Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 5**

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

***з розділу***: «Файли. Системи числення. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек»

***Виконала:***

студентка групи ШІ-14

Михальчук Віра Іванівна

# **Тема роботи:**

Бінарні та текстові файли. Робота з файлами у С++. Символи та рядкові змінні. Зчитування з файлу та запис у файл. Створення й використання бібліотек

# **Мета роботи:**

Ознайомитися з файлами та з деталями роботи з ними у С++. Опрацювати зчитування та запис у файл. Ознайомитися з символами та рядковими змінним

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Файли.
* Тема №2: Бібліотеки.

1. Індивідуальний план опрацювання теорії:

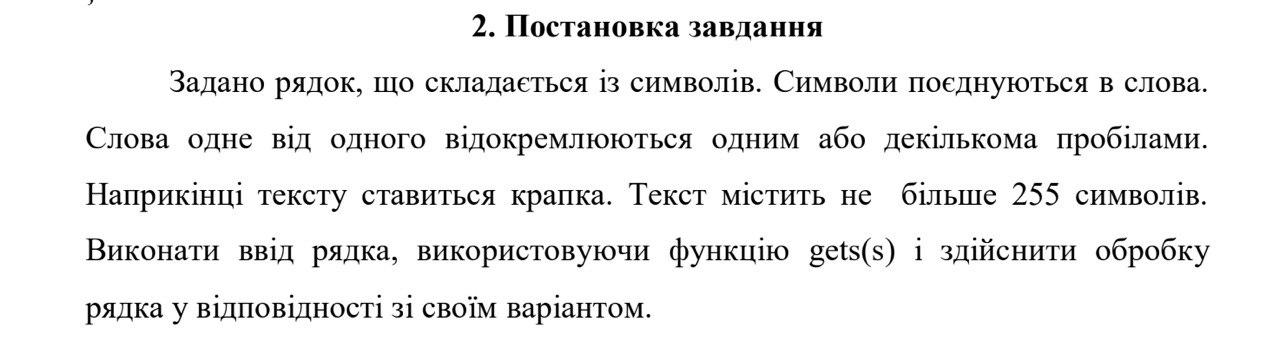
* Тема №1: Файли.
  + Джерела Інформації:
    - <https://acode.com.ua/urok-220-bazovyj-fajlovyj-vvid-i-vyvid/>
    - <http://cpp.dp.ua/ponyattya-fajla/>
    - <https://www.bestprog.net/uk/2019/09/11/examples-of-using-c-tools-for-working-with-files-ua/>
  + Що опрацьовано:
    - Опрацьовано файли, їх види, особливості роботи з ними, потоковий ввід та вивід.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 18.12
* Звершення опрацювання теми: 20.12
* Тема №2 : Бібліотеки.
  + Джерела Інформації
    - <https://acode.com.ua/statychni-i-dynamichni-biblioteky/>
  + Що опрацьовано:
    - Опрацьовано бібліотеки та їх види.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 18.12
  + Звершення опрацювання теми: 20.12

