби**



*Рисунок 7: результат виконання програми 7*

Час затрачений на виконання завдання: 20 хвилин

# **Висновки:**

Під час виконання даного епіку я навчився працювати з файлами (записувати та зчитувати з них інформацію), строками (перевести, наприклад, в char або звернутися до окремого елемента), та використовувати функції (було цікаво змінювати динамічний масив (ветктор) в функції видалення, використовуючи & для передавання посилання на об’єкт)