МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №11
З навчальної дисципліни
“Базові методології та технології програмування “
“Реалізація програмних засобів оброблення динамічних структур даних та бінарних файлів”

Виконав
Студенти академічної групи КН-22
Осадчий В.К
Півненко О.М
Рудь І.В

Перевірив
Викладач кафедри кібербезпеки
Та програмного забезпечення
О.СОБІНОВ

Кропивницький-2023

Мета роботи полягає у набутті ґрунтовних вмінь і практичних навичок командної (колективної) реалізації програмного забезпечення, розроблення функцій оброблення динамічних структур даних, використання стандартних засобів C++ для керування динамічною пам’яттю та бінарними файловими потоками.

https://github.com/odorenskyi/**Rud-Ihor**-KN22
https://github.com/odorenskyi/**Pivnenko-Oleksandr**-KN22
https://github.com/odorenskyi/**Osadchyi-Vladyslav**-KN22

Варіант 3

— *Завдання на розроблення програмного забезпечення* —

Створити **електронний довідник індексів та відділень поштового зв’язку України УДППЗ “Укрпошта”** (п’ятизначний індекс, область, район, населений пункт, ВПЗ, що обслуговує).
За вибором користувача (оператора) програма забезпечує:

- *пошук* запису в довіднику за введеним індексом або назвою населеного пункта;
- *виведення* всього довідника на екран або у текстовий файл;
- *додавання* нового запису в довідник;
- *вилучення* заданого оператором запису з довідника;
- *завершення* роботи програми з автоматичним записом даних у файл.

Дані довідника автоматично завантажуються з файлу під час запуску програми.

Поштові індекси та відділення поштового зв’язку України

Пошук за індексом

25006 Показати Пошук здійснюється за п`ятизначним індексом

Пошук за місцем розташування

Пошук за назвою населеного пункту

Знайдено 3 записи

Індекс	Область	Район	Населений пункт	До переіменування*	ВПЗ, що обслуговує
25006	Кіровоградська		Кропивницький	Кіровоград	Кропивницький 6
25006	Кіровоградська		Кропивницький	Кіровоград	ЦехОПП м.Кропивницький
25006	Кіровоградська		Кропивницький	Кіровоград	ЦОС № 5 м. Кропивницький

Розподіл праці над задачею:

1)Півненко О.М-реалізація логіки та модуля задачі;

2)Рудь І.В-реалізація основної програми для користувача;

3)Осадчий В.К-реалізація тестування програми та написання звіту.

1)Реалізація завдання Півненка О.М

Файл.h: hearder.hpp

```
#ifndef HEADER_HPP_INCLUDED
#define HEADER_HPP_INCLUDED
#include <string>

class Department
{
   public:
      unsigned int index;
      std::string area;
      std::string district;
      std::string township;
      std::string postOffice;

      Department(unsigned int i, std::string ar, std::string dist, std::string tsh, std::string
office){
         if(ar.length() == 0 || tsh.length() == 0 || office.length() == 0)
            throw std::exception();

         index = i; district = dist; postOffice = office;
         area = ar; township = tsh;
      }

      Department(){
      }
};

Department* readDeparts(std::string path,int* length);

void addDepartToFile(std::string path, Department* deps, int length_deps);

void removeDepartFromFile(std::string path, Department* dep);

void createFile(std::string path);

bool isFindDep(std::string path, Department* dep);

#endif // HEADER_HPP_INCLUDED
```

