МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

автоматизованих систем обробки інформації та управління

**КУРСОВА РОБОТА**

з дисципліни

“ Технології розробки програмного забезпечення ”

на тему

"Розробка сервера пропускного приладу на базі бібліотеки Spring"

|  |  |  |
| --- | --- | --- |
| Керівник : Іванова Л.М. |  | Виконав : Язенок Михайло Сергійович |
| Допущений до захисту |  | студент гр. ІП-61,ФІОТ |
| І\_\_\_І \_\_\_\_\_\_\_\_ 201\_ \_\_\_\_\_\_  підпис |  | 3 курс  № IП-6130 |
| Захистив з оцінкою  \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  оцінка підпис  І\_\_\_І \_\_\_\_\_\_\_\_\_ 201\_ |  | Підпис: |

Київ 2019

Національний технічний університет України “КПІ”

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Основи програмування

Напрям "Інженерія програмного забезпечення"

Курс 3 Група ІП-61 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента**

Язенка Михайла Сергійовича

1. Тема роботи: розробка сервера пропускного приладу на базі бібліотеки Spring.
2. Строк здачі студентом закінченої роботи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Вихідні дані до роботи: вихідні дані відсутні.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

1. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень ): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

1. Дата видачі завдання: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Календарний план

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи |  |  |
| 2. | Підготовка ТЗ |  |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи |  |  |
| 4. | Розробка алгоритму вирішення задачі |  |  |
| 6. | Узгодження алгоритму з керівником |  |  |
| 5. | Розробка сценарію роботи програми |  |  |
| 6. | Узгодження сценарію роботи програми з керівником |  |  |
| 7. | Узгодження з керівником інтерфейсу користувача |  |  |
| 8. | Розробка програмного забезпечення |  |  |
| 9. | Налагодження розрахункової частини програми |  |  |
| 10. | Розробка та налагодження інтерфейсної частини програми |  |  |
| 11. | Узгодження з керівником набору тестів для контрольного прикладу |  |  |
| 12. | Тестування програми |  |  |
| 13. | Підготовка пояснювальної записки |  |  |
| 14. | Здача курсової роботи на перевірку |  |  |
| 15. | Захист курсової роботи |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  |  |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| Керівник |  |  | Іванова Л.М. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

Анотація

Пояснювальна записка до курсової роботи: 128 сторінок, 12 рисунків, 9 таблиць, 2 посилання.

Об’єкт дослідження: Розробка сервера пропускного приладу на базі бібліотеки Spring.

Мета роботи: дослідження методів створення сервера на базі бібліотеки Spring, створення програмного забезпечення для реалізації графічного інтерфейсу пропускного приладу та ключової логіки.

У теоретичних викладках було розглянуто поняття «Пропускного пристрою» та методи створення цього програмного забезпечення.

У розділі «Опис алгоритмів» було розписано покроковий алгоритм реалізації функціоналу.

В описі програмного забезпечення було продемонстровано основну структуру проекта за допомогою UML діаграм. Також було описано стандартні функції і методи класів, які були застосовані в реалізації.

В розділі тестування було проведено низку тестів, які підтвердили надійність програмного забезпечення.

В інструкції користувача було вказано як користуватися функціоналом, який повинен був бути реалізований згідно з технічним завданням.

Виконана програмна реалізація пропускного пристрою та панелі адміністратора.м

СЕРВЕР ПРОПУСНОГО ПРИЛАДУ НА БАЗІ БІБЛІОТЕКИ SPRING

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра автоматизованих систем обробки інформації та управління

|  |  |  |
| --- | --- | --- |
| **РОЗРОБИВ** |  | **ЗАТВЕРДИВ** |
| **Студент**  \_\_\_\_\_\_\_\_\_\_ *М.С. Язенок*  (підпис) (ініціали, прізвище)  “10” листопада 2017 р. |  | **Керівник**  \_\_\_\_\_\_\_\_\_\_ Л.М. Іванова  (підпис) (ініціали, прізвище)  “11” листопада 2017 р. |

Планувальник задач

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

з дисципліни:

«Об'єктно-орієнтоване програмування»

Шифр КПІ ІП-6130.ТЗ

на 15 сторінках

Київ 2017

Зміст

[1 Найменування та область застосування 4](#_Toc10308406)

[2 Підстави для розробки 5](#_Toc10308407)

[2.1 Перелік документів, на підставі яких ведеться розробка 5](#_Toc10308408)

[2.2 Найменування і умовне позначення теми розробки 5](#_Toc10308409)

[3 Призначення розробки 6](#_Toc10308410)

[3.1 Функціональне призначення розробки 6](#_Toc10308411)

[3.2 Експлуатаційне призначення 6](#_Toc10308412)

[4 Вимоги до програмного забезпечення 7](#_Toc10308413)

[4.1 Вимоги до функціональних характеристик 7](#_Toc10308414)

[4.1.1 Вимоги до складу виконуваних функцій 7](#_Toc10308415)

[4.1.2 Вимоги до організації вхідних даних 8](#_Toc10308416)

[4.1.3 Вимоги до організації вихідних даних 8](#_Toc10308417)

[4.2 Вимоги до надійності 8](#_Toc10308418)

[4.2.1 Вимоги до надійного функціонування програми 8](#_Toc10308419)

[4.2.2 Час відновлення після відмови 8](#_Toc10308420)

[4.2.3 Відмови через некоректні дії користувача 9](#_Toc10308421)

[4.3 Умови експлуатації 9](#_Toc10308422)

[4.3.1 Кліматичні умови експлуатації 9](#_Toc10308423)

[4.3.2 Вимоги до видів обслуговування 9](#_Toc10308424)

[4.3.3 Вимоги до кількості та кваліфікації персоналу 9](#_Toc10308425)

[4.4 Вимоги до складу і параметрів технічних засобів 9](#_Toc10308426)

[4.5 Вимоги до інформаційної і програмної сумісності 9](#_Toc10308427)

[4.6 Вимоги до маркування та упаковки 10](#_Toc10308428)

[4.7 Вимоги до транспортування та зберігання 10](#_Toc10308429)

[5 Вимоги до програмної документації 11](#_Toc10308430)

[5.1 Попередній склад програмної документації 11](#_Toc10308431)

[6 Техніко-економічні показники 12](#_Toc10308432)

[7 Стадії та етапи розробки 13](#_Toc10308433)

[7.1 Стадії розробки 13](#_Toc10308434)

[7.2 Етапи розробки 13](#_Toc10308435)

[7.3 Зміст роботи по етапам 13](#_Toc10308436)

[8 Порядок контролю і прийому 14](#_Toc10308437)

[8.1 Види досліджень 14](#_Toc10308438)

[8.2 Загальні вимоги до прийняття роботи 14](#_Toc10308439)

Найменування та область застосування

Найменування – «Сервер пропускного приладу на базі бібліотеки Spring».

Данне програмне забезпечення призначене для симуляції роботи приладу для пропуску в приміщення за допомогою перепусток, які мають користувачі, та коду доступу.

На данному приладі можна змінити код доступу і код контролю за допомогою коду контролю.

Також в приладі передбачена база даних для збереження даних карток користувачів, журналу відвідувань, журналу блокування карток і даних адмніністраторів, які можуть отримати доступ до всіх вище зазначених даних, окрім таблиці інших адміністраторів.

Основною застосування розробленого програмного забезпечення є тестування роботи симуляції пропускного приладу.

Підстави для розробки

Перелік документів, на підставі яких ведеться розробка

Підставою для проведення розробки являється наказ № 4 від 5 травня 2017 року. Наказ затверджено на засіданні кафедри АСОІУ, далі Замовник і прийнято до виконання студентом Язенком Михайлом Сергійовичем, далі Виконавець.

Найменування і умовне позначення теми розробки

Найменування теми розробки – «Розробка планувальника задач».

Умовне позначення теми розробки (шифр теми) – «РСППББS 1.0».

