

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра обчислювальної техніки

(повна назва кафедри, циклової комісії)

КУРСОВА РОБОТА

з дисципліни «Об'єктно-орієнтоване програмування»

(назва дисципліни)

на тему: «Створення програми обробки табличних даних»

Студента II курсу групи ІП-64

напряму підготовки _____

спеціальності Програмна інженерія

Вінницького В.А.

(прізвище та ініціали)

Керівник: ст. викл. Порєв В. М.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна оцінка _____

Кількість балів: _____

Оцінка: ECTS _____

Члени комісії

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2017 рік

ЗМІСТ

ЗМІСТ	1
ЗАВДАННЯ	3
ОБГРУНТУВАННЯ ПРОЕКТНОГО РІШЕННЯ	5
АНАЛІЗ МОЖЛИВИХ ВАРІАНТІВ РІШЕННЯ ЗАДАЧІ	10
ТЕОРЕТИЧНІ ПОЛОЖЕННЯ	11
ОПИС РЕАЛІЗАЦІЇ ПРОЕКТУ	18
ВИСНОВКИ	Ошибка! Закладка не определена.
ПЕРЕЛІК ПОСИЛАНЬ	21
ВИХІДНІ ТЕКСТИ ВСІХ МОДУЛІВ ПРОГРАМИ	22
ДІАГРАМА КЛАСІВ	37
ДІАГРАМА #INCLUDE-ІЄРАРХІЇ МОДУЛІВ	38
СКРІНШОТИ РОБОТИ ПРОГРАМИ	39

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 43 сторінок, 11 рисунків, 8 посилань.

Об'єкт дослідження: редагування табличних даних.

Мета роботи: дослідження методів роботи з табличними даними, які задаються в текстовому файлі.

Дана курсова робота включає в себе: аналіз методів, за допомогою яких можна вирішити поставлену задачу, опис оптимального методу вирішення поставленої задачі, теоретичні відомості до цих методів, код програми вирішення перерахованих вище методів на мові програмування C++, діаграму класів, include-ієрархію, описання детального процесу розв'язання поставленої задачі, за допомогою обраного методу, скріншоти виконання програми.

ЗАВДАННЯ

Завдання 2. Запрограмувати програму обробки табличних даних.

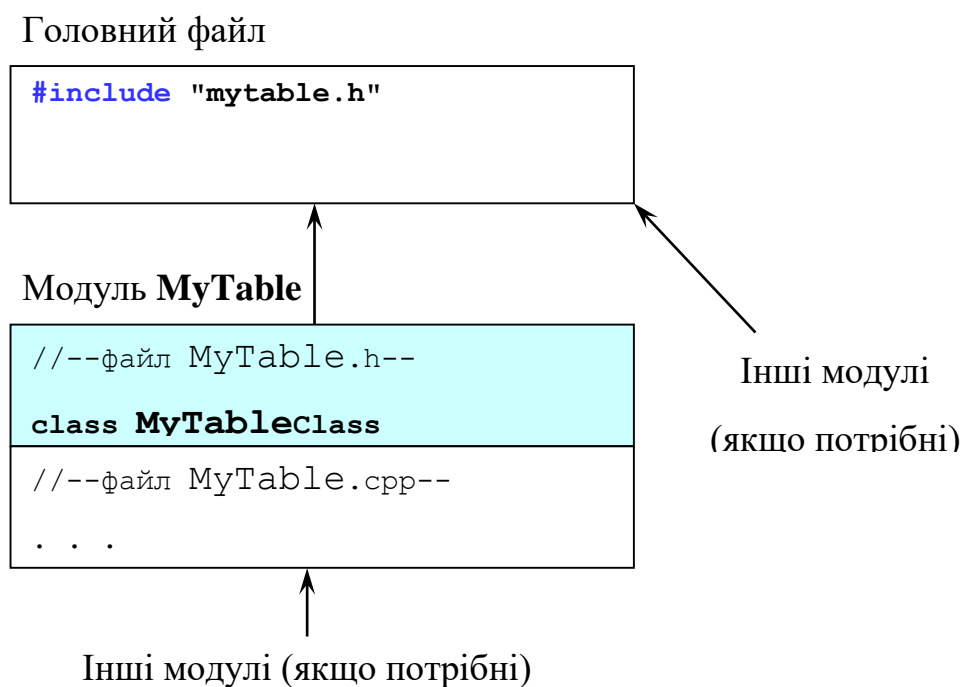
Вимоги

Функціональні вимоги:

1. Таблиця повинна відображатися у головному вікні програми. Потрібно передбачити скролінг по горизонталі та вертикалі, якщо зображення таблиці більше клієнтської частини вікна. Передбачити коректне реагування на можливу зміну розмірів головного вікна.
2. Таблиця завантажується так: після вибору меню "Файл – Відкрити. . ." спочатку з'являється стандартне діалогове вікно (GetOpenFileName), у якому показується список папок та файлів. У цьому вікні вибирається потрібний текстовий файл, який зберігає таблицю. Дані з файлу завантажуються у пам'ять і потім показуються у вигляді таблиці.
3. Програма повинна вміти записувати таблицю у текстовий файл. Для цього передбачити пункти меню "Файл–Записати", та "Файл–Записати як. . .". У останньому випадку вибір нового імені файлу через стандартне вікно (GetSaveFileName).
4. Програма читає та записує таблиці у текстовому форматі "Delimited ASCII", у якому кожний рядок комірок таблиці – це рядок текстових значень, розділених символами табуляції ('\t'). Перший рядок файлу – це назви стовпчиків, у наступних рядках записуються рядки таблиці.
5. Користувач може редагувати кожну комірку таблиці у головному вікні програми. Курсором миші вказується потрібна комірка і далі прямо на площині таблиці відкривається невеличке вікно-комірка вводу тексту. Якщо натиснути "Enter", то зміни відбудуться, якщо "Esc" – то змін не буде.
6. Передбачити можливість відміни останньої операції редагування – натисканням клавіш "Ctrl + Z" або меню "Редагування – Відміна".

Вимоги щодо програмування:

1. Запрограмувати таблицю у вигляді класу `MyTableClass`, програмний код його повинен міститися у **окремому модулі**, який складатиметься з двох файлів
 - файл `MyTable.cpp`
 - файл `MyTable.h`
2. Інтерфейс модуля у вигляді оголошення класу `MyTableClass` у файлі `MyTable.h`.
3. Ієрархія модулів повинна бути такою:



4. Методи класу `MyTableClass` повинні забезпечувати основні функції показу та редагування таблиць.
5. За необхідності, інтерфейс модуля **може бути** зроблений у вигляді **декількох класів**, наприклад, двох. Тоді це потрібно обгрунтувати.
6. Деякі вимоги (функціональні та реалізації) повинні узгоджуватися з викладачем

ОБГРУНТУВАННЯ ПРОЕКТНОГО РІШЕННЯ

Моя тема курсової роботи, а саме програма для обробки даних у вигляді таблиці є досить актуальною та необхідною на сьогодні.

Майже кожен день ми можемо спостерігати, як оброблюються якісь дані в табличній формі. Це може бути будь-що, як дані статистики, так і дані про оцінки з предметів. Найчастіше ж це, на мою думку, використовується в обробці баз даних, словників, реєстрів, каталогів.

Це є дуже зручно, адже в таблиці простіше орієнтуватися і сприймати інформацію користувачу. Також таблицю можна редагувати: змінювати її вміст. Саме в таких випадках, мій проект буде зручно використовувати.

Метою розробки програми є відображення даних у вигляді таблиці за допомогою WIN32API. Для розробки даної програми було використано технологію Win32API, яка дозволяє розробляти віконні додатки для платформи Windows. Win32API – це загальна назва цілого набору базових функцій інтерфейсів програмування додатків на операційних системах Microsoft Windows корпорації Microsoft. Windows API має переваги та недоліки. Першою і найбільш вагомою перевагою є швидкість, так як це функції безпосередньо самої системи Windows. Друга перевага – це надійність і стабільність роботи(якщо без використання сторонніх бібліотек). З недоліків можна виділити те, що це не є кросплатформена технологія, на відміну від тих же Windows Form або Qt.

Для обробки результатів програми було використано стандартний контейнер vector.

Vector знаходиться в заголовному файлі <vector>. Як і всі стандартні компоненти він знаходиться в просторі імен std. Даний інтерфейс емулює роботу стандартного масиву C, а також деякі додаткові можливості, наприклад автоматичної зміни розміру при додаванні або видаленні елементів. Всі

елементи вектора повинні бути одного типу. Наприклад, неможна зберігати разом дані типу `char` і `int` в одному і тому ж екземплярі вектора. `Vector` має стандартний набір методів для доступу до елементів, додавання і видалення, а також отримання кількості зберігаємих елементів.

Для реалізації відміни зміни було використано стек.

Стек в інформатиці та програмуванні — різновид лінійного списку, структура даних, яка працює за принципом (дисципліною) «останнім прийшов — першим пішов» (LIFO, англ. *last in, first out*). Всі операції (наприклад, видалення елементу) в стеку можна проводити тільки з одним елементом, який знаходиться на *верхівці стеку* та був введений в стек останнім.

