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:*** «Epic 5. Файли. Системи числення. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек»

***Виконав:***

студент групи ШІ-14

Михальчишин Лук’ян Павлович

**Тема роботи:**

Робота з файлами, створення бінарних файлів і робота з ними, також робота з таки бібліотеками як “fstream”

**Мета роботи:**

Навчитись працювати з бібліотекою “fstream”, також навчитись працювати з файлами для різних потреб користувача, розуміти що таке символьний і рядковий тип змінних

**Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Theory Education Activities
* Тема №2:  Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-9)
* Тема №3: Lab# programming: VNS Lab 6
* Тема №4: Lab# programming: VNS Lab 8
* Тема №5  Lab# programming: VNS Lab 9
* Тема №6 Lab# programming: Algotester Lab 4
* Тема №7 Lab# programming: Algotester Lab 6
* Тема №8 Practice# programming: Class Practice Task
* Тема №9 Practice# programming: Self Practice Task

1. Індивідуальний план опрацювання теорії:

* **Тема №1: Theory Education Activities**

1. Джерела Інформації

Статті: бібліотека <fstream> <https://purecodecpp.com/archives/2751>

Рядкові типи змінних: <https://studfile.net/preview/7079855/page:8/>

Символьні типи змінних: <https://studfile.net/preview/7079855/page:7/>

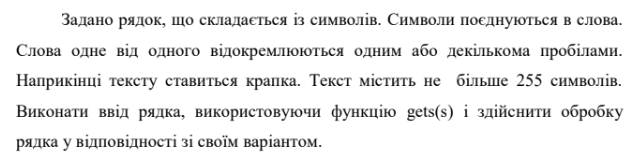
Курс: enum - <https://youtu.be/ifdk1fYSk1Y?si=FgJ_d3WcWUA15hYJ>

Структури - <https://youtu.be/999IE-6b7_s?si=u5E_zH_SRS_F7ChH>

1. Що опрацьовано: Що таке бібліотека <fstream>, як використовувати файли для реалізації завдань, для чого потрібні символьні і рядкові типи змінних, що таке enum, використання структур
2. Статус: Ознайомлений

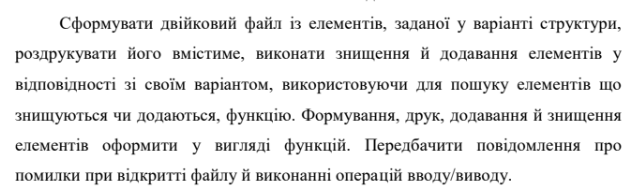
**Виконання роботи:**

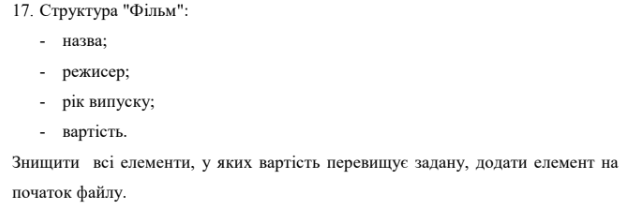
* 1. **Опрацювання завдання та вимог до програм та середовища:**
* Task 2 - Requirements management and design activities with Draw.io and Google Docs
  + Створити блок-схему для 1 з програмного коду
* Task №3: Lab# programming: VNS Lab 6