Призначення розробки

Функціональне призначення розробки

Функціональним призначенням розробки програми є надання користувачам можливості отримати доступ до приміщення за допомогою перепустки і коду доступу.

Експлуатаційне призначення

Кінцевими користувачами програми є будь-які люди, яким необхідно протестувати симуляцію роботи пропусного пристрою.

Вимоги до програмного забезпечення

Вимоги до функціональних характеристик

Вимоги до складу виконуваних функцій

Програма повинна забезпечувати можливість виконання нижче наведених функцій:

1. можливість розблокувати двері за допомогою наступних кроків:
   1. вказати ідентифікатор картки користувача (симуляція вставки картки). Пристрій перевіряє існування такої картки в базі даних. Якщо картка не існує або поточний час не співпадає з графіком відвідання, який вказано в картці, то на дисплеї буде зображено повідомлення про заборону доступу, інакше на дисплеї буде зображено ім'я користувача та буде запропоновано ввести код доступу.
   2. ввести код доступу. Дається три спроби (кількість спроб можна налаштовувати в конфігураційному файлі) для введення коду доступу. Якщо всі спроби вичерпано, то картка блокується (в базі даних картка позначається як заблокована, а також вноситься запис в журнал блокування карток про відповідну дію). Якщо код введено вірно, то замок відчиняється на 4 секунди (кількість секунд налаштовується в конфігураційному файлі) і робиться запис в журнал відвідувань в базі даних.
2. можливість змінити код доступу за допомогою наступних кроків:
   1. при відкритих дверях натиснути кнопку контролю (при цьому двері закриваються)
   2. ввести код контролю. Якщо код контролю введено не вірно, то пропускний пристрій повертається до початкового стану.
   3. ввести новий код доступу
3. можливість змінити код контролю за допомогою наступних кроків:
   1. при відкритих дверях затиснути кнопку виклику
   2. ввести код контролю. Якщо код контролю введено не вірно, то пропускний пристрій повертається до початкового стану.
   3. ввести новий код контролю
4. можливість подати звуковий сигнал, при початковому стані пристрою, за допомогою кнопки виклику
5. можливість перейти на панель адміністратора за допомогою логіну та паролю. Панель адміністратора надає наступні можливості:
   1. переглянути таблицю всіх карток доступу
   2. змінити картку доступу
   3. видалити картку доступу
   4. переглянути журнал відвідувань
   5. переглянути журнал блокування карток
   6. можливість вийти з панелі адміністратора

Вимоги до організації вхідних даних

Вхідні дані до програми є конфігураційними файлами, в яких можна вказати наступні властивості:

1. час звучання звукового сигналу в секундах (2 за замовчуванням)
2. час на який відкриваються двері в секундах (4 за замовчуванням)
3. довжину коду доступу (4 за замовчуванням)
4. код доступу (1234 за замовчуванням)
5. довжину коду контроля (4 за замовчуванням)
6. код контроля (4321 за замовчуванням)
7. кількість спроб для введення коду доступу (3 за замовчуванням)

Також вхідним файлом є початкова база даних

Вимоги до організації вихідних даних

Вихідними даними є таблиці карток, журнал відвідувань, журнал блокування карток.

Вимоги до надійності

Вимоги до надійного функціонування програми

Надійне функціонування програми повинно бути забезпечене виконанням Замовником сукупності організаційно-технічних заходів, перелік яких наведено нижче:

1. організацією безперебійного струмопостачання технічних засобів;
2. використанням ліцензійного програмного забезпечення;
3. використанням своєчасно оновленого програмного забезпечення;

Час відновлення після відмови

Час відновлення після відмови, що спричинена неполадкою технічних засобів, крахом операційної системи, не повинно перевищувати час, який необхідний для ліквідації неполадок технічних засобів та переустановлення програмних засобів.

Відмови через некоректні дії користувача

Відмови програми можливі внаслідок некоректних дій користувача системи.

Некоректними діями вважається порушення структури конфігураційних файлів.

Задля запобігання відмов програми через вище вказані причини слід обмежити доступ до конфігураційних файлів пропускного пристрою.

Умови експлуатації

Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, що заявлені до технічних засобів зокрема до умов їх експлуатації.

Вимоги до видів обслуговування

Див. Вимоги до забезпечення надійного функціонування програми.

Програма не потребує будь-яких видів обслуговування.

Вимоги до кількості та кваліфікації персоналу

Мінімальна кількість персоналу, що необхідний для роботи програми, складає 1 штатну одиницю – адміністратор, який визначить список адміністраторів та внесе відповідні зміни до конфігураційних файлів.

Вимоги до складу і параметрів технічних засобів

До технічного засобу, на якому має бути розгорнуто застосування, висуваються наступні вимоги:

1. 32-розрядний процесор з тактовою частотою не нижче 2.4 ГГц;
2. достатній об’єм оперативної пам’яті (не менше 512 Мб);
3. достатній об’єм жорсткого диску (не менше 3 Мб).

Вимоги до інформаційної і програмної сумісності

На технічному засобі, де буде застосовуватися програма, необхідно наступне ПЗ:

1. Java 8;

Вимоги до маркування та упаковки

Вимоги до маркування та упаковки не висуваються.

Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

Вимоги до програмної документації

Попередній склад програмної документації

Склад програмної документації повинен включати в себе:

1. технічне завдання;
2. керівництво користувача.

Техніко-економічні показники

Орієнтовна економічна ефективність не обчислюється.

Стадії та етапи розробки

Стадії розробки

Розробка повинна бути проведена в дві стадії:

1. розробка технічного завдання;
2. робоче проектування.

Етапи розробки

На стадії розробки технічного завдання повинен бути виконаний етап розробки, узгодження і затвердження технічного завдання.

На стадії робочого проектування повинні бути виконаний наведений нижче перелік робіт:

1. розробка програми;
2. тестування програми;
3. розробка програмної документації.

Зміст роботи по етапам

На етапі розробки технічного завдання повинні бути виконані наступні роботи:

1. постановка задачі;
2. визначення і уточнення вимог до технічних засобів;
3. визначення вимог до програми;
4. визначення стадій, етапів і строків розробки програми та документації для неї;
5. узгодження та затвердження технічного завдання.

На етапі розробки програми повинні бути виконані роботи по програмуванню і налагодженню програми.

На етапі тестування повинні бути проведені ряд тестів для визначення та усунення уразливих сторін програмного забезпечення.

На етапі розробки програмної документації повинна бути виконана розробка програмних документів.

Порядок контролю і прийому

Види досліджень

Приймально-здавальні випробування програмного продукту мають проводитися згідно з розробленою виконавцем і узгодженою із замовником “Програмою та методикою випробувань” на об’єкті замовника.

Загальні вимоги до прийняття роботи

На основі проведених досліджень Виконавець спільно з Замовником підписують Акт прийому-здачі програми в експлуатацію.

**Пояснювальна записка  
до курсової роботи**

|  |  |
| --- | --- |
| на тему: | Розробка сервера пропускного приладу на базі бібліотеки Spring |
|  |  |
|  | |

Київ 2019

Зміст

[Вступ 3](#_Toc10316973)

[1 Постановка задачі 4](#_Toc10316974)

[2 Теоретичні викладки 6](#_Toc10316975)

[2.1 Поняття пропускного пристрою 6](#_Toc10316976)

[2.2 Поняття бібліотеки Spring 6](#_Toc10316977)

[3 Опис алгоритмів 7](#_Toc10316978)

[3.1 Алгоритм пропускного пристрою 7](#_Toc10316979)

[4 Опис програмного забезпечення 9](#_Toc10316980)

[4.1 Опис діаграми класів програмного забезпечення 9](#_Toc10316981)

[4.2 Опис класів та їх методів 14](#_Toc10316982)

[4.2.1 Користувацькі функції 14](#_Toc10316983)

[4.2.2 Стандартні функції 22](#_Toc10316984)

[5 Тестування програмного забезпечення 24](#_Toc10316985)

[5.1 План тестування 24](#_Toc10316986)

[5.2 Приклади тестування 24](#_Toc10316987)

[6 Інструкція користувача 30](#_Toc10316988)

[6.1 Призначення програми 30](#_Toc10316989)

[6.2 Вимоги до системи 30](#_Toc10316990)

[6.3 Інструкція по роботі з програмою 30](#_Toc10316991)

[6.4 Склад програмного забезпечення 32](#_Toc10316992)

[Висновки 33](#_Toc10316993)

[Перелік посилань 34](#_Toc10316994)

[Додаток А 35](#_Toc10316995)

# Вступ

Завдання даної курсової роботи є Розробка сервера пропускного приладу на базі бібліотеки Spring, який буде виконувати симуляцію для отримання доступу в приміщення за допомогою картки користувача та коду доступу

Дане програмне забезпечення може бути використано будь-ким, кому необхідно протестувати роботу симуляції пропускного приладу.

