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. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.»

***Виконала:***

Студентка групи ШІ-13

Жмуд Анастасія

# **Тема роботи:**

Робота із двійковими файлами. Ввід-вивід рядків.

# **Мета роботи:**

Ознайомитись з організацією вводу-виводу структурованої інформації із зовнішніх носіїв. Навчитися працювати з текстовими файлами, ввід-вивід текстової інформації і її зберігання на зовнішніх носіях.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Потоковий ввід-вивід
* Тема №2: Блоковий ввід-вивід
* Тема №3: Вектори в С++

1. Індивідуальний план опрацювання теорії:

* Тема №1: Потоковий ввід-вивід.
  + Джерела Інформації
    - <http://cpp.dp.ua/potokove-vvedennya-vyvedennya/>
    - <https://acode.com.ua/urok-215-potoky-vvodu-i-vyvodu/>
    - http://www.kievoit.ippo.kubg.edu.ua/kievoit/2016/47\_C++/index.html
  + Що опрацьовано:
    - Ввід/вивід в C++
    - Стандартні потоки С++
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.12.23
  + Звершення опрацювання теми: 02.12.23
* Тема №2: Блоковий ввід-вивід
  + Джерела Інформації:
    - http://um.co.ua/3/3-17/3-176444.html
  + Що опрацьовано:
    - Функції, що використовуються для блочного вводу та виводу
  + Статус: Ознайомлений
  + Початок опрацювання теми: 02.12.23
  + Звершення опрацювання теми: 02.11.23

* Тема №3: Вектори в С++
  + Джерела Інформації:
    - http://www.kytok.org.ua/post/vector-u-cplusplus
    - https://www.bestprog.net/uk/2021/10/12/c-the-vector-class-methods-that-define-and-modify-the-general-characteristics-of-array-ua/
    - https://acode.com.ua/urok-99-vvedennya-v-std-vector/
  + Що опрацьовано:
    - Інформація про базові операції з векторами
    - Зміна елементів вектора та доступ до них
  + Статус: Ознайомлений
  + Початок опрацювання теми: 03.12.23
  + Звершення опрацювання теми: 04.12.23

**Виконання роботи:**

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

**Завдання №1 Опрацювання теорії**

* Деталі завдання:

Опрацювання різноманітних матеріалів, аналіз та дослідження відео, статей, книг на задані теми. Організація та структурування отриманих даних для можливості ефективніше засвоювати отримані знання на практиці. Вивчення найважливіших моментів.

**Завдання №2 Перегляд вимог та проектування**

* Деталі завдання

Введення завдань в Trello, перегляд задач, та створення математичної моделі. Проектування блок-схем в Draw.io.

**Завдання №3 VNS Lab 6**

* Варіант завдання: 23
* Деталі завдання :

Задано рядок, що складається із символів. Символи поєднуються в слова.

Слова одне від одного відокремлюються одним або декількома пробілами.

Наприкінці тексту ставиться крапка. Текст містить не більше 255 символів.

**Завдання №4 VNS Lab 8**

* Варіант завдання: 23
* Деталі завдання:

Сформувати двійковий файл із елементів, заданої у варіанті структури,

роздрукувати його вмістиме, виконати знищення й додавання елементів у

відповідності зі своїм варіантом.

**Завдання №5 VNS Lab 9**

* Варіант завдання: 23
* Деталі завдання :

Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього інформацію. Виконати завдання задане у варіанті.

**Завдання №6 Algotester Lab 4**

* Варіант завдання: 1
* Деталі завдання:

Вам дано 2 цiлих чисел масиви, розмiром N та M.

Ваше завдання вивести:1. Рiзницю N-M; 2. Рiзницю M-N; 3. Їх перетин; 4. Їх обєднання; 5. Їх симетричну рiзницю.

- Важливі деталі для врахування при імплементації програми:

Для розв’язку необхідно використовувати засоби STL

**Завдання №7 Algotester Lab 6**

* Варіант завдання: 1
* Деталі завдання :

Вам дано N слiв та число K. Ваше завдання перечислити букви в словах, якi зустрiчаються в текстi бiльше-рiвне нiж K разiв

- Важливі деталі для врахування при імплементації програми:

Великi та маленькі букви вважаються однаковими, виводити необхiдно малi, посортованi вiд останньої до першої у алфавiтi.

**Завдання №8 Class Practice Task**

* Деталі завдання :

1) Реалізувати функцію створення файла і запису в нього даних

2) Реалізувати функцію створення файла і запису в нього даних

**Завдання №9 Self Practice Task**

**-** Деталі завдання:

Нарешті ми його дочекалися, 256-го дня в році (дня програміста)... Кожен зі студентів факультету прикладної математики та інформатики святкували його по-різному: дехто вдома, дехто в «Picasso», дехто в гуртожитку... Не дивно, що Зеник із Марічкою залишилися святкувати його в гуртожитку. Вони організували «mega party». Свято вдалося...

Наступного дня, гуляючи з Марічкою, Зенику стало цікаво, скільки ж було випито різної випивки?!

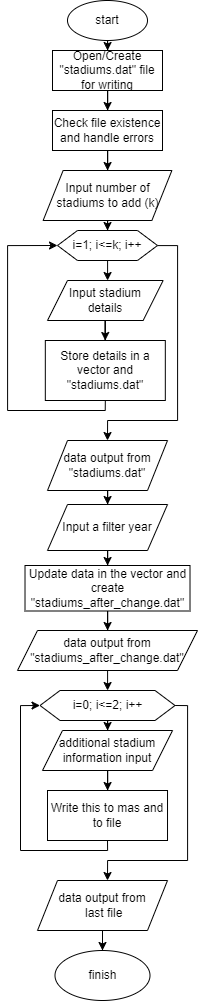
Марічка змогла згадати n назв випивок, а Зеник — m.