Алгоритм:
1)Оголошення директив препроцесора для включення заголовочного файлу та попереднього визначення маркера включення заголовочного файлу.
2)Оголошення класу "Department", який містить п'ять публічних членів даних: "index" - беззнакове ціле число, "area" - рядок, "district" - рядок, "township" - рядок та "postOffice" - рядок.
3)Визначення конструктора з п'ятьма параметрами, який приймає значення для кожного з членів даних, перевіряє, чи довжина рядків "area", "township" та "postOffice" більше 0, та викидає виключення, якщо хоча б один з них має довжину 0. Якщо перевірка успішна, то встановлюються значення членів даних.
4)Визначення конструктора за замовчуванням без параметрів, який не робить нічого.
5)Оголошення функцій "readDeparts", "addDepartToFile", "removeDepartFromFile", "createFile" та "isFindDep".
6)"readDeparts" - функція, яка приймає рядок "path" та вказівник на ціле число "length", і повертає вказівник на об'єкт "Department". Функція читає з файлу за заданим шляхом "path" департаменти та повертає їх як вказівник на масив об'єктів "Department". Розмір масиву повертається через параметр "length".
7)"addDepartToFile" - функція, яка приймає рядок "path", вказівник на масив об'єктів "Department" "deps" та ціле число "length_deps". Функція додає масив департаментів до файлу за заданим шляхом "path".
8)"removeDepartFromFile" - функція, яка приймає рядок "path" та вказівник на об'єкт "Department" "dep". Функція видаляє департамент з файлу за заданим шляхом "path", вказаний у параметрі "dep".
9)"createFile" - функція, яка приймає рядок "path". Функція створює файл за заданим шляхом "path".
10)"isFindDep" - функція, яка приймає рядок "path" та вказівник на об'єкт "Department" "dep". Функція перевіряє, чи є департамент з вказаними значеннями членів даних в файлі за заданим шляхом "path". Якщо такий департамент існує, функція повертає true, інакше повертає false.
11)Закінчення визначення директиви препроцесора "#endif".