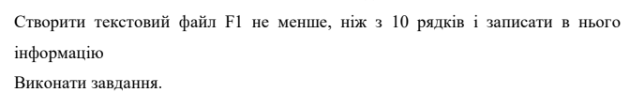


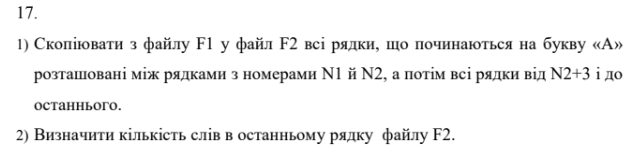
* Task №4: Lab# programming: VNS Lab 8



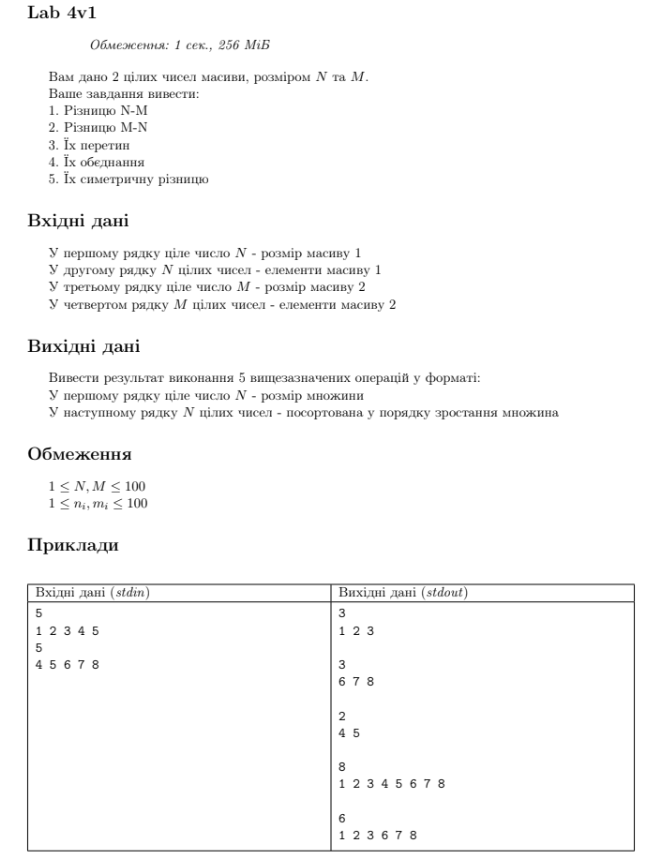


* Task №5  Lab# programming: VNS Lab 9





* Task №6 Lab# programming: Algotester Lab 4

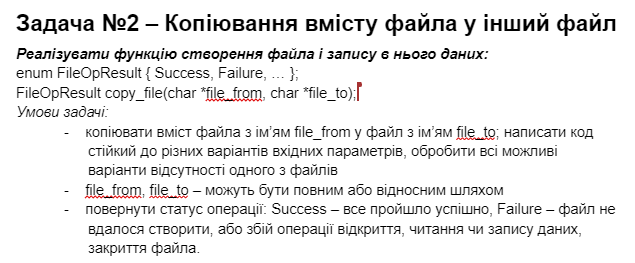


* Task №7 Lab# programming: Algotester Lab 6



* Task №8 Practice# programming: Class Practice Task





* Task №9 Practice# programming: Self Practice Task

Створити файл, в якому буде виводитись вся інформація про користувача, яку він про себе надав

