**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ**

**Кафедра інформаційних технологій та прикладної математики**

**Індивідуальна робота**

з дисципліни «Технологія комп’ютерного проектування»

на тему: «Розробка програми ведення ділового щоденника»

Студентки ІІ курсу групи \_\_\_\_\_\_

напряму підготовки

**6.050101 – «Комп’ютерні науки»**

спеціальності \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка:

ECTS \_\_\_\_\_

Члени комісії

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

м.Київ – 2018 рік

**Зміст**

[**1.** **Вступ** 3](#_Toc516054727)

[**2.** **Аналіз предметної області** 4](#_Toc516054728)

[**Постановка задачі** 4](#_Toc516054729)

[**3.** **Проектування проекту за допомогою діаграм** 6](#_Toc516054730)

[3.1. UML-діаграма прецедентів 6](#_Toc516054731)

[3.2. UML-діаграма класів 10](#_Toc516054732)

[3.3. Опис системних операцій та поведінки програми у вигляді UML-діаграм послідовностей 11](#_Toc516054733)

[**4.** **Опис структури програми у вигляді UML-діаграми компонентів** 12](#_Toc516054734)

[**5.** **Опис тестових прикладів виконання програми** 13](#_Toc516054735)

[**6.** **Висновки** 15](#_Toc516054736)

[**7.** **Список літератури** 16](#_Toc516054737)

# **Вступ**

В наш час люди постійно зайняті. Сучасній активній людині доводиться утримувати в голові багато інформації. В результаті чого мозок стає перевантажений. А важливі справи можуть випадково забуті.

Звернемось до думки людей, яких уважають «зібраними».

Вони одностайно вказують нам на чотири основні причини:

* нам не вдається правильно визначити пріоритети;
* ми витрачаємо масу часу на зовсім непотрібні речі;
* ми беремо на себе більше, ніж можемо зробити;
* ми намагаємось у всьому, чим займаємось, досягти повної досконалості.

Заведи собі планер або щоденник. І в ньому записуй все, що тобі необхідно встигнути зробити за день. Візуалізація списку справ – перший крок на шляху до їх виконання.

Маленький лайфхак: коли виконаєш одне із завдань, викресли його яскравим маркером. Значно приємніше братися за наступну справу, коли бачиш, що список завдань зменшується на очах! Якщо ти думаєш, що носити з собою щоденник незручно, веди планер у телефоні: або в нотатках, або в одному зі спеціальних додатків, яких зараз безліч!

Доручивши зберігання інформації щоденнику легко можливо вчасно про неї згадати.

Але не слід за бувати за структуру запису даних. Найчастіше відсутність будь-якої структури призводить до абсолютного хаосу в цих записах. При необхідності знайти потрібну інформацію швидко і без праці в результаті стає неможливо.

# **Аналіз предметної області**

Людина яка веде діловий щоденник встигає зробити за день набагато більше, ніж без нього. Навчається правильно розподіляти свій час на виконання тих чи інших завдань.

Переваги електронного варіанту:

* Можна працювати в темний час доби без освітлення;
* Знайти потрібний запис не складе труднощів і не займе багато часу;
* Існують шаблони записів;
* Не потрібні ручки і олівці для роботи.

## **Постановка задачі**

Необхідно розробити програму, яка призначена для ведення ділового щоденника. Програма повинна мати наступні функціональні можливості:

- збереження інформації у файл;

- читання інформації з файлу;

- видалення події з файлу;

- додавання події в файл (з перевіркою проти накладання двох подій);

- виведення додаткової інформації про подію;

- пошук за датою;

- пошук за типом;

Основної функцією буде – ведення списку подій (ділового щоденника). Розглянемо дерево функцій ІС (рис. 1).



Рисунок 1. Дерево функцій

Виділимо основні сутності предметного середовища:

1. Сутність – подія, яка призначена для зберігання даних про конкретну подію. Дана сутність буде мати наступні атрибути:

- Назва події;

- Тип події;

- Дата проведення події;

- Час проведення події;

- Опис події;

2. Список подій – сутність яка призначена зберігати інформацію про всі заплановані події. Має наступні атрибути:

- Кількість подій у списку.