Стек можна розглядати як певну аналогію до стопки тарілок, з якої можна взяти верхню, і на яку можна покласти верхню тарілку (інша назва стеку — «магазин», за аналогією з принципом роботи магазину в автоматичній зброї).

Операції зі стеком

`push` («заштовхнути елемент»): елемент додається в стек та розміщується в його верхівці. Розмір стеку збільшується на одиницю. При перевищенні розміру стека граничної величини, відбувається переповнення стека (англ. `stack overflow`).

`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, а розмір стеку зменшується на одиницю. При намаганні «виштовхнути» елемент з вже пустого стеку, відбувається ситуація «незаповнення» стеку (англ. `stack underflow`).

Кожна з цих операцій зі стеком виконується за фіксований час $O(1)$ і не залежить від розміру стеку.

Додаткові операції:

isEmpty: перевірка наявності елементів в стеку; результат: істина (true), коли стек порожній.

isFull: перевірка заповненості стека. Результат: істина, коли додавання нового елемента неможливе.

clear: звільнити стек (видалити усі елементи).

top: отримати верхній елемент (без виштовхування).

size: отримати розмір (кількість елементів) стека.

swap: поміняти два верхніх елементи місцями.

Організація в пам'яті комп'ютера

Стек може бути організований як масив або множина комірок в певній області комп'ютера з додатковим зберіганням ще й вказівника на верхівку стека. Заштовхування першого елемента в стек збільшує адресу вказівника, виштовхування елемента зменшує її. Таким чином, адреса вказівника завжди відповідає комірці масиву, в якій зараз знаходиться верхівка стеку.

Багато процесорів ЕОМ мають спеціалізовані регістри, які використовуються як вказівники на верхівку стеку, або використовують деякі з регістрів загального вжитку для цієї спеціальної функції в певних режимах адресації пам'яті.

Приклади застосування

Калькулятори, які використовують зворотну польську нотацію, використовують стек для збереження даних обчислень.

Існують «стеко-орієнтовані» мови програмування (Forth, PostScript), які використовують стек як базову структуру даних при виконанні багатьох операцій (арифметичних, логічних, вводу-виводу тощо).

Стеко-орієнтованими є деякі з віртуальних машин, наприклад віртуальна машина Java.

Компілятори мов програмування використовують стек для передавання параметрів в процесі виклику підпрограм, процедур та функцій. Спеціалізований стек використовується також для збереження адрес повернення з підпрограм.

Реалізація базових алгоритмів

На мовах програмування високого рівня, стек може бути реалізований за допомогою масиву та додаткової змінної: Для зберігання елементів стеку резервується масив $S[1..n]$ певного розміру та додаткова змінна $top[S]$, яка буде зберігати індекс верхівки стеку.

Операції push та pop тоді можуть бути записані так (без перевірки на переповнення та «незаповнення»):

PUSH (S, x)

1 $top[S] := top[S] + 1$ // збільшення індексу на 1

2 $S[top[S]] := x$ // запис нового елемента у верхівку стека

POP (S)

1 $top[S] := top[S] - 1$ // зменшення індексу на 1

```
2 return S[top[S] + 1] // повернення колишньої верхівки стеку
```

Результати роботи програми виводяться у вікно засобами GDI. GDI – це один з трьох основних компонентів, що разом з ядром і Windows API складають користувацький інтерфейс, розроблений для представлення графічних об'єктів і передачі їх на пристрої відображення, таких як монітори і принтери. Відповідає за растеризацію ліній і кривих, відображення шрифтів і обробку палітри.

АНАЛІЗ МОЖЛИВИХ ВАРІАНТІВ РІШЕННЯ ЗАДАЧІ

Поставлена переді мною задача, а саме відобразити дані у вигляді таблиці має багато різних варіантів. Ми можемо використати різні мови програмування, як і Java, C++, C#. Можемо також використовувати різні технології, алгоритми, бібліотеки. Але в даному випадку задача повинна бути вирішена за допомогою використання багатофункціональної мови програмування, з якою ми вже добре знайомі з першого курсу, а саме C++ та технології WIN32API.

Все-таки, зупинимось на альтернативах даного проектного рішення реалізованих на мові C++. Для написання віконного додатку можна було б використати кросплатформенну бібліотеку Qt або ж Windows Forms, розроблені корпорацією Microsoft.

Я в своїй курсовій використав контейнер vector, але також міг би використати і масив, в якому зберігалась би інформація з кожної комірки, але це є зовсім не зручно і не практично, також міг би використати масив контролів textbox, але це також є незручним і не практичним.

Також я не використовував стандартну бібліотеку listview, бо як на мене таблиця, яка будується з використанням цього класу є менш красивою та зручною для сприймання даних.

Ще для реалізації відміни змінення в таблиці я використав stack, що є також надзвичайно зручним підходом. Тому що можна було б використати запасний vector, в який би копіювалися всі попередні значення комірок при редагуванні таблиці. Але це забрало б багато ресурсів та часу.

Я вважаю, що я використав оптимальний та якісний підхід до виконання задачі, яку поставили переді мною.

ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Windows API — загальне найменування для цілого набору базових функцій інтерфейсів програмування застосунків операційних систем сімейства Windows корпорації Майкрософт. Є найпрямішим способом взаємодії застосунків з Windows. Для створення програм, що використовують Windows API, Майкрософт випускає SDK, який називається Platform SDK і містить документацію, набір бібліотек, утиліт і інших інструментальних засобів.

Windows API був спочатку спроектований для використання в програмах, написаних на мові C (або C++). Робота через Windows API — це найближчий до системи спосіб взаємодії з нею прикладних програм. Нижчий рівень доступу, необхідніший тільки для драйверів пристроїв, в поточних версіях Windows надається через Windows Driver Model.

Версії:

- **Win16** — перша версія Windows API для 16-розрядних версій Windows. Спочатку називався просто Windows API, потім почав називатися Win16 для розрізнення із Win32.
- **Win32s** — підмножина Win32, що встановлюється на сімейство 16-розрядних систем Windows 3.x і реалізовує обмежений набір функцій Win32 API для цих систем.
- **Win32** — 32-розрядний API для сучасних версій Windows. Найпопулярніша нині версія. Базові функції цього API реалізовані в DLL **kernel32.dll** і **advapi32.dll**; базові модулі GUI — в **user32.dll** і **gdi32.dll**. Win32 з'явився разом з Windows NT і потім був перенесений (у декілька обмеженому вигляді) в системи серії Windows 9x. У сучасних версіях Windows, що походять від Windows NT, роботу Win32

забезпечують два модулі: *csrss.exe (client/server Runtime Server Subsystem)*, що працює в призначеному для користувача режимі, і *win32k.sys* у режимі ядра.

- **win64** — 64-розрядна версія Win32, що містить додаткові функції для використання на 64-розрядних комп'ютерах. Win64 API можна знайти тільки в 64-розрядних версіях Windows XP і Windows Server 2003.

Graphics Device Interface, GDI — один з трьох основних компонентів або «підсистем», що разом з ядром і Windows API складають користувацький інтерфейс (віконний менеджер GDI) Microsoft Windows.

GDI — це інтерфейс Microsoft Windows для представлення графічних об'єктів і передачі їх на пристрої відображення, таких як монітори і принтери.

GDI відповідає за растеризацію ліній і кривих, відображення шрифтів і обробку палітри. Він не відповідає за растеризацію вікон, меню тощо. Це завдання закріплене за користувацькою підсистемою, що розташоване в *user32.dll* і ґрунтуються на GDI. GDI схожий з класичним QuickDraw у Apple.

Одна з найбільших переваг використання GDI замість методів прямого доступу до обладнання — це можливість розширення функцій над різними пристроями. Використовуючи GDI, можна легко малювати на кількох різних пристроях, таких як екран або принтер, і досягти практично однакового відображення на них. Ця можливість лежить у центрі всіх WYSIWYG застосунків для Microsoft Windows.

Прості ігри, які не потребують швидкої графіки, використовують GDI (отже, GDI - повільний задля швидкої графіки). Однак GDI не може забезпечити якісну анімацію (немає можливості синхронізації з кадровим буфером). Також GDI не вистачає растеризації для відмальовування 3D-графіки. Сучасні ігри використовують DirectX чи OpenGL, що дає програмістам доступ до більшої кількості апаратних можливостей.

В програмуванні, **послідовний контейнер** це ціла група шаблонів класів стандартної бібліотеки мови C++, які реалізують логіку контейнера, що виконує функцію зберігання елементів даних. Будучи шаблонами, вони можуть використовуватися для зберігання довільних елементів, як для цілих чисел так і для користувацьких класів. Спільною властивістю всіх послідовних контейнерів в тому, що доступ до елементів відбувається послідовно. Як і всі інші стандартні компоненти бібліотеки, вони знаходяться в просторі імен *std*.

В останньому стандарті C++ визначені наступні контейнери: `array`, `vector`, `list`, `forward_list`, `deque`. Кожен з цих контейнерів реалізує різні алгоритми зберігання даних, це означає, що вони мають різну швидкодію при виконанні різних операцій:

`array` реалізує масив незмінного розміру, що створюється під час компіляції.