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

Блок-схема

* Task №6 Lab# programming: Algotester Lab 4

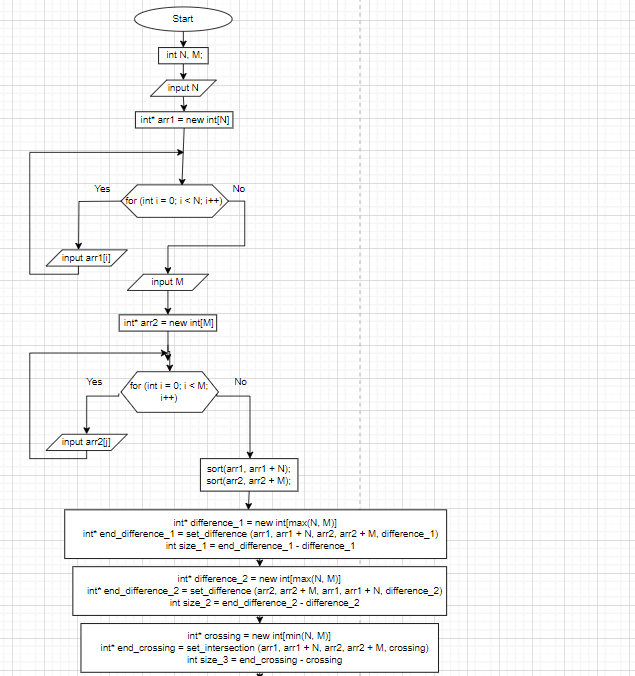


Рисунок 1: Блок схема до програми №1

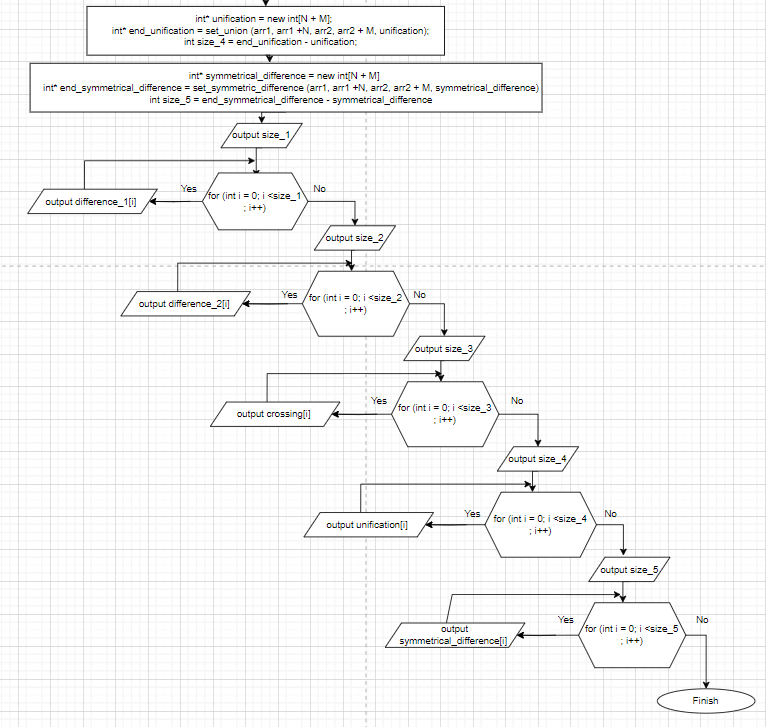
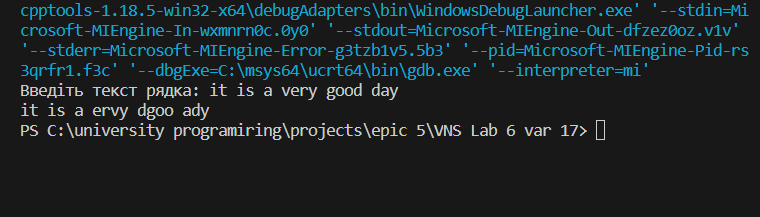
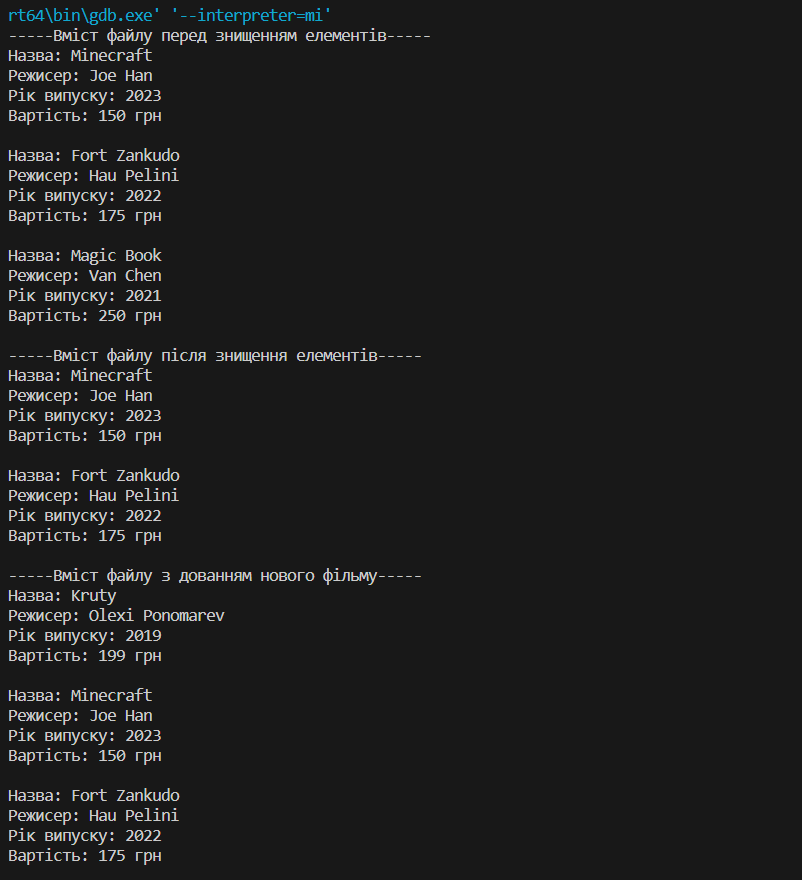