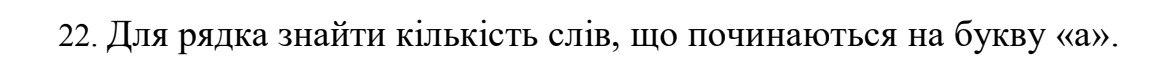
# **Виконання роботи:**

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

Завдання №1 VNS Lab 6

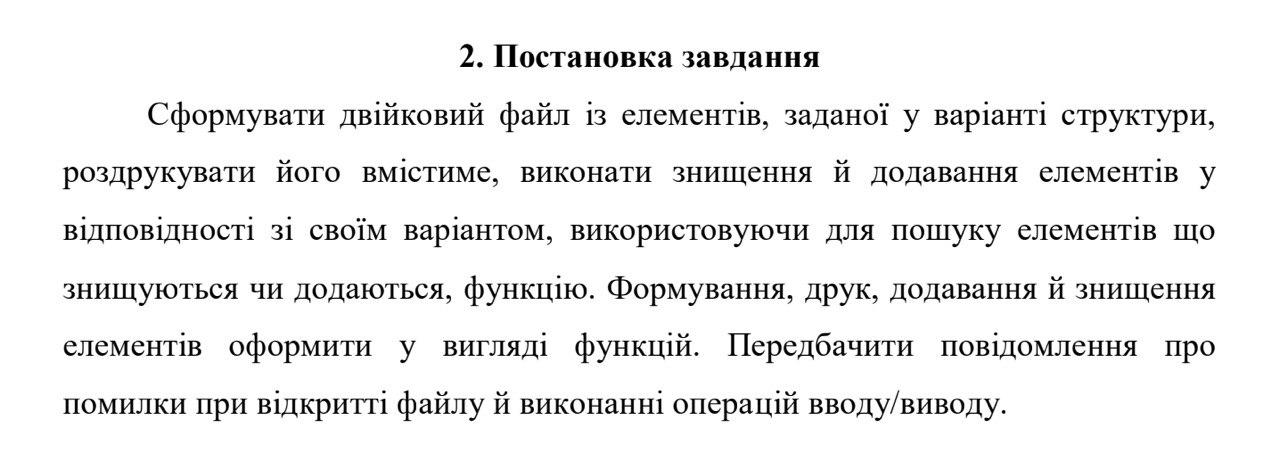
* Варіант 22
* Деталі завдання

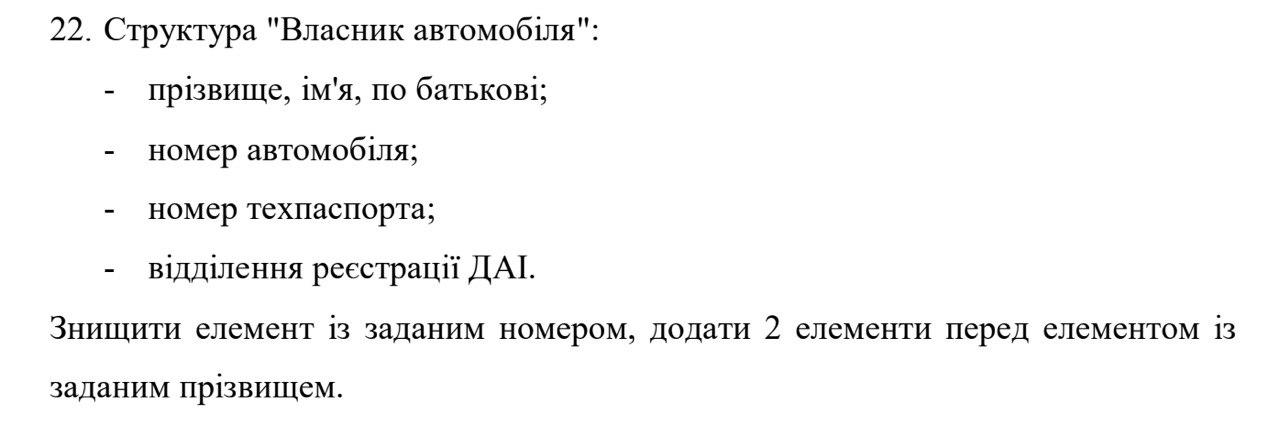




Завдання №2 VNS Lab 8

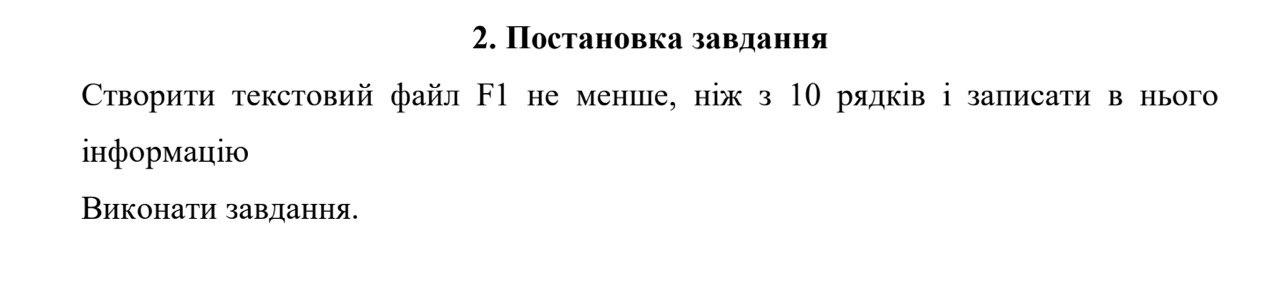
* Варіант 22
* Деталі завдання

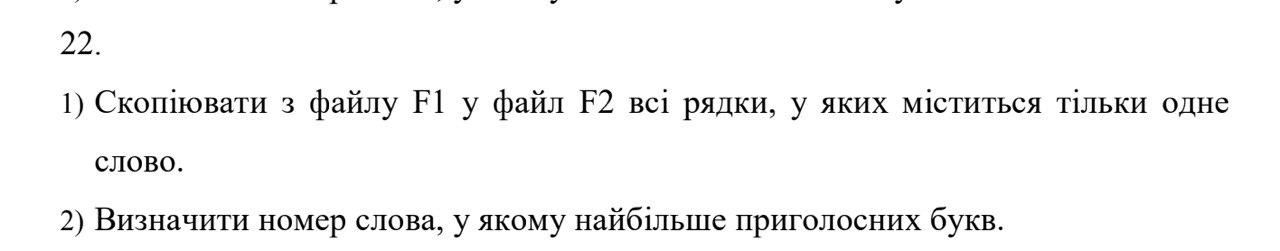




Завдання №3 VNS Lab 9

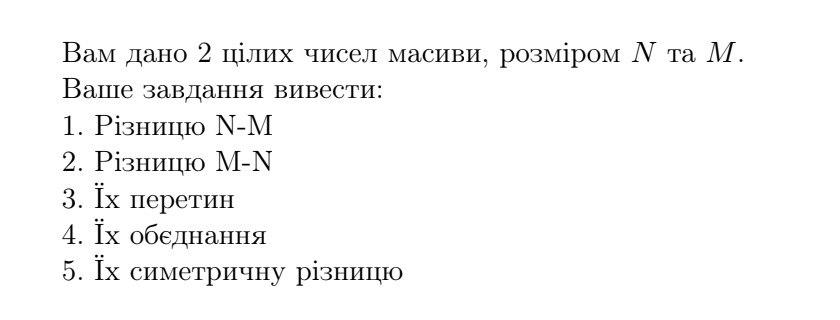
* Варіант 22
* Деталі завдання





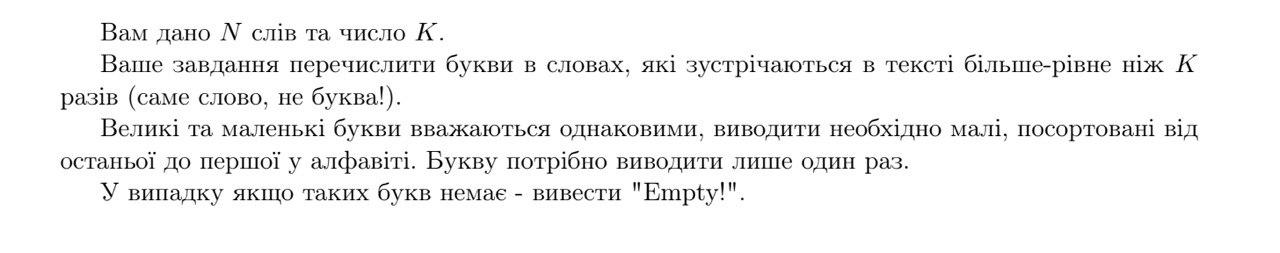
Завдання №4 Algotester Lab 4

* Варіант 1
* Деталі завдання