Вам потрібно написати програму, яка порахує скільки ж було різної випивки на святі, яку змогли згадати Зеник з Марічкою.

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

**Програма №1 VNS Lab 8**

* Блок-схема:



* Планований час на реалізацію: 6 годин

**Програма №2 VNS Lab 9**

* Планований час на реалізацію: 3 години

**Програма №3 Algotester Lab 4**

* Планований час на реалізацію: 3 години
* Важливі деталі для врахування в імплементації: Для розв’язку необхідно використовувати засоби STL

**Програма №4 VNS Lab 6**

* Планований час на реалізацію: 5 годин

**Програма №5 Algotester Lab 6**

* Планований час на реалізацію: 5 годин

- Важливі деталі для врахування в імплементації:

Великi та маленькі букви вважаються однаковими, виводити необхiдно малi, посортованi вiд останньої до першої у алфавiтi.

**Програма №6 Class Practice Task**

* Планований час на реалізацію: 3 години

**Програма №7 Self Practice Task**

- Планований час на реалізацію: 1 година

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

**Завдання №1 VNS Lab 8**

#include<iostream>

#include<string>

#include<vector>

#include<algorithm>

using namespace std;

 struct stadium

 {

    string name;

    int year;

    int playgrounds;

    string kindofsport;

 };

 int main(){

     int k=0,n=0, elements=0;

     stadium m;

     FILE\* sdiums;

     sdiums = fopen("stadiums.dat", "wb");

     if ((sdiums=fopen("stadiums.dat", "wb"))==NULL){

      cout<<"Eror!";

      exit(1);

     }

     cout<<"How much stadiums do you want to add?   ";

     cin>>k;

     vector<stadium> mas;

     for(int i=1; i<=k; i++){

      cout<<"Stadium "<<i<<endl;

      cout<<"Enter a name:  \n";

      cin>>m.name;

      cout<<"Enter a year:   \n";

      cin>>m.year;

      cout<<"Enter a number of playgrounds:   \n";

      cin>>m.playgrounds;

      cout<<"Enter a kind of sports:   \n";

      cin.ignore();

      getline(cin, m.kindofsport);

      mas.push\_back(m);

      fwrite(&m, sizeof(stadium), 1, sdiums);

        if (ferror(sdiums)) {

            exit(1);

        }

     }

     fclose(sdiums);

     sdiums = fopen("stadiums.dat", "rb");

     if ((sdiums=fopen("stadiums.dat", "rb"))==NULL){

      cout<<"Eror!";

      exit(1);}

     for(int i=0; i<k; i++){

      fread(&m, sizeof(stadium), 1, sdiums);

      cout<<"Stadium "<<i+1<<endl;

      cout<<mas[i].name<<endl;

      cout<<mas[i].year<<endl;

      cout<<mas[i].playgrounds<<endl;

      cout<<mas[i].kindofsport<<endl<<endl;

     }

     fclose(sdiums);

     int filtr=0;

     cout<<"Enter the year from which you want to see the stadiums:  ";

     cin>>filtr;

     for(int i=0; i<k; i++){

      if(mas[i].year<filtr){

            mas.erase(mas.begin() + i);

            k--;

            i--;

            n++;

      }

     }

      FILE\* sdiums\_after1;

      sdiums\_after1=fopen("stadiums\_after\_change.dat", "wb");

      if ((sdiums\_after1=fopen("stadiums\_after\_change.dat", "wb"))==NULL){

      cout<<"Eror!";

      exit(1);}

      int ns = mas.size();

      for (int i = 0; i <ns; i++) {

        fwrite(&m, sizeof(stadium), 1, sdiums\_after1);

      }

      fclose(sdiums\_after1);

      cout<<"File after first change:  \n"<<endl;

      sdiums\_after1 = fopen("stadiums\_after\_change.dat", "rb");

      if ((sdiums\_after1=fopen("stadiums\_after\_change.dat", "rb"))==NULL){

      cout<<"Eror!";

      exit(1);}

      for (int i = 0; i <ns; i++) {

        fread(&m, sizeof(stadium), 1, sdiums\_after1);

        cout << "Stadium " << i + 1 << endl;

        cout << mas[i].name << endl;

        cout << mas[i].year << endl;

        cout << mas[i].playgrounds << endl;

        cout << mas[i].kindofsport << endl << endl;

    }

     fclose(sdiums\_after1);

     FILE\* sdiums\_after2;

     sdiums\_after2 = fopen("stadiums\_after\_change\_1.dat", "wb");

     if ((sdiums\_after2=fopen("stadiums\_after\_change\_1.dat", "wb"))==NULL){

      cout<<"Eror!";

      exit(1);}

     cout<<"Enter the number of the stadium before which you want to put the entered stadiums:   \n";

     cin>>elements;

     cout<<"\nEnter a stadiums you want to add:  \n";

     for(int i=1; i<=2; i++){

      cout<<"Stadium "<<i<<endl;

      cout<<"Enter a name:  \n";

      cin>>m.name;

      cout<<"Enter a year:   \n";

      cin>>m.year;

      cout<<"Enter a number of playgrounds:   \n";

      cin>>m.playgrounds;

      cout<<"Enter a kind of sports:   \n";

      cin.ignore();

      getline(cin, m.kindofsport);

      mas.emplace(mas.begin()+(elements-1), m);

      fwrite(&m, sizeof(stadium), 1, sdiums\_after2);

     }

     fclose(sdiums\_after2);

     int newSize = mas.size();

     sdiums\_after2 = fopen("stadiums\_after\_change\_1.dat", "rb");

     if ((sdiums\_after2=fopen("stadiums\_after\_change\_1.dat", "rb"))==NULL){

      cout<<"Eror!";

      exit(1);}

     for(int i=0; i<newSize; i++){

      fread(&m, sizeof(stadium), 1, sdiums\_after2);

      cout<<"Stadium "<<i+1<<endl;

      cout<<mas[i].name<<endl;

      cout<<mas[i].year<<endl;

      cout<<mas[i].playgrounds<<endl;

      cout<<mas[i].kindofsport<<endl<<endl;