# **Проектування проекту за допомогою діаграм**

## UML-діаграма прецедентів



Рисунок 2. Діаграма прецедентів

Опишемо основні прецеденти:

1. Перегляд списку подій

**Основний виконавець.** Користувач.

**Зацікавлені особи та їх вимоги.** Користувач. Може переглянути список подій.

**Передумови.** Користувач запустив програму.

**Результат (Постумова).** На екрані користувача з‘явилася головна форма програми зі списком подій.

**Основний успішний сценарій (або основний процес):**

Користувач переглядає список подій.

2. Перегляд додаткової інформації

**Основний виконавець.** Користувач.

**Зацікавлені особи та їх вимоги.** Користувач. Може переглядати додаткову інформацію.

**Передумови.** Користувач обрав подію і натиснув кнопку «дізнатися більше».

**Результат (Постумова).** На екрані користувача з‘явилася додаткова форма програми з текстовим полем, що містить короткий опис події.

**Основний успішний сценарій (або основний процес):**

Користувач переглядає додаткову інформацію.

3. Робота з файлом зі списком

**Основний виконавець.** Користувач.

**Зацікавлені особи та їх вимоги.** Користувач. Може працювати з файлом зі списком подій.

**Передумови.** Користувач запустив програму.

**Результат (Постумова).** На екрані користувача з‘явилася головна форма програми зі списком подій.

**Основний успішний сценарій (або основний процес):**

Користувач починає роботу з файлом зі списком подій.

4. Зчитування з файлу

**Основний виконавець.** Програма.

**Зацікавлені особи та їх вимоги.** Програма. Може читати список з файлу.

**Результат (Постумова).** Список подій прочитано з файлу і відображено на головній формі.

**Основний успішний сценарій (або основний процес):** Список успішно прочитано з файлу.

5. Запис у файл

**Основний виконавець.** Програма.

**Зацікавлені особи та їх вимоги.** Програма. Може записати список з файлу.

**Передумови.** Користувач натиснув кнопку «записати».

**Результат (Постумова).** Список подій записано у файл і відображено на головній формі.

**Основний успішний сценарій (або основний процес):** Список успішно записано у файл.

6. Додавання події

**Основний виконавець.** Користувач.

**Зацікавлені особи та їх вимоги.** Користувач. Може додати заплановану подію.

**Передумови.** Користувач заповнив всі поля («Назва», «Тип», «Дата», «Час» та «Опис») і натиснув кнопку «додати».

**Результат (Постумова).** На екрані користувача з‘явився оновлений список.

**Основний успішний сценарій (або основний процес):**

Користувач переглядає оновлений щоденник.

7. Видалення події

**Основний виконавець.** Користувач.

**Зацікавлені особи та їх вимоги.** Користувач. Може видалити заплановану подію.

**Передумови.** Користувач обрав подію і натиснув кнопку «видалити подію».

**Результат (Постумова).** На екрані користувача з‘явився оновлений список.

**Основний успішний сценарій (або основний процес):**

Користувач переглядає оновлений щоденник.

8. Пошук за типом

**Основний виконавець.** Користувач.

**Зацікавлені особи та їх вимоги.** Користувач. Може шукати подію, що відповідає обраному типу.

**Результат (Постумова).** Список подій, що відповідають обраному типу.

**Основний успішний сценарій (або основний процес):**

1. Користувач обирає у полі необхідний тип.

2. Користувач натискає кнопку «пошук».

3. Система виводить список подій, що належать до заданого типу.

9. Пошук за датою

**Основний виконавець.** Користувач.

**Зацікавлені особи та їх вимоги.** Користувач. Може шукати подію, що відбувається у заданий день.

**Результат (Постумова).** Список подій, що відбуваються в обраний день.

**Основний успішний сценарій (або основний процес):**

1. Користувач обирає у полі дату.