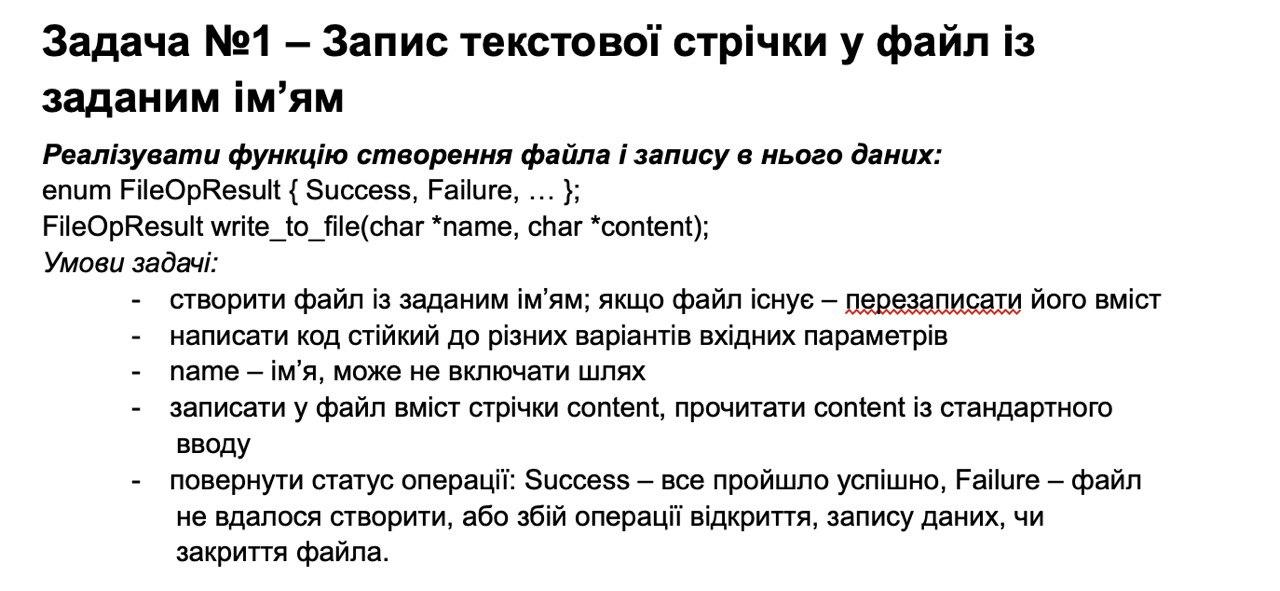
Завдання №5 Algotester Lab 6

* Варіант 1
* Деталі завдання



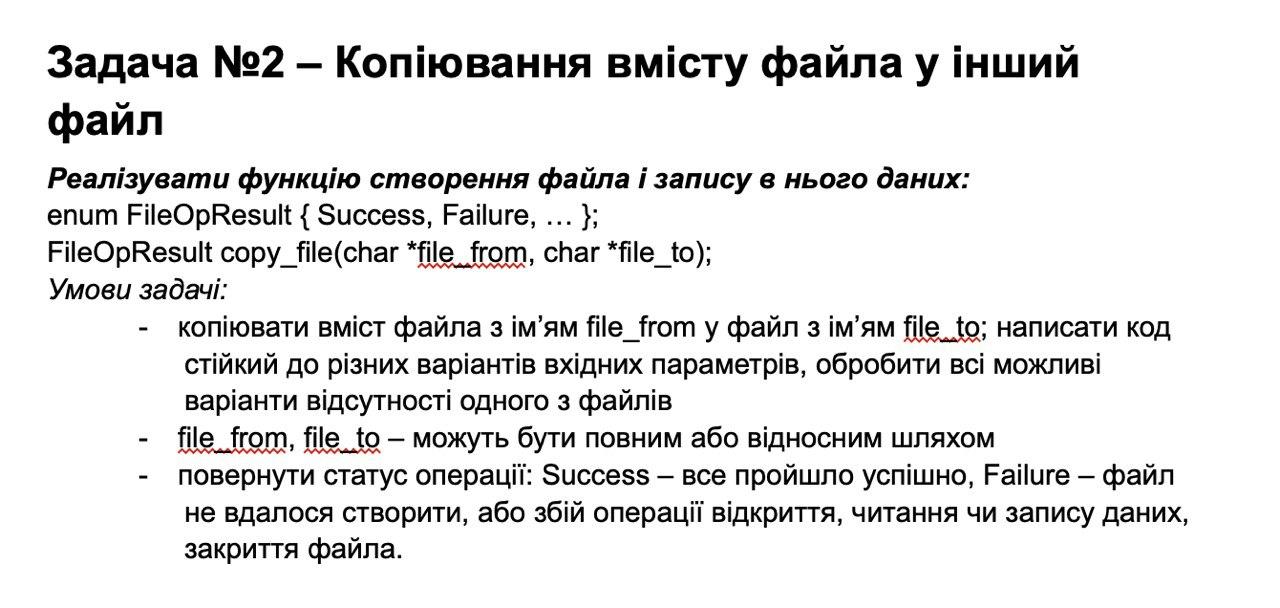
Завдання №6 Class Practice Work 1

* Варіант завдання
* Деталі завдання



Завдання №7 Class Practice Work 2

* Варіант завдання
* Деталі завдання



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

Програма №1 VNS Lab 6

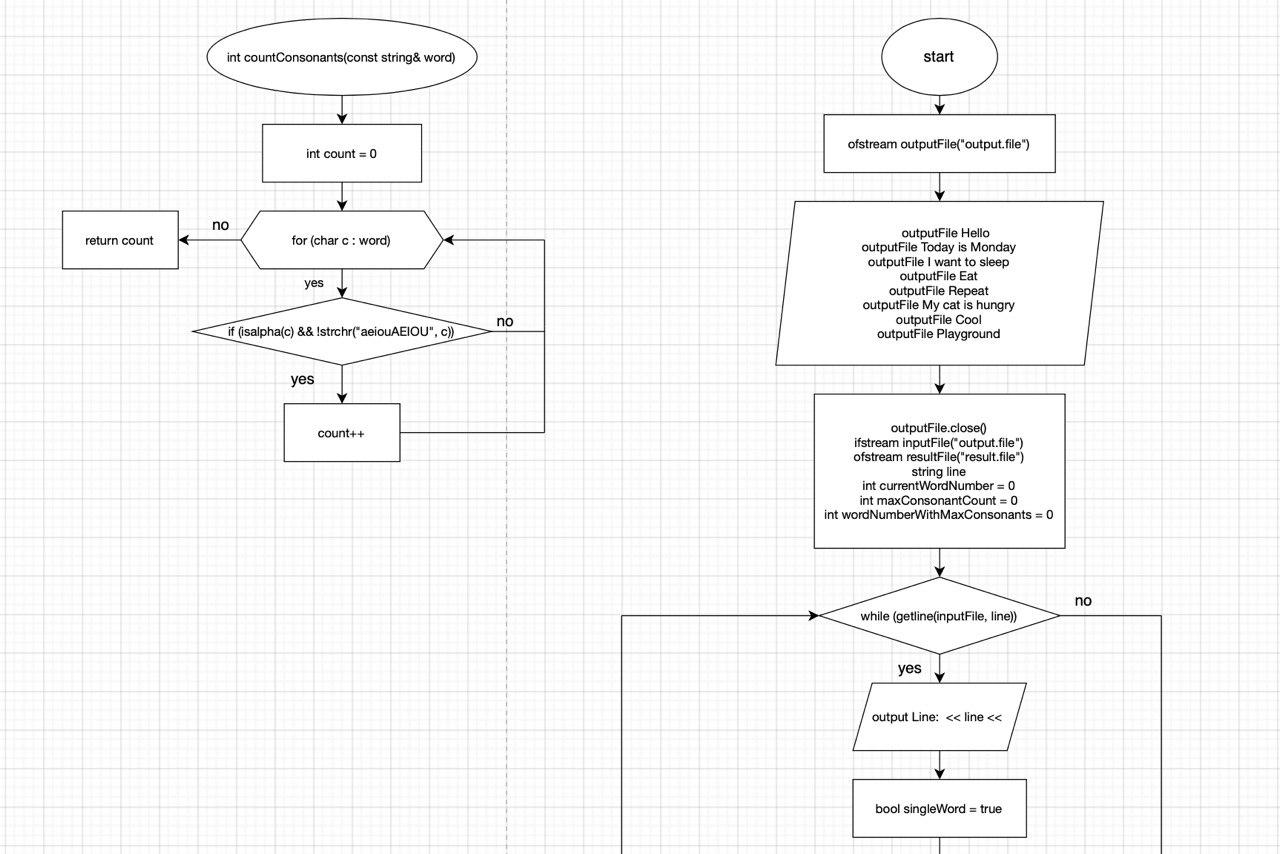
* Планований час на реалізацію 1 година

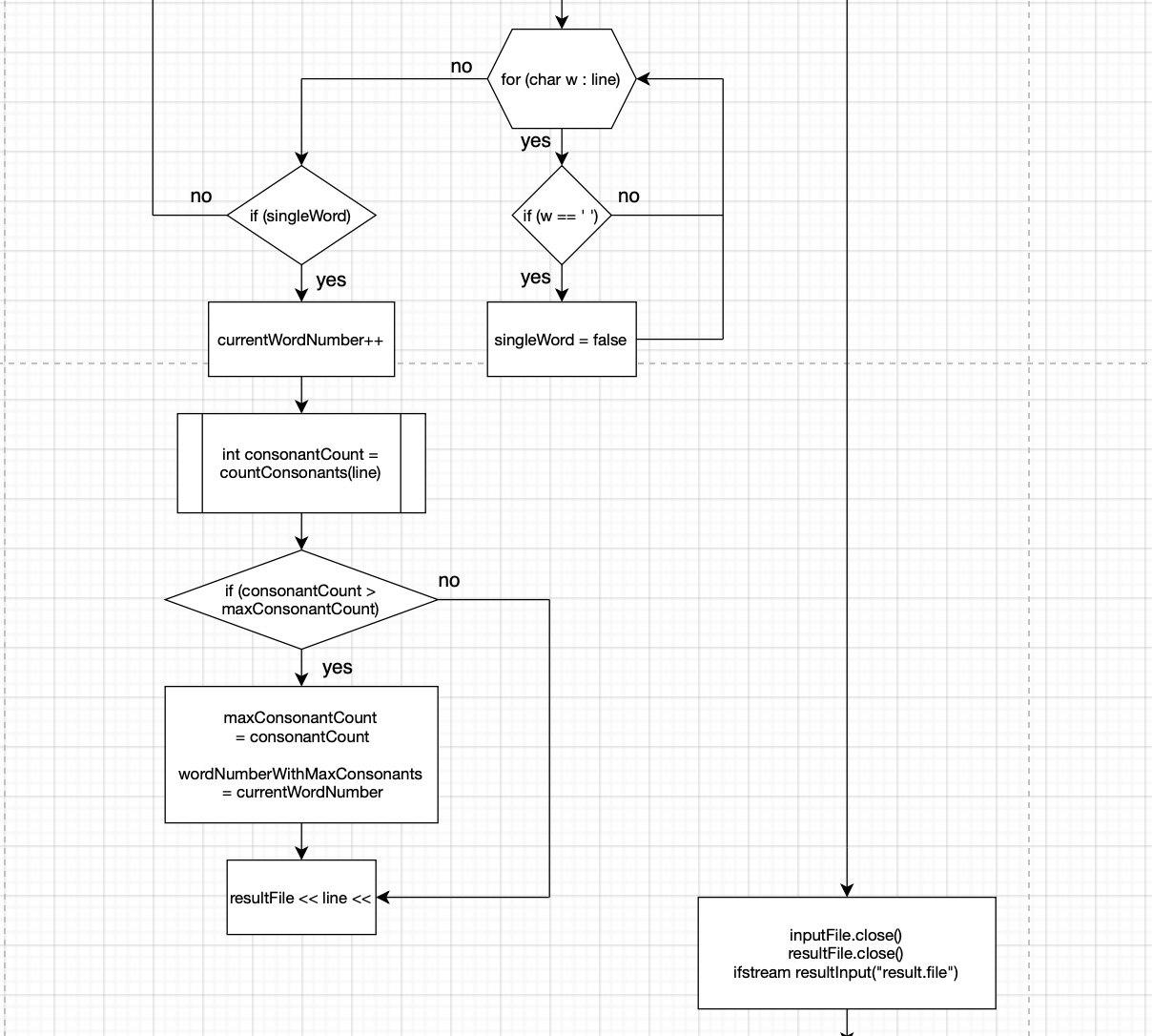
Програма №2 VNS Lab 8

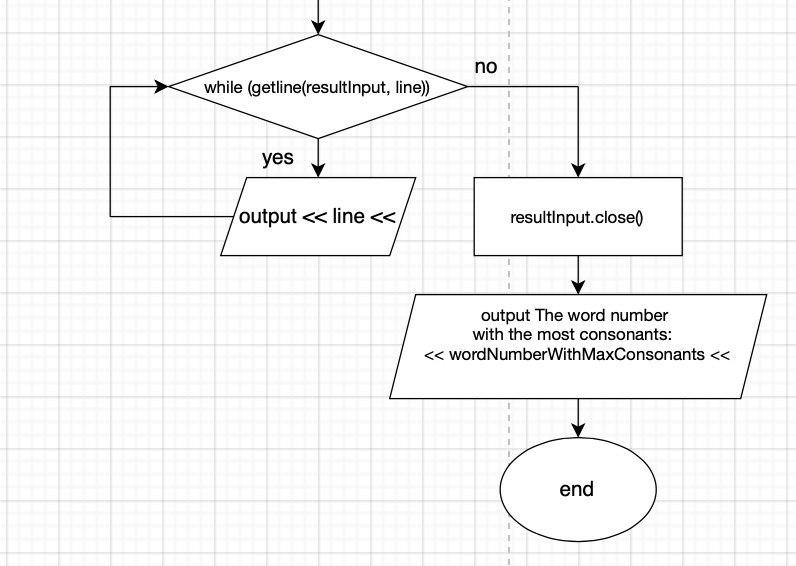
* Планований час на реалізацію 1 година

Програма №3 VNS Lab 9

* Блок-схема







* Планований час на реалізацію 1.5 години

Програма №4 Algotester Lab 4

* Планований час на реалізацію 2 години

Програма №5 Algotester Lab 6

* Планований час на реалізацію 2 години

Програма №6 Class Practice Work 1

* Планований час на реалізацію 1 година

Програма №7 Class Practice Work 2

* Планований час на реалізацію 1.5 години

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

Завдання №1 VNS Lab 6

[vns\_lab\_6\_task\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_5_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_5_practice_and_labs_vira_mykhalchuk/vns_lab_6_task_vira_mykhalchuk.cpp)