Файл.cpp: main.cpp

```
#include "header.hpp"
```

```
// ######################################################
// Private functions. Only use in module. (PRIVATE)
// ######################################################

bool comparing(Department* dep, Department* dep_sec){
   if(dep_sec->index == dep->index && dep_sec->area == dep->area && dep_sec->district
== dep->district && dep_sec->township == dep->township && dep_sec->postOffice ==
dep->postOffice)
      return true;

   return false;
}

void changeCount(std::string path, int cnt, bool cnt_or_not){
   int i = 0;
   std::string cnt_before = "";
   std::string line_content = "";

   std::ifstream reader;
   reader.open(path, std::ios_base::in);

   getline(reader, line_content);

   while(line_content[i] != '&' && cnt_before.length() <= 15){
      cnt_before += line_content[i];
      i += 1;
   }

   std::string full_count = "";

   if(cnt_or_not == true)
      full_count = std::to_string(cnt + std::stoi(cnt_before));
   else
      full_count = std::to_string(cnt);

   std::string toWrite = full_count + "&";

   while(!reader.eof()){
      getline(reader, line_content); //

      toWrite += "\n" + line_content;
   }

   reader.close();

   std::ofstream writer;
   writer.open(path, std::ios_base::out);
```

```
    writer << toWrite;

    writer.close();
}

int countObjects(std::string path){
    std::ifstream reader;
    std::string line_content;
    std::string temp_line = "";
    int i = 0;
    reader.open(path, std::ios_base::in);

    getline(reader, line_content);

    reader.close();

    while(line_content[i] != '&' && temp_line.length() < 15){
        temp_line += line_content[i];
        i += 1;
    }

    if(temp_line.length() < 15)
        return std::stoi(temp_line);

    return -1;
}

// This is also a private function, do not touch. For delete use "removeDepartFromFile"

Department* removeDep(Department* deps, int index, int dep_size){
    int output_index = 0;
    int new_size = dep_size - 1;
    Department* output = new Department[new_size];

    for(int i = 0; i < dep_size; i += 1){
        if(i == index){
            continue;
        }
        output[output_index] = deps[i];

        output_index += 1;
    }

    return output;
}

Department analyzeObject(std::string obj, bool* flagCrash){
```

```
    int sem_col = 0;
    int length = obj.length();
    std::string index = "";
    std::string area = "";
    std::string district = "";
    std::string township = "";
    std::string postOffice = "";

    for(int i = 0; i < length; i += 1){
        if(obj[i] == ';'){
            sem_col += 1;
            continue;
        }

        switch(sem_col){
            case 0:
                index += obj[i];
                break;
            case 1:
                area += obj[i];
                break;
            case 2:
                district += obj[i];
                break;
            case 3:
                township += obj[i];
                break;
            case 4:
                postOffice += obj[i];
                break;
        }
    }

    if(sem_col != 4)
        (*flagCrash) = true;

    return Department(std::stoi(index), area, district, township, postOffice);
}

//
//  ##################################################
//  Public functions. Can use in any place. (PUBLIC)
//  ##################################################
//

// Read all departs in file.

Department* readDeparts(std::string path, int* length){
```

```
    std::string obj_line = "";
    bool IsCrashed = false;
    int obj_index = 0;

    (*length) = countObjects(path);

    if((*length) == -1)
        throw std::exception(); // The file is corrupt - the reading did not go through.
    else if(*length == 0)
        return nullptr; // The file is empty. There are no objects. We don't do calculations.

    Department* output_objects = new Department[(*length)];

    std::ifstream reader;
    reader.open(path, std::ios_base::in);
    getline(reader, obj_line); obj_line = "";

    while(!reader.eof()){
        getline(reader, obj_line);

        Department dep_obj = analyzeObject(obj_line, &IsCrashed);

        if(IsCrashed == true || obj_index >= (*length))
            throw std::exception(); // if object is corrupt - throw exception.

        output_objects[obj_index] = dep_obj;
        obj_index += 1;
    }

    reader.close();

    return output_objects; // return all read objects
}

// Adds an array of deps objects to the file.
// The size of the array is specified with length_deps
// ATTENTION. UP COMMENTS!!!

void addDepartToFile(std::string path, Department* deps, int length_deps){
    if(length_deps <= 0)
        throw std::exception();

    changeCount(path, length_deps, true);

    std::ofstream writer;
    writer.open(path, std::ios_base::app);

    for(int i = 0; i < length_deps; i += 1){
```

```
        Department toAdd_dep = deps[i];

        writer << "\n";
        writer << toAdd_dep.index;
        writer << ";" + toAdd_dep.area + ";" + toAdd_dep.district + ";" + toAdd_dep.township +
";" + toAdd_dep.postOffice;
    }

    writer.close();
}

// Delete the dep object from the file.
// ATTENTION, ONLY ONE (1) OBJECT! NOT ARRAY.

void removeDepartFromFile(std::string path, Department* dep){
    int length = -1;
    int cnt_dels = 0;
    bool isFind = false;

    Department* read_deps = readDeparts(path, &length);

    for(int i = 0; i < length; i += 1){
        if(comparing(&read_deps[i], dep)){
            read_deps = removeDep(read_deps, i, length - cnt_dels);
            cnt_dels += 1;
            isFind = true;
            i = -1;
        }
    }
    if(isFind == true){
        std::ofstream writer;
        writer.open(path, std::ios_base::out);

        writer << (length - cnt_dels) << "&";

        for(int i = 0; i < length - cnt_dels; i += 1){
            writer << "\n" + std::to_string(read_deps[i].index) + ";" + read_deps[i].area + ";" +
read_deps[i].district + ";" + read_deps[i].township + ";" + read_deps[i].postOffice;
        }

        writer.close();
    }
}

bool isFindDep(std::string path, Department* dep){
    int length = 0;
    Department* all_deps = readDeparts(path, (&length));
```

```
    for(int i = 0; i < length; i += 1){
        if(comparing(&all_deps[i], dep))
            return true;
    }

    return false;
}

// If the file does not exist, initialize it.

void createFile(std::string path){
    std::ofstream creator;
    creator.open(path, std::ios_base::out);

    creator << "0&";
    creator.close();
}
```

Алгоритм:

Алгоритм, реалізований у даному коді, дозволяє зчитувати, записувати та опрацьовувати дані про департаменти в файлі. Програма містить набір функцій, які забезпечують таку можливість. Файл містить інформацію про департаменти, яка записується у вигляді текстових рядків. Кожен рядок описує один департамент і містить наступну інформацію: індекс департаменту, назва області, району, населеного пункту та поштового відділення.

Основні функції, які використовуються в програмі:

1)comparing - функція порівнює два департаменти за індексом, назвою області, району, населеного пункту та поштовим відділенням.

2)readDeparts - функція зчитує дані про департаменти з файлу та повертає масив об'єктів типу Department, який містить ці дані. Функція також повертає кількість департаментів, які були зчитані.

3)addDepartToFile - функція додає новий масив департаментів у файл. Кількість департаментів, які додаються, визначається параметром length_deps.

4)countObjects - функція рахує кількість департаментів, які містяться в файлі.

5)changeCount - функція змінює значення лічильника департаментів у файлі.

6)removeDep - функція видаляє департамент з масиву департаментів.