     }

     fclose(sdiums\_after2);

}

Для початку під’єднуємо усі необхідні бібліотеки: «iostream», «string», «vector» та «algorithm». Далі оголошуємо структуру «stadium». В функції main відриваємо файл "stadiums.dat" у бінарному режимі для запису та зчитування даних про стадіони. Далі іде зчитування даних про стадіони з клавіатури. Інформація про стадіони записується у вектор mas та у файл "stadiums.dat". Після йде читання та виведення інформації про стадіони з файлу "stadiums.dat"

Наступний крок- користувач вводить рік, після якого всі стадіони будуть видалені з вектора та файлу. Оновлений вектор записується у новий файл "stadiums\_after\_change.dat" та виводиться на екран. Після, Користувач вводить номер стадіону перед яким потрібно вставити два нові стадіони та інформацію про них, яка вставляються у вектор та файл "stadiums\_after\_change\_1.dat" перед вказаним елементом. Оновлений вектор з новими стадіонами записується у файл "stadiums\_after\_change\_1.dat" та виводиться на екран.

**Завдання №2 VNS Lab 9**

#include<iostream>

#include<fstream>

#include<vector>

#include<string>

#include <algorithm>

using namespace std;

bool twowords(string line){

    vector<string> s;

    int position=0,len=0, k;

    for (char c : line) {

        if (c == ' ') ++len;

    }

    while(true){

        k=line.find(' ', position);

        s.push\_back(line.substr(position, k-position));

        position = k + 1;

        if(k == string::npos){

            break;

        }

    }

    k=0;

    for(int i=0; i<s.size(); i++){

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

            if(s[i]==s[j] && i!=j) return 1;

        }

    }

    return 0;

}

int number(vector<string> s){

    int a=0, count=0, rez=0;

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

        for (char c : s[i])

        if (c == 'A' || c == 'a') a++;

        if(a>count){

            count = a;

            rez = i+1;

        }

        a=0;

}

return rez;

}

int main(){

    int n;

    int position=0,k=0;

    vector<int> A;

    vector<string> s;

    string line;

    ifstream inFile("F1.txt");

    ofstream outFile("F2.txt");

    if (!inFile.is\_open() || !outFile.is\_open()) {

        cout<< "Error!" << endl;

        return 1;

    }

    while(getline(inFile, line)){

    if(twowords(line)) outFile << line << endl;

    while(true){

        k=line.find(' ', position);

        s.push\_back(line.substr(position, k-position));

        position = k + 1;

        if(k == string::npos){

            break;

        }

    }

    }

    cout<<"the word with the largest number of letters a has a number:  "<<number(s);

    inFile.close();

    outFile.close();

}

Першим кроком є під’єднання бібліотек «iostream», «string», «vector», «fstream» та «algorithm». Функція twowords перевіряє рядок на наявність однакових слів у ньому. Вона розділяє рядок на окремі слова за допомогою пробілів, зберігає ці слова у векторі s, а потім перевіряє, чи є однакові слова серед них. Якщо такі слова знайдені, функція повертає true, в іншому випадку – false.

Функція number приймає вектор рядків та знаходить рядок, у якому кількість літер 'A' або 'a' найбільша. Вона проходить усі рядки у векторі, рахує кількість літер 'A' або 'a' в кожному рядку та повертає номер рядка, де ця кількість найбільша.

У функції main створюються необхідні змінні: n (для збереження номеру рядка), position (позиція в рядку), k (для збереження позиції пробілу), вектори A і s для зберігання чисел і рядків відповідно, а також рядок line для зчитування рядків з файлу. Відкриваються вхідний файл "F1.txt" для читання та вихідний файл "F2.txt" для запису. Якщо файли не вдалося відкрити, виводиться повідомлення про помилку. Зчитується кожен рядок з вхідного файлу. У цьому ж циклі проводиться перевірка наявності двох однакових слів у рядку, а потім відбувається розділення рядка на окремі слова за допомогою пробілів та їх збереження у векторі s. Виводиться номер рядка, де знайдено найбільшу кількість літер 'A' або 'a' за допомогою функції number(s). Після, файли закриваються.

**Завдання №3 Algotester Lab 4**

#include <iostream>

#include <vector>

#include <algorithm>

#include <set>

using namespace std;

int main() {

    int N, K,n;

    vector<int> ar1;

    vector<int> ar2;

    cin>>N;

    for(int i=0; i<N; i++) {

        cin>>n;

        ar1.push\_back(n);

        n=0;}

    cin>>K;

    for(int i=0; i<K; i++) {

        cin>>n;

        ar2.push\_back(n);

        n=0;}

    sort(ar1.begin(), ar1.end());

    sort(ar2.begin(), ar2.end());

    vector<int> dif1;

    set\_difference(ar1.begin(), ar1.end(), ar2.begin(), ar2.end(), back\_inserter(dif1));

    cout<<dif1.size()<<endl;

    for (int i=0; i<dif1.size(); i++) {

         cout<<dif1[i]<<" ";

    }

    cout<<endl<<endl;

    vector<int> dif2;

    set\_difference(ar2.begin(), ar2.end(), ar1.begin(), ar1.end(), back\_inserter(dif2));

    cout<<dif2.size()<<endl;

    for (int i=0; i<dif2.size(); i++) {

         cout<<dif2[i]<<" ";

    }

    cout<<endl<<endl;

    vector<int> in;

    set\_intersection(ar1.begin(), ar1.end(), ar2.begin(), ar2.end(), back\_inserter(in));

    cout<<in.size()<<endl;

    for (int i=0; i<in.size(); i++) {

         cout<<in[i]<<" ";