#include <iostream>

#include <cstring>

using namespace std;

int Aword(const char\* inputString) {

int count = 0;

//strtok розділення рядка на слова

char\* token = strtok(const\_cast<char\*>(inputString), " ");

while (token != nullptr) {

if (token[0] == 'a' or token[0] == 'A') {

count++;

}

token = strtok(nullptr, " ");

}

return count;

}

int main() {

const int maxLength = 255;

char inputString[maxLength];

// Ввід рядка

cout << "Enter your text: ";

gets(inputString);

int result = Aword(inputString);

cout << result << " words begin with letter \"a\" \n";

return 0;

}

Завдання №2 VNS Lab 8

[vns\_lab\_8\_task\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_5_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_5_practice_and_labs_vira_mykhalchuk/vns_lab_8_task_vira_mykhalchuk.cpp)

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

const int maxOwners = 100;

struct OwnerOfCar

{

string lastName;

string firstName;

string fathersName;

string plateOfCar;

long numberOfTechPassport;

int numberOfRegistraionOffice;

};

void printInformation(OwnerOfCar owners[], int size)

{

cout << '\n';

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

{

cout << "The last name: " << owners[i].lastName << '\n';

cout << "The first name: " << owners[i].firstName << '\n';

cout << "The middle name: " << owners[i].fathersName << '\n';

cout << "The car plate: " << owners[i].plateOfCar << '\n';

cout << "The technical passport number: " << owners[i].numberOfTechPassport << '\n';

cout << "The number of registration office: " << owners[i].numberOfRegistraionOffice << '\n';

cout << "\n";

}

}

void deleteOwner(OwnerOfCar owners[], int& size, const string& plateOfCarToDelete)

{

cout << '\n';

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

{

if (owners[i].plateOfCar == plateOfCarToDelete)

{

for (int j = i; j < size - 1; ++j)

{

owners[j] = owners[j + 1];

}

--size;

break;

}

}

}

void addOwners(OwnerOfCar owners[], int& size, const string& LastName, const OwnerOfCar& owner1, const OwnerOfCar& owner2)

{

cout << '\n';

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

{

if (owners[i].lastName == LastName)

{

for (int j = size - 1; j >= i; --j)

{

owners[j + 2] = owners[j];

}

owners[i] = owner1;

owners[i + 1] = owner2;

size += 2;

break;

}

}

}

int main()

{

OwnerOfCar carOwners[maxOwners];

int numberOfOwners = 0;

ifstream file("car\_owners");

if (file.is\_open())

{

OwnerOfCar tempOwner;

while (file >> tempOwner.lastName >> tempOwner.firstName >> tempOwner.fathersName >> tempOwner.plateOfCar >> tempOwner.numberOfTechPassport >> tempOwner.numberOfRegistraionOffice)

{

carOwners[numberOfOwners++] = tempOwner;

}

file.close();

}

else

{

cout << "File wasn't opened. New file was created.\n";

}

OwnerOfCar owner1 = {"Ivanov", "Ivan", "Ivanovych", "BC1111KK", 123456, 1};

OwnerOfCar owner2 = {"Petrov", "Petro", "Petrovych", "BC5656CC", 234567, 2};

OwnerOfCar owner3 = {"Mykhalchuk", "Dmytro", "Ivanivna", "BC6504OE", 000001, 100};

carOwners[numberOfOwners++] = owner1;

carOwners[numberOfOwners++] = owner2;

carOwners[numberOfOwners++] = owner3;

ofstream file\_output("car\_owners");

if (file\_output.is\_open())

{

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

{

file\_output << carOwners[i].lastName << " " << carOwners[i].firstName << " " << carOwners[i].fathersName << " "

<< carOwners[i].plateOfCar << " " << carOwners[i].numberOfTechPassport << " " << carOwners[i].numberOfRegistraionOffice << '\n';

}

file\_output.close();

}

cout << "\nThe list of car owners before editing:\n";

printInformation(carOwners, numberOfOwners);

string carPlateToDelete;

cout << "Enter car plate to delete: ";

cin >> carPlateToDelete;

deleteOwner(carOwners, numberOfOwners, carPlateToDelete);

cout << "The list after deleting\n";

printInformation(carOwners, numberOfOwners);

OwnerOfCar newOwner1 = {"Romanov", "Roman", "Romanovych", "BC1025DD", 345678, 5};

OwnerOfCar newOwner2 = {"Mykolov", "Mykola", "Mykolovych", "BC2923AA", 456789, 21};

string lastNameToInsertAfter;

cout << "Enter the last name from the list to insert after new two owners: ";

cin.ignore();

getline(cin, lastNameToInsertAfter);

addOwners(carOwners, numberOfOwners, lastNameToInsertAfter, newOwner1, newOwner2);

cout << "The final list: ";

printInformation(carOwners, numberOfOwners);

ofstream finalOutputFile("car\_owners");

if (finalOutputFile.is\_open())

{

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

{

finalOutputFile << carOwners[i].lastName << " " << carOwners[i].firstName << " " << carOwners[i].fathersName << " "

<< carOwners[i].plateOfCar << " " << carOwners[i].numberOfTechPassport << " " << carOwners[i].numberOfRegistraionOffice << '\n';

}

finalOutputFile.close();

cout << "The final list has been saved in the 'car\_owners' file." << endl;

}

else

{

cout << "Error: Unable to open the file for writing." << endl;

}

return 0;

}

Завдання №3 VNS Lab 9

[vns\_lab\_9\_task\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_5_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_5_practice_and_labs_vira_mykhalchuk/vns_lab_9_task_vira_mykhalchuk.cpp)

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

int countConsonants(const string& word) {

int count = 0;

for (char c : word) {

if (isalpha(c) && !strchr("aeiouAEIOU", c)) {

count++;

}

}

return count;