# Постановка задачі

Розробити сервер на базі бібліотеки Spring для симуляції роботи пропускного приладу з наступними можливостями:

1. можливість розблокувати двері за допомогою наступних кроків:
   1. вказати ідентифікатор картки користувача (симуляція вставки картки). Пристрій перевіряє існування такої картки в базі даних. Якщо картка не існує або поточний час не співпадає з графіком відвідання, який вказано в картці, то на дисплеї буде зображено повідомлення про заборону доступу, інакше на дисплеї буде зображено ім'я користувача та буде запропоновано ввести код доступу.
   2. ввести код доступу. Дається три спроби (кількість спроб можна налаштовувати в конфігураційному файлі) для введення коду доступу. Якщо всі спроби вичерпано, то картка блокується (в базі даних картка позначається як заблокована, а також вноситься запис в журнал блокування карток про відповідну дію). Якщо код введено вірно, то замок відчиняється на 4 секунди (кількість секунд налаштовується в конфігураційному файлі) і робиться запис в журнал відвідувань в базі даних.
2. можливість змінити код доступу за допомогою наступних кроків:
   1. при відкритих дверях натиснути кнопку контролю (при цьому двері закриваються)
   2. ввести код контролю. Якщо код контролю введено не вірно, то пропускний пристрій повертається до початкового стану.
   3. ввести новий код доступу
3. можливість змінити код контролю за допомогою наступних кроків:
   1. при відкритих дверях затиснути кнопку виклику
   2. ввести код контролю. Якщо код контролю введено не вірно, то пропускний пристрій повертається до початкового стану.
   3. ввести новий код контролю
4. можливість подати звуковий сигнал, при початковому стані пристрою, за допомогою кнопки виклику
5. можливість перейти на панель адміністратора за допомогою логіну та паролю. Панель адміністратора надає наступні можливості:
   1. переглянути таблицю всіх карток доступу
   2. змінити картку доступу
   3. видалити картку доступу
   4. переглянути журнал відвідувань
   5. переглянути журнал блокування карток
   6. можливість вийти з панелі адміністратора

Також дане програмне забезпечення повинно встановлювати зв’язок з таблицями бази даних по патерну DAO.

# Теоретичні викладки

## Поняття пропускного пристрою

Пропускний пристрій – це спеціальний пристрій для забезпечення доступу до деякого об’єкту, наприклад приміщення, за допомогою карток користувача та введеного коду доступу.

Картка користувача представляє собою засіб для перенесення електронної інформації. На картці зберігаються дані про:

1. ідентифікатор картки користувача
2. статус картки (заблокована, розблокована)
3. дані про користувача
   1. ім’я
   2. прізвище
4. дані про графік відвідування
   1. час починаючи з якого користувач може відвідувати приміщення
   2. час до якого користувач може відвідувати приміщення

Код доступу представляє собою набір цифр від 0 до 9 будь-якої довжини (можна налаштувати в конфігураційному файлі).

На введення коду доступу користувачу дається 3 спроби (можна налаштувати в конфігураційному файлі). Після останньої спроби картка користувача блокується та до бази даних пристрою в журнал блокування карток вноситься відповідний запис.

Якщо користувач правильно ввів код доступу, то замок відкривається на 4 секунди (можна налаштувати в конфігураційному файлі) і до бази даних пристрою в журнал відвідувань вноситься відповідний запис.

## Поняття бібліотеки Spring

Бібліотека Spring надає можливість спростити спосіб створення серверу для будь-яких задач. За допомогою сучасних анотацій бібліотека генерує велику кількість коду, що значно пришвидшує процес написання програмного забезпечення на базі клієнт-серверної архітектури.

# Опис алгоритмів

## Алгоритм пропускного пристрою

1. ПОЧАТОК
2. КРОК 1
   1. Вставити картку доступу
      1. Валідувати картку.
         1. Перевірити наявність картки в базі даних
         2. Перевірити належність поточного часу до графіку вказаному в картці
      2. ЯКЩО картка валідована, то перейти до КРОКУ 2, ІНАКШЕ перейти в КІНЕЦЬ.
3. КРОК 2
   1. Вивести на екран повідомлення про необхідність вводу коду доступу
   2. Встановити кількість спроб = 3
   3. Ввести код доступу
      1. ЯКЩО код доступу вірний ТО перейти до КРОКУ 3, ІНАКШЕ:
         1. Зменшити кількість спроб
         2. ЯКЩО кількість спроб > 0 ТО перейти на 3.3, ІНАКШЕ:
            1. Заблокувати картку в базі даних
            2. Внести запис в базу даних про блокування картки
            3. Вивести відповідне повідомлення. Перейти в КІНЕЦЬ
4. КРОК 3
   1. Відчини двері на чотири секунди
   2. Вивести повідомлення про відчинення дверей
5. КРОК 4
   1. ЯКЩО натиснута кнопка контролю, ТО
      1. Заблокувати двері
      2. Вивести повідомлення про необхідність введення коду контролю для зміни коду доступу
   2. Ввести код контролю
   3. ЯКЩО код контролю введено неправильно, ТО перейти в КІНЕЦЬ, ІНАКШЕ:
      1. Вивести повідомлення про необхідність введення нового коду доступу
      2. Введення нового коду доступу
      3. Збереження нового коду доступу, перейти на КРОК 3
6. КРОК 5
   1. ЯКЩО зажата кнопка виклику, ТО
      1. Заблокувати двері
      2. Вивести повідомлення про необхідність введення коду контролю для зміни коду контролю
   2. Ввести код контролю
   3. ЯКЩО код контролю введено неправильно, ТО перейти в КІНЕЦЬ, ІНАКШЕ:
      1. Вивести повідомлення про необхідність введення нового коду контролю
      2. Введення нового коду контроля
      3. Збереження нового коду контроля, перейти на КРОК 3
7. КІНЕЦЬ

# Опис програмного забезпечення

## Опис діаграми класів програмного забезпечення

Клас контролер зображено на рисунку 4.1.