Рисунок 2: Блок схема до програми №1

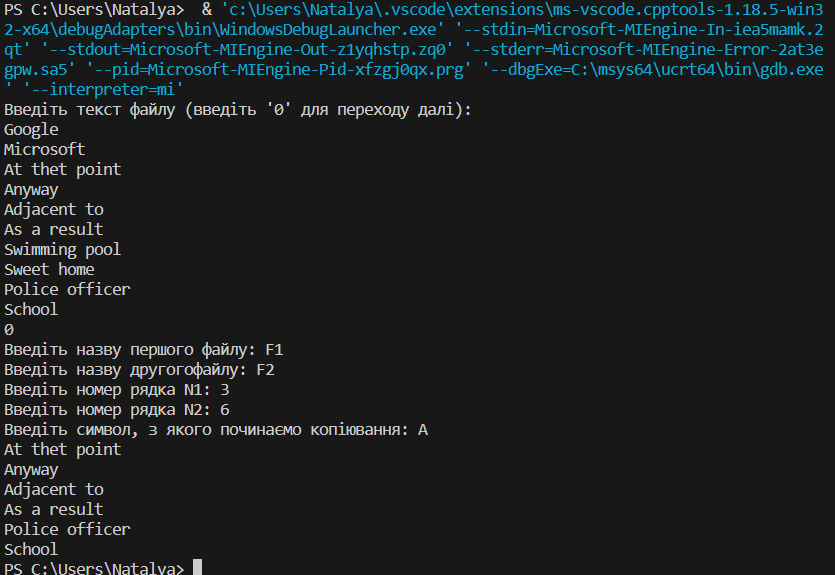
* 1. **Код програм з посиланням на зовнішні ресурси:**
* Task №3: Lab# programming: VNS Lab 6
* #include <iostream>
* #include <string>
* #include <algorithm>
* using namespace std;
* int main()
* {
* cout << "Введіть текст рядка: ";
* char simvol;
* string word;
* while (cin.get(simvol))
* {
* if (simvol == ' ' || simvol == '\n')
* {
* sort (word.begin(), word.end());
* cout << word << " ";
* word.clear();
* }
* else
* {
* word += simvol;
* }
* if (simvol == '\n')
* {
* break;
* }
* }
* return 0;
* }
* Task №4: Lab# programming: VNS Lab 8
* #include <iostream>
* #include <fstream>
* #include <vector>
* #include <algorithm>
* using namespace std;
* struct Film
* {
* char title [150];
* char director [150];
* int year;
* int cost;
* };
* void print\_film (const Film& film)
* {
* cout << "Назва: " << film.title << endl;
* cout << "Режисер: " << film.director << endl;
* cout << "Рік випуску: " << film.year << endl;
* cout << "Вартість: " << film.cost << " грн\n\n";
* }
* void file\_information (const string& filename)
* {
* ifstream file (filename, ios::binary); //відкриття бінарного файлу
* if (!file.is\_open())
* {
* cerr << "Помилка відкриття файлу для читання\n";
* return;
* }
* Film film; //створення Film для збережених даних які вже зчитані
* while (file.read (reinterpret\_cast <char\*> (&film), sizeof(Film)))
* {
* print\_film (film);
* }
* file.close();
* }
* void add\_new\_movie (const string& filename, const Film& newFilm)
* {
* vector <Film> films; //зберігаєм фільми в векторі
* ifstream inFile (filename, ios::binary);
* if (inFile.is\_open())
* {
* Film currentFilm; //зчитуєм фільми які вже є в вектор
* while (inFile.read (reinterpret\_cast <char\*> (&currentFilm), sizeof (Film)))
* {
* films.push\_back (currentFilm);
* }
* inFile.close();
* }
* // Додаєм новий фільм на початок вектору
* films.insert (films.begin(), newFilm);
* // перезапис файлу
* ofstream outFile (filename, ios::binary | ios::trunc);
* if (outFile.is\_open())
* {
* for (const Film& film : films) // записуєм дані з вектору в файл
* {
* outFile.write (reinterpret\_cast <const char\*> (&film), sizeof (Film));
* }
* outFile.close();
* }
* else
* {
* cerr << "Помилка відкриття файлу для запису\n";
* }
* }
* void deleting\_Movie (const string& filename, int max\_cost\_film)
* {
* ifstream inFile (filename, ios::binary);
* if (!inFile.is\_open())
* {
* cerr << "Помилка відкриття файлу для читання\n";
* return;
* }
* vector <Film> films; //створення вектора в яку збережем обов'язкові фільми
* Film currentFilm;
* while (inFile.read (reinterpret\_cast <char\*> (&currentFilm), sizeof (Film))) //додавання  фільмів у вектор якщо не перевищено ціну
* {
* if (currentFilm.cost <= max\_cost\_film)
* {
* films.push\_back (currentFilm);
* }
* }
* inFile.close();
* ofstream outputFile(filename, ios::binary | ios::trunc); // перезаписуєм файл
* if (!outputFile.is\_open())
* {
* cerr << "Помилка відкриття файлу для запису\n";
* return;
* }
* for (const Film& film : films) // запис фільму що лишився
* {
* outputFile.write (reinterpret\_cast <const char\*> (&film), sizeof (Film));
* }
* outputFile.close();
* }
* int main() {
* const string filename = "films.bin";
* vector<Film> films;
* Film f1 = {"Minecraft", "Joe Han", 2023, 150};
* Film f2 = {"Fort Zankudo", "Hau Pelini", 2022, 175};
* Film f3 = {"Magic Book", "Van Chen", 2021, 250};
* films.push\_back(f1); //додавання Film до вектору films
* films.push\_back(f2);
* films.push\_back(f3);
* // Запис даних в файл
* ofstream file (filename, ios::binary | ios::trunc);
* if (file.is\_open())
* {
* for (const Film& film : films)
* {
* file.write (reinterpret\_cast <const char\*> (&film), sizeof (Film)); //запис даних про кожен фільм
* }
* file.close();
* }
* else
* {
* cerr << "Помилка відкриття файлу для запису\n";
* return 1;
* }
* cout << "-----Вміст файлу перед знищенням елементів-----\n";
* file\_information (filename);
* int max\_cost\_film = 240;
* // Знищення елементів
* deleting\_Movie (filename, max\_cost\_film);
* cout << "-----Вміст файлу після знищення елементів-----\n";
* file\_information (filename);
* // Новий фільм, який додається на початок файлу
* Film newFilm = {"Kruty", "Olexi Ponomarev", 2019, 199};
* // Додавання нового фільму на початок файлу
* add\_new\_movie (filename, newFilm);