    }

    cout<<endl<<endl;

    vector<int> uni;

    set\_union(ar1.begin(), ar1.end(), ar2.begin(), ar2.end(), back\_inserter(uni));

    cout<<uni.size()<<endl;

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

         cout<<uni[i]<<" ";

    }

    cout<<endl<<endl;

    vector<int> sym\_dif;

    set\_symmetric\_difference (ar1.begin(), ar1.end(), ar2.begin(), ar2.end(), back\_inserter(sym\_dif));

    cout<<sym\_dif.size()<<endl;

    for (int i=0; i<sym\_dif.size(); i++) {

         cout<<sym\_dif[i]<<" ";

    }

    return 0;

}

Для початку під’єднуємо усі необхідні бібліотеки: «iostream», «set», «vector» та «algorithm». Після, відбувається створення змінних N, K і n, а також векторів ar1 і ar2 для зберігання цілих чисел. Далі, відбувається зчитування значення N та значень вектора arr1, і аналогічно значення K та вектора arr2. Наступним етапом є сортування значень у векторах.

Після, з використанням функцій set\_difference, set\_intersection, set\_union, set\_symmetric\_difference з STL для виконання відповідних операцій над векторами чисел. Результати цих операцій записуються у відповідні нові вектори: dif1, dif2, in, uni та sym\_dif.

Далі йде виведення результатів операцій на екран, включаючи розмір кожного з векторів dif1, dif2, in, uni та sym\_dif, а також їхні вмісти за допомогою циклу for.

**Завдання №4 VNS Lab 6**

#include <iostream>

#include <string>

using namespace std;

int shotrest(string s){

    int k=0,length=0;

    for(int i=0; i<s.length(); i++){

        if(s[i]==' '){

            if(length==0){

                length=k;

            }

            if(k<length){

                length=k;

            }

            k=0;

        }

        else{

            k++;

        }

    }

    return length;

}

int main(){

    string s;

    cout<<"Enter a string: ";

    getline(cin, s);

    cout<<shotrest(s);

}

Перш за все оголошуємо бібліотеки «iostream», «string». Далі оголошується функція shortest, яка приймає рядок s та обчислює довжину найкоротших слів у цьому рядку. Функція проходить по кожному символу у рядку та рахує кількість символів до зустрічі пробілу (якщо такий є). Якщо вона знаходить слово, яке коротше за попереднє найкоротше слово, вона зберігає його довжину. В функції main викликається функція shotrest і те що вона повертає виводиться на екран.

**Завдання №5 Algotester Lab 6**

#include <iostream>

#include <unordered\_map>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

    int N, K;

    cin >> N >> K;

    unordered\_map<string, int> c;

    vector<char> k;

    for (int i = 0; i < N; ++i) {

        string word;

        cin >> word;

        for (int j=0; j<word.length(); j++) {

            word[j]= tolower(word[j]);

        }

        c[word]++;

    }

    for (auto n = c.begin(); n != c.end(); n++) {

    const auto& entry = \*n;

    if (entry.second >= K) {

        string word = entry.first;

        for (auto it = word.begin(); it != word.end(); ++it) {

            char f = \*it;

            if (find(k.begin(), k.end(), f) == k.end()) {

                k.push\_back(f);

            }

        }

    }

}

    if (k.empty()) {

        cout << "Empty!";

    } else {

        cout<<k.size()<<endl;

        sort(k.begin(), k.end(), greater<char>());

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

            cout << k[i]<< " ";

        }

    }

    return 0;

}

Спочатку оголошуємо усі необхідні бібліотеки, у нашому випадку це: бібліотек «iostream», «vector», «unordered\_map» та «algorithm». Далі, зчитуємо два числа: N та K. Далі створюємо unordered\_map для зберігання слів та їх кількості вхіджень. Ключ у цій unordered\_map - це слово у нижньому регістрі, а значення - кількість входжень слова.

Після, відбувається введення N слів з клавіатури та збереження їх у unordered\_map. Кожне слово переводиться у нижній регістр за допомогою tolower(). Після цього кожне слово додається у unordered\_map, і його кількість входжень збільшується на одиницю. Далі проходимось по unordered\_map та якщо кількість входжень слова більша або дорівнює K, то кожна унікальна літера цього слова додається до вектора k.

Якщо вектор k порожній (тобто усі слова, які зустрілись принаймні K разів, не містять унікальних літер), виводиться повідомлення "Empty!". Якщо ж у векторі містяться елементи, то спочатку виводиться їхня кількість, а потім упорядковані у спадному порядку унікальні літери з цього вектора розділені пробілами.

**Завдання №6 Class Practice Task**

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

enum FileOpResult {Success, Failure};

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

ofstream file(name);

    if (!file.is\_open()) {

        return Failure;

    }

    file << content;

    if (file.fail()) {

        return Failure;

    }

    file.close();

    if (file.fail()) {

        return Failure;

    }

    return Success;

}

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

    ifstream from(file\_from);

    if (!from.good()) {

        return Failure;

    }

    if (!from.is\_open()) {

        return Failure;}

    ofstream to("file\_to.txt");

    if (!to.is\_open()) {

        from.close();

        return Failure;

    }

    to<<from.rdbuf();

    if(to.fail()) {

        to.close();

        from.close();

        return Failure;

        }

    else {

        to.close();

        from.close();

        return Success;

}

}

int main(){

    char\* name= new char[100];

    char\* content = new char[1000];

    name="practice1.txt";

    cout << "Enter a content: ";

    cin.getline(content, 1000);

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

    delete[] name;

    delete[] content;

    if (result == Success) {

        cout << "Y" << endl;

    } else {

        cout << "n" << endl;

    }

    char\* file\_from = new char [100000];

    file\_from = "class\_pr\_2.txt";

    char\* file\_to = new char [100000];

    FileOpResult rez2 =copy\_file(file\_from, file\_to);

    delete[] file\_from;

    delete[] file\_to;

    if (rez2 == Success) {

        cout << "Yes" << endl;

    } else {

        cout << "No" << endl;

    }

}

Перш за все під’єднуються бібліотеки «iostream», «fstream», «string». Далі оголошується функція FileOpResult write\_to\_file(char \*name, char \*content) призначена для запису даних у файл. Отримуючи два параметри: name - назва файлу та content - вміст файлу. Вона відкриває файл для запису, записує вміст у файл та повертає значення Success, якщо виконання успішне, або Failure, якщо виникла помилка під час виконання.