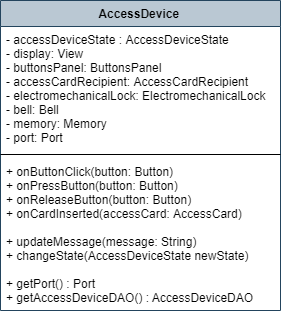


Рисунок 4.1 – Клас контролер

Він включає наступні класи:

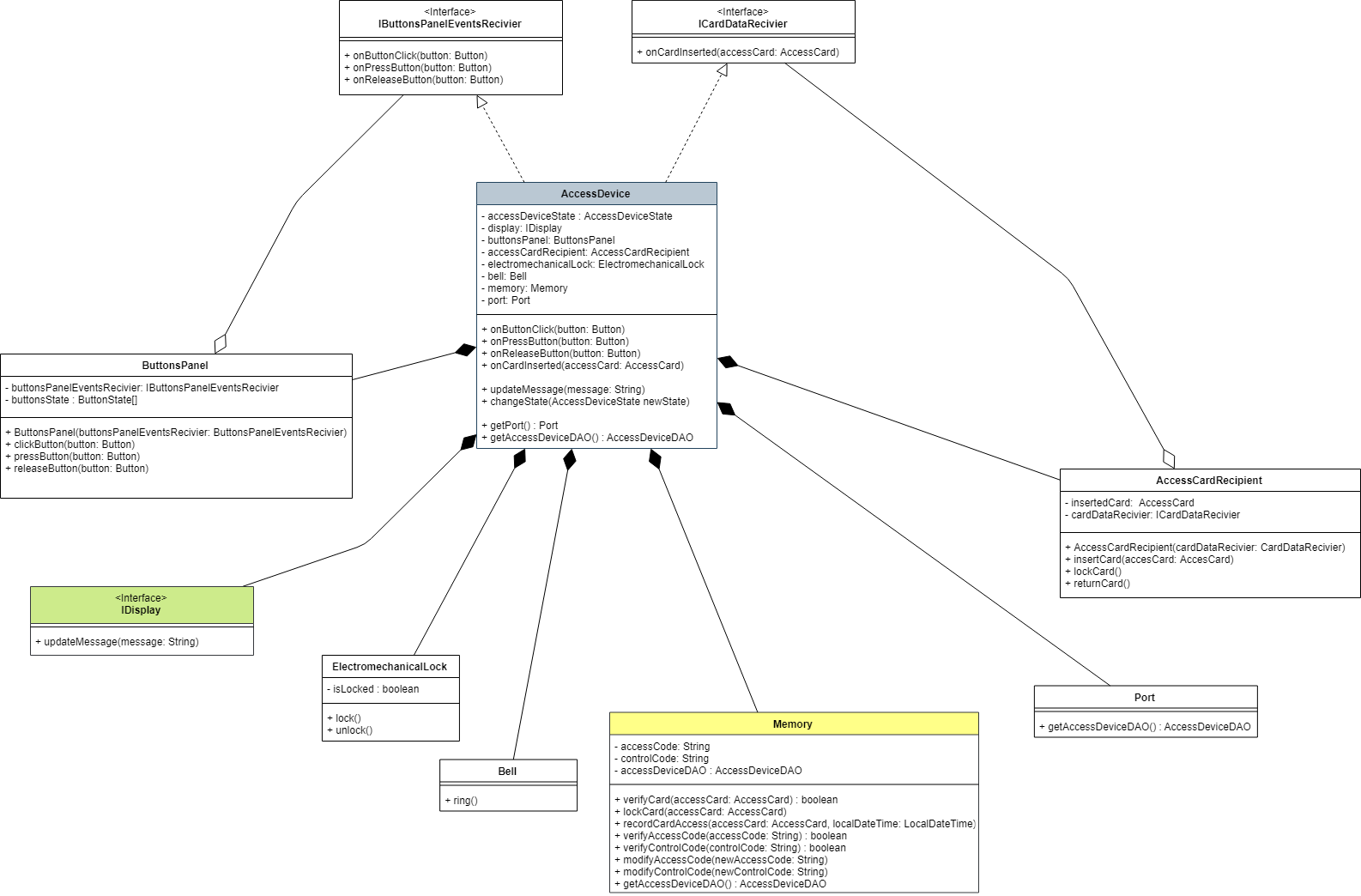


Рисунок 4.2 – Ключові елементи котролера

Display

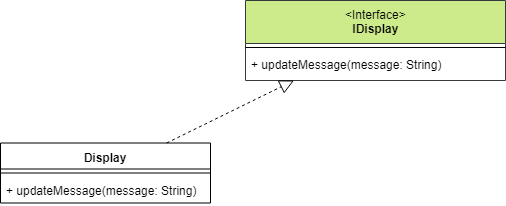


Рисунок 4.3 – Клас Display

**Display** (помічено зеленим) – інтерфейс графічного дисплею, яка виступає у ролі *View*.

Memory

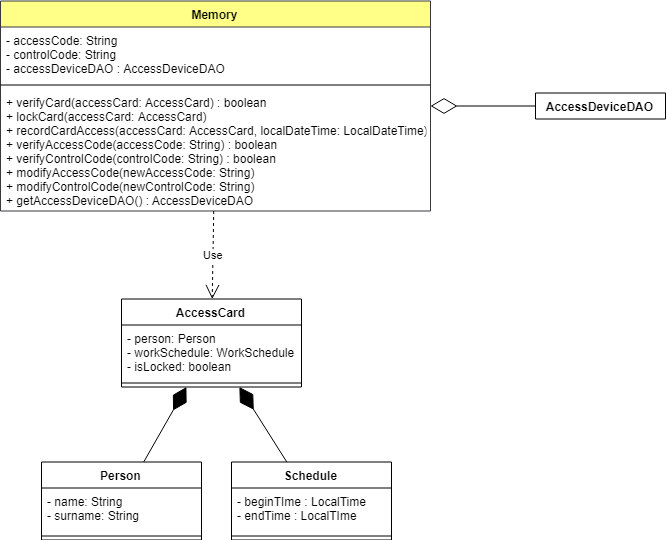


Рисунок 4.4 – Клас Memory

**Memory** (помічено жовтим) – клас внутрішньої пам’яті пристрою, який виступає у ролі *Model*.

Відповідає за взаємодію з базою даних (через Database) та зберігання кодів (коду доступу і коду контроля). Дані картки зберігаються у класі AccessCard (дані про людину та розклад за яким ця людина може відвідувати приміщення)

ButtonsPanel

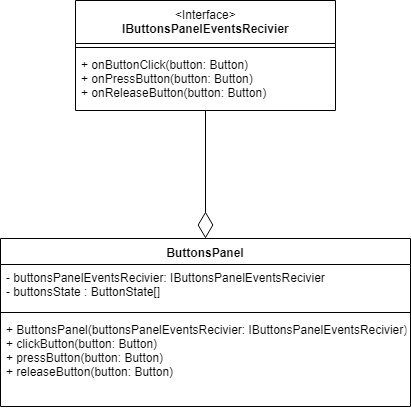


Рисунок 4.5 – Клас ButtonsPanel

**ButtonsPanel** – клас панелі керування з кнопками, відповідає за введення даних і повідомлення контролеру (через інтерфейс **ButtonsPanelEventsRecivier**) про натиснуту/зажату/віджату кнопку. Також контролює стан кнопок (допомагає уникнути ситуації, коли одну і ту саму кнопку двічі зажимають).

AccessCardRecipient

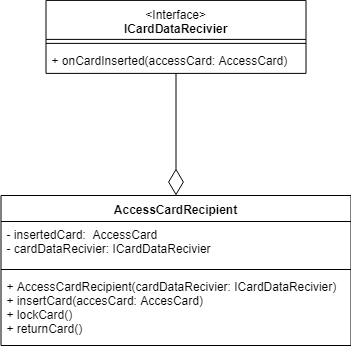


Рисунок 4.6 – Клас AccessCardRecipient

**AccessCardRecipient** – клас приймача карток, відповідає за отримання картки (та повідомлення контролер через **CardDataRecivier** про отримання картки), повернення картки та блокування картки.