* cout << "-----Вміст файлу з дованням нового фільму-----\n";
* file\_information (filename);
* return 0;
* }
* Task №5  Lab# programming: VNS Lab 9
* #include <iostream>
* #include <fstream>
* #include <string>
* #include <vector>
* using namespace std;
* void copying (const string& enter\_filename, const string& output\_filename, char start\_copy, int raw\_1, int raw\_2)
* {
* ifstream enter\_file (enter\_filename);
* ofstream outputFile (output\_filename);
* if (!enter\_file.is\_open() || !outputFile.is\_open())
* {
* cerr << "Помилка відкриття файлів." << endl;
* return;
* }
* vector <string> lines;
* string line;
* // Читаємо всі рядки з файлу
* while (getline (enter\_file, line))
* {
* lines.push\_back(line);
* }
* // Копіюємо рядки, які починаються на 'A' і розташовані між 1 і N2
* for (int i = raw\_1 - 1; i < raw\_2 && i < lines.size(); ++i)
* {
* if (!lines[i].empty() && lines[i][0] == start\_copy)
* {
* cout << lines[i] << endl;
* outputFile << lines[i] << endl;
* }
* }
* // Додаємо всі рядки від N2+3 до останнього
* for (int i = raw\_2 + 2; i < lines.size(); ++i)
* {
* cout << lines[i] << endl;
* outputFile << lines[i] << endl;
* }
* enter\_file.close();
* outputFile.close();
* }
* int main()
* {
* string enter\_text;
* string output\_filename\_1, output\_filename\_2;
* char start\_copy;
* int raw\_1, raw\_2;
* cout << "Введіть текст файлу (введіть '0' для переходу далі):\n";
* // Ввід тексту для файла
* while (true)
* {
* string line;
* getline(cin, line);
* if (line == "0")
* {
* break;
* }
* enter\_text += line + '\n';
* }
* cout << "Введіть назву першого файлу: ";
* cin >> output\_filename\_1;
* // Збереження введеного тексту в перший файл
* ofstream inputFile(output\_filename\_1);
* inputFile << enter\_text;
* inputFile.close();
* cout << "Введіть назву другогофайлу: ";
* cin >> output\_filename\_2;
* // Введення рядка N1
* cout << "Введіть номер рядка N1: ";
* cin >> raw\_1;
* // Введення рядка N2
* cout << "Введіть номер рядка N2: ";
* cin >> raw\_2;
* // Введення символу, з якого починаємо копіювання
* cout << "Введіть символ, з якого починаємо копіювання: ";
* cin >> start\_copy;
* // Викликаємо функцію для копіювання та додавання рядків
* copying (output\_filename\_1, output\_filename\_2, start\_copy, raw\_1, raw\_2);
* return 0;
* }
* Task №6 Lab# programming: Algotester Lab 4
* #include <iostream>
* #include <algorithm>
* using namespace std;
* int main()
* {
* int N, M;
* cin >> N;
* int\* arr1 = new int[N];
* for (int i = 0; i < N; i++)
* {
* cin >> arr1[i];
* }
* cout << endl;
* cin >> M;
* int\* arr2 = new int[M];
* for (int i = 0; i < M; i++)
* {
* cin >> arr2[i];
* }
* sort(arr1, arr1 + N);
* sort(arr2, arr2 + M);
* int\* difference\_1 = new int[max(N, M)];
* int\* end\_difference\_1 = set\_difference (arr1, arr1 + N, arr2, arr2 + M, difference\_1);
* int size\_1 = end\_difference\_1 - difference\_1;
* int\* difference\_2 = new int[max(N, M)];
* int\* end\_difference\_2 = set\_difference (arr2, arr2 + M, arr1, arr1 + N, difference\_2);
* int size\_2 = end\_difference\_2 - difference\_2;
* int\* crossing = new int[min(N, M)];
* int\* end\_crossing = set\_intersection (arr1, arr1 + N, arr2, arr2 + M, crossing);
* int size\_3 = end\_crossing - crossing;
* int\* unification = new int[N + M];
* int\* end\_unification = set\_union (arr1, arr1 +N, arr2, arr2 + M, unification);
* int size\_4 = end\_unification - unification;
* int\* symmetrical\_difference = new int[N + M];
* int\* end\_symmetrical\_difference = set\_symmetric\_difference (arr1, arr1 +N, arr2, arr2 + M, symmetrical\_difference);
* int size\_5 = end\_symmetrical\_difference - symmetrical\_difference;
* cout << endl << size\_1 << endl;
* for (int i = 0; i < size\_1; i++)
* {
* cout << difference\_1[i] << " ";
* }
* cout << endl << endl << size\_2 << endl;
* for (int i = 0; i < size\_2; i++)
* {
* cout << difference\_2[i] << " ";
* }
* cout << endl << endl << size\_3 << endl;
* for (int i = 0; i < size\_3; i++)
* {
* cout << crossing[i] << " ";
* }
* cout << endl << endl << size\_4 << endl;
* for (int i = 0; i < size\_4; i++)
* {
* cout << unification[i] << " ";
* }
* cout << endl << endl << size\_5 << endl;
* for (int i = 0; i < size\_5; i++)
* {
* cout << symmetrical\_difference[i] << " ";
* }
* delete []arr1;
* delete []arr2;
* delete []difference\_1;
* delete []difference\_2;
* delete []crossing;
* delete []unification;