`vector` реалізує масив із швидким довільним доступом і можливістю автоматичної зміни розміру при додаванні елементів.

`deque` реалізує двобічну чергу з порівняно швидким довільним доступом до елементів.

`list` реалізує двобічно зв'язаний список.

`forward_list` реалізує одnobічно зв'язаний список.

Оскільки кожен з контейнерів потребує можливості копіювати свої елементи для правильного функціонування, тип даних елементу має виконувати вимоги CopyConstructible і Assignable (мати конструктор копій і оператор присвоювання). У даного контейнера, всі елементи мають бути одного типу. Наприклад, не можливо одночасно зберігати дані типу `char` і `int` в одному контейнері.

Структура **OPENFILENAME** содержит информацию, которую используют функции **GetOpenFileName** и **GetSaveFileName**, чтобы инициализировать диалоговое окно **Открыть (Open)** или **Сохранить как (Save As)**. После того как пользователь закроет диалоговое окно, в этой структуре система возвращает информацию о выборе пользователя.

Синтаксис

```
typedef struct tagOFN {
```

```
    DWORD      lStructSize;
```

```
    HWND      hwndOwner;
```

```
    HINSTANCE  hInstance;
```

```
    LPCTSTR    lpstrFilter;
```

```
    LPTSTR     lpstrCustomFilter;
```

```
    DWORD      nMaxCustFilter;
```

```
    DWORD      nFilterIndex;
```

```
    LPTSTR     lpstrFile;
```

```
    DWORD      nMaxFile;
```

```
    LPTSTR     lpstrFileTitle;
```

```
    DWORD      nMaxFileTitle;
```

```
    LPCTSTR    lpstrInitialDir;
```

```
    LPCTSTR    lpstrTitle;
```

```
    DWORD      Flags;
```

```

WORD      nFileOffset;

WORD      nFileExtension;

LPCTSTR   lpstrDefExt;

LPARAM    lCustData;

LPOFNHOOKPROC lpfnHook;

LPCTSTR   lpTemplateName;

#if (_WIN32_WINNT >= 0x0500)

    void *    pvReserved;

    DWORD    dwReserved;

    DWORD    FlagsEx;

#endif      // (_WIN32_WINNT >= 0x0500)

} OPENFILENAME, *LPOpenFileName;

```

Композиція у програмуванні або ж **Об'єктна композиція**, також **Агрегація** та **включення** - це створення об'єктів існуючих класів як елементів інших класів. Про композицію також часто говорять як про «відношення приналежності» за принципом у «у машини є корпус, колеса і двигун».

Вкладені об'єкти нового класу зазвичай оголошуються закритими, що робить їх недоступними для прикладних програмістів, що працюють з класом. З іншого боку, творець класу може змінювати ці об'єкти, не порушуючи роботи існуючого клієнтського коду. Крім того, заміна вкладених об'єктів на стадії

виконання програми дозволяє динамічно змінювати її поведінку. Механізм наслідування такої гнучкості не має, оскільки для похідних класів встановлюються обмеження, що перевіряються на стадії компіляції.

На відміну від наслідування, в композиції тип відносин є *Has*-а тобто *має* (машина має двигун). В наслідуванні ж тип відносин між породженим об'єктом і батьківським є *Is*-а зв'язком, тобто якщо об'єкт *кішка* породжено від *тварина*, то кішка є тварина (*cat is a pet*).

fstream (скорочення від «*FileStream*») — заголовний файл з стандартної бібліотеки C++, який включає набір класів, методів і функцій, які представляють собою інтерфейс для читання/запису даних в файл. Для маніпуляції з даними файлів використовуються об'єкти, які називаються потоками («stream»).

Функції, включені в даний файл, дозволяють здійснювати читання файлів як побайтово, так і блоками, записувати так само двома способами. В комплект включені всі необхідні функції для управління послідовністю доступу до даних файлів, а також допоміжні функції.

Загальнодоступні функції:

Це базові функції, які не ввійшли ні в один з основних класів. Вони використовуються доволі часто, так як можуть бути викликані для всіх об'єктів потоків в кожному з класів.

- `open()`. Цим методом можна відкрити заданий файл, співставивши його з одним з об'єктів потоку. В залежності від аргументів, файл може бути відкритий для читання, для запису(або повної, або для доповнення), як бінарний або текстовий файл.

- `is_open()`. Функція, яка визначає чи файл відкритий. Повертає логічне значення. Використовується, в основному для попередження помилок доступу при спробі відкрити вже використовуваний файл. Без аргументів.
- `close()`. Функція закриває файл, тобто припиняє доступ до нього, таким чином звільняючи його для інших функцій або програм.

Базові класи

<code>ios_base</code>	« <i>InputStream_Base</i> », базовий клас всієї ієрархії класів потоків. Містить загальні функції, типи і класи, в основному, індикатори. Ці флаги використовуються підкласами <code>fstream</code> и можуть бути визначені за допомогою функцій <code>ios_base</code> .
<code>ios</code>	« <i>InputStream</i> », основний підклас, разом з <code>ios_base</code> , який визначає всі інші підкласи бібліотеки. Містить функції флаги форматування і обробки помилок, а також деякі функції унаслідовані від <code>ios_base</code> .
<code>ifstream</code>	« <i>InputFileStream</i> », організує читання даних з файлу.
<code>ofstream</code>	« <i>OutputFileStream</i> », організує запис даних в файл.

ОПИС РЕАЛІЗАЦІЇ ПРОЕКТУ

Для ефективного вирішення поставленої задачі, а саме: відображення даних у вигляді таблиці, було розроблено проект в середовищі Visual Studio 2013 на мові C++, з використанням WinAPI. На рис 1.1 зображено дерево проекту:

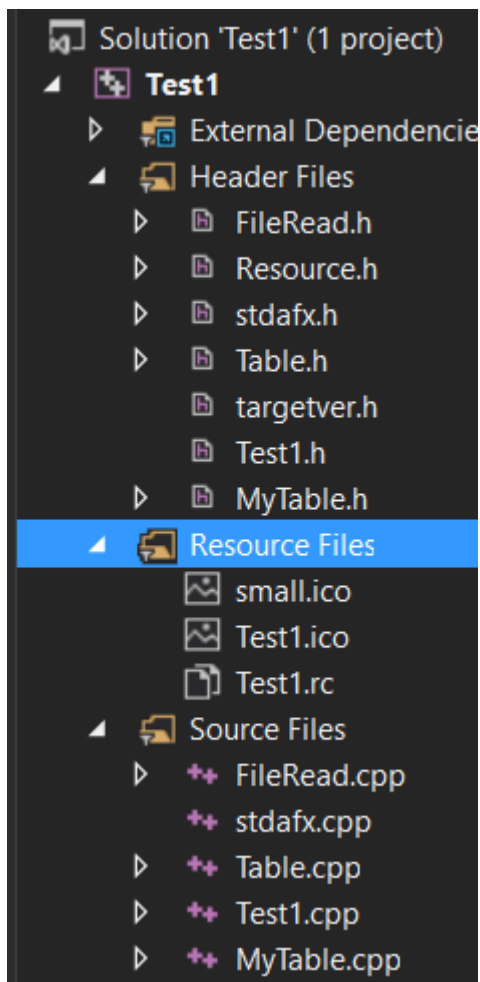


Рис 1.1 – Дерево проекту

Визначимо призначення файлів та класів проекту, опишемо функції, які виконує кожен файл:

- 1) FileRead.h, FileRead.cpp – файли, в яких ввiдбувається зчитування даних з файлу в таблицю(в клас Table)

- 2) Table.h, Table.cpp – файли, в яких зберігається головна інформація про таблицю, а саме кількість стовпців і рядків та сам текст з комірок.
- 3) MyTable.h, MyTable.cpp – файли, в яких відбувається відрисовка таблиці, визначення максимальної довжини таблиці, редагування та відміна.
- 4) Test1.h, Test1.cpp – головні файли програми, в яких відбувається обробка повідомлень головного вікна

ВИСНОВКИ

У результаті виконання курсової роботи було розроблено програмний комплекс, який дозволяє створювати таблицю на основі даних з файлу та редагувати її, з використанням технології Windows API та стандартних бібліотек C++.

Також були проведені деякі дослідження, щодо даного завдання, а саме дослідження на оптимальний метод розробки програми.

В ході виконання було досліджено роботу з допоміжними стандартними діалогами для вибору файлів, був вивчений і використаний популярний контейнер vector, було застосовано агрегацію та стек, досліджена робота з вікнами, з засобами відрисовки GDI.

Я вважаю, що даний проект є необхідним на сьогоднішній день, адже саме так зручніше тримати інформацію про якісь бази даних, список у таблиці, адже це дозволяє добре сприймати інформацію з екрана.