Database

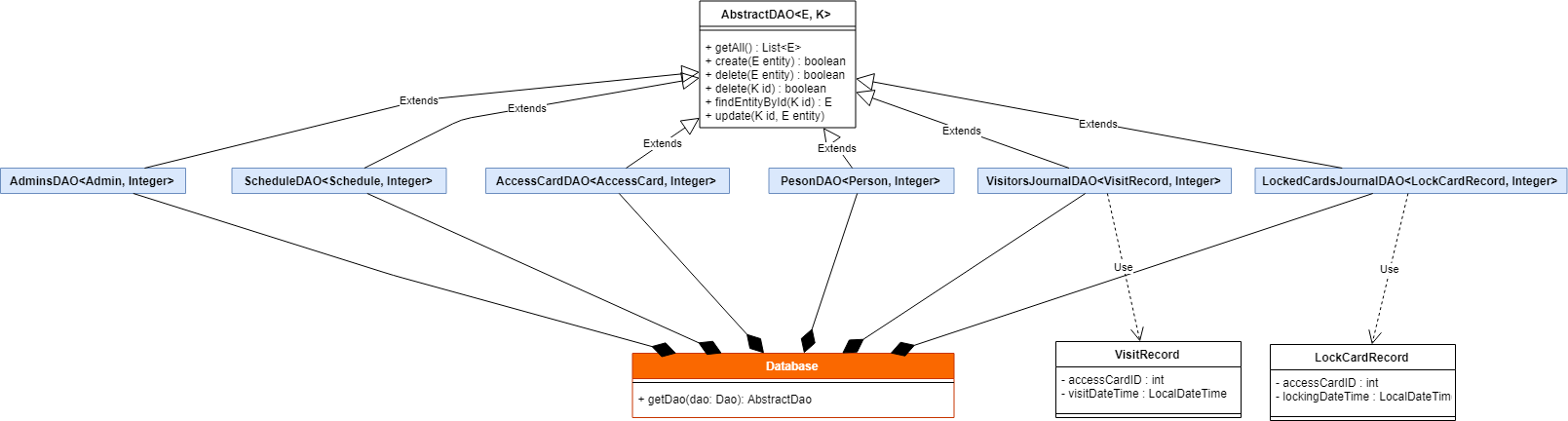


Рисунок 4.7 – Клас Database

**Database** (помічено помаранчевим) – клас для доступу до бази даних, який включає в себе класи:

**AccessCardDAO** - для доступу до карток користувачів

**ScheduleDAO** – для доступу до таблиці розкладу конкретного користувача

**PersonDAO** – для доступу до таблиці користувачів

**VisitorsJournalDAO** - для доступу до журналу відвідувань користувачами приміщення

**LockedCardsJournalDAO** – для доступу до журналу блокування карток

Схема бази даних

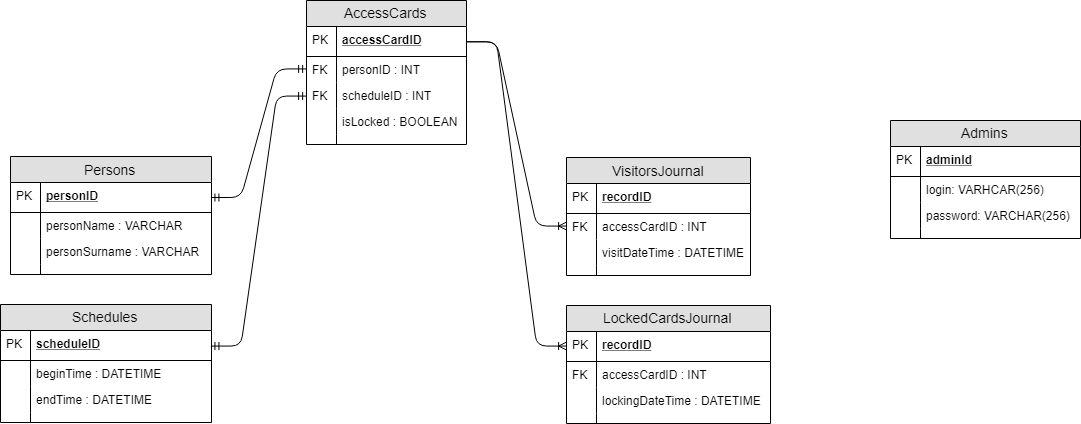


Рисунок 4.8 – Схема бази даних

## Опис класів та їх методів

### Користувацькі функції

Особливості та деталі кожної функції наведені у таблиці 4.1.

Таблиця 4.1 - Специфікація функцій

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | MainForm | MainForm | Конструктор | hide – чи потрібно сховати форму  entrance – чи потрібно активувати ENTRANCE тригер |  | MainForm.h |
| 2 | MainForm | refresh | Оновити список задач |  |  | MainForm.h |
| 3 | MainForm | InitializeComponent | Ініціалізація компонентів |  |  | MainForm.h |
| 4 | MainForm | RefreshButton\_Click | Подія натискання на кнопку оновлення |  |  | MainForm.h |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 5 | MainForm | check\_show\_cycle | Перевірка чи потрібно показати форму |  |  | MainForm.h |
| 6 | CreateTaskForm | CreateTaskForm | Конструктор |  |  | CreateTaskForm.h |
| 7 | CreateTaskForm | InitializeComponent | Инициалізація компонентів |  |  | CreateTaskForm.h |
| 8 | CreateTaskForm | InputHeader | Зчитування заголовку задачі |  | Заголовок задачі | CreateTaskForm.h |
| 9 | CreateTaskForm | InputTrigger | Зчитування тригеру |  | Покажчик на тригер | CreateTaskForm.h |
| 10 | CreateTaskForm | InputAct | Зчитування дії |  | Покажчик на дію | CreateTaskForm.h |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 11 | Form\_Exception | operator std::string | Отримати виключення як string |  | Виключення, як рядок | Form\_Exception.h |
| 12 | Form\_Exception | Get\_as\_string\_with\_solution | Отримати виключення з рішенням |  | Виключення, як рядом з розв’язком | Form\_Exception.h |
| 13 | Task\_Manager | Task\_Manager | Конструктор | file\_name\_ - назва конфігураційного файлу  entrance – чи потрібно активувати ENTRANCE |  | Task\_Manager.h |
| 14 | Task\_Manager | create\_task | Створити задачу | header – заголовок задачі  trigger – покажчик на тригер  act – покажчик на дію |  | Task\_Manager.h |
| 15 | Task\_Manager | delete\_task | Видалити задачу | id\_ - ідентифікатор задачі |  | Task\_Manager.h |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 16 | Task\_Manager | import\_task | Імортувати задачу | import\_file\_name\_ - назва файла для імпорту |  | Task\_Manager.h |
| 17 | Task\_Manager | export\_task | Експортувати задачу | export\_file\_name\_ - шлях для експорту файла  id\_ - ідентифікатор задачі |  | Task\_Manager.h |
| 18 | Task\_Manager | Get\_task\_info | Отримати вектор описів задач |  | Вектор описів задач | Task\_Manager.h |
| 19 | Task | Task | Конструктор | header\_ - заголовок задачі  trigger\_ - покажчик на тригер  act\_ - покажчик на дію |  | Task\_Manager.h |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 20 | Task | calculate\_time\_left | Визначення часу до наступного спрацювання задачі | c\_time – час для якого потрібно провести розрахунки | false – якщо потрібно видалити задачу | Task.h |
| 21 | Task | make\_act | Виконати задачу |  |  | Task.h |
| 22 | Task\_act | make\_act | Виконати задачу |  |  | Task\_act.h |
| 23 | Task\_trigger | calculate\_time\_left | Визначити час до наступного спрацювання тригера | c\_time – час для якого потрібно провести розрахунки | false – якщо потрібно видалити задачу | Task\_trigger.h |
| 24 | Time | Set\_next\_weekday | Збільшити поточну дату, так щоб день тижня = wday\_ | wday\_ - день тижня |  | \_Time.h |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 25 | Time | Set\_next\_month\_day | Збільшити дату, зважаючи на вектор місяців та вектор днів | month\_list – вектор місяців;  days\_list – вектор днів місяця. |  | \_Time.h |
| 26 | Alert\_Manager | add\_alert | Додати повідомлення в чергу | header – заголовок повідомлення;  message – текст повідомлення. |  | Alert\_Manager.h |
| 27 | Task\_Exception | operator std::string | Отримати виключення як рядок |  | Виключення як рядок | graph.hpp |
| 28 | Task\_Exception | Get\_as\_string\_with\_solution | Отримати виключення з рішенням |  | Виключення, як рядом з розв’язком | graph.hpp |
| 29 | thread | thread | Створення новогу потоку |  |  | boost/thread.hpp |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 30 | thread | bind | Передача параметра в функцію | Покажчик на функцію;  Список параметрів для функції. | Функція з встановленими занченнями | boost/thread.hpp |
| 31 | thread | join | Об’єднання двох потоків |  |  | boost/thread.hpp |
| 32 | fstream | fstream | Відкриття файлу | Назва фала;  Параметри відкриття файла. |  | fstream.h |
| 33 | fstream | is\_open | Перерірка чи відкритий файл |  | true – якщо відкритий | fstream.h |
| 34 | fstream | read | Зчитати з бінарного файла | Покажчик на символьний буфер  Кількість байт для зчитування |  | fstream.h |
| 35 | fstream | write | Запис у бінарний файл | Покажчик на символьний буфер  Кількість байт для запису |  | fstream.h |
| 36 | fstream | rdstate | Стан зчитування |  | Поточний стан зчитування | fstream.h |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 37 | mutex | lock | Заблокувати mutex |  |  | boost/thread.hpp |
| 38 | mutex | unlock | Розблокувати mutex |  |  | boost/thread.hpp |
| 39 | date | date | Створення дати | Рік;  місяць;  день. |  | boost/date\_time/gregorian/gregorian.hpp |
| 40 | date | end\_of\_month | Отримати останній день цього місяця |  | Дата, в якій день встановлений як останній день місяця | boost/date\_time/gregorian/gregorian.hpp |