Після йде функція FileOpResult copy\_file(char\* file\_from, char\* file\_to) призначена для копіювання вмісту з одного файлу у інший. Вона відкриває вихідний файл для читання та файл-приймач для запису. Потім вона копіює вміст одного файлу в інший за допомогою from.rdbuf(). Функція також повертає значення Success, якщо операція пройшла успішно, або Failure, якщо виникла помилка під час виконання.

Далі, в функції main створюються дві динамічні масиви name та content типу char. Потім в name зберігається назва файлу "practice1.txt", а в content зберігається вміст, введений користувачем з клавіатури за допомогою cin.getline(). Після чого викликається функція write\_to\_file(name, content) для запису вмісту у файл. Після виконання цієї операції динамічні масиви name та content очищуються. Далі створюються два динамічні масиви file\_from та file\_to, в які записуються назви вихідного файлу "class\_pr\_2.txt" та файлу-приймача "file\_to.txt". Потім викликається функція copy\_file(file\_from, file\_to) для копіювання вмісту з вихідного файлу у файл-приймач. Після виконання операції ці динамічні масиви також очищуються.

**Завдання №7 Self Practice Task**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main()

{

    int n, m;

    cin >> n >> m;

    vector<string> N;

    vector<string> M;

    for (int i = 0; i < n; i++) {

        string H;

        cin >> H;

        N.push\_back(H);

    }

    for (int i = 0; i < m; i++) {

        string H;

        cin >> H;

        M.push\_back(H);

    }

    vector<string> rez;

    for (int i = 0; i < n; i++) {

        rez.push\_back(N[i]);

    }

    for (int i = 0; i < m; i++) {

        rez.push\_back(M[i]);

    }

    sort(rez.begin(), rez.end());

    rez.erase(unique(rez.begin(), rez.end()), rez.end());

    cout << rez.size();

    return 0;