ПЕРЕЛІК ПОСИЛАНЬ

1. MSDN [https://msdn.microsoft.com/en-us/library/windows/desktop/ff818516\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ff818516(v=vs.85).aspx)
2. Вікіпедія https://en.wikipedia.org/wiki/Bubble_sort
3. Cppstudio.com <http://cppstudio.com>
4. Habrahabr.ru
<https://habrahabr.ru/post/130093/>
5. <http://stackoverflow.com/>
6. <http://stackoverflow.com/questions/2298242/callback-functions-in-c>
7. <http://stackoverflow.com/questions/14189440/c-class-member-callback-simple-examples>
8. <http://stackoverflow.com/questions/8966330/creating-a-table-using-win32-api>

ВИХІДНІ ТЕКСТИ ВСІХ МОДУЛІВ ПРОГРАМИ

Лістинг файлу Test1.h

```
#pragma once

#include <windows.h>
#include "resource.h"
```

Лістинг файлу Test1.cpp

```
// Test1.cpp : Defines the entry point for the application.
//

#include "stdafx.h"
#include "Test1.h"
#include "MyTable.h"

#define MAX_LOADSTRING 100

using namespace std;
// Global Variables:
//
//Table tb;
Files FO;
MyTable Vw;

HINSTANCE hInst;                                // current instance
TCHAR szTitle[MAX_LOADSTRING];                 // The title bar text
TCHAR szWindowClass[MAX_LOADSTRING];           // the main window class name

// Forward declarations of functions included in this code module:
ATOM                MyRegisterClass(HINSTANCE hInstance);
BOOL                InitInstance(HINSTANCE, int);
LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK    About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY _tWinMain(_In_ HINSTANCE hInstance,
                      _In_opt_ HINSTANCE hPrevInstance,
                      _In_ LPTSTR    lpCmdLine,
                      _In_ int       nCmdShow)
{
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);

    // TODO: Place code here.
    MSG msg;
    HACCEL hAccelTable;

    // Initialize global strings
    LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadString(hInstance, IDC_TEST1, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    // Perform application initialization:
    if (!InitInstance (hInstance, nCmdShow))
    {
        return FALSE;
    }
}
```

```

    }

    hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_TEST1));

    // Main message loop:
    while (GetMessage(&msg, NULL, 0, 0))
    {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }

    return (int) msg.wParam;
}

//
// FUNCTION: MyRegisterClass()
//
// PURPOSE: Registers the window class.
//
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    WNDCLASSEX wcex;

    wcex.cbSize = sizeof(WNDCLASSEX);

    wcex.style          = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc    = WndProc;
    wcex.cbClsExtra     = 0;
    wcex.cbWndExtra     = 0;
    wcex.hInstance     = hInstance;
    wcex.hIcon          = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_TEST1));
    wcex.hCursor        = LoadCursor(NULL, IDC_ARROW);
    wcex.hbrBackground  = (HBRUSH)(COLOR_WINDOW+1);
    wcex.lpszMenuName   = MAKEINTRESOURCE(IDC_TEST1);
    wcex.lpszClassName  = szWindowClass;
    wcex.hIconSm        = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));

    return RegisterClassEx(&wcex);
}

//
// FUNCTION: InitInstance(HINSTANCE, int)
//
// PURPOSE: Saves instance handle and creates main window
//
// COMMENTS:
//
//      In this function, we save the instance handle in a global variable and
//      create and display the main program window.
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    HWND hWnd;

    hInst = hInstance; // Store instance handle in our global variable

```



```

    hWnd = CreateWindow(szWindowClass, "Table stylesheet", WS_OVERLAPPEDWINDOW |
WS_VSCROLL | WS_HSCROLL,
    CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);
    RegisterHotKey(hWnd, 0, MOD_CONTROL, 0x5A);
    if (!hWnd)
    {
        return FALSE;
    }

    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

    return TRUE;
}

//
// FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)
//
// PURPOSE: Processes messages for the main window.
//
// WM_COMMAND      - process the application menu
// WM_PAINT        - Paint the main window
// WM_DESTROY      - post a quit message and return
//
//
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HDC hdc;

    switch (message)
    {
        case WM_SIZE:
            Vw.SetFlag(true);
            break;
        case WM_CREATE:
            SetScrollRange(hWnd, SB_HORZ, 0, 0, false);
            SetScrollRange(hWnd, SB_VERT, 0, 0, false);
            break;
        case WM_HSCROLL:
            Vw.HScroll(hWnd, wParam);
            break;
        case WM_VSCROLL:
            Vw.VScroll(hWnd, wParam);
            break;
        case WM_HOTKEY:
            Vw.Cansel(hWnd);
            break;
        case WM_LBUTTONDOWN:
            Vw.Edit(hWnd);
            break;
        case WM_COMMAND:
            wmId    = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            // Parse the menu selections:
            switch (wmId)
            {
                case IDM_ABOUT:
                    DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);

```

```

        break;
    case ID_FILE_OPEN:
        Vw.SetTable(FO.FileOpen(hWnd), FO.GetF(), hWnd);
        //FO.FileOpen(tb, hWnd);

        break;
    case ID_FILE_SAVE:
        Vw.WriteFile(FO.GetF());
        //FO.WriteFile(tb);
        break;
    case ID_FILE_SAVEAS:
        Vw.WriteFileAs(hWnd);

        //FO.WriteFileAs(hWnd, tb);
        break;
    case ID_EDIT_BACK:
        Vw.Cansel(hWnd);
        break;
    case IDM_EXIT:

        DestroyWindow(hWnd);
        break;
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    break;
case WM_PAINT:
    Vw.Paint(hWnd);
    hdc = BeginPaint(hWnd, &ps);
    // TODO: Add any drawing code here...
    EndPaint(hWnd, &ps);
    break;
case WM_DESTROY:
    PostQuitMessage(0);
    break;
default:
    return DefWindowProc(hWnd, message, wParam, lParam);
}
return 0;
}

// Message handler for about box.
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    UNREFERENCED_PARAMETER(lParam);
    switch (message)
    {
    case WM_INITDIALOG:
        return (INT_PTR)TRUE;

    case WM_COMMAND:
        if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
            return (INT_PTR)TRUE;
        }
        break;
    }
    return (INT_PTR)FALSE;
}
}

```

Лістинг файлу MyTable.h

```

#include "Table.h"
#include "FileRead.h"
#include "resource.h"
#include <Commdlg.h>
#include <fstream>
#include <stack>

#define let 10.8
#define keys 20

using namespace std;

class MyTable {
private:
    struct Info{
        char _text[150];
        int nC;
        int nI;
        int max;
    };

    SCROLLINFO HscrollInfo;
    SCROLLINFO VscrollInfo;
    int coord(int j);
    int numberC;
    int numberI;
    HWND hwndEdit = NULL;
    HWND hwnd;
    int numI(int y);
    int numC(int x);

    Table* tab;
    int* maxlc;
    int maxl;
    stack<Info> stc;
    bool f = false;
    WNDPROC oldEditProc;
    static LRESULT CALLBACK EditProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM
lParam);
    void SetScrollsPropertys(HWND hwnd);

    LRESULT CALLBACK RealEditProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam);

public:
    void SetTable(Table* tb, char* filename, HWND hwnd);
    void Paint(HWND);
    void Edit(HWND hwnd);
    void SetFlag(bool);
    void WriteFile(char*);
    void WriteFileAs(HWND hwnd);
    void HScroll(HWND hwnd, WPARAM wParam);
    void VScroll(HWND hwnd, WPARAM wParam);
    void Cansel(HWND);
};

```

Лістинг файлу MyTable.cpp

```

#include "stdafx.h"
#include "MyTable.h"

```

```

void MyTable::SetTable(Table* tb, char* filename, HWND hWnd)
{
    EnableMenuItem(GetMenu(hWnd), ID_FILE_SAVEAS, false);
    EnableMenuItem(GetMenu(hWnd), ID_FILE_SAVE, false);
    DestroyWindow(hWndEdit);
    if (tb != NULL)
    {
        SetWindowTextA(hWnd, filename);
        tab = tb;
        while (!stc.empty())
        {
            stc.pop();
        }

        f = true;
        int qc = tb->GetQC();
        int qi = tb->GetQI();

        maxlc = new int[tb->GetQC()];

        for (int j = 0; j < tb->GetQC(); j++)
        {
            maxlc[j] = 0;
        }

        for (int j = 0; j < tb->GetQC(); j++)
        {
            for (int i = 0; i < tb->GetQI(); i++)
            {
                if (maxlc[j] < strlen(tb->GetVct(i, j))) maxlc[j] = strlen(tb-
>GetVct(i, j));
            }
        }
        InvalidateRect(hWnd, 0, TRUE);
    }
}

void MyTable::SetFlag(bool bu)
{
    f = bu;
}

void MyTable::Paint(HWND hWnd)
{
    if (tab != NULL)
    {
        PAINTSTRUCT ps;
        HDC hdc;

        hdc = BeginPaint(hWnd, &ps);
        maxl = 0;
        for (int i = 0; i < tab->GetQC(); i++)
        {
            maxl += maxlc[i];
        }
        if (f)
        {
            SetScrollsPropertys(hWnd);
            f = false;
        }

        Rectangle(hdc, 0 - HscrollInfo.nPos, 0 - VscrollInfo.nPos, (maxl*let
+ 2 * tab->GetQC() *keys) - HscrollInfo.nPos, tab->GetQI() * 60 - VscrollInfo.nPos);
    }
}