### Стандартні функції

Стандартні функції, що використані у проекті наведені у таблиці 4.2.

Таблица 4.2 – Специфікація функцій

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Бібліотека |
| 1 | msclr::interop::marshal\_as<std::string>() | Перетворює String^ у char. | msclr\marshal\_cppstd.h |

Продовження таблиці 4.2

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Бібліотека |
| 2 | atoi() | Перетворює char у int. |  |
| 3 | msclr::interop::marshal\_as<String^> | Перетворює string у String^. | msclr\marshal\_cppstd.h |
| 4 | time() | Отримує поточний час. | time.h |
| 5 | localtime() | Приведення time\_t до tm | time.h |
| 6 | sort() | Сортування вектору | algorithm |
| 7 | lower\_bound() | Пошук елементу вектора | algorithm |
| 8 | MessageBoxA() | Показати повідомлення | Windows.h |
| 9 | Sleep() | Призупинити роботу потоку | Windows.h |

# Тестування програмного забезпечення

## План тестування

Для уникнення будь-яких помилок в ході використання користувачем створеної програми, необхідно провести тестування програмного забезпечення, передбачивши будь-які ситуації. Саме на цьому базується план тестування наведений нижче.

1. Тестування правильності роботи пропускного пристрою
   1. Тестування проходу по картці, яка є заблокованою
   2. Тестування проходу по картці, яка не проходить по зазначеному графіку
   3. Тестування при введенні коду доступу для відкриття дверей
   4. Тестування при зміні коду доступу
   5. Тестування при зміні коду контролю
2. Тестування правильності роботи панелі адміністратора
   1. Тестування введення неіснуючої пари логіна та пароля
   2. Тестування зміни картки доступу
   3. Тестування створення картки доступу
   4. Тестування видалення картки доступу

## Приклади тестування

Тестування програмного забезпечення було проведено згідно з планом, наведеним у підрозділі 5.1. Були отримані наступні результати.

1. Тестування правильності роботи пропускного пристрою.

Робота програми представлена у вигляді скріншотів наведених на рисунках 5.1, 5.2, 5.3 та у вигляді таблиць 5.1, 5.2 та 5.3.

Таблиця 5.1 – Приклади роботи пропускного пристрою

|  |  |
| --- | --- |
| Мета тесту | Перевірити правильність роботи пропускного пристрою |
| Початковий стан програми | Відкритий сайт симуляції |

Продовження таблиці 5.1

|  |  |
| --- | --- |
| Вхідні дані | Картка, яка є заблокованою |
| Схема проведення тесту | Зповнення текстового поля картки. |
| Очікуваний результат | Повідомлення про заборону доступу |
| Стан програми після проведення випробувань | Видано помилку «Access denied» |



Рисунок 5.1 - Тестування проходу по картці, яка є заблокованою

Таблиця 5.2 – Приклади роботи програми при проходу по картці, яка не проходить по зазначеному графіку

|  |  |
| --- | --- |
| Мета тесту | Перевірити правильність роботи пропускного пристрою |
| Початковий стан програми | Відкритий сайт симуляції |
| Вхідні дані | Картка яка не проходить по зазначеному графіку |
| Схема проведення тесту | Зповнення текстового поля картки. |
| Очікуваний результат | Повідомлення про заборону доступу |
| Стан програми після проведення випробувань | Видано помилку «Access denied» |



Рисунок 5.2 - Тестування проходу по картці, яка не проходить по зазначеному графіку

Таблиця 5.3 – Приклади роботи програми при введенні коду доступу для відкриття дверей

|  |  |
| --- | --- |
| Мета тесту | Перевірити правильність роботи пропускного пристрою |
| Початковий стан програми | Відкритий сайт симуляції |
| Вхідні дані | Правильний код доступу |
| Схема проведення тесту | Введення коду доступу |
| Очікуваний результат | Повідомлення про відчиненння дверей на 4 секунди |
| Стан програми після проведення випробувань | Виведено повідомлення «Door unlocked on 4 seconds» |

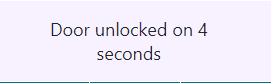


Рисунок 5.3 - Тестування при введенні коду доступу для відкриття дверей

Таблиця 5.4 – Приклади роботи програми при зміні коду доступу

|  |  |
| --- | --- |
| Мета тесту | Перевірити правильність роботи пропускного пристрою |
| Початковий стан програми | Відкритий сайт симуляції з відкритими дверями |
| Вхідні дані | Код контролю |
| Схема проведення тесту | Натиснути кнопку контролю, ввести код контроля |
| Очікуваний результат | Повідомлення про необхідність введення нового коду доступу |
| Стан програми після проведення випробувань | Видано повідомлення «Input new access code» |



Рисунок 5.4 - Тестування при зміні коду доступу

Таблиця 5.5 – Приклади роботи програми при зміні коду котроля

|  |  |
| --- | --- |
| Мета тесту | Перевірити правильність роботи пропускного пристрою |
| Початковий стан програми | Відкритий сайт симуляції з відкритими дверями |
| Вхідні дані | Код контролю |
| Схема проведення тесту | Зажати кнопку виклику, ввести код контроля |
| Очікуваний результат | Повідомлення про необхідність введення нового коду котроля |
| Стан програми після проведення випробувань | Видано повідомлення «Input new control code» |



Рисунок 5.5 - Тестування при зміні коду котроля

1. Тестування правильності роботи панелі адміністратора. (рисунок 5.4, таблиця 5.4)

Таблиця 5.6 – Приклади роботи програми при введенні неіснуючої пари логіна та пароля

|  |  |
| --- | --- |
| Мета тесту | Перевірити правильність роботи панелі адміністратора |
| Початковий стан програми | Відкритий сайт симуляції |
| Вхідні дані | Неіснуюча пара логіна та паролю |
| Схема проведення тесту | Заповнити текстові поля |
| Очікуваний результат | Повідомлення про невірний логін або пароль |

Продовження таблиці 5.6

|  |  |
| --- | --- |
| Стан програми після проведення випробувань | Видано помилку «Wrong login or password» |



Рисунок 5.6 - Тестування введення неіснуючої пари логіна та пароля

Таблиця 5.7 – Приклади роботи програми при зміни картки доступу

|  |  |
| --- | --- |
| Мета тесту | Перевірити правильність роботи панелі адміністратора |
| Початковий стан програми | Відкрита панель адміністратора |
| Вхідні дані | Нове ім'я «NewName» |
| Схема проведення тесту | Заповння текстових полів |
| Очікуваний результат | Таблиця зі зміненими даними |
| Стан програми після проведення випробувань | Картка з ім’ям «NewName» |