7)analyzeObject - функція аналізує текстовий рядок та повертає об'єкт типу Department, що містить інформацію про департамент.
8)AddDepartToFile- функція додає нові департаменти до файлу, що знаходиться за вказаним шляхом path. У якості параметрів приймається масив департаментів deps, який необхідно додати, та його довжина length_deps.
9)removeDepFromFile - це функція, яка видаляє департамент з файлу. Вона приймає параметр index, який вказує на індекс департаменту, який потрібно видалити.
10)isFindDep перевіряє, чи містить файл, що знаходиться за шляхом path, департамент dep. Спочатку викликається функція readDeparts, яка повертає масив департаментів та їх кількість з файлу. Потім перевіряється, чи існує в масиві департамент, що має такі ж характеристики як департамент dep, за допомогою функції comparing. Якщо такий департамент знайдено, функція повертає true, якщо ні - false.
11)createFile створює файл за заданим шляхом path, якщо він ще не існує. Файл ініціалізується строкою "0&", що означає, що файл містить 0 департаментів. Для створення файлу використовується об'єкт std::ofstream, який дозволяє записувати дані у файл за заданим шляхом та з певними параметрами (у даному випадку - std::ios_base::out). Після виконання операції з файлом, об'єкт закривається за допомогою методу close().

2)Реалізація завдання Рудя І.В

```
#include <iostream>
#include "header.hpp"
#include <exception>
using namespace std;

int main()
{
   setlocale(LC_ALL, “ukr”);
    try{
       string fileName;
       // Запитуємо у користувача чи хоче він створити файл чи працювати з
відомим файлом
       char yesOrNo;
       cout << "Введіть ім'я файлу: " ;
       cin >> fileName;
       cout << "Ви хочете на основі цього імені створити файл?(Y|N)" << endl;
       cin >> yesOrNo;
```

```
    if(yesOrNo == 'y' || yesOrNo == 'Y'){
        createFile(fileName);
        cout << "Файл створено!" << endl;
    }
    unsigned short n;
    cout << "Введіть кількість записів, яку ви хочете додати до довідника: ";
    cin >> n;
    // Ініціюємо масив departments з розміром n
    Department *departments = new Department[n];

    int i;
    // Ініціюємо поля класу Department
    unsigned int index;
    string area;
    string district;
    string township;
    string postOffice;

    /* В циклі проходимось n-ну кількість, та створюємо об'єкти Department
    *Просимо користувача ввести відповідні поля для створення запису у файл
    */
    for(i = 0; i < n;i++){
        cout << "Введіть індекс:";
        cin >> index;
        cout << "Введіть область:";
        cin >> area;
        cout << "Введіть район:";
        cin >> district;
        cout << "Введіть населений пункт:";
        cin >> township;
        cout << "Введіть ВПЗ, що обслуговує:";
        cin >> postOffice;
        departments[i] = Department(index,area,district,township,postOffice);
    }
    // Додавання массиву Департаметів до ФАЙЛУ
    addDepartToFile(fileName,departments,n);
    // Звільнюємо дим. пам'ять
    delete[] departments;

    // Запитуємо у користувача чи хоче він отримати всі Департаменти з
файлу?
    cout << "Хочете вивести всі департаменти таблицею?(Y|N)";
    yesOrNo = '\0';
    cin >> yesOrNo;
```

```
        if(yesOrNo == 'y' || yesOrNo == 'Y'){
        // Кількість Департаментів
        int quanityOfDepartments = 0;
        // Отримуємо список Департаментів
        Department *departmentsFromFile =
readDeparts(fileName,&quanityOfDepartments);
        // Виводимо усі Департаменти (типу таблицею)
        for(int i = 0;i < quanityOfDepartments;i++){
            cout << "Індекс"
            << "\t" << "Область"
            << "\t\t" << "Район"
            << "\t" << "Населений пункт"
            << "\t" << " ВПЗ, що обслуговує"
            << endl;

            cout << departmentsFromFile[i].index
            << "\t" << departmentsFromFile[i].area
            << "\t" << departmentsFromFile[i].district
            << "\t" << departmentsFromFile[i].township
            << "\t\t" << departmentsFromFile[i].postOffice
            << endl;
            }
        }
        // Запитуємо у користувача чи хоче він перевірити чи існує Департамент?
        yesOrNo = '\0';
        cout << "Хочете перевірити чи існує Департамент?(Y|N)";
        cin >> yesOrNo;
        if(yesOrNo == 'y' || yesOrNo == 'Y'){
            fileName = ""; index=0; area="a"; district="b"; township="c"; postOffice="d";
            cout << "Введіть ім'я файлу: " ;
            cin >> fileName;
            cout << "Добре! Будь ласка введіть всю інформацію про Департамент:"
<< endl;
            cout << "Введіть індекс:";
            cin >> index;
            cout << "Введіть область:";
            cin >> area;
            cout << "Введіть район:";
            cin >> district;
            cout << "Введіть населений пункт:";
            cin >> township;
            cout << "Введіть ВПЗ, що обслуговує:";
            cin >> postOffice;
            Department depart (index,area,district,township,postOffice);
```

```
            bool isFind = isFindDep(fileName,&depart);
            if(isFind){
                cout << "Так, такий запис є у файлі" << endl;
            }else{
                cout << "Такого запису немає у файлі. Перевірте введену інформацію."
<< endl;
            }
        }
        // Видалення Департаменту з файлу.
        yesOrNo = '\0';
        cout << "Хочете видалити Департамент з файлу?(Y|N)";
        cin >> yesOrNo;
        if(yesOrNo == 'y' || yesOrNo == 'Y'){
            fileName = ""; index=0; area=""; district=""; township=""; postOffice="";
            cout << "Введіть ім'я файлу: " ;
            cin >> fileName;
            cout << "Добре! Будь ласка введіть всю інформацію про Департамент:"
<< endl;
            cout << "Введіть індекс:";
            cin >> index;
            cout << "Введіть область:";
            cin >> area;
            cout << "Введіть район:";
            cin >> district;
            cout << "Введіть населений пункт:";
            cin >> township;
            cout << "Введіть ВПЗ, що обслуговує:";
            cin >> postOffice;
            Department depart (index,area,district,township,postOffice);
            removeDepartFromFile(fileName,&depart);
            cout << "Успішно видалено!" << endl;
        }
    }catch(const exception& ex){
        cerr << ex.what() << endl;
    }
    return 0;
}
```