2. Користувач натискає кнопку «пошук».

3. Система виводить список подій, що відбуваються у обрану дату.

## UML-діаграма класів

Розглянемо діаграму класів системи (рис.3).



Рисунок 3. Діаграма класів

Опишемо кожен з класів більш докладніше.

**1. Клас Podiya**

Клас призначений для відображення даних про подію.

Даний клас має наступні поля:

- public string naz – атрибут призначений для збереження назви події;

- public string typ – атрибут призначений для збереження типу події;

- public string data – атрибут призначений для збереження дати події;

- public string chas – атриибут призначений для збереження часу події;

- public string opys – атрибут призначений для збереження опису події;

Даний клас має конструктори:

- Podiya () - конструктор без параметрів, у якому усім полям буде присвоєно значення за замовченням.

- Podiya (string naz, string typ, string data, string chas, string opys) – конструктор з параметрами, у якому атрибутам присвоюється значеня параметрів.

**2. Клас PodiyaList**

Клас призначений для роботи зі списком подій.

Даний клас має наступні поля:

- private XmlSerializer sr – атрибут для проведення серіалізації, при збереженні та читанні списку з файлу.

- public list<Podiya> Shodennyk – атрибут списку об’єктів класу Podiya.

Даний клас має наступні методи:

- public void Add(string n, string t, string d, string c, string o, dataGridView dg) – метод для додавання події;

- public void Show(dataGridView dg) - метод для виведення списку на екран;

- public void Reading() – метод для читання списку з файлу;

- public void Deleting(int i, dataGridView dg) – метод для видалення події;

- public void Writing(string f) – метод для запису списку в текстовий файл;

- public void Searching(string s, dataGridView dg) – метод для пошуку подій за критерієм, який обрав користувач;

## Опис системних операцій та поведінки програми у вигляді UML-діаграм послідовностей

Опишемо поведінку системи у вигляді діаграми послідовностей для основних операцій системи, а саме:

1. Виведення додаткової інформації (рис. 4)



Рисунок 4. Операція виведення додаткової інформації

2. Додавання події (рис. 5).



Рисунок 5. Операція додавання події

# **Опис структури програми у вигляді UML-діаграми компонентів**

Опишемо структуру програми у вигляді діаграми компонентів.

Розроблена програма включає два компонента:

- Головна форма;

- Форма для додаткової інформації.

Діаграма компонентів наведена на рис. 6.

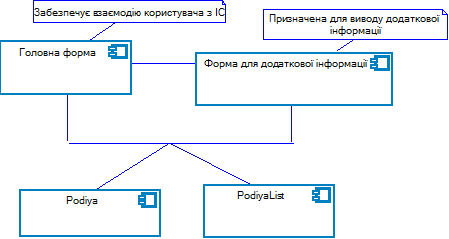
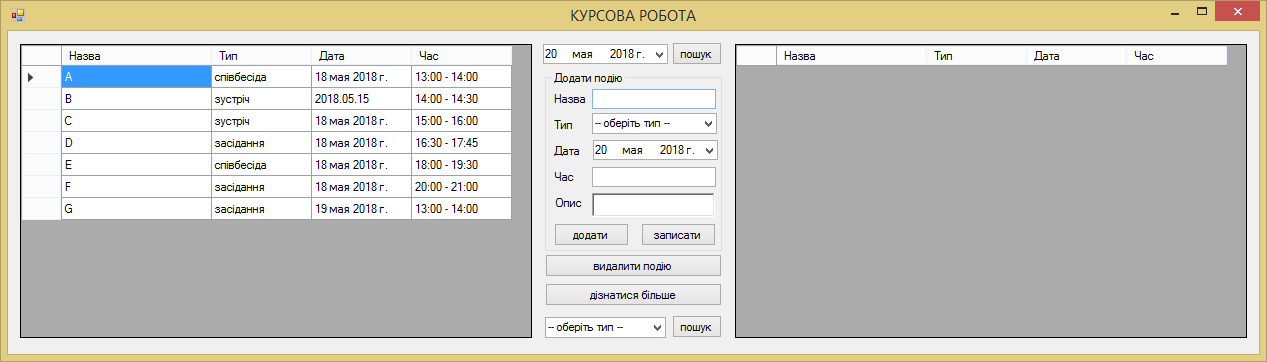