Рисунок 5.7 - Тестування зміни картки доступу

Таблиця 5.8 – Приклади роботи програми при створенні картки доступу

|  |  |
| --- | --- |
| Мета тесту | Перевірити правильність роботи панелі адміністратора |
| Початковий стан програми | Відкрита панель адміністратора |
| Вхідні дані | Дані для нової картки |
| Схема проведення тесту | Заповнити текстові поля |
| Очікуваний результат | Таблиця з новою карткою |
| Стан програми після проведення випробувань | Таблиця з новою карткою |



Рисунок 5.8 - Тестування створення картки доступу

Таблиця 5.9 – Приклади роботи програми при введенні неіснуючої пари логіна та пароля

|  |  |
| --- | --- |
| Мета тесту | Перевірити правильність роботи панелі адміністратора |
| Початковий стан програми | Відкрита панель адміністратора |
| Вхідні дані | Відсутні |
| Схема проведення тесту | Натиснути кнопку видалення |
| Очікуваний результат | Таблиця без видаленої картки |
| Стан програми після проведення випробувань | Таблиця без видаленої картки |

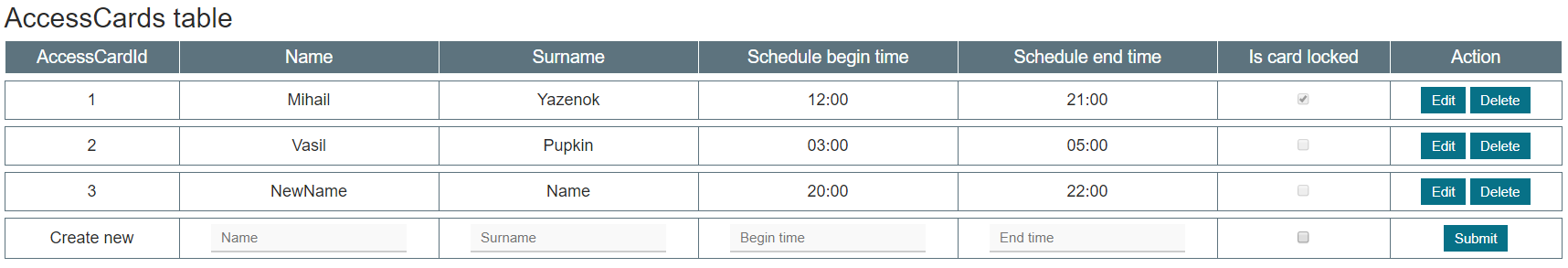


Рисунок 5.9 - Тестування видалення картки доступу

# Інструкція користувача

## Призначення програми

Функціональним призначенням розробки програми є надання користувачам можливості отримати доступ до приміщення за допомогою перепустки і коду доступу.

## Вимоги до системи

- Intel® Core 2 або AMD Athlon® 64 процесор; частота процесора 2 GHz або швидше

- Microsoft Windows 7 with Service Pack 1, Windows 8.1, або Windows 10

- 2 GB або більше RAM (8 GB рекомендовано)

- 2 GB або більше вільного місця на диску

- 1024 x 768 дисплей (1920x1080 рекомендовано)

- Java 8

## Інструкція по роботі з програмою

Інтерфейс пропускного пристрою наведений на рисунку 6.1.

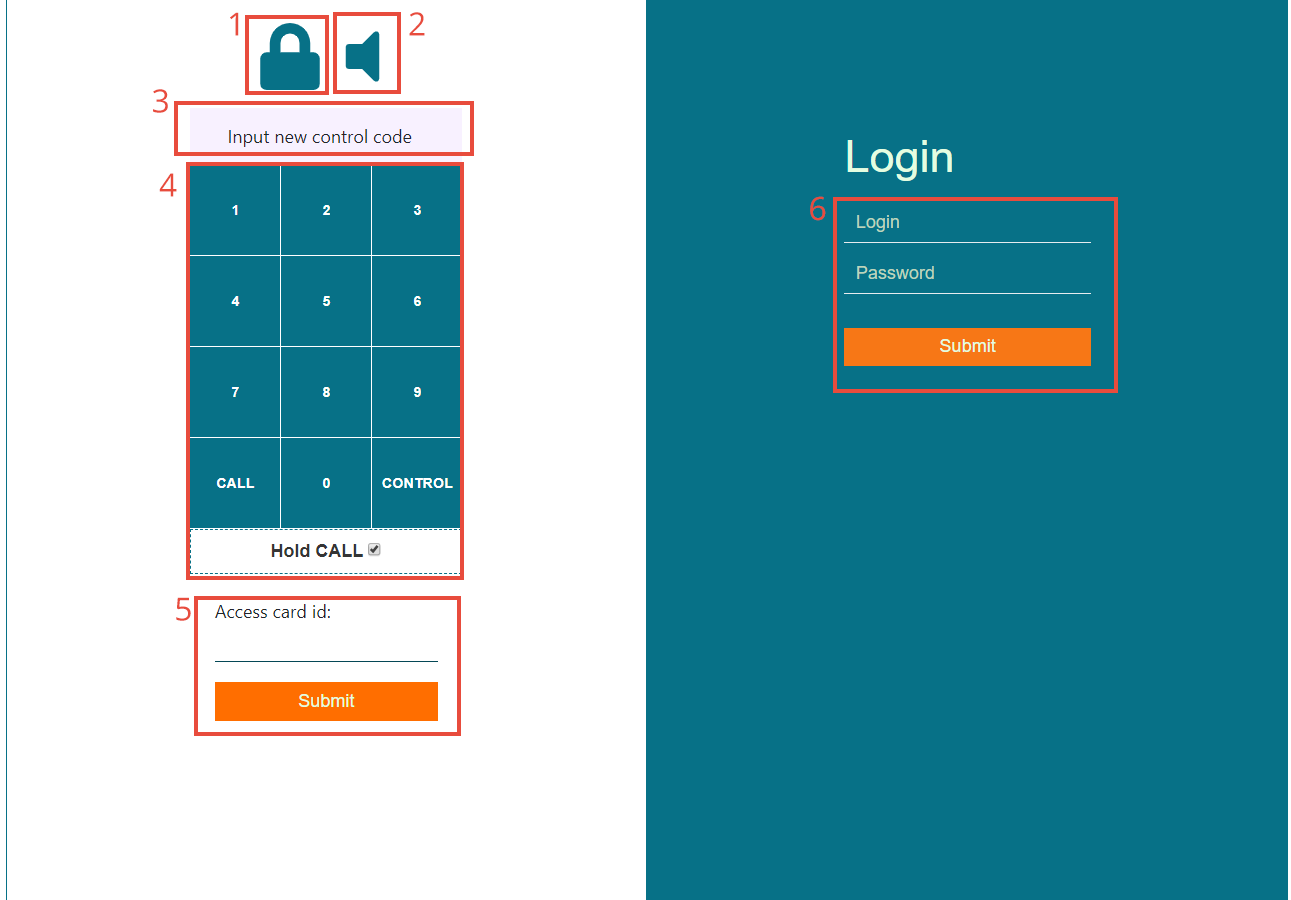


Рисунок 6.1 - Інтерфейс пропускного пристрою

1 – статус замка на дверях (закритий/відкритий)

2 – статус дзвінка

3 – дисплей, де виводяться повідомлення пропускного пристрою

4 – панель для введення кодів, зміни кодів та вмикання дзвінка

5 – поле для введення ідентифікатора картки

6 – поля для входу адміністратора

Інтерфейс панелі адміністратора наведений на рисунку 6.2.