}

int main() {

ofstream outputFile("output.file");

outputFile << "Hello\n";

outputFile << "Today is Monday\n";

outputFile << "I want to sleep\n";

outputFile << "Eat\n";

outputFile << "Repeat\n";

outputFile << "My cat is hungry\n";

outputFile << "Cool\n";

outputFile << "Playground\n";

outputFile.close();

ifstream inputFile("output.file");

ofstream resultFile("result.file");

string line;

int currentWordNumber = 0;

int maxConsonantCount = 0;

int wordNumberWithMaxConsonants = 0;

while (getline(inputFile, line)) {

cout << "Line: " << line << "\n";

bool singleWord = true;

for (char w : line) {

if (w == ' ') {

singleWord = false;

break;

}

}

if (singleWord) {

currentWordNumber++;

int consonantCount = countConsonants(line);

if (consonantCount > maxConsonantCount) {

maxConsonantCount = consonantCount;

wordNumberWithMaxConsonants = currentWordNumber;

}

resultFile << line << "\n";

}

}

inputFile.close();

resultFile.close();

cout << "\n";

ifstream resultInput("result.file");

while (getline(resultInput, line)) {

cout << line << "\n";

}

resultInput.close();

cout << "\n";

cout << "The word number with the most consonants: " << wordNumberWithMaxConsonants << "\n";

return 0;

}

Завдання №4 Algotester Lab 4

[algotester\_lab\_4\_v\_1\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_5_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_5_practice_and_labs_vira_mykhalchuk/algotester_lab_4_v_1_vira_mykhalchuk.cpp)

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int sizeN, sizeM;

cin >> sizeN;

int arrayN[100];

for (int i = 0; i < sizeN; ++i) {

cin >> arrayN[i];

}

cin >> sizeM;

int arrayM[100];

for (int i = 0; i < sizeM; ++i) {

cin >> arrayM[i];

}

sort(arrayN, arrayN + sizeN);

sort(arrayM, arrayM + sizeM);

int diffNM[100];

auto itDiffNM = set\_difference(arrayN, arrayN + sizeN, arrayM, arrayM + sizeM, diffNM);

int sizeDiffNM = distance(diffNM, itDiffNM);

cout << sizeDiffNM << "\n";

for (int i = 0; i < sizeDiffNM; ++i) {

cout << diffNM[i] << " ";

}

cout << "\n";

int diffMN[100];

auto itDiffMN = set\_difference(arrayM, arrayM + sizeM, arrayN, arrayN + sizeN, diffMN);

int sizeDiffMN = distance(diffMN, itDiffMN);

cout << sizeDiffMN << "\n";

for (int i = 0; i < sizeDiffMN; ++i) {

cout << diffMN[i] << " ";

}

cout << "\n";

int intersection[100];

auto itIntersection = set\_intersection(arrayN, arrayN + sizeN, arrayM, arrayM + sizeM, intersection);

int sizeIntersection = distance(intersection, itIntersection);

cout << sizeIntersection << "\n";

for (int i = 0; i < sizeIntersection; ++i) {

cout << intersection[i] << " ";

}

cout << "\n";

int Union[200];

auto itUnion = set\_union(arrayN, arrayN + sizeN, arrayM, arrayM + sizeM, Union);

int sizeUnion = distance(Union, itUnion);

cout << sizeUnion << "\n";

for (int i = 0; i < sizeUnion; ++i) {

cout << Union[i] << " ";

}

cout << "\n";

int symmetricDifference[200];

auto itSymmetricDiff = set\_symmetric\_difference(arrayN, arrayN + sizeN, arrayM, arrayM + sizeM, symmetricDifference);

int sizeSymmetricDiff = distance(symmetricDifference, itSymmetricDiff);

cout << sizeSymmetricDiff << "\n";

for (int i = 0; i < sizeSymmetricDiff; ++i) {

cout << symmetricDifference[i] << " ";

}

cout << "\n";

return 0;

}

Завдання №5 Algotester Lab 6

[algotester\_lab\_6\_v\_1\_vira\_mykhalchuk.cpp.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_5_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_5_practice_and_labs_vira_mykhalchuk/algotester_lab_6_v_1_vira_mykhalchuk.cpp.cpp)

#include <iostream>

#include <unordered\_map>

#include <set>

#include <algorithm>

using namespace std;

int main() {

int N, K;

cin >> N >> K;

unordered\_map<string, int> wordCount;

set<char> repeatedLetters;

for (int i = 0; i < N; ++i) {

string word;

cin >> word;

for (char& c : word) {

c = tolower(c);

}

if (++wordCount[word] == K) {

for (char c : word) {

repeatedLetters.insert(c);

}

}

}

if (repeatedLetters.empty()) {

cout << "Empty!" << endl;

} else {

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

for (auto it = repeatedLetters.rbegin(); it != repeatedLetters.rend(); ++it) {

cout << \*it << ' ';

}

cout << endl;

}

return 0;

}

Завдання №6 Class Practice Work 1

[class\_ptactice\_task\_1\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_5_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_5_practice_and_labs_vira_mykhalchuk/class_ptactice_task_1_vira_mykhalchuk.cpp)

#include <iostream>

#include <fstream>

using namespace std;

enum FileOpResult { success, failure };

FileOpResult write\_to\_file(const char\* name, const char\* content) {

ofstream file(name);

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

return failure;

}

file << content;

return success;

}

int main() {

char filename[15];

char content[250];

cout << "Enter the name of the file: ";

cin >> filename;

cout << "Enter your content: ";

cin.ignore();

cin.getline(content, sizeof(content));

FileOpResult result = write\_to\_file(filename, content);

if (result == success) {

cout << "Your file is succeddfully created!\n";

} else {

cout << "ERROR\n";

}

return 0;

}

Завдання №7 Class Practice Work 2

[class\_practice\_task\_2\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_5_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_5_practice_and_labs_vira_mykhalchuk/class_practice_task_2_vira_mykhalchuk.cpp)

#include <iostream>

#include <fstream>

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult copy\_file(char\* file\_from, char\* file\_to) {

ifstream from\_file(file\_from, ios::binary);

if (!from\_file.is\_open()) {

return Failure;

}

ofstream to\_file(file\_to, ios::binary);

if (!to\_file.is\_open()) {

return Failure;

}

to\_file << from\_file.rdbuf();

if (from\_file.fail() || to\_file.fail()) {

return Failure;

}

return Success;

}

int main() {

char from\_file[250];

char to\_file[250];

cout << "Enter the name of file from which you want to copy content: ";

cin >> from\_file;

cout << "Enter the name of file to which you want to copy content: ";

cin >> to\_file;

FileOpResult result = copy\_file(from\_file, to\_file);

if (result == Success) {

cout << "Content of file: " << "\"" << from\_file << "\"" << " is successfully copied to file " << "\"" << to\_file << "\"."<< "\n";