}

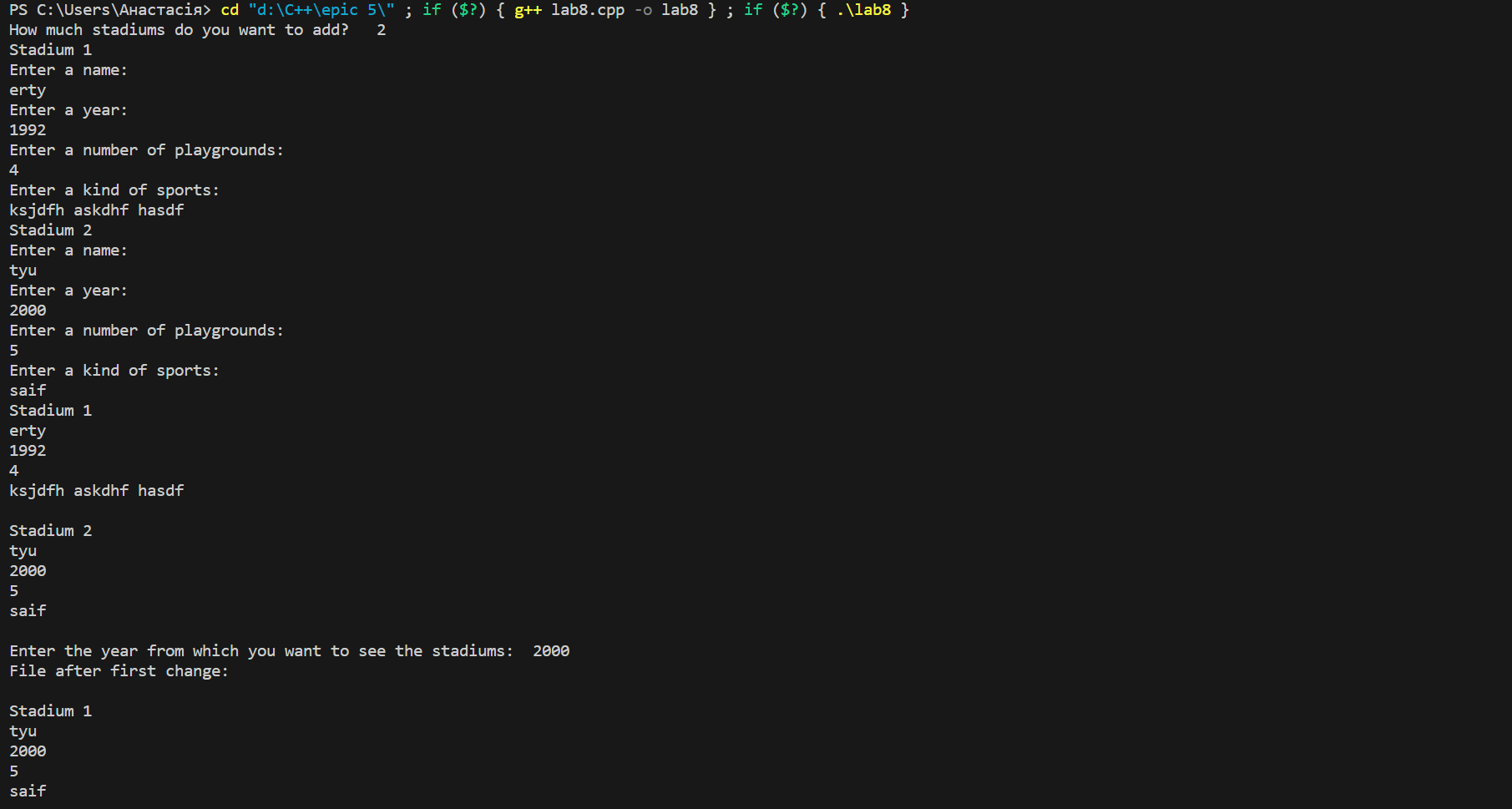
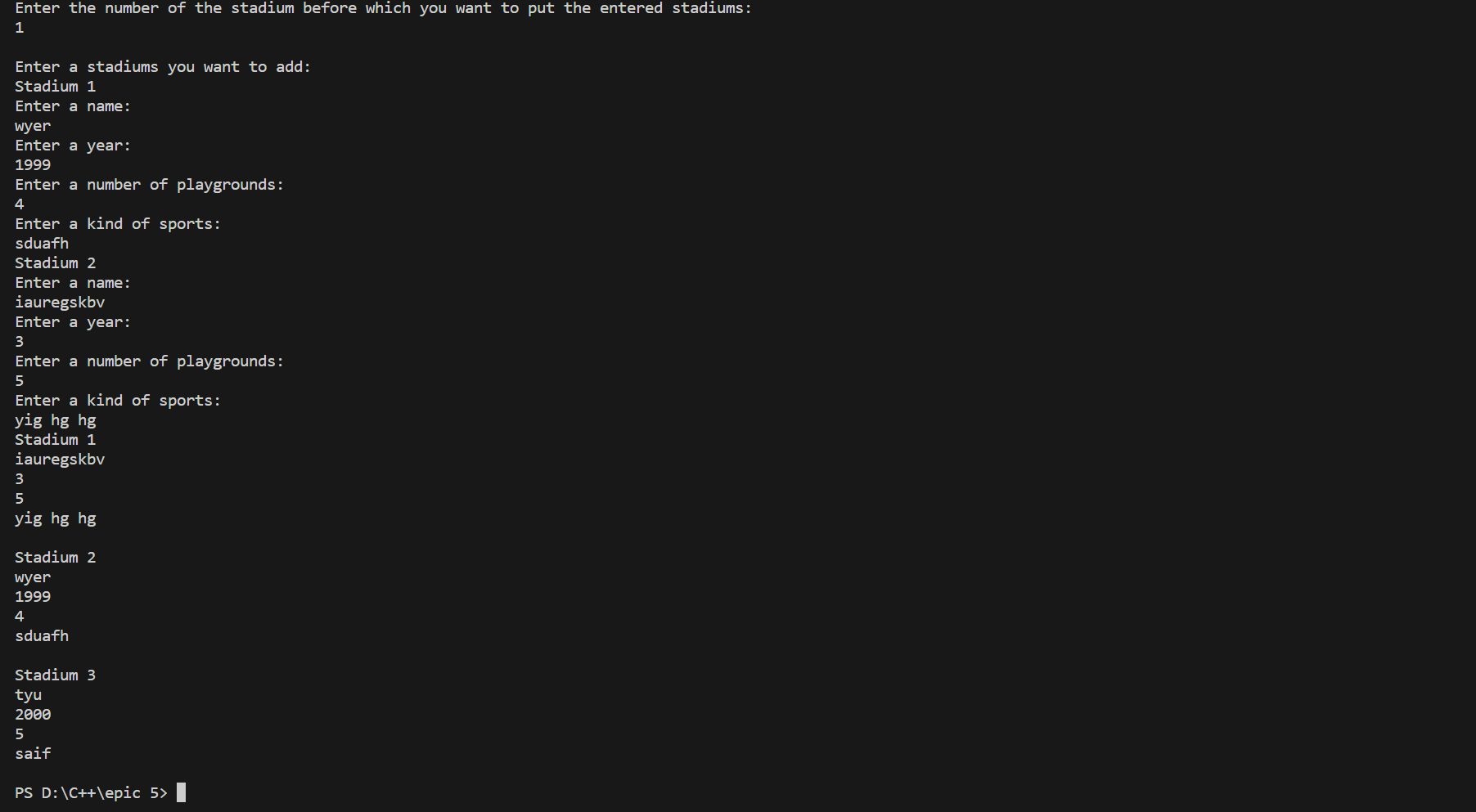
Коли я запускаю програму, першим кроком є введення двох чисел n та m, що відповідають розмірам векторів N та M. Це дає мені уявлення про те, скільки елементів я буду вводити для кожного з векторів. Потім я ввожу елементи для кожного з векторів N та M. Це відбувається у двох циклах, де я вводжу значення для кожного елемента окремо для обох векторів.

Після введення всіх значень для обох векторів я створюю третій вектор rez. Я копіюю всі елементи з вектору N у вектор rez, а потім копіюю всі елементи з вектору M у той же вектор rez. Після я використовую функцію sort для сортування елементів у векторі rez. Потім використовую функцію unique, яка видаляє суміжні повторення елементів у відсортованому векторі rez. Результат цього видалення повторюваних елементів зберігається у векторі rez.

Після відбувається виведення розміру вектора rez, що і є кінцевим результатом виконання програми.