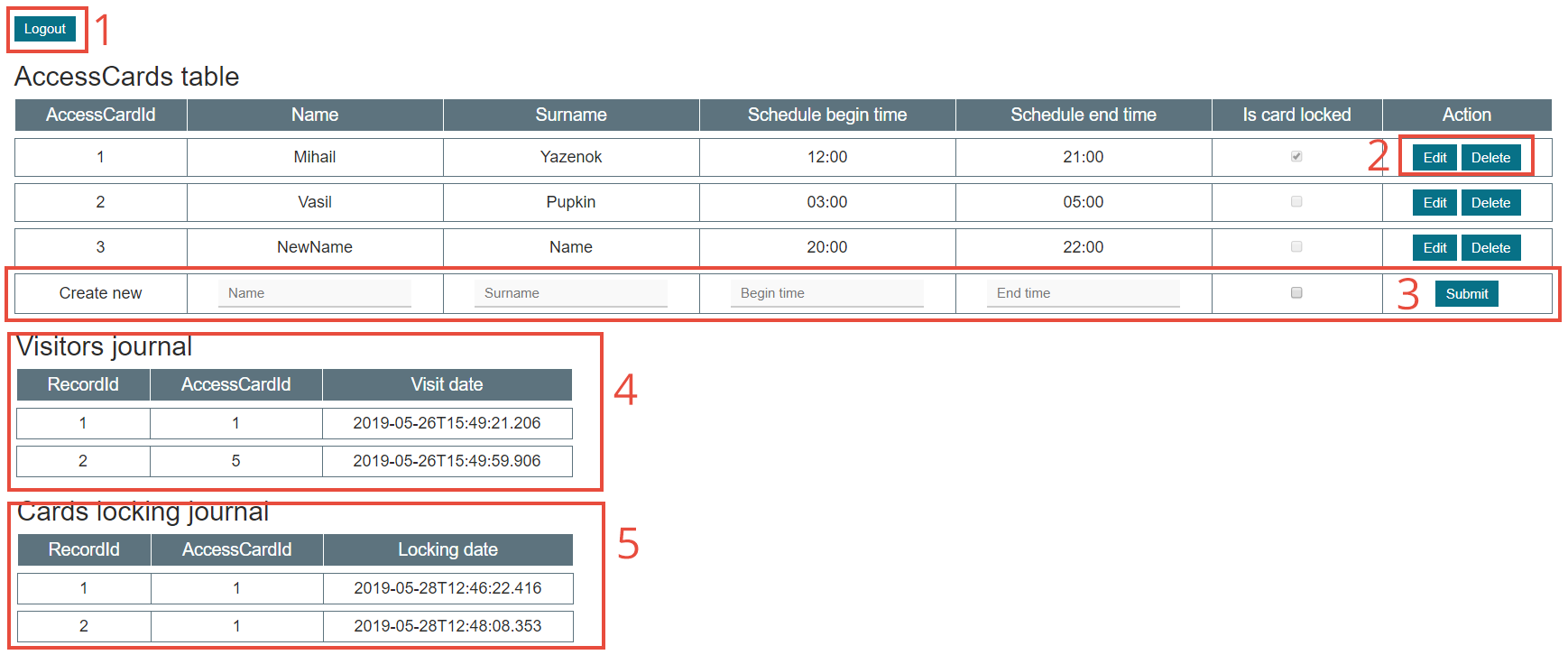


Рисунок 6.2 - Інтерфейс панелі адмністратора

1 – кнопка для виходу з панелі адмністратора

2 – кнопки для редагування/видалення картки

3 – інтерфейс для створення нової картки

4 – таблиця відвідувань

5 – таблиця заблокованих карток

## Склад програмного забезпечення

Готовий програмний продукт являє собою папку web-застосунку, який можна розгорнути на сервері. В папку входять ресурси конфігураційних файлів та бази даних. За допомогою конфігураційних файлів можна налаштувати пропускний пристрій, як було зазначено в попередніх розділах. В базу даних можна внести адміністратора для швидкого доступу до панелі адміністратора з сайту сервера.

# Висновки

На аналітичному етапі розробки програми були розібрані способи створення програмного забезпечення з використанням бібліотеки Spring, яка повинна була забезпечити легкий спосіб побудови клієнт-серверної архітектури.

При проектуванні програмного забезпечення були ретельно продумані структура майбутнього проекту та графічний інтерфейс, за допомогою якого користувач мав би змогу взаємодіяти з пропускним пристроєм, а адміністратор з панеллю адміністрування.

На етапі по дослідження була проведена низка експериментів, які дали змогу побачити та усунути слабкі сторони створеного програмного забезпечення, також було проведено опитування серед користувачів, на основі якого можна сказати, що інтерфейс «Пропускного пристрою» досить інтуїтивний і не потребує додаткових пояснень.

Метою курсової роботи було дослідження методів створення сервера на базі бібліотеки Spring, створення програмного забезпечення для реалізації графічного інтерфейсу пропускного приладу та ключової логіки, тому я вважаю, що я досяг поставленої мети.

# Перелік посилань

1. Ентоні Уільямс Паралельне програмування на С++ в дії. Практика розробки багатопотокових програм — М.: «ДМК Пресс», 2012.
2. Мая Посч Mastering C++ Multithreading — М.: «Packt Publishing», 2017.
3. Wikipedia [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – [Планировщик задач]. – Режим доступу: <https://ru.wikipedia.org/wiki/> Планировщик\_задач (дата звернення 10.11.2017). – Назва з екрана.

# Додаток А

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення планувальника задач*

(Вид носія даних)

*CD-RW*

(Обсяг програми (документа), арк., Кб)

*76 арк, 2017 Кб*

*студента групи ІП-61 ІІ курсу*

Язенка М.С.

Київ 2017

1. main.cpp

#include "stdafx.h"

#include "MainForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

int main(array<String^>^ args)

{

std::vector<std::string> argv;

for (int i = 0; i < args->Length; ++i)

argv.push\_back(msclr::interop::marshal\_as<std::string>(args[i]->ToString()));

if (std::find(argv.begin(), argv.end(), "--help") != argv.end())

{

printf("\

Task Manager help\n\

=================\n\

This program was developed for creating and running tasks.\n\

\

\n\tTaskManager [parameters]\n\

\

\nparameters:\n\

\t--run, --r - run Task Manager\n\

\t--s, --show - show Task Manager after run (work only with \"--r\")\n\

\t--e, --entrance - make all entrance tasks (work only with \"--r\")\n\

");

return 0;

}

#ifndef SHOW\_FORM

if (std::find(argv.begin(), argv.end(), "--r") != argv.end() || std::find(argv.begin(), argv.end(), "--run") != argv.end())

{

bool hide, entrance;

if (std::find(argv.begin(), argv.end(), "--s") != argv.end() || std::find(argv.begin(), argv.end(), "--show") != argv.end())

hide = false;

else

hide = true;

if (std::find(argv.begin(), argv.end(), "--e") != argv.end() || std::find(argv.begin(), argv.end(), "--entrance") != argv.end())

entrance = true;

else

entrance = false;

#endif // !SHOW\_FORM

#ifdef SHOW\_FORM

bool hide, entrance;

hide = false;

entrance = true;

#endif // SHOW\_FORM

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Application::Run(gcnew TaskManagerForm::MainForm(hide, entrance));

#ifndef SHOW\_FORM

}

else

{

boost::interprocess::shared\_memory\_object obj;

try

{

obj = boost::interprocess::shared\_memory\_object(boost::interprocess::open\_only, "ShowTaskManagerBool", boost::interprocess::read\_write);

}

catch (boost::interprocess::interprocess\_exception &e)

{

if (e.get\_error\_code() == boost::interprocess::not\_found\_error)

{

printf("\nError:\n\tShowTaskManagerBool was not created.\nSolution:\n\tBe sure that Task Manager was already started.\n\tIf not: start Task Manager with parameters \"--r --s\"\n");

return 0;

}

}

boost::interprocess::mapped\_region region(obj, boost::interprocess::read\_write);

bool \*show = static\_cast<bool\*>(region.get\_address());

\*show = true;

return 0;

}

#endif // !SHOW\_FORM

return 0;

}

1. MainForm.h

#pragma once

#include "stdafx.h"

#include "CreateTaskForm.h"

#define CONFIG\_FILE\_NAME "C:/Users/mikle/Documents/Visual Studio 2017/Projects/GitHub/Task\_manager/Coursework/Debug/TasksConfig.tm"

namespace TaskManagerForm {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

public ref class MainForm : public System::Windows::