} else {

cout << "ERROR\n";

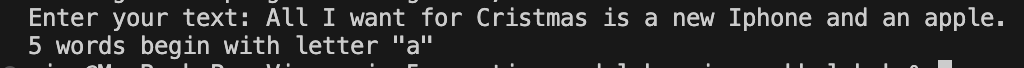
}

return 0;

}

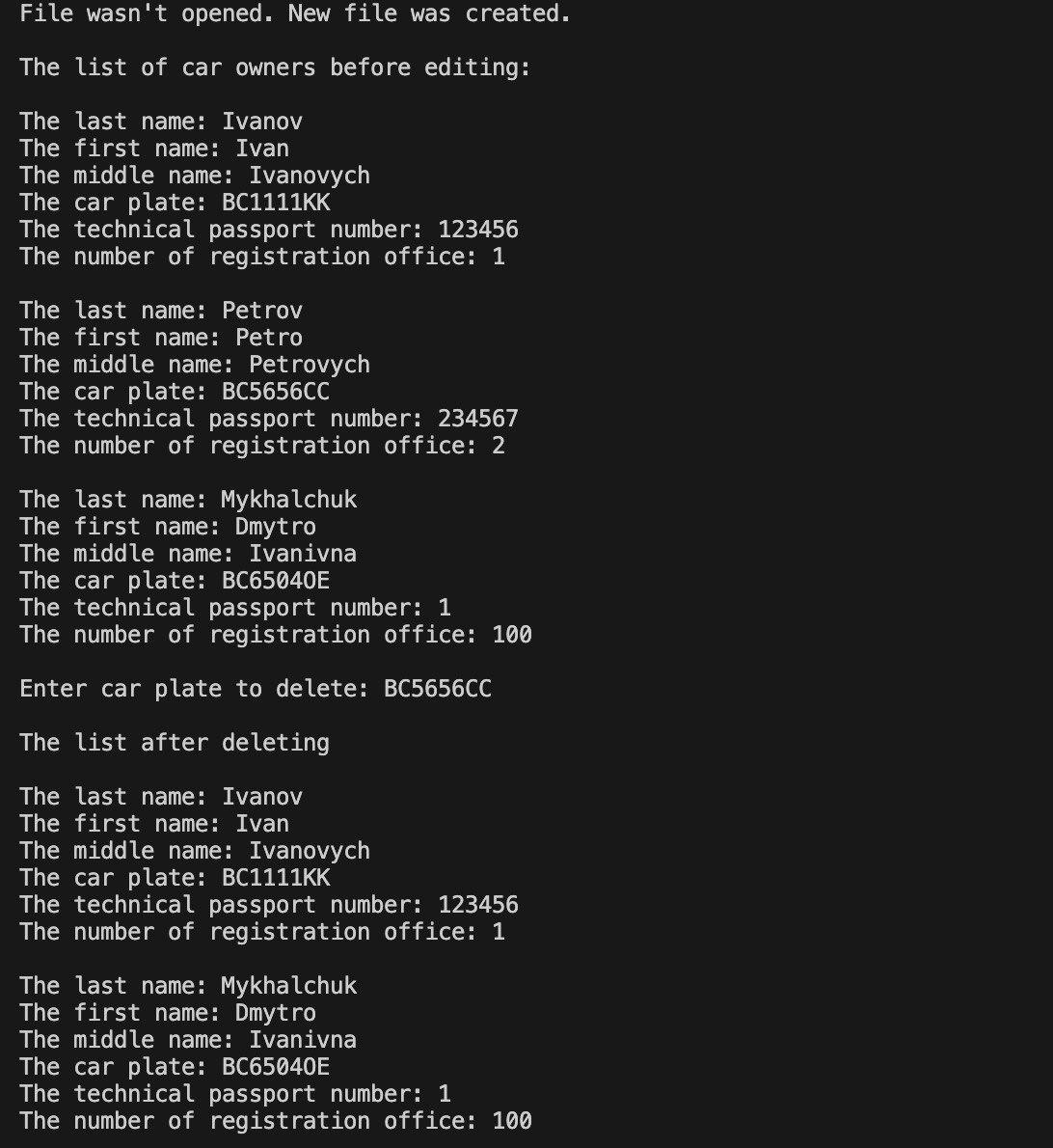
## **4. Результати виконання завдань, тестування та фактично затрачений час:**