```

```

        for (int i = 0; i < tab->GetQI(); i++)
        {
            MoveToEx(hdc, 0 - HscrollInfo.nPos, i * 60 - VscrollInfo.nPos,
NULL);
            LineTo(hdc, maxl*let + 2 * tab->GetQC() * keys -
HscrollInfo.nPos, i * 60 - VscrollInfo.nPos);
            for (int j = 0; j < tab->GetQC(); j++)
            {
                if (i == 0)
                {
                    MoveToEx(hdc, coord(j) - keys, 0 -
VscrollInfo.nPos, NULL);
                    LineTo(hdc, coord(j) - keys, tab->GetQI() * 60 -
VscrollInfo.nPos);
                }
                TextOutA(hdc, coord(j), 20 + i * 60 - VscrollInfo.nPos,
tab->GetVct(i, j), strlen(tab->GetVct(i, j)));
            }
        }
        EndPaint(hWnd, &ps);
    }

}

void MyTable::HScroll(HWND hWnd, WPARAM wParam)
{
    int xCurrentScroll = GetScrollPos(hWnd, SB_HORZ);
    int xDelta;
    int xNewPos;
    switch (LOWORD(wParam))
    {
        case SB_PAGERIGHT:
            xNewPos = xCurrentScroll + 50;
            break;
        case SB_PAGELEFT:
            xNewPos = xCurrentScroll - 50;
            break;
        case SB_LINERIGHT:
            xNewPos = xCurrentScroll + 5;
            break;
        case SB_LINELEFT:
            xNewPos = xCurrentScroll - 5;
            break;
        case SB_THUMBPOSITION:
            xNewPos = HIWORD(wParam);
            break;
        default:
            xNewPos = xCurrentScroll;
    }
    if (!IsWindow((HWND)hWndEdit))
    {
        SetScrollPos(hWnd, SB_HORZ, xNewPos, true);
        HscrollInfo.nPos = xNewPos;
        InvalidateRect(hWnd, 0, TRUE);
    }
}

void MyTable::VScroll(HWND hWnd, WPARAM wParam)
{
    int yCurrentScroll = GetScrollPos(hWnd, SB_VERT);
    int yDelta;
    int yNewPos;

```

```

switch (LOWORD(wParam))
{
case SB_PAGEDOWN:
    yNewPos = yCurrentScroll + 10;
    break;
case SB_PAGEUP:
    yNewPos = yCurrentScroll - 10;
    break;
case SB_LINEDOWN:
    yNewPos = yCurrentScroll + 5;
    break;
case SB_LINEUP:
    yNewPos = yCurrentScroll - 5;
    break;
case SB_THUMBPOSITION:
    yNewPos = HIWORD(wParam);
    break;
default:
    yNewPos = yCurrentScroll;
}
if (!IsWindow((HWND)hwndEdit))
{
    SetScrollPos(hwnd, SB_VERT, yNewPos, true);
    VscrollInfo.nPos = yNewPos;
    InvalidateRect(hwnd, 0, TRUE);
}
}

void MyTable::SetScrollsPropertys(HWND hwnd)
{
    RECT rc;
    GetClientRect(hwnd, &rc);
    if (rc.right < (maxl * let + 2 * tab->GetQC() * keys))
    {
        HscrollInfo.cbSize = sizeof(SCROLLINFO);
        HscrollInfo.nMin = 0;
        HscrollInfo.nMax = (maxl*let + 2 * tab->GetQC() *keys) + 20;
        HscrollInfo.fMask = SIF_ALL;
        if (!HscrollInfo.nPos) HscrollInfo.nPos = 0;
        HscrollInfo.nPage = rc.right;
        SetScrollInfo(hwnd, SB_HORZ, &HscrollInfo, false);
    }
    else
        SetScrollRange(hwnd, SB_HORZ, 0, 0, false);
    if (rc.bottom < tab->GetQI() * 60)
    {
        VscrollInfo.cbSize = sizeof(SCROLLINFO);
        VscrollInfo.nMin = 0;
        VscrollInfo.nMax = tab->GetQI() * 60 + 20;
        VscrollInfo.fMask = SIF_ALL;
        if (!VscrollInfo.nPos) VscrollInfo.nPos = 0;
        VscrollInfo.nPage = rc.bottom;
        SetScrollInfo(hwnd, SB_VERT, &VscrollInfo, false);
    }
    else
        SetScrollRange(hwnd, SB_VERT, 0, 0, TRUE);
}

```

```

LRESULT CALLBACK MyTable::EditProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)

```

```

{
    MyTable* v = (MyTable*)(GetWindowLongPtr(hwnd, GWLP_USERDATA));
    if (v) return v->RealEditProc(hwnd, msg, wParam, lParam);
    return DefWindowProc(hwnd, msg, wParam, lParam);
}

LRESULT CALLBACK MyTable::RealEditProc(HWND hwndEdit, UINT msg, WPARAM wParam, LPARAM
lParam)
{
    char bufEdit[150];
    Info kek1;
    switch (msg)
    {
    case WM_KEYDOWN:
        switch (wParam)
        {
        case VK_RETURN:
            kek1.nC = numberC;
            kek1.nI = numberI;
            kek1.max = maxlc[numberC];
            strcpy(kek1._text, tab->GetVct(numberI, numberC));
            stc.push(kek1);
            GetWindowText(hwndEdit, bufEdit, 150);
            DestroyWindow(hwndEdit);
            if (bufEdit[0] == '\\0') strcpy(bufEdit, " ");
            tab->ChangeItem(numberI, numberC, bufEdit);
            if (maxlc[numberC] < strlen(bufEdit)) maxlc[numberC] =
strlen(bufEdit);

            maxl = 0;
            for (int i = 0; i < tab->GetQC(); i++)
            {
                maxl += maxlc[i];
            }
            SetScrollsPropertys(hwnd);
            InvalidateRect(hwnd, 0, TRUE);
            break; //or return 0; if you don't want to pass it further to
def proc

            //If not your key, skip to default:
        case VK_ESCAPE:
            DestroyWindow(hwndEdit);
            break;
        }
    default:
        return CallWindowProc(oldEditProc, hwndEdit, msg, wParam, lParam);
    }
    return 0;
}

void MyTable::Edit(HWND hwnd)
{
    if (tab != NULL)
    {
        if (!IsWindow((HWND)hwndEdit))
        {
            int ryadprop = 0;
            int otny = 0;
            POINT pt;
            long x, y;
            GetCursorPos(&pt);
            ScreenToClient(hwnd, &pt);

```

```

        x = pt.x;
        y = pt.y;
        if ((x + HscrollInfo.nPos < (maxl*let + 2 * tab->GetQC() *keys)) &&
            (y + VscrollInfo.nPos < tab->GetQI() * 60))
        {
            //x -= ostx();
            numberC = numC(x + HscrollInfo.nPos);
            numberI = numI(y + VscrollInfo.nPos);

            hwndEdit = CreateWindowEx(0,
                "EDIT", // predefined class
                NULL, // no window title
                WS_CHILD | WS_VISIBLE | WS_BORDER |
                ES_LEFT,
                coord(numberC) - 20, (numberI) * 60 - VscrollInfo.nPos
, maxlc[numberC] * let + 2 * keys, 60, // set size in WM_SIZE message
                hwnd, // parent window
                (HMENU)1450, // edit control ID
                (HINSTANCE)GetWindowLong(hwnd, GWL_HINSTANCE),
                NULL); // pointer not needed
            SetWindowText(hwndEdit, tab->GetVct(numberI, numberC));
            SetFocus(hwndEdit);
            SetWindowLongPtr(hwndEdit, GWLP_USERDATA, (long)this);
            hwnd = hwndEdit;
            oldEditProc = (WNDPROC)SetWindowLongPtr(hwndEdit,
GWLP_WNDPROC, (LONG_PTR)EditProc);
            //oldEditProc = (WNDPROC)SetWindowLongPtr(hwndEdit,
GWLP_WNDPROC, (LONG_PTR)EditProc);

        }
    }
}

int MyTable::coord(int j)
{
    int max = 0;
    int x = 0;
    for (int i = 0; i < j; i++)
    {
        max += maxlc[i];
    }
    x = max*let + 2 * j * keys + keys;
    return x - HscrollInfo.nPos;
}

int MyTable::numC(int x)
{
    int max = 0;
    int i = 0;
    while (max < x)
    {
        max += maxlc[i] * let + 2 * keys;
        i++;
    }

    return i - 1;
}

int MyTable::numI(int y)
{
    int max = 0;