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

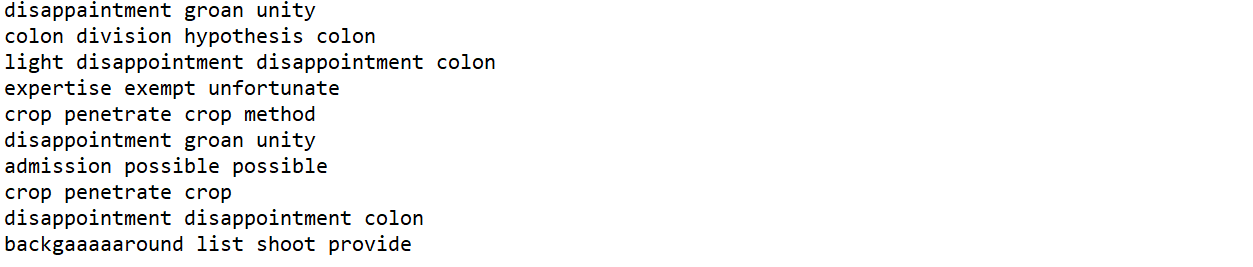
**Завдання №1** **VNS Lab 8**

****

Тест програми VNS Lab 8

Час затрачений на виконання завдання: 10 годин

**Завдання №2** **VNS Lab 9**



Файл 1

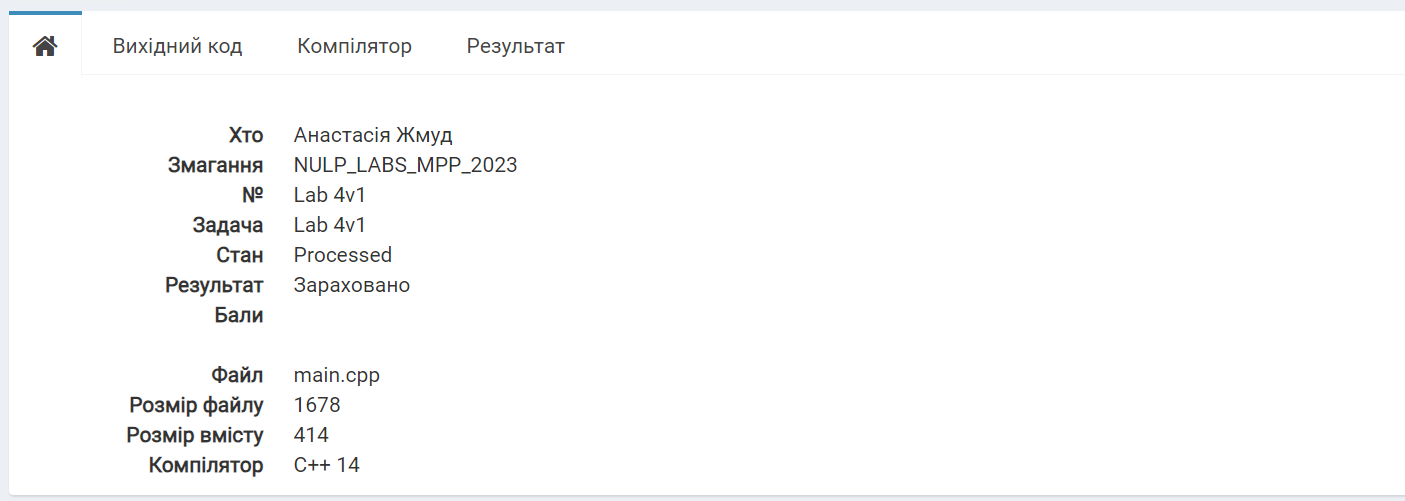


Файл 2



Тест програми VNS Lab 9

**Завдання №3 Algotester Lab 4**

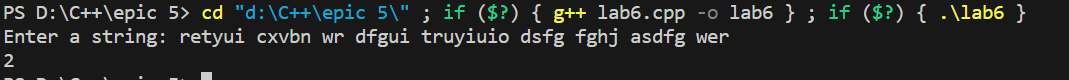


Зарахування Algotester Lab 4v1

**Завдання №4 VNS Lab 6**

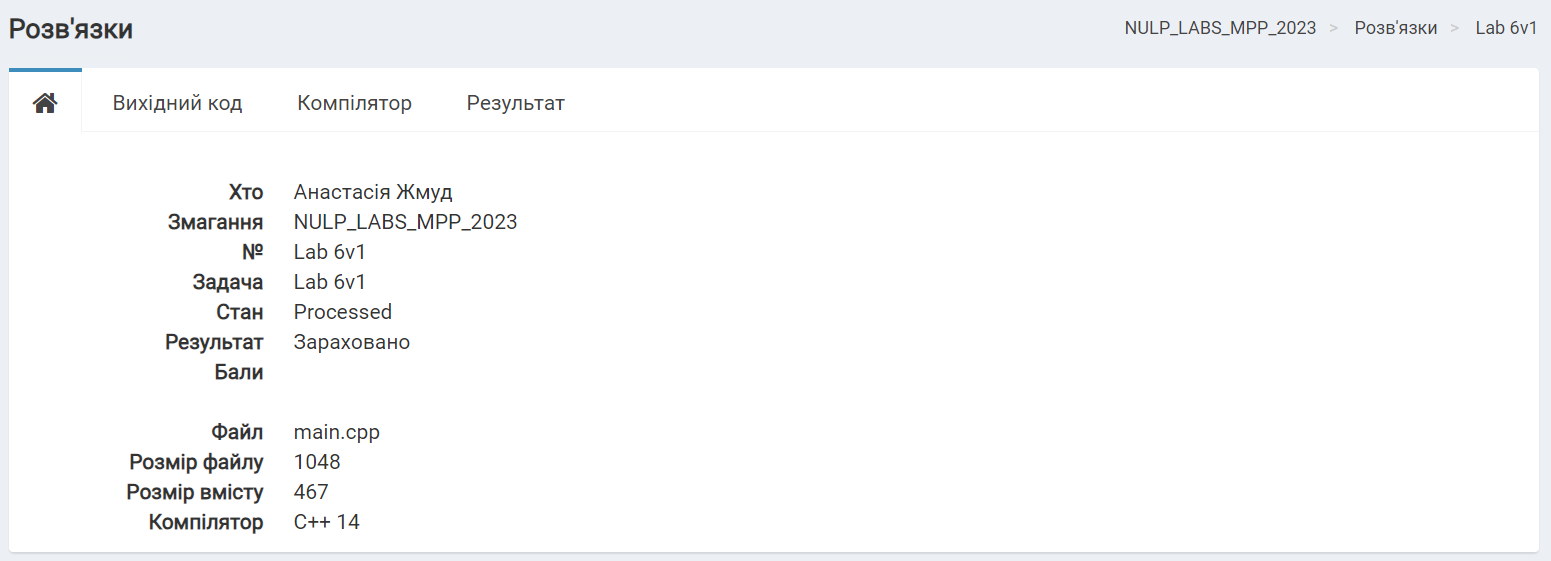


Тест №1



Тест №2

**Завдання №5 Algotester** **Lab 6**



Зарахування Algotester Lab 6

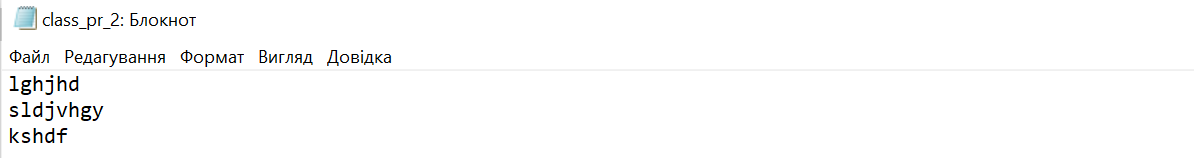
**Завдання №6 Class Practice Task**

****

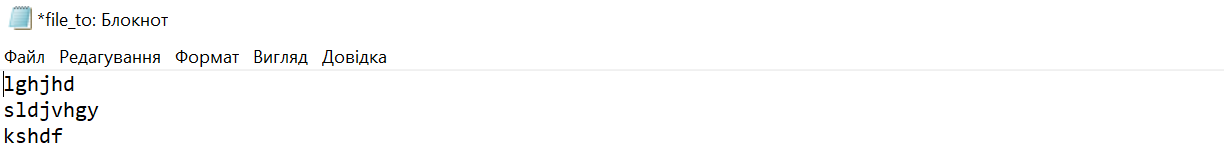
Виконання коду