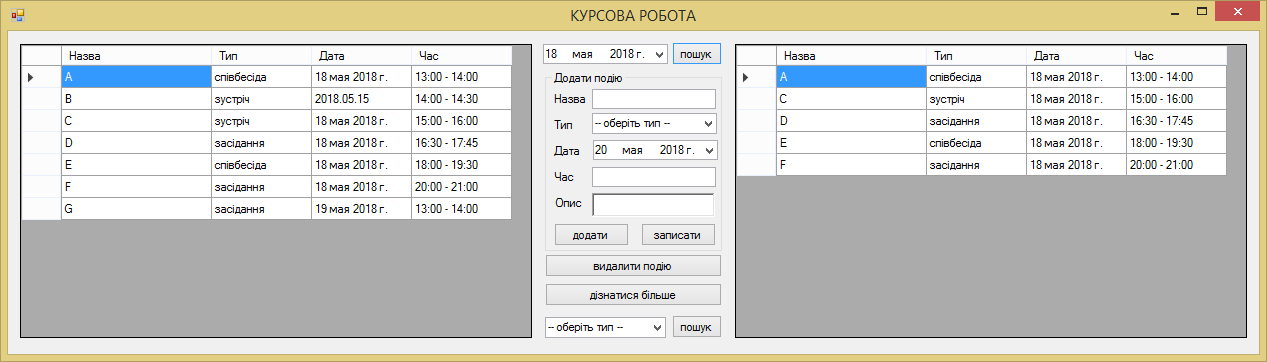
Рисунок 6. Діаграма компонентів

# **Опис тестових прикладів виконання програми**

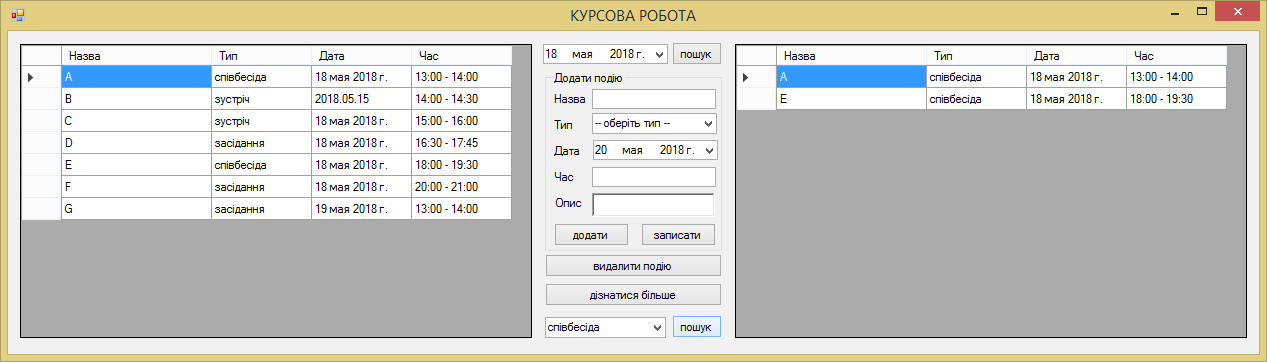
При запуску програми:



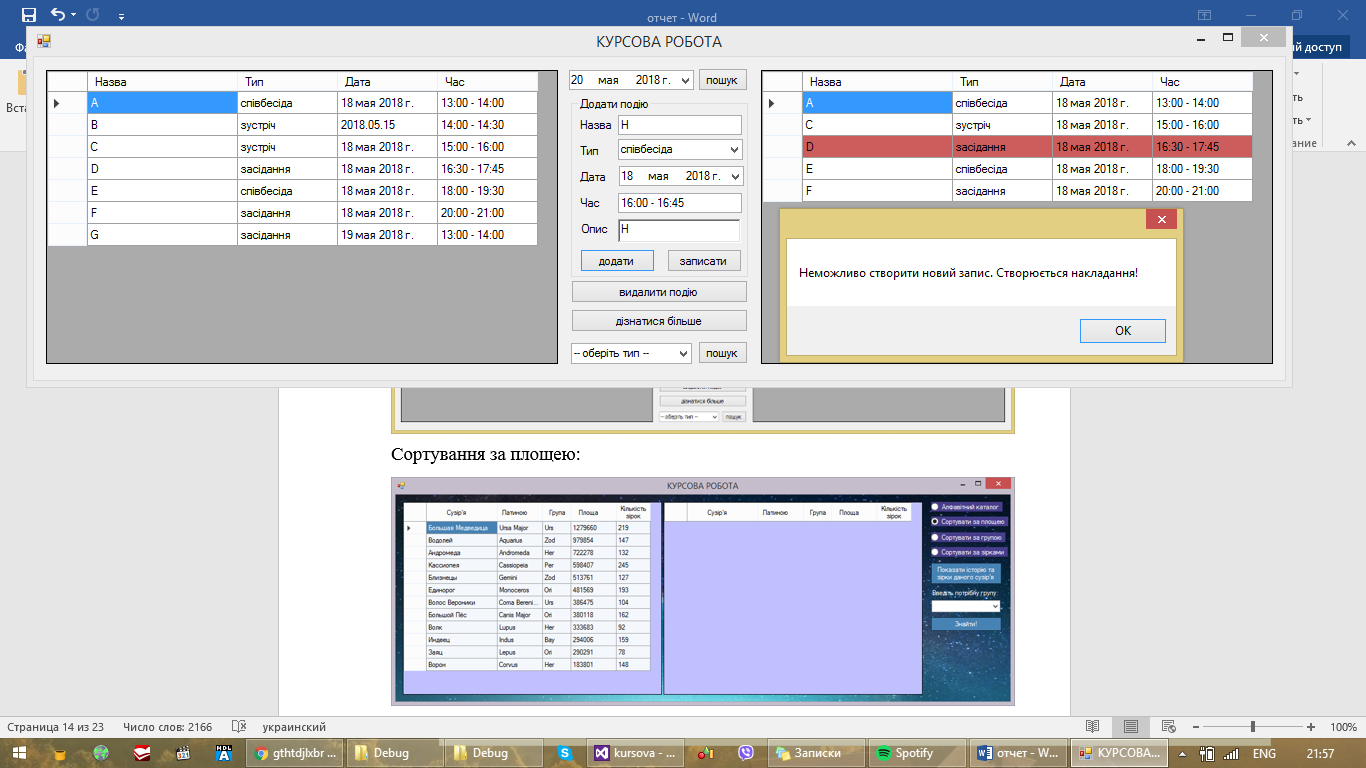
Пошук за датою:

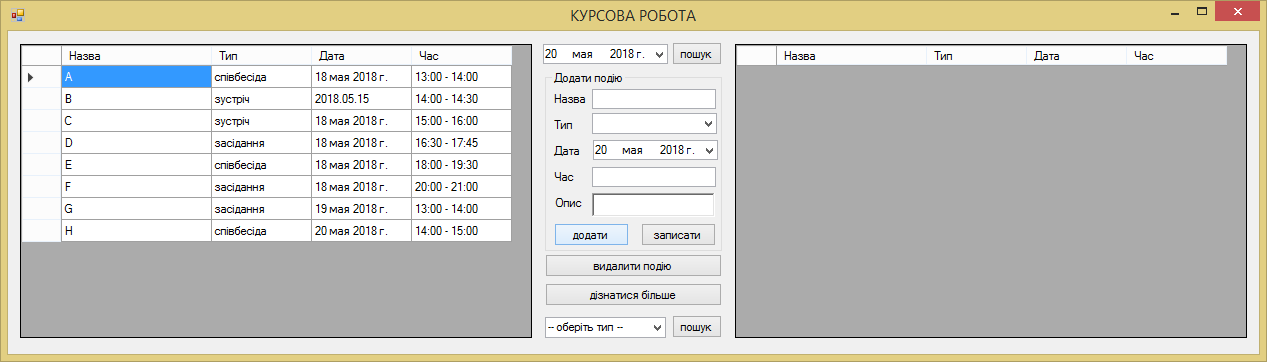


Пошук за типом:

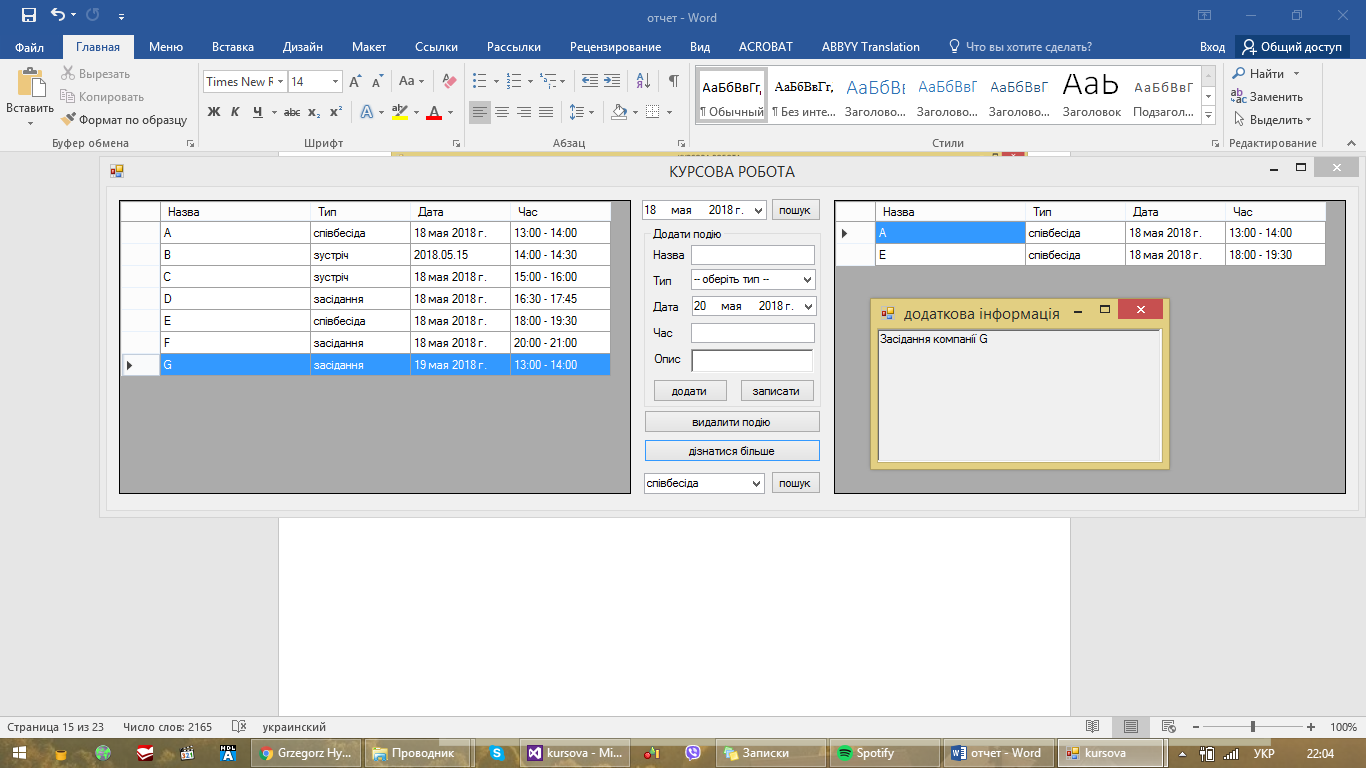


Додавання нового запису:

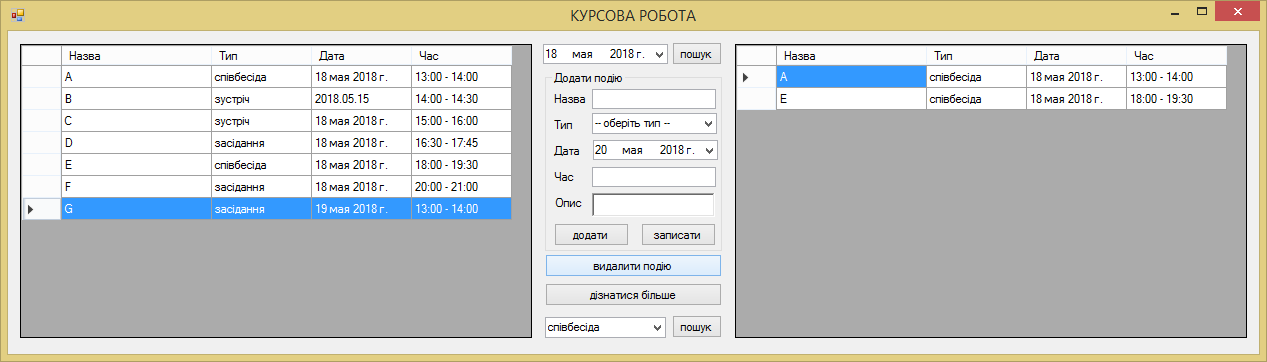


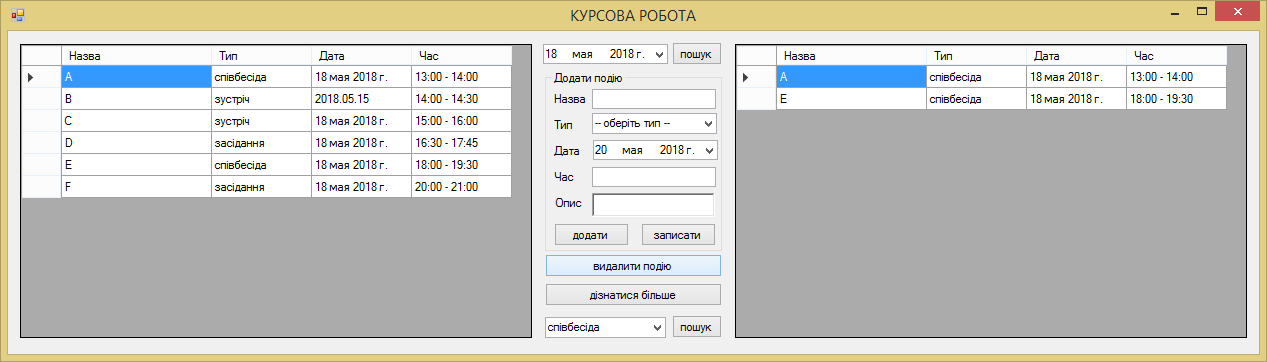


Виведення додаткової інформації:



Видалення події:





# **Висновки**

В даному проекті була розроблена програма для ведення ділового щоденника. Програма була написана на об’єктно-орієнтованій мові програмування С#. Дана програма була розроблена для того, аби мати можливість виконати більше справ за день.

Програма складається з 2 етапів: створення списку подій, пошук подій за конкретною датою. Перший етап потрібен для формування списку справ / подій, які потрібно виконати в певний день та час. Другий – для перегляду сформованого списку запланованих справ.

Отже, дана програма є дуже актуальною в наш час. Адже кожній людині потрібно робити безліч справ кожного дня, а щоб встигнути виконати, якщо ж не всі, то хоча б декілька. Потрібно правильно спланувати день. Програма є дуже простою у використанні. Тому вона буде зручною для користування.

# **Список літератури**

* А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд «Язык программирования С#»
* Э. Стиллмэн, Дж. Грин «Изучаем С#»