```



```

    int i = 0;

    while (max < y)
    {
        max += 60;
        i++;
    }

    return i - 1;
}

void MyTable::Cansel(HWND hWnd)
{
    Info inf;
    if (!stc.empty())
    {
        inf = stc.top();
        tab->ChangeItem(inf.nI, inf.nC, inf._text);
        maxlc[inf.nC] = inf.max;
        stc.pop();
        InvalidateRect(hWnd, 0, TRUE);
    }
}

void MyTable::WriteFile(char* filename)
{
    if (tab != NULL)
    {
        ofstream fout;
        fout.open(filename, ios::out);
        for (int i = 0; i < tab->GetQI(); i++)
        {
            for (int j = 0; j < tab->GetQC(); j++)
            {
                fout << tab->GetVct(i, j);

                if (j != tab->GetQC() - 1) fout << "\t";
            }
            if (i != tab->GetQI() - 1) fout << "\n";
        }
        fout.close();
    }
}

void MyTable::WriteFileAs(HWND hWnd)
{
    if (tab != NULL)
    {
        OPENFILENAME ofn;
        char szFile[100];
        ZeroMemory(&ofn, sizeof(ofn));
        ofn.lStructSize = sizeof(ofn);
        ofn.hwndOwner = hWnd;
        ofn.lpstrFile = szFile;
        ofn.lpstrFile[0] = '\0';
        ofn.nMaxFile = sizeof(szFile);
        ofn.lpstrFilter = "Text\0*.TXT\0";
        ofn.nFilterIndex = 1;
        ofn.lpstrFileTitle = NULL;
        ofn.nMaxFileTitle = 0;
        ofn.lpstrInitialDir = NULL;
        ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST | OFN_HIDEREADONLY;
    }
}

```

```

        if (GetSaveFileName(&ofn));
        {
            ofstream fout;
            fout.open(ofn.lpstrFile, ios::out);
            for (int i = 0; i < tab->GetQI(); i++)
            {
                for (int j = 0; j < tab->GetQC(); j++)
                {
                    fout << tab->GetVct(i, j);
                    if (j != tab->GetQC() - 1) fout << "\t";
                }
                if (i != tab->GetQI() - 1) fout << "\n";
            }
            fout.close();
        }
    }
}

```

Лістинг файлу FileRead.h

```

#include "Table.h"

using namespace std;

class Files
{
private:
    char filename[150];

public:
    Table* FileOpen(HWND);
    char* GetF();
};

```

Лістинг файлу FileRead.cpp

```

#include "stdafx.h"
#include "FileRead.h"

Table* Files::FileOpen(HWND hWnd)
{
    Table* tb = new Table;
    OPENFILENAME ofn;
    char szFile[100];
    char buff[500];
    int indQC = 0;
    int indQI = 0;
    // open a file name
    ZeroMemory(&ofn, sizeof(ofn));
    ofn.lStructSize = sizeof(ofn);
    ofn.hwndOwner = hWnd;
    ofn.lpstrFile = szFile;
    ofn.lpstrFile[0] = '\0';
    ofn.nMaxFile = sizeof(szFile);
    ofn.lpstrFilter = "Text\0*.TXT\0";
    ofn.nFilterIndex = 1;
    ofn.lpstrFileTitle = NULL;
    ofn.nMaxFileTitle = 0;
    ofn.lpstrInitialDir = NULL;
    ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
}

```

```

if (GetOpenFileName(&ofn))
{
    vector<char*> Vct;

    strcpy(filename, ofn.lpstrFile);
    //filename = ofn.lpstrFile;
    ifstream fin;
    fin.open(ofn.lpstrFile, ios::in);
    fin.getline(buff, 500);
    char* pch = strtok(buff, "\\t");
    while (pch != NULL)                                // пока есть лексемы
    {
        indQC++;
        pch = strtok(NULL, "\\t");
    }

    if (indQC != 0)
    {
        indQI++;

        while (!fin.eof())
        {
            fin.getline(buff, 500);
            indQI++;
        }

        Vct.resize(indQI*indQC);
        int qic = indQI;
        //tb->SetVct(NULL, NULL, NULL, indQI*indQC, NULL);

        indQI = 0;

        fin.seekg(ios::beg);
        while (!fin.eof())
        {
            if (indQI == 82)
            {
                int mam = 0;
                mam++;
            }
            int ind = 0;
            fin.getline(buff, 500);
            char* pch = strtok(buff, "\\t");
            /*tb->SetVct(indQI, -1, NULL, -1, indQC);*/
            Vct[indQI] = new char*[indQC];
            while (pch != NULL)                            // пока есть
                лексемы
            {
                /*tb->SetVct(indQI, ind, NULL, -1, 150);*/
                Vct[indQI][ind] = new char[150];
                /*tb->SetVct(indQI, ind, pch, -1, -1);*/
                strcpy(Vct[indQI][ind], pch);

                //if (strlen(pch) > maxlc[ind]) maxlc[ind] =
                strlen(pch);
            }
        }
    }
}

```

```

        ind++;
        pch = strtok(NULL, "\\t");
    }
    indQI++;
}
}
fin.close();
if (indQC)
{
    tb->SetQI(indQI);
    tb->SetQC(indQC);
    tb->SetVct(Vct);
}
return tb;
}
else
{
    return NULL;
}
}

char* Files::GetF()
{
    return filename;
}

```

Лістинг файлу Table.h

```

#pragma once

#include <vector>
#include <fstream>
#include <Commdlg.h>

using namespace std;

class Table {
private:
    vector<char*> vct;
    int qc = 0;
    int qi = 0;
public:
    char* GetVct(int, int);
    int GetQC();
    int GetQI();
    void ChangeItem(int i, int j, char* str);
    void SetVct(vector<char*>);
    void SetQC(int);
    void SetQI(int);
};

```

Лістинг файлу Table.cpp

```

#include "stdafx.h"
#include "Table.h"

char* Table::GetVct(int i, int j)
{
    return vct[i][j];
}

```

```
int Table::GetQC()
{
    return qc;
}

int Table::GetQI()
{
    return qi;
}

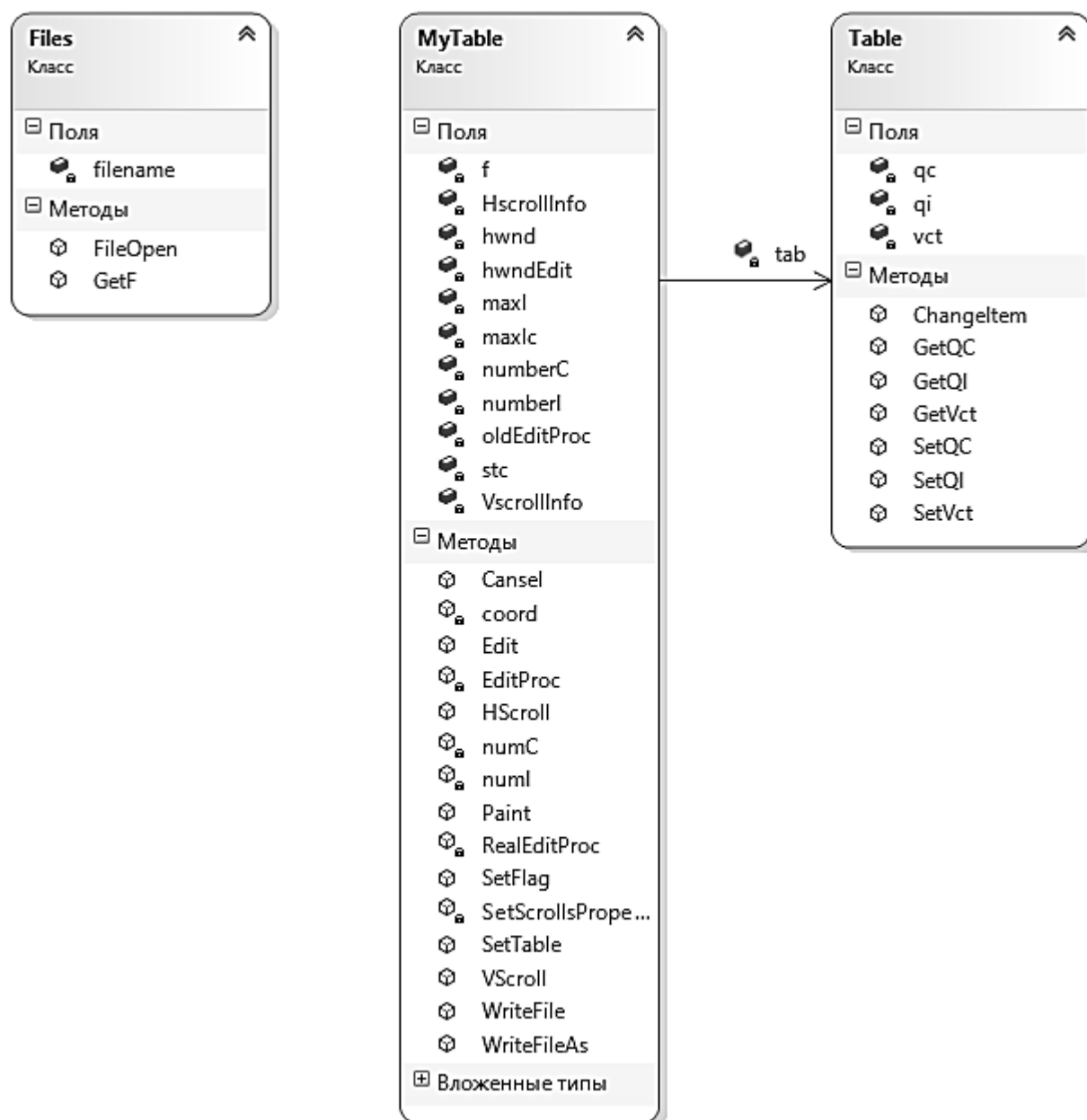
void Table::SetVct(vector<char**> Vcts)
{
    vct = Vcts;
}

void Table::ChangeItem(int i, int j, char* str)
{
    strcpy(vct[i][j], str);
}

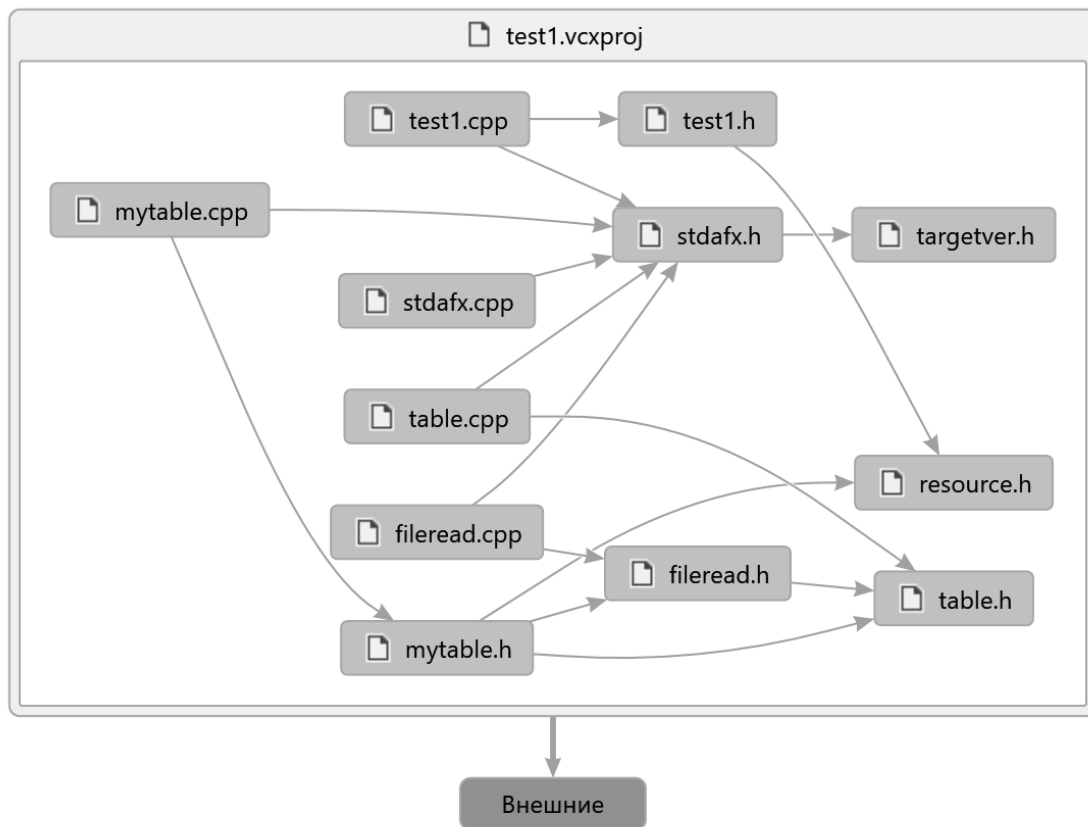
void Table::SetQI(int QI)
{
    qi = QI;
}

void Table::SetQC(int QC)
{
    qc = QC;
}
```