* delete []symmetrical\_difference;
* return 0;
* }
* Task №7 Lab# programming: Algotester Lab 6
* #include <iostream>
* #include <vector>
* #include <string>
* #include <map>
* #include <set>
* #include <cctype>
* #include <algorithm>
* using namespace std;
* int main()
* {
* int all\_words, min\_words;
* cin >> all\_words;
* cin >> min\_words;
* map <string, int> words\_amount; // структура words\_amount, яка зберігає кількість слів
* vector <string> words (all\_words);
* for (int i = 0; i < all\_words; ++i)
* {
* cin >> words[i];
* transform (words[i].begin(), words[i].end(), words[i].begin(), ::tolower); //перетворення слова в нижній регістр
* words\_amount[words[i]]++;
* }
* set <char> unique\_letters;
* for (const auto & word : words)
* {
* if (words\_amount[word] >= min\_words) // додавання букв з слів, які зустрілись min\_words разів
* {
* unique\_letters.insert (word.begin(), word.end());
* }
* }
* if (unique\_letters.empty())
* {
* cout << "Empty!\n";
* }
* else
* {
* cout << unique\_letters.size() << endl;
* for (auto reverse = unique\_letters.rbegin(); reverse != unique\_letters.rend(); ++ reverse) // зворотній порядок виводу
* {
* cout << \*reverse;
* if (next (reverse) != unique\_letters.rend())
* {
* cout << " ";
* }
* }
* cout << endl;
* }
* return 0;
* }
* Task №8 Practice# programming: Class Practice Task
* #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())
* {
* return Failure;
* }
* file << content;
* if (!file)
* {
* file.close();
* return Failure;
* }
* file.close();
* return Success;
* }
* FileOpResult copy\_file (const char \*file\_from, const char \*file\_to)
* {
* ifstream source (file\_from, ios::binary);
* ofstream end (file\_to, ios::binary);
* if (!source. is\_open() || !end.is\_open())
* {
* return Failure;
* }
* end << source.rdbuf();
* if (!end)
* {
* source.close();
* end.close();
* return Failure;
* }
* source.close();
* end.close();
* return Success;
* }
* int main()
* {
* char file\_name [40], file\_content [500];
* char file\_from [500], file\_to [500];
* cout << "-----Ім'я файлу повинно мати 40 символів!-----\n";
* cout << "-----Текст повиннен містити 500 символів!-----\n\n";
* cout << "Введіть назву файла - ";
* cin.getline (file\_name, sizeof(file\_name));
* cout << "Введіть текст, для збереження в файлі - ";
* cin.getline (file\_content, sizeof(file\_content));
* FileOpResult result\_1 = write\_to\_file (file\_name, file\_content);
* cout << endl << (result\_1 == Success ? "Все Збережено" : "Помилка збереження");
* cout << endl << endl << "-----Процес копіювання файлу!-----\n" << endl;
* cout << "Введіть шлях до файлу який потрібно скопіювати - ";
* cin.getline (file\_from, sizeof (file\_from));
* cout << "Введіть шлях до нового файлу - ";
* cin.getline (file\_to, sizeof (file\_to));
* FileOpResult result\_2 = copy\_file (file\_from, file\_to);
* cout << (result\_2 == Success ? "Скопійовано успішно" : "Помилка копіювання");
* return 0;
* }
* Task №9 Practice# programming: Self Practice Task
* #include <iostream>
* #include <string>
* #include <fstream>
* using namespace std;
* struct Person
* {
* string name;
* int age;
* string work;
* int sallary;
* string preferences;
* };
* int main()
* {
* Person user;
* cout << "Введіть своє ім'я: ";
* getline(cin, user.name);
* cout << "Введіть свій вік: ";
* cin >> user.age;
* cin.ignore();
* cout << "Введіть ким ви працюєте: ";
* getline(cin, user.work);
* cout << "Введіть свою зарплату: ";
* cin >> user.sallary;
* cin.ignore();
* cout << "Введіть свої уподобання: ";
* getline(cin, user.preferences);
* cout << endl << "------ Дані користувача ------";
* cout << endl << "Ім'я: " << user.name << endl;
* cout << "Вік: " << user.age << endl;
* cout << "Робота: " << user.work << endl;
* cout << "Зарплата: " << user.sallary << " $ " << endl;
* cout << "Уподобання: " << user.preferences << endl;
* ofstream File ("user information.txt");
* if (File.is\_open())
* {
* File << "Ім'я: " << user.name << endl;
* File << "Вік: " << user.age << endl;
* File << "Робота: " << user.work << endl;
* File << "Зарплата: " << user.sallary << " $ " << endl;
* File << "Уподобання: " << user.preferences << endl;
* File.close();
* cout << "Все збережено в File.txt\n";
* }
* else
* {
* cout << "Помилка відкриття File.txt\n";
* }
* return 0;
* }
  1. **Результати виконання завдань, тестування та фактично затрачений час:**
* Task №3: Lab# programming: VNS Lab 6