1)Оголошення змінних типу string (fileName) та char (yesOrNo).

2) Запит у користувача введення імені файлу.

3)Запит у користувача, чи він хоче створити файл на основі введеного імені, чи працювати з існуючим файлом.

4)Якщо користувач вирішив створити новий файл, то викликається функція createFile() з введеним іменем файлу.

5)Запит у користувача кількості записів, які необхідно додати до файлу.
6)Створення динамічного масиву об'єктів Department розміром n.
7)Заповнення полів класу Department шляхом проходження циклу n-ну кількість разів та введення необхідних даних користувачем.
8)Додавання створеного масиву Департаментів до файлу за допомогою функції addDepartToFile().
9)Звільнення виділеної динамічної пам'яті.
10)Запит у користувача, чи він бажає отримати всі департаменти з файлу та вивести їх у вигляді таблиці.
11)Якщо користувач вирішив вивести таблицю департаментів, то за допомогою функції readDeparts() отримуємо список департаментів з файлу та виводимо їх у вигляді таблиці.
12)Запит у користувача, чи він бажає перевірити, чи існує департамент у файлі.
13)Якщо користувач погодився перевірити існування департаменту, то запитуємо у користувача ім'я файлу та всю необхідну інформацію про департамент.
14)Перевіряємо чи існує департамент у файлі за допомогою функції isDepartmentExists().

3)Реалізація завдання Осадчого В.К

Назва тестового набору Test Suite Description	TestSuit 11
Name of Project / Software	(Ім’я та фамілія розробників)-KN22/temporary_dvd/TestSuit/TestSuit 11
Рівень тестування Level of Testing	Модульне тестування
Автор тест-сьюта Test Suite Author	Осадчий Владислав Костянтинович
Виконавець Implementer	Осадчий Владислав Костянтинович