ДІАГРАМА КЛАСІВ



ДІАГРАМА #INCLUDE-ІЄРАРХІЇ МОДУЛІВ



СКРІНШОТИ РОБОТИ ПРОГРАМИ

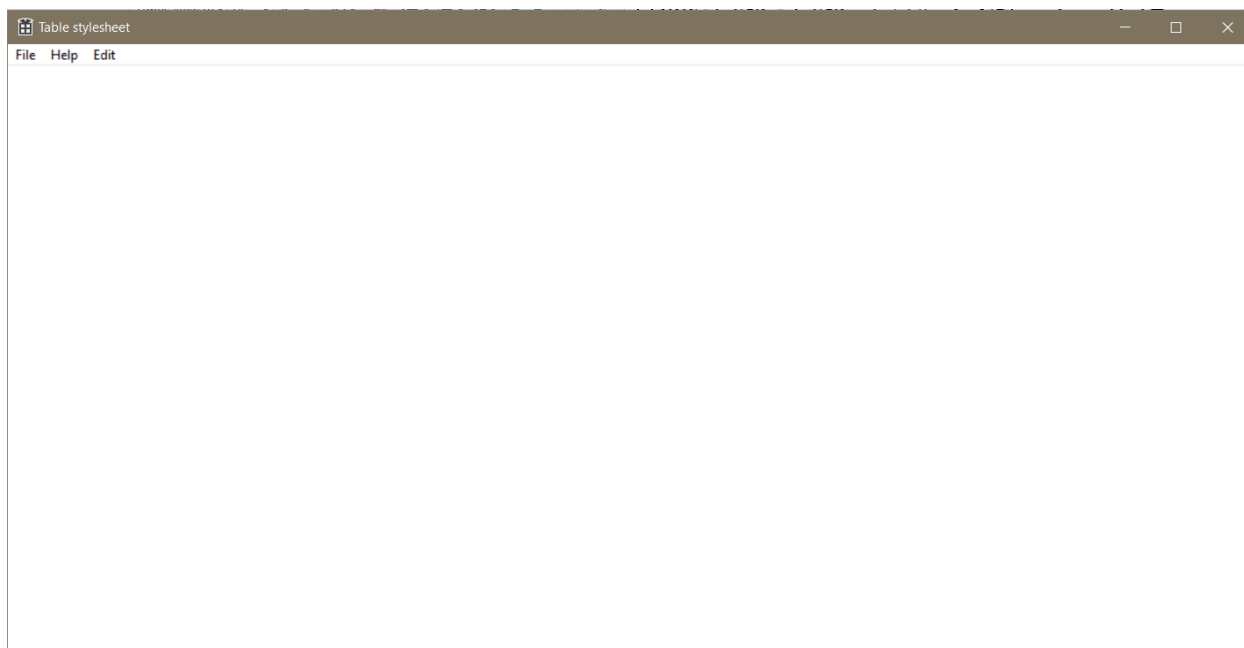


Рис 2.1 - Вибір файлу зчитування

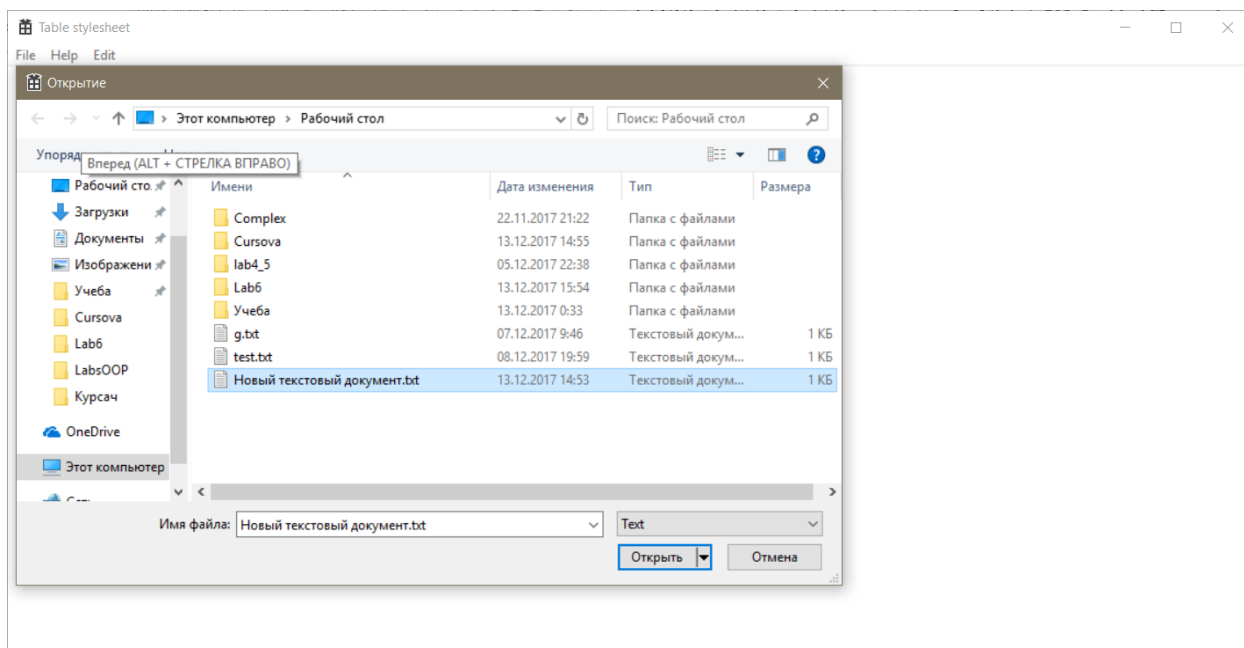
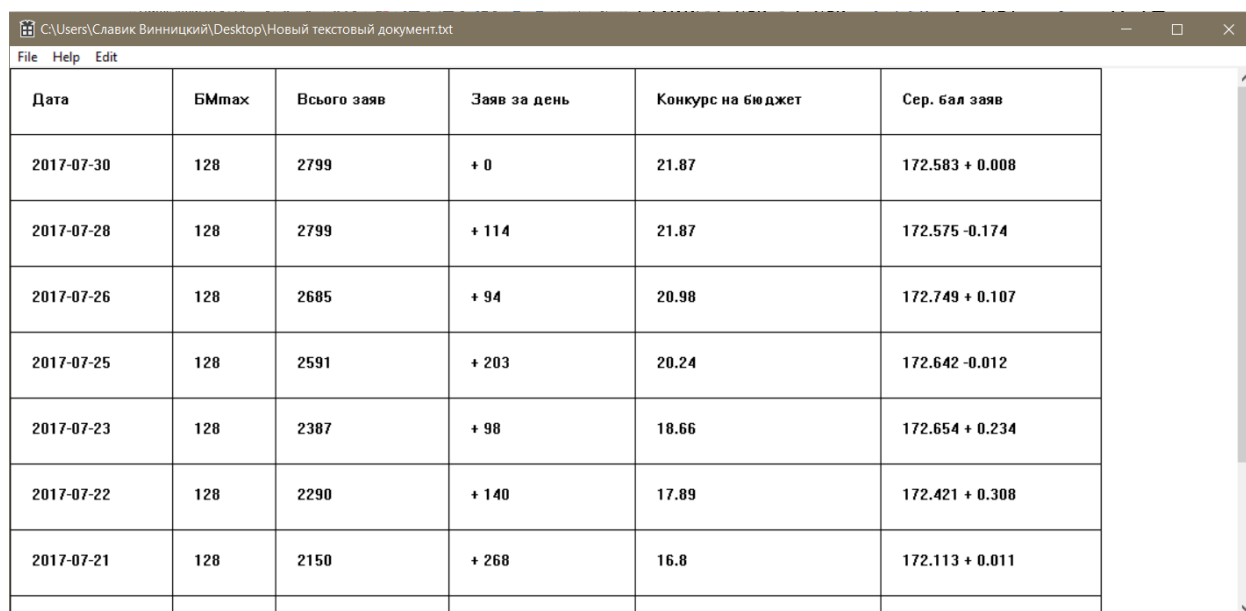
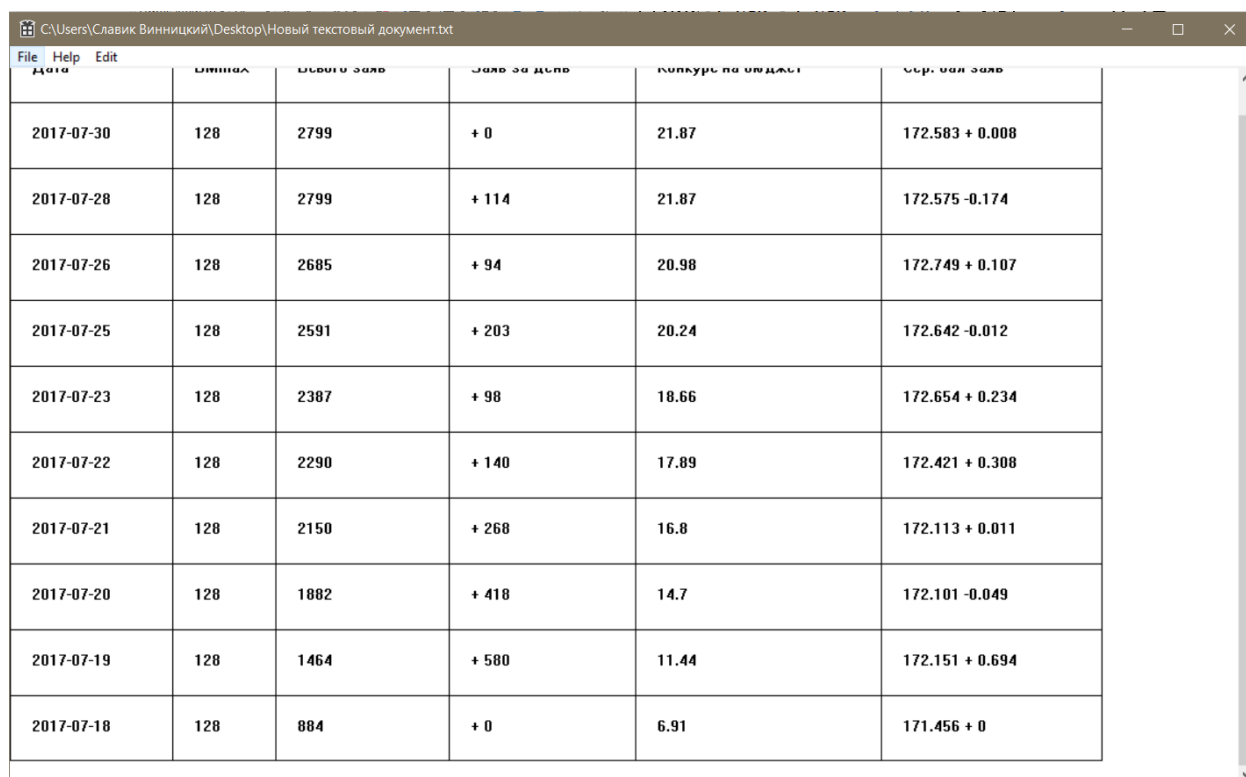


Рис 2.2 – Відрисування таблиці



Дата	БМmax	Всего заяв	Заяв за день	Конкурс на бюджет	Сер. бал заяв
2017-07-30	128	2799	+ 0	21.87	172.583 + 0.008
2017-07-28	128	2799	+ 114	21.87	172.575 -0.174
2017-07-26	128	2685	+ 94	20.98	172.749 + 0.107
2017-07-25	128	2591	+ 203	20.24	172.642 -0.012
2017-07-23	128	2387	+ 98	18.66	172.654 + 0.234
2017-07-22	128	2290	+ 140	17.89	172.421 + 0.308
2017-07-21	128	2150	+ 268	16.8	172.113 + 0.011

Рис 2.3 – Реагування на скролінг



Дата	БМmax	Всего заяв	Заяв за день	Конкурс на бюджет	Сер. бал заяв
2017-07-30	128	2799	+ 0	21.87	172.583 + 0.008
2017-07-28	128	2799	+ 114	21.87	172.575 -0.174
2017-07-26	128	2685	+ 94	20.98	172.749 + 0.107
2017-07-25	128	2591	+ 203	20.24	172.642 -0.012
2017-07-23	128	2387	+ 98	18.66	172.654 + 0.234
2017-07-22	128	2290	+ 140	17.89	172.421 + 0.308
2017-07-21	128	2150	+ 268	16.8	172.113 + 0.011
2017-07-20	128	1882	+ 418	14.7	172.101 -0.049
2017-07-19	128	1464	+ 580	11.44	172.151 + 0.694