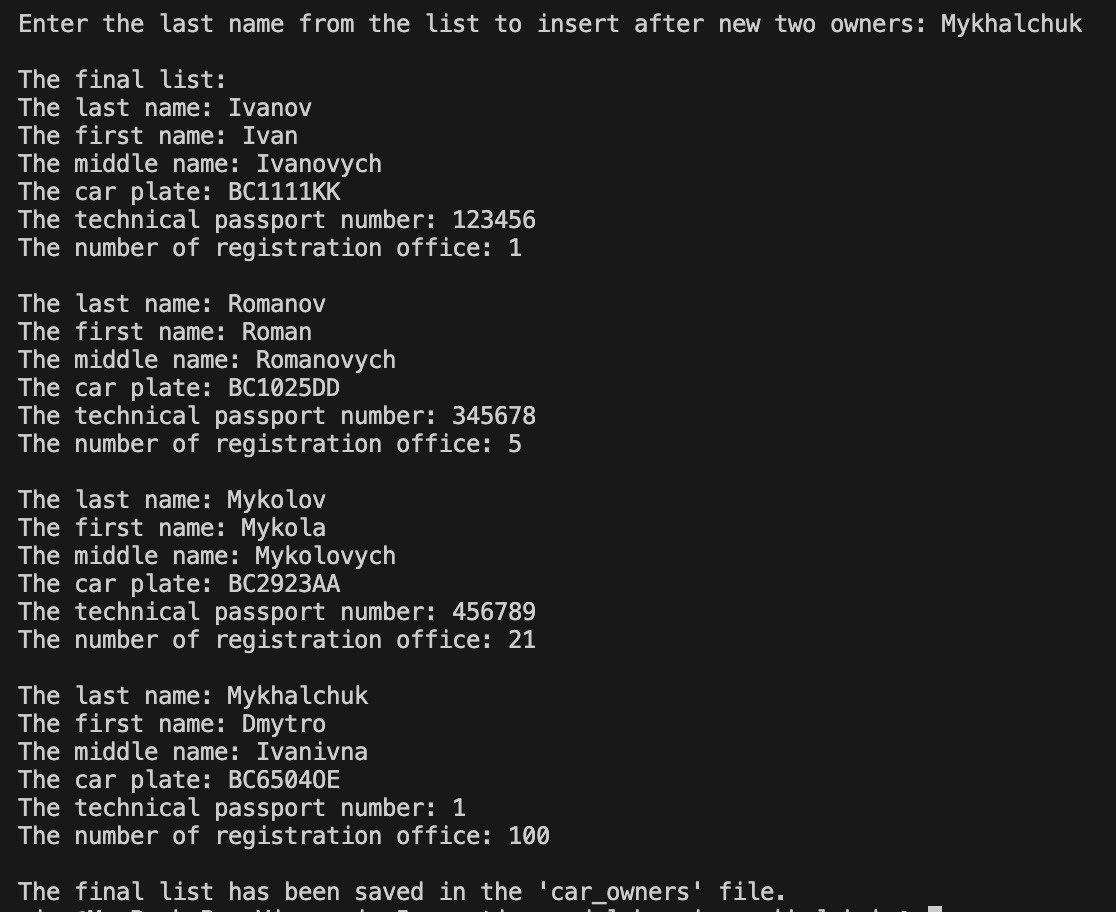
Завдання №1 VNS Lab 6

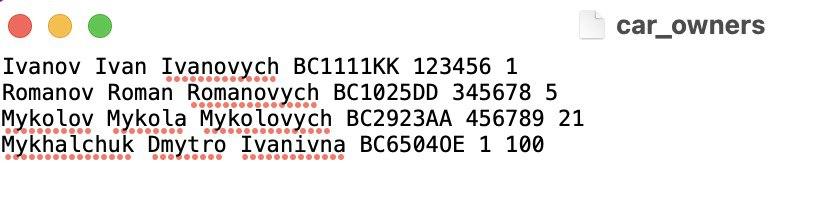


Час затрачений на виконання завдання 40 хв

Завдання №2 VNS Lab 8

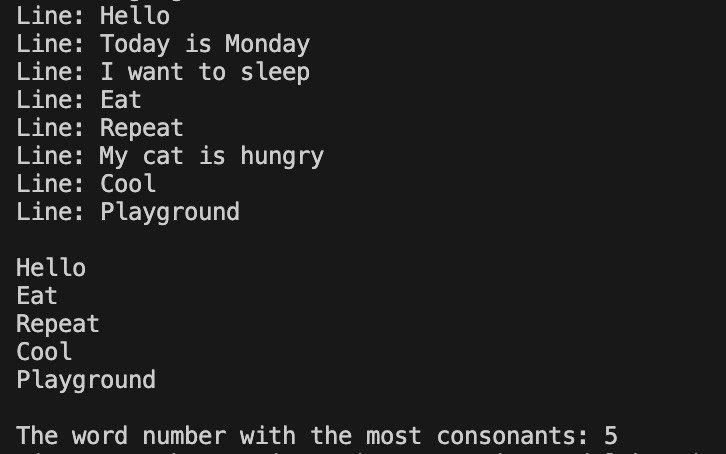


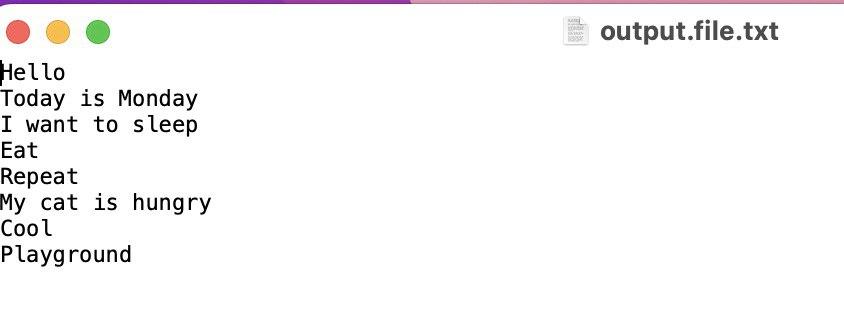




Час затрачений на виконання завдання 1 година

Завдання №3 VNS Lab 9







Час затрачений на виконання завдання 1.5 години

Завдання №4 Algotester Lab 4



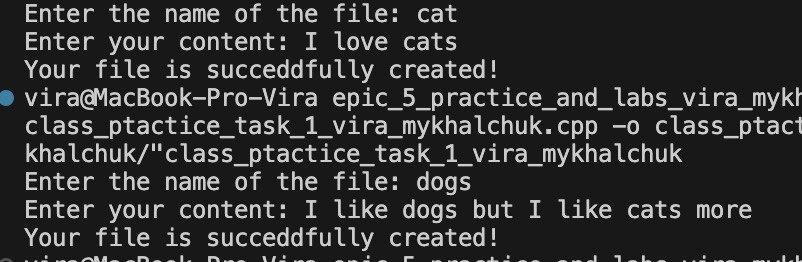
Час затрачений на виконання завдання 1.5 години

Завдання №5 Algotester Lab 6

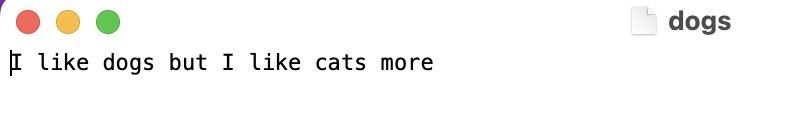


Час затрачений на виконання завдання 1.5 години

Завдання №6 Class Practice Work 1

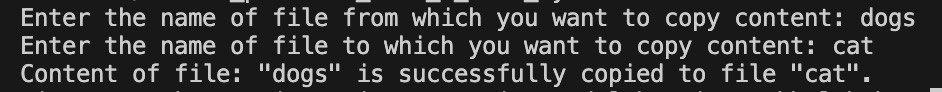


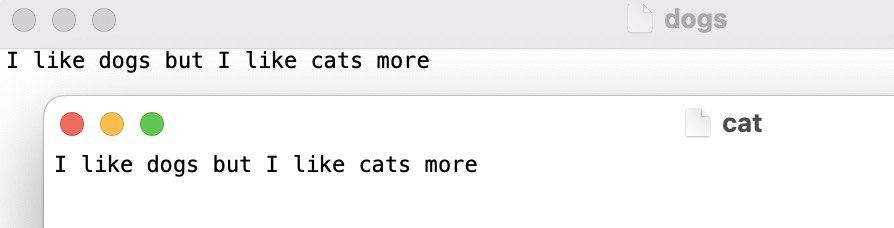




Час затрачений на виконання завдання 1 година

Завдання №7 Class Practice Work 2





Час затрачений на виконання завдання 1 година

# **Висновки:**

Деталі по результатам виконання робіт та висновки згідно тем та завдань