**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

**ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ**

**ЗВІТ**

**З навчальної практики**

**Про виконання проектів**

**Виконали:**

**Студенти 2-го курсу**

**Групи ПМп-22**

**Команда "vVv":**

**Настосяк Віталій**

**Лозинський Степан**

**Стахів Юлія**

**Гаєвська Ірина**

**Львів-2017**

**ЗМІСТ**

**-----------------------------------------------------**

Проект №1

**--- 1)Тема -----------------------------------------------------5 ст.**

**--- 2)Загальна мета ----------------------------------------5 ст.**

**--- 3)Постановка завдання -------------------------------5 ст.**

**--- 4)Шляхи реалізації ------------------------------------6 ст.**

**--- 5)Хід виконання ----------------------------------------6 ст.**

**--- 6)Особливості продукту ------------------------------6 ст.**

**--- 7)Перелік команд --------------------------------------7 ст.**

**--- 8)Інструкція з використання -----------------------8 ст.**

**--- 9)Презентація результатів ---------------------------9 ст.**

**--- 10)Список учасників, та їх внески ----------------9 ст.**

--------------------------------------------------------------------------

Проект №2

**--- 1)Тема -----------------------------------------------------9 ст.**

**--- 2)Загальна мета ----------------------------------------9 ст.**

**--- 3)Постановка завдання -------------------------------9 ст.**

**--- 4)Шляхи реалізації ------------------------------------10 ст.**

**--- 5)Хід виконання ----------------------------------------10 ст.**

**--- 6)Особливості продукту ------------------------------10 ст.**

**--- 7)Перелік команд --------------------------------------12 ст.**

**--- 8)Інструкція з використання -----------------------12 ст.**

**--- 9)Презентація результатів ---------------------------12 ст.**

**--- 10)Список учасників, та їх внески ----------------12 ст.**

--------------------------------------------------------------------------

Проект №3

**--- 1)Тема -----------------------------------------------------13 ст.**

**--- 2)Загальна мета ----------------------------------------13 ст.**

**--- 3)Постановка завдання -------------------------------13 ст.**

**--- 4)Шляхи реалізації ------------------------------------14 ст.**

**--- 5)Хід виконання ----------------------------------------15 ст.**

**--- 6)Особливості продукту ------------------------------16 ст.**

**--- 7)Перелік команд --------------------------------------16 ст.**

**--- 8)Інструкція з використання -----------------------16 ст.**

**--- 9)Презентація результатів ---------------------------17 ст.**

**--- 10)Список учасників, та їх внески ----------------17 ст.**

--------------------------------------------------------------------------

Проект №4

**--- 1)Тема -----------------------------------------------------17 ст.**

**--- 2)Загальна мета ----------------------------------------18 ст.**

**--- 3)Постановка завдання -------------------------------18 ст.**

**--- 4)Шляхи реалізації ------------------------------------18 ст.**

**--- 5)Хід виконання ----------------------------------------19 ст.**

**--- 6)Особливості продукту ------------------------------20 ст.**

**--- 7)Перелік команд --------------------------------------20 ст.**

**--- 8)Інструкція з використання -----------------------21 ст.**

**--- 9)Презентація результатів ---------------------------22 ст.**

**--- 10)Список учасників, та їх внески ----------------22 ст.**

--------------------------------------------------------------------------

Проект №5

**--- 1)Тема -----------------------------------------------------1 ст.**

**--- 2)Загальна мета ----------------------------------------1 ст.**

**--- 3)Постановка завдання -------------------------------1 ст.**

**--- 4)Шляхи реалізації ------------------------------------1 ст.**

**--- 5)Хід виконання ----------------------------------------1 ст.**

**--- 6)Особливості продукту ------------------------------1 ст.**

**--- 7)Перелік команд --------------------------------------1 ст.**

**--- 8)Інструкція з використання -----------------------1 ст.**

**--- 9)Презентація результатів ---------------------------1 ст.**

**--- 10)Список учасників, та їх внески ----------------1 ст.**

--------------------------------------------------------------------------

Проект №6

**--- 1)Тема -----------------------------------------------------22 ст.**

**--- 2)Загальна мета ----------------------------------------22 ст.**

**--- 3)Постановка завдання -------------------------------22 ст.**

**--- 4)Шляхи реалізації ------------------------------------23 ст.**

**--- 5)Хід виконання ----------------------------------------23 ст.**

**--- 6)Особливості продукту ------------------------------24 ст.**

**--- 7)Перелік команд --------------------------------------24 ст.**

**--- 8)Інструкція з використання -----------------------24 ст.**

**--- 9)Презентація результатів ---------------------------24 ст.**

**--- 10)Список учасників, та їх внески ----------------24 ст.**

--------------------------------------------------------------------------

**Висновок -----------------------------------------------------25 ст.**

**Проект № 1**

**Тема:** Реалізація відомих методів сортування масиву.

**Загальна мета:** Виконати поставлене завдання шляхом командної взаємодії, пригадати та реалізувати відомі алгоритми сортування, здобути навички роботи в GitHub.

**Постановка завдання:**

Реалізувати програму, що дає змогу відсортувати масив різними алгоритмами сортування.

**Шляхи реалізації:**

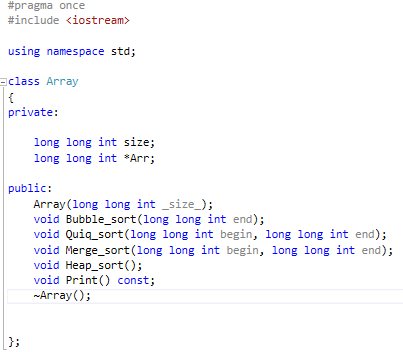
Поставлене завдання ми вирішили виконати в одному класі, де були створені методи, які сортують масив за різними алгоритмами. Також, в головному меню користувач має повинен обрати спосіб сортування.

Було реалізовано сортування бульбашкою, швидке, пірамідальне, а також сортування злиттям.

Крім цього, ми вирішили додади пункти створення нового масиву і виходу з програми. Для візуалізації результатів, також ми вирішили обраховувати за який час виконується сортування.

**Хід виконання:**

Спочатку, був створений клас, консткуктор якого створює масив за допомогою генератору чисел. Далі добавлялися методи, які сортують масив. На завершення, створили меню з вибором команд.



На рисунку зображені прототипи методів класу.

**Особливості продукту:**

Наша програма може відсортувати масив, однак не зберігає не відсортований (послідовність не відсортованого). Оскільки, масив чисел задається генератором чисел, то кожен новий масив, буде містити інші елементи.

**Перелік команд:**

Enter N:

1) Quiq sort;

2) Marge sort

3) Heap sort

4) Bubble sort

5) New array

6) Exit.