Файл 1

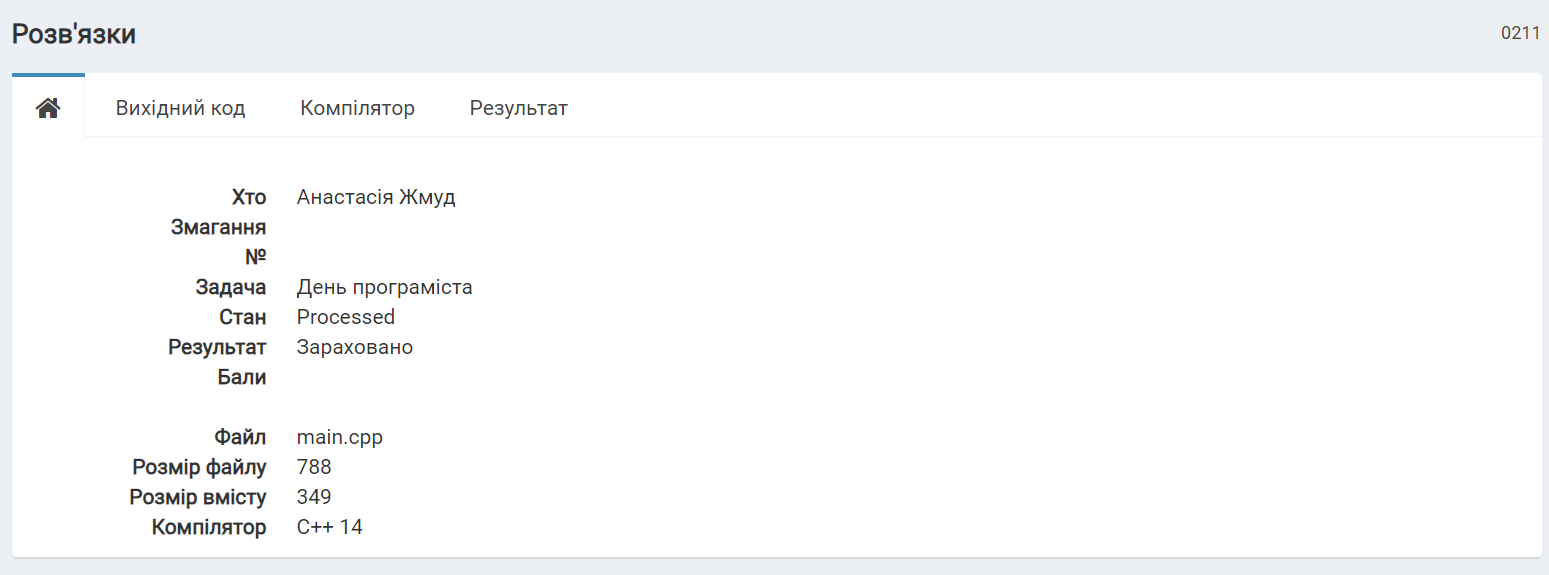


Файл 2



Файл 3

**Завдання №7 Self Practice Task**



Зарахування Algotester «День програміста»

# **Висновки:**

У ході виконання лабораторних і практичних робіт, присвячених роботі з двійковими файлами та ввіду-виводу рядків, було успішно досягнуто мети ознайомлення з організацією вводу-виводу структурованої інформації з зовнішніх носіїв. Працюючи над даним епіком я набула навичок у роботі з текстовими файлами, ввідом-виводом текстової інформації та її зберіганням на зовнішніх носіях виявилося надзвичайно корисним. Ця робота надала розуміння процесу зберігання та обробки інформації в різних форматах, зокрема, вивчення особливостей двійкових файлів і текстових рядків.