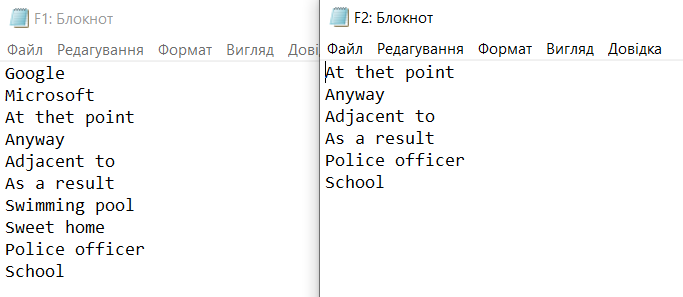


* Task №4: Lab# programming: VNS Lab 8

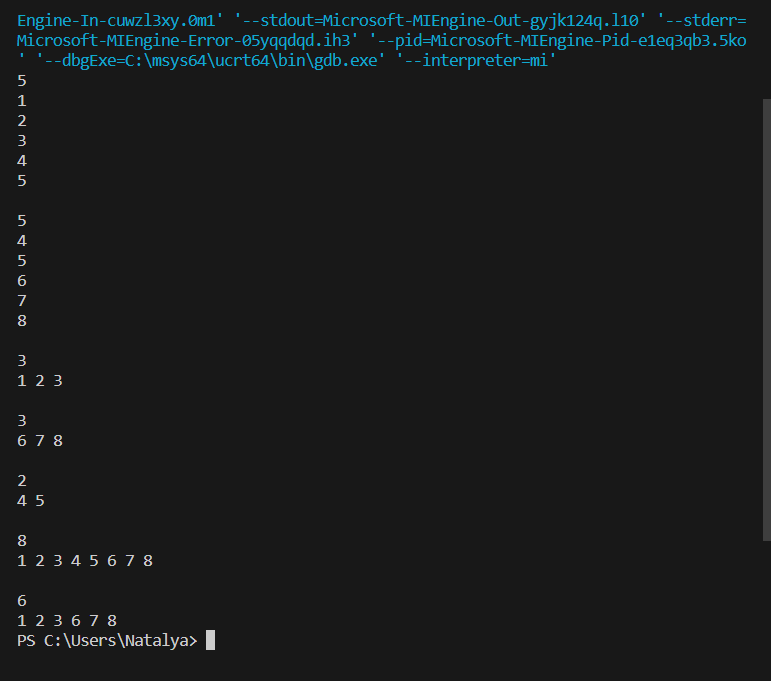


* Task №5  Lab# programming: VNS Lab 9

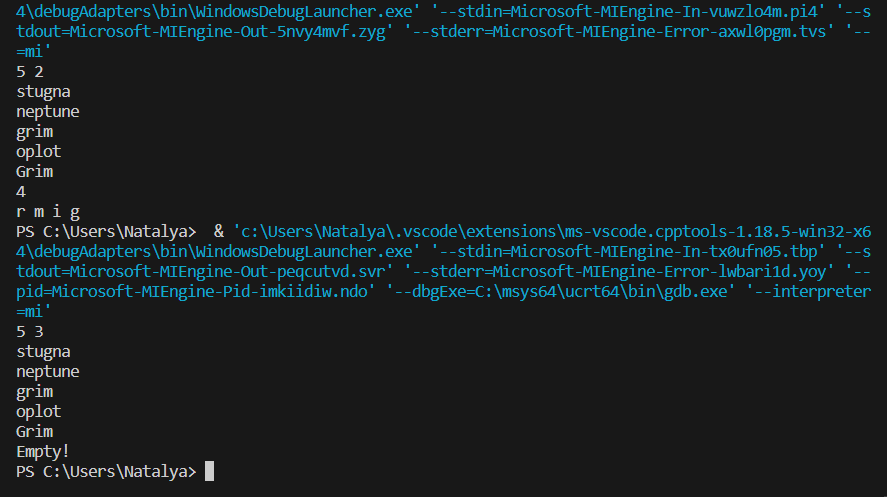




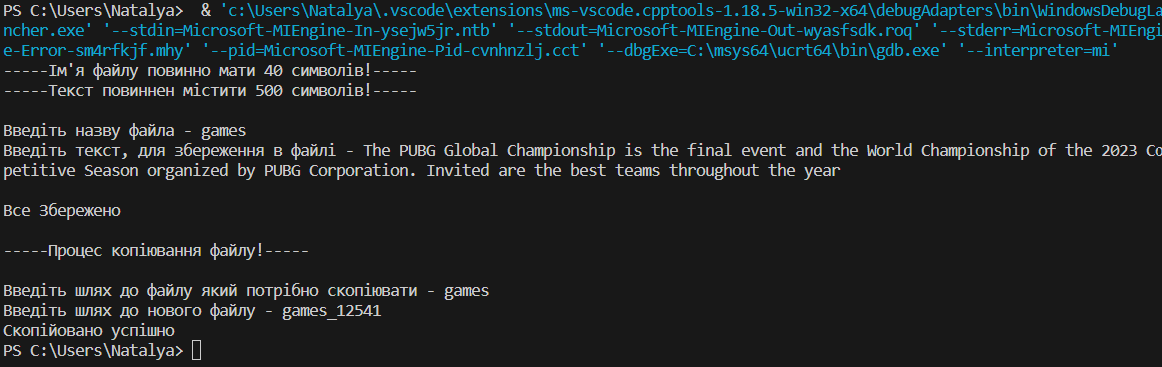
* Task №6 Lab# programming: Algotester Lab 4