Action	Expected Result	Test Result(passed/failed/blocked)
Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська Бажаємо вивести таблицею(Y) та перевіряємо чи існує він	Вхідні дані: -test12ва1.txt Очікуваний результат: Індекс 25006 Область Кіровоградська Район (порожній) Населений пункт (порожній) ВПЗ, що обслуговує (порожній)	failed
Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська Бажаємо вивести таблицею та перевіряємо чи існує він	Вхідні дані: -test12ва1.txt Очікуваний результат: File is not found	passed
Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська “-” -”-” Бажаємо вивести таблицею(Y)	Вхідні дані: -test1.txt Очікуваний результат: Індекс 25006 Область Кіровоградська Район - Населений пункт - ВПЗ, що обслуговує (порожній)	passed
Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська -“-” -”-” -”-” Бажаємо вивести таблицею(Y)	Вхідні дані: -test1.txt Очікуваний результат: Індекс 25006 Область Кіровоградська Район - Населений пункт - ВПЗ, що обслуговує -	passed

Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська -Миколаївський район -Кропивницький -Кроп-4 Бажаємо вивести таблицею(Y)	Вхідні дані: -test12ва1.txt Очікуваний результат: Індекс 25006 Область Кіровоградська Район Миколаївський район Населений пункт Кропивницький ВПЗ, що обслуговує Кроп-4	passed
Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська -Миколаївський район -Кропивницький -Кроп-4 Бажаємо вивести таблицею(Y) та перевіряємо чи існує він	Вхідні дані: -test1.txt –25006 -Кіровоградська -Миколаївський район -Кропивницький –”-” Очікуваний результат: Такого запису немає у файлі. Перевірте введену інформацію.	passed
Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська -Миколаївський район -Кропивницький -Кроп-4 Бажаємо вивести таблицею(Y) та перевіряємо чи існує він	Вхідні дані: -test1.txt –25006 -Кіровоградська -Донецький район -Кропивницький –Кроп-4 Очікуваний результат: Такого запису немає у файлі. Перевірте введену інформацію.	passed
Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська -Миколаївський район -Кропивницький -Кроп-4 Бажаємо вивести таблицею(Y) та перевіряємо чи існує він	Вхідні дані: -test1.txt –25006 -Кіровоградська -Миколаївський район -Кропивницький –Кроп-4 Очікуваний результат: Так, такий запис є у файлі	passed

Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська -Миколаївський район -Кропивницький -Кроп-4 Бажаємо вивести таблицею(Y) Перевіряємо чи існує він Бажаємо видалити Департамент з файлу(Y)	Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська -Миколаївський район -Кропивницький -Кроп-4 Очікуваний результат: File is not found	passed
Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська -Миколаївський район -Кропивницький -Кроп-4 Бажаємо вивести таблицею(Y) Перевіряємо чи існує він Бажаємо видалити Департамент з файлу(Y)	Вхідні дані: -Y -test1.txt -1 –25006 -Кіровоградська -Миколаївський район -Кропивницький -Кроп-4 Очікуваний результат: Успішно видалено!	passed

Результат:
Помилка у першій теці була виправлена та відредагована.
Результат можна побачити у другій теці.

Висновок:

Мета цієї роботи полягала у набутті практичних вмінь і навичок командної (колективної) реалізації програмного забезпечення, розроблення функцій оброблення динамічних структур даних, тестування програмного забезпечення, використання стандартних засобів C++ для керування динамічною пам’яттю та бінарними файловими потоками.Сам проект був реалізований у середовищі Code::Blocks.
Хід нашої роботи:
-Завдання були розподілені між учасниками цього проекту;
-Почалася реалізація кожного блоку послідовно за іншим;
-Півненко Олександр реалізував своє завдання;

-Саме завдання було скомпільоване та протестоване для знаходження певних помилок, після чого вони були виправлені і занесені у фінальну версію нашого коду;
-Після реалізації першої частини завдання Рудь Ігор взяв саму бібліотеку та починав реалізувати власну частину роботи:
-Після того, як Рудь Ігор реалізував своє завдання воно було скомпільоване та протестоване для знаходження певних помилок, після чого вони були виправлені і занесені у фінальну версію нашого коду;
-Останньою частину завдання реалізував Осадчий Влад;
-Кожну частину застосунку, що був створений Ігорем Рудем, було протестоване і помилки в реплізації були виправлені;
-Фінальну версію коду було додано до звіту;
-Після чого фінальну версію завдання було запушено в git.