2017-07-18	128	884	+ 0	6.91	171.456 + 0

Рис 2.4 – Редагування таблиці

Дата	Витрата	Сума за день	Сума за день	Конкурс на область	Ср. бал за день
2017-07-30	128	2799	+ 0	21.87	172.583 + 0.008
2017-07-28	128	2799	+ 114	21.87	172.575 -0.174
2017-07-26	128	2685	+ 94	20.98	172.749 + 0.107
2017-07-25	128	2591	+ 203	20.24	172.642 -0.012
2017-07-23	128	2387	+ 98123235	18.66	172.654 + 0.234
2017-07-22	128	2290	+ 140	17.89	172.421 + 0.308
2017-07-21	128	2150	+ 268	16.8	172.113 + 0.011
2017-07-20	128	1882	+ 418	14.7	172.101 -0.049
2017-07-19	128	1464	+ 580	11.44	172.151 + 0.694
2017-07-18	128	884	+ 0	6.91	171.456 + 0

2.5 Відредагована таблиця на весь екран

Дата	Витрата	Сума за день	Сума за день	Конкурс на область	Ср. бал за день
2017-07-30	128	2799	+ 0	21.87	172.583 + 0.008
2017-07-28	128	2799	+ 114	21.87	172.575 -0.174
2017-07-26	128	2685	+ 94	20.98	172.749 + 0.107
2017-07-25	128	2591	+ 203	20.24	172.642 -0.012
2017-07-23	128	2387	+ 98123235	18.66	172.654 + 0.234
2017-07-22	128	2290	+ 140	17.89	172.421 + 0.308
2017-07-21	128	2150	+ 268	16.8	172.113 + 0.011
2017-07-20	128	1882	+ 418	14.7	172.101 -0.049
2017-07-19	128	1464	+ 580	11.44	172.151 + 0.694
2017-07-18	128	884	+ 0	6.91	171.456 + 0

Рис 2.6 – Вибір файлу для збереження