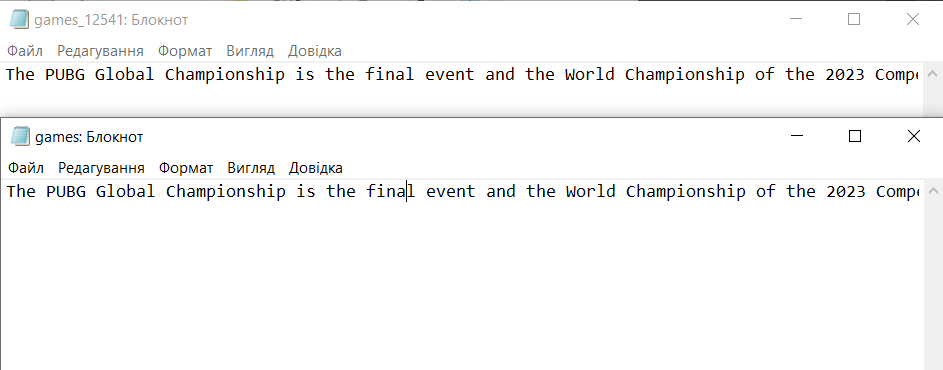


* Task №7 Lab# programming: Algotester Lab 6

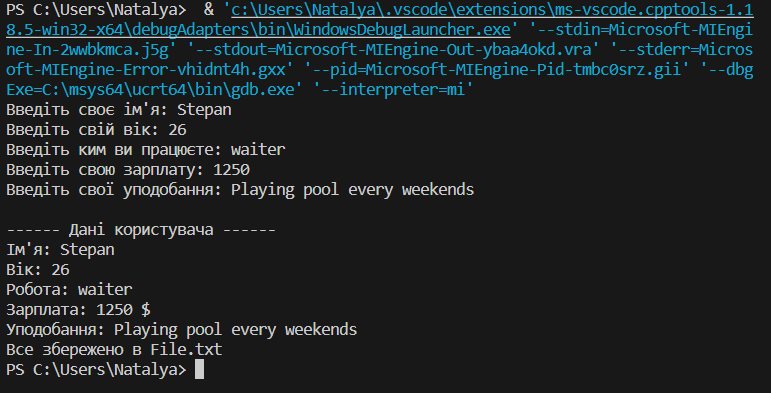


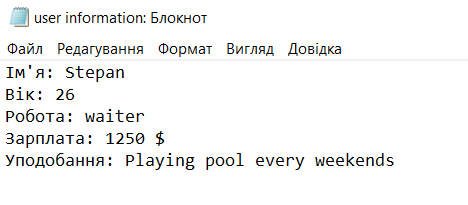
* Task №8 Practice# programming: Class Practice Task



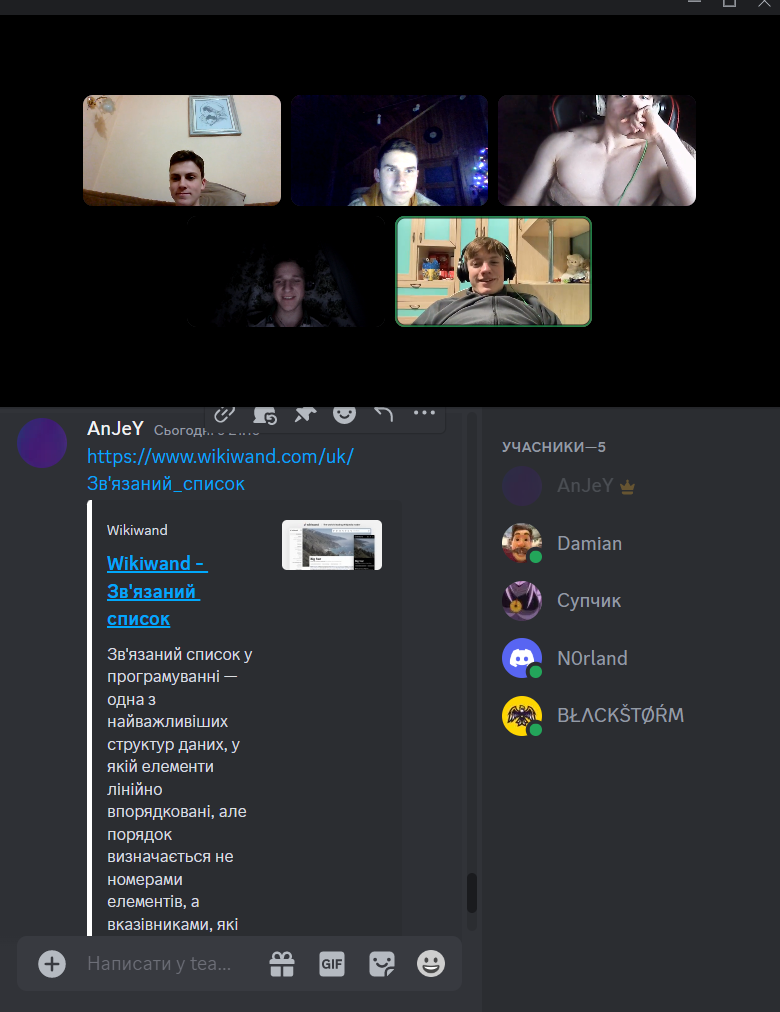


* Task №9 Practice# programming: Self Practice Task





* 1. **Онлайн зустріч з командою**

****

**Посилання на pull request**

[**https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/884**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/884)

**Висновки:** Навчився використовувати файли для реалізації програм під поставленні задачі, також розібрався для чого потрібні символьні і рядкові типи змінних, взнав як працює бібліотека <fstream>. Зрозуміти як структури спрощують програмування