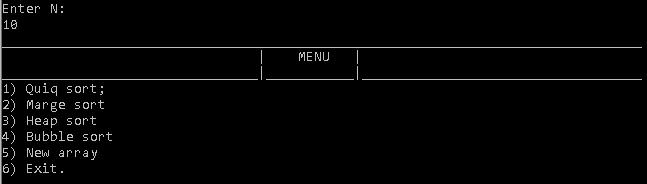
**Інструкція з використання:**

При запуску програми вводимо кількість елементів



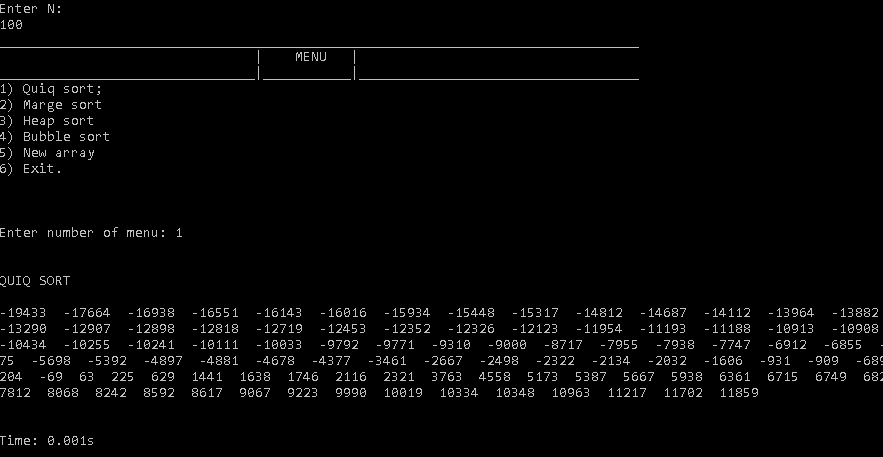
масиву "Enter N:",

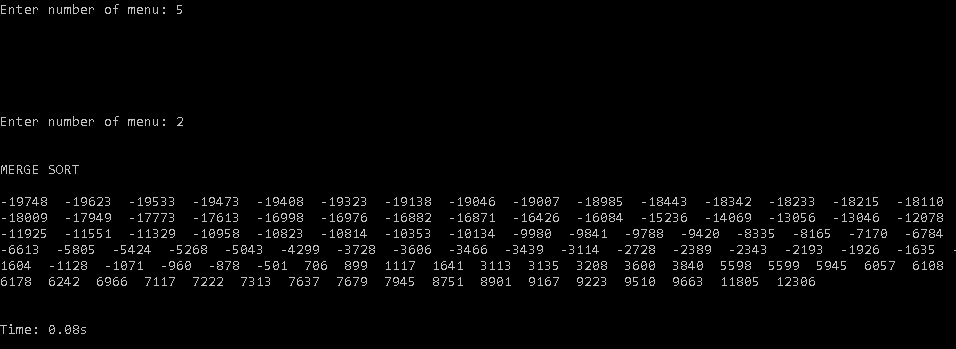
а потім обираємо спосіб сортування 1-4,

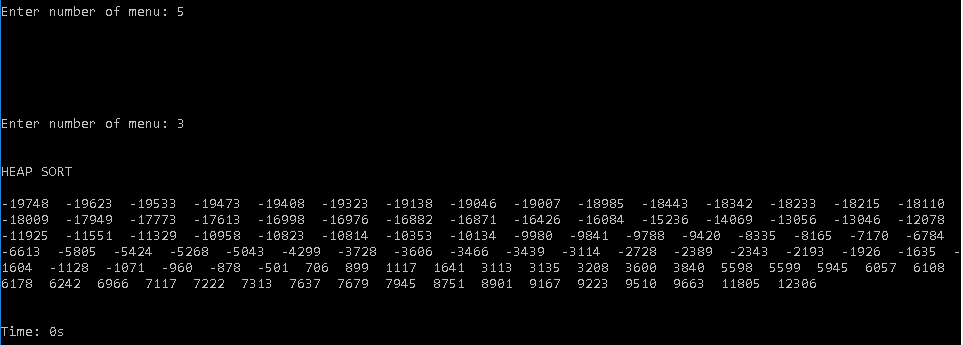


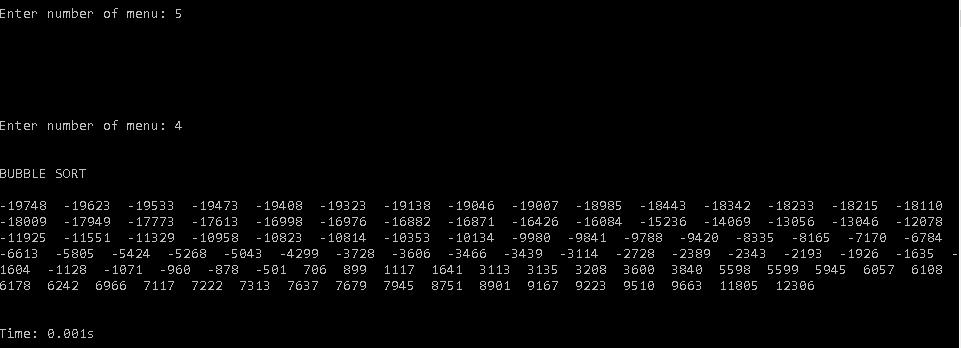
або 5 створення нового масиву. Пункт 6 завершує роботу програми.

**Презентація результатів:**







**Список учасників, та їх внески:**

Настосяк Віталій - пірамідальне сортування (Heap\_sort).

Стахів Юлія - швидке сортування (Quiq\_sort).

Гаєвська Ірина - сортування бульбашкою (Bubble\_sort).

Лозинський Степан - сортування злиттям (Merge\_sort).

**Проект № 2**

**Тема:** Робота з BMP файлами.

**Загальна мета:** Виконати поставлене завдання шляхом командної взаємодії, навчитися одержувати необхідну інформацію з файлу.

**Постановка завдання:** Закодувати та розкодувати слово з BMP файлу.

**Шляхи реалізації:**

Основними кроками виконання завдання є знаходження інформації про кожен піксель, кодування та декодування з картинки.

**Хід виконання:**

Спочатку обирали картинку необхідного розширення, враховували розмір (від цього залежить скільки інформації можна закодувати у пікселі). Show\_info - ми вирішили показувати інформацію про даний BMP файл.Також реалізували алгоритм перетворення слова у двійковий код. Змінювали останні коди пікселів для кодування, і відтворювали для декодування (для цієї команди важливою є кількість закодованої інформації після введення слова користувачем).

Прототипи функцій:

void get\_info() const;

void PrintMassage() const;

void DeCoder();

void Coder(char\* \_Massage\_, char\* way);

**Особливості продукту:**

Користувач має змогу закодувати і декодувати лише одне слово. Також для декодування необхідно знати кількість змінених бітів , або довжину закодованого слова.

Кінець введеного слова ~ для кодування!

**Перелік команд:**

1) Show\_info;

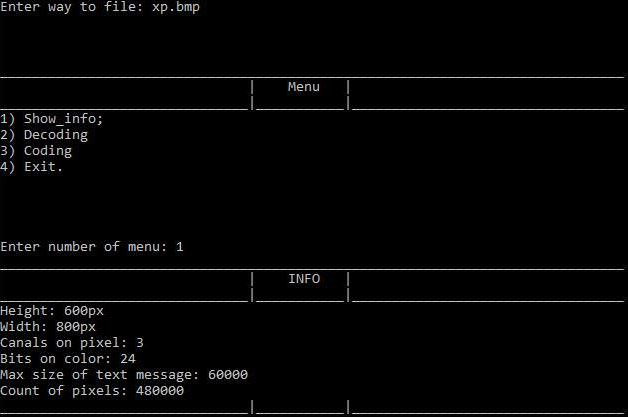
2) Decoding;

3) Coding;

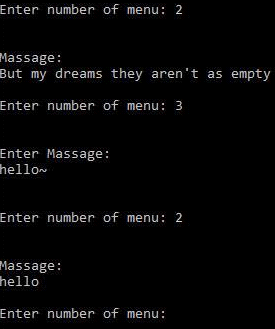
4) Exit.

**Інструкція з використання:**

Спочатку необхідно перевірити наявність файлу для виконання операції. Вибираємо шлях до файлу. Вибираємо 1 , для того щоб переконатись кількість інформації яку можна закодувати.



Вибираємо 2 і закодовуємо слово (кінець слова ~), 3 і декодуємо , переконуємось у правильності виконаних операції.



4 - завершуємо роботу.

**Список учасників, та їх внески:**

Настосяк Віталій - робота з отриманням інформації про доступні "комірки" для кодування.

Стахів Юлія - кодування.

Гаєвська Ірина - тестування.

Лозинський Степан - декодування.

**Проект № 3**

**Тема:** Створення плейлиста.

**Загальна мета:** Виконати поставлене завдання шляхом командної взаємодії, створювати підбірки вибраних пісень з папок, роботам з mp3 файлами.

**Постановка завдання:** Створити консольну програму, яка повинна проходитись по документах (папках) і знаходити , читати , виділяти інформацію з mp3 файлу. Також, однією з функцій повинно бути створення плейлиста на основі вибраних програм.

**Шляхи реалізації:**

Першим етапом виконання завдання є створення панелі керування, яка дозволяє вибирати папку, заходити в неї, відображати наявні папки, сканувати на наявність mp3 файлів.

Другим етапом є створення менеджеру плейлистів, який повинен вибирати пісні, добавляти їх до створеного плейлисту, обирати інший плейлист.

Також, ми вирішили втілити відтворення інформації пісні, а саме назву,автора і дата створення.

**Хід виконання:**

**Спочатку були створені менеджер з опрацюванням каталогів (обирання папок, відтворнення їх вмісту і тд.).**

**Далі**

**Особливості продукту:**

**Перелік команд:**

**Інструкція з використання:**

**Презентація результатів:**

**Список учасників, та їх внески:**

Настосяк Віталій - .

Стахів Юлія - .

Гаєвська Ірина - .

Лозинський Степан - .

**Проект № 4**

**Тема:** Робота з менеджером польотів.

**Загальна мета:** Виконати поставлене завдання шляхом командної взаємодії, навчитись створювати менеджер польотів для авіакомпанії.

**Постановка завдання:**

Реалізувати програму, яка дає змогу користувачеві керувати даними про продаж квитків для здійснення авіаперевезень.

Головною вимогою є створення замовлень на певний авіарейс, внесенням інформації у базу даних, редагування інформації.

Головні положення:

а) користувач робить замовлення, добавляючи інформацію про

- назву літака;

- бортовий номер літака;

- назву місця і час (відліт – приліт)

- дані про пасажира, аеропорт, рейс, посадкове місце;

- ціна квитка;

б) користувач повинен мати змогу створювати і скасовувати замовлення;

в) інформація повинна також міститися у файлах (зчитування-запис);

г) потрібно обчислювати загальний прибуток компанії від здійснення всіх рейсів.

**Шляхи реалізації:**

Обговорюючи поставлене завдання, ми прийшли до висновку, що зручним способом виконання поставленого завдання є реалізація кожного важливого пункту через окремий клас. Для цього, ми виділили такі приорітетні керуючи дані:

а) пасажир;

б) аеропорт;

в) літак;

г) рейс;

д) місце.

Ці дані ми реалізували вже у самому класі Замовлення (або ж квиток).

**Хід виконання:**

Спочатку створювались класи, які будуть перелічені нижче, добавлялись необхідні поля і методи. Згодом все це скомбінувати в менеджері, який дає змогу користувачеві використовувати дану програму без труднощів.

**Особливості продукту:**

Перш за все, наш продукт дає змогу доступитися до важливої інформації. Оскільки даний проект реалізовано через класи,

то Ви маєте змогу з легкістю можете знайти потрібну для себе інформацію доступившись до полів потрібного вам класу.

Наприклад:

-клас пасажир - інформація про пасажира, необхідну для реєстрації (ім'я та прізвище пасажира, стать , національність , номер мобільного телефону, серія та номер паспорта);

-клас Plane- інформація про назву бортового номера , та кількість місць ;

-клас Airport - інформація про назву, країну , місто , контактний номер;

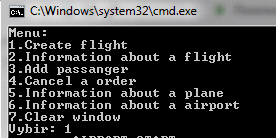
-клас Place - інформація про номер місця та його статус (вільно-зайнято);

-клас Flight - інформація про місце і час приліт-відліт.

Так само , користувач має змогу змінити цю ж інформацію. Крім того, інформація зберігається у текстових документах.

**Інструкція з використання:**

При запуску програми з’являється вікно з меню:



Користувач вводить певний пункт меню і виконує інші кроки, які будуть з’являтися на екран.

**Список учасників, та їх внески:**

Настосяк Віталій –класи Plane, Flight, менеджер.

Стахів Юлія - клас Place.

Гаєвська Ірина – клас Passenger.

Лозинський Степан –клас Airport.

**Проект № 5**

**Тема:**

**Загальна мета:**

**Постановка завдання:**

Написати бібліотеку, що дає змогу працювати з масивами, одно- і двозв'язними списками,стеками, чергами, передбачити створення, додавання/вилучення елемента в кінець, на початок,на конкретну позицію, дописування одного до іншого, обертання порядку елементів, сортування,пошук. Для деяких типів, наприклад, стека, не всі ці операції можливі, тому необхідно обдумати,що має зміст реалізовувати, а що - ні.

**Шляхи реалізації:**

Для того, щоб реалізувати програму, необхідно розуміти, що саме являє собою стек, черга, одно- та двозв’язні списки.

Отож:

Стек — різновид лінійного списку, структура даних, яка працює за принципом (дисципліною) «останній прийшов — перший пішов».

Черга (англ. queue) — динамічна структура даних, що працює за принципом «перший прийшов — перший пішов».

Однозв'язний список — вид зв'язаного списку, який складається з вузлів, кожен з яких містить у собі данні (інформаційну частину) та посилання на наступний вузол.

Двозв'язний список — вид зв'язаного списку, у якому посилання в кожному вузлі вказують на попередній і на подальший вузол у списку.

**Інструкція з використання:**

Отже, ця програма дає змогу працювати із однозв'язними списками, двоз'язними списками, стеками та чергами.

В програмі описані такі операції із стеками:

\* pushBack («заштовхнути елемент»): елемент додається в стек та розміщується в його верхівці. Розмір стеку збільшується на одиницю.

\* pop («виштовхнути елемент»): отримує елемент з верхівки стеку. При цьому він видаляється зі стеку і його місце в верхівці стеку займає наступний за ним відповідно до правила LIFO(Last In, First Out), а розмір стеку зменшується на одиницю.

Функції оголошені у файлі Stack.h, а їх опис у файлі Stack.cpp.

Щоб доступитися до цих функцій, в Source.cpp необхідно створити об'єкт типу Stack, і доступатися до функцій через крапочку (".").

Клас Stack шаблонний (це означає, що ви можете задавати різні типи даних).

\* \* \*

Операції із чергами такі ж як і з стеками, і доступаються до них аналогічно.

Функції оголошені у файлі Queue.h, а їх опис у файлі Queue.cpp.

\* \* \*

В програмі описані такі операції із однозв'язними списками:

\* pushBack - елемент додається в список та розміщується в його верхівці.

\* pop - отримує елемент з верхівки списку.

\* leftpop - видаляє перший елемент зліва.

\* clear - видаляє всі елементи списку.

\* head\_on\_tail - міняє місцями перший елемент списку з останнім.

\* remove - видаляє елемент списку, вибраний користувачем (користувач задає елемент який хоче видалити в одиничних лапках, наприклад pushback('4')).

\* insert - вставляє елемент у список на певну позицію (користувач задає елемент(в одиничних лапках), який хоче вставити у список і номер позиції після якої цей елемент повинен стояти; наприклад insert ('T',1), внаслідок цієї функції елемент Т вставиться на 2 позицію у списку, при цьому розмір списку збільшиться на 1).

\* reverse - розвертає список.

\* sort - сортує список (методом швидкого сортування).

Також можливе додавання 2 списків.

Функції оголошені в файлі List.h, а їх опис у файлі List.cpp.

Доступаємося до цих функцій, так як і з чергами та стеками.

\* \* \*

Операції їз двозв'язними списками такі ж як із однозв'язними.

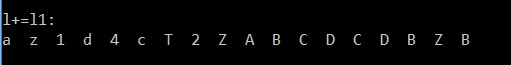
Функції описані в файлі Doubly\_list.h, а їх опис у файлі Doubly\_list.cpp.

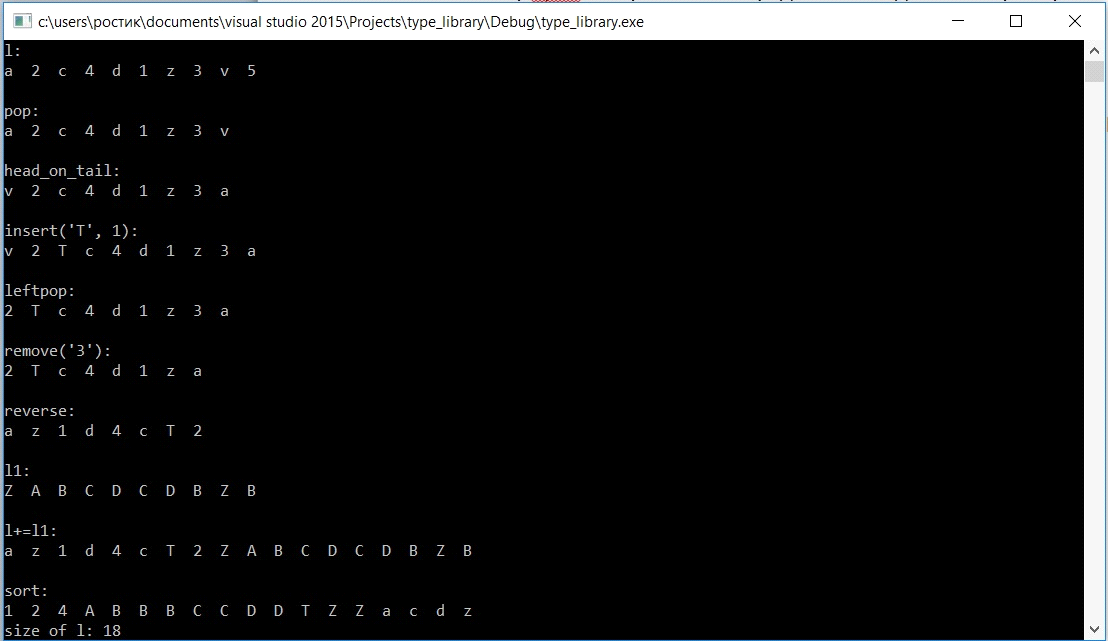
**ПРЕЗЕНТАЦІЯ РЕЗУЛЬТАТІВ:**

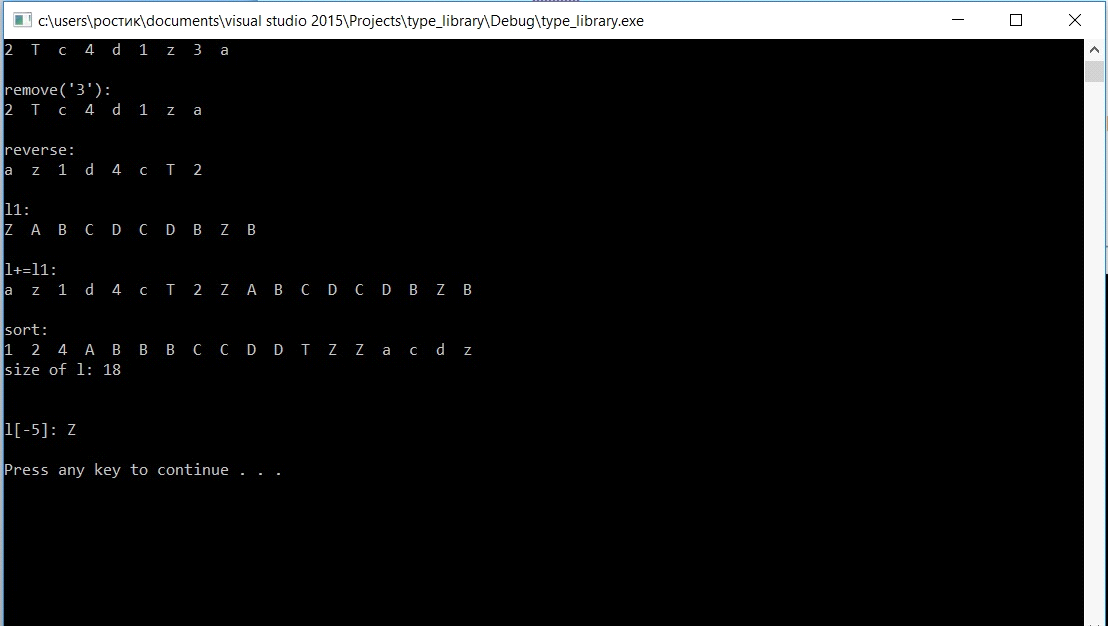
Приклад реалізації програми.

Створений однозв’язний список ***l*** і продемонстровані всі операції, які можна виконувати із цим списком.

Також створений ще один список ***l1*** для представлення функції додавання 2 списків.







**Список учасників, та їх внески:**

Лозинський Степан реалізував роботу із чергами.

Стахів Юлія реалізувала роботу із стеками.

Настосяк Віталій реалізував роботу із одно- та двозв’язними списками і звів програму до робочого стану.

**Проект № 6**

**Тема:** Опрацювання многокутників.

**Загальна мета:** Виконати поставлене завдання шляхом командної взаємодії, а також втілити за власною ідеєю відображення результатів на кооринатній площині.

**Постановка завдання:**

Створити програмний продукт для роботи з двовимірними многокутниками. Проектування включає опис класів, які необхідні для імплементації необхідної функціональності (очевидно, що, окрім класу "многокутник", як мінімум, необхідно буде ввести клас "точка").

Функціональність:

* Додавання многокутників: результат - новий многокутник ;
* Віднімання многокутників: результат - новий многокутник ;
* Клонування многокутника: результат - новий об'єкт многокутника, що має абсолютно такі ж параметри '
* Масштабування многокутників: результат - "розтягнутий" або "стиснений" многокутник відносно центру мас;
* Масштабування многокутників з клонуванням: результат - "розтягнутий" або "стиснений" многокутник відносно центру мас ;
* Перенесення многокутника: результат - зміщений на певні величини по X та Y многокутник ;
* Перенесення з клонуванням многокутника: результат - зміщений на певні величини по X та Y многокутник .

**Шляхи реалізації:**

Для створення многокутника нам потрібні точки , та лінії , тому ми створили 3 класи : Polygon, Line, Point , - які є взаємоповязані. Многокутник складається з ліній , а лінії з точок (вершин многокутника). Тому при створенні об'єкта многокутника, ми використовуємо створення таких об'єктів як ліній та точок, які входять у цей многокутник. Фактично, ми реалізували створення многокутника за масивом ліній. Для відображення самого многокутника, ми вирішили використати стандартну бібліотеку "windows.h".

Деякі функції, такі як множення на скаляр (розтягування та стиснення), клонування, зміщення , ми реалізували як перевантаження операторів \*, = ,+=.

Також ми вирішили створити консольне меню, яке дозволяє виконувати необхідні операції, за введенням команд. Список цих команд відображається, при введені "help".

**Хід виконання:**

Спочатку було створені наш основний "кістяк, або скелет" - це самі класи Polygon, Line, Point. Так у класі Point були реалізовані методи необхідні для визначення положення точки (координати X та Y); у класі Line реалізовано вже безпоседеднє використання двох точок для створення лінії (A і B відповідно початок і кінець лінії), також було реалізована функція для її відображення; у класі Polygon було створено масив ліній створеного многокутника, за якими він відображався.

Для завершення даного завдання було створено наступні методи необхідні для повної функціональності програми: move x або y, move point, zoom, draw.

Для зручності використання було створене консольне меню зі застосуванням цих команд (реалізовано у функції devide).

**Особливості продукту:**

Як раніше згадувалось, многокутник будується на основі масиву ліній. Відображається многокутника відбувається на створеній координатній площині відповідно від знайденого центру консолі. Реалізовано консольне використання команд для простішого застосування.

**Перелік команд:**

**"--**p new" - створення многокутника;

"--p move\_x" - зміщення по X;

"--p move\_y" - зміщення по Y;

"--p move\_point" - зміщення по X і Y;

"--p zoom" - масштабування;

"--c draw" - відображення на координатній площині.

**Інструкція з використання:**

Після запуску програми, за допомогою введення команди "help" переглядаємо доступні команди, та як їх використовувати. **ЗАУВАЖЕННЯ:** вводити з точністю до знаків!

При введенні "--p new" необхідно ввести кількість вершин створеного многокутника, та ввести їх координати відповідно як x1,y1 потім x2,y2 і т.д. до xn,yn, де n - кількість вершин.

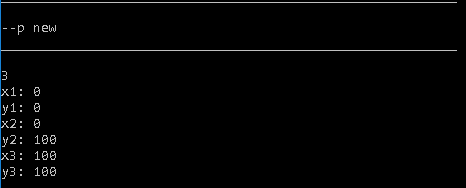
Далі вводимо необхідні команди.

"--p move\_x" та "--p move\_y" вводимо число зміщення.

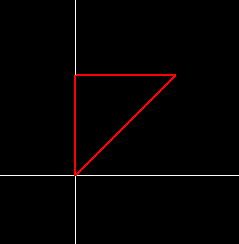
"--p move\_point" вводимо координати зміщення (пара чисел).

**Презентація результатів:**

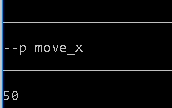
Створення многокутника



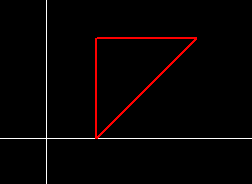
Зображення



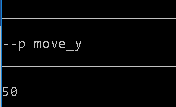
Зміщення по X



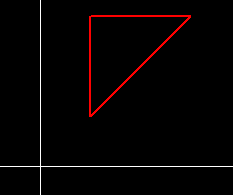
Зображення



Зміщення по Y



Зоображення



**Список учасників, та їх внески для реалізації:**

Настосяк Віталій - класи Polygon, Line, Point, функції devide і меню, draw.

Стахів Юлія - operator +, operator += .

Гаєвська Ірина - operator \*, operator \*=.

Лозинський Степан - operator =, меню.

**ВИСНОВОК:** В результаті виконання даних проектів, ми навчилися командній взаємодії та співпраці, ефективному розподілу завдань, виконувати завдання завдяки GitHub, що дозволяє виконувати зміни у програмному коді кожному члену команди, не впливаючи на негативні зміни, руйнування проекту. Також, ми навчилися частково розподіляти завдання, втілювати поєднання ідей у створенні проектів**.**

Дані завдання закріпили наші навички роботи у програмному середовищі, навички програмування мови C ++ , використання різних типів та структур даних, обробка та зчитування інформацій з файлів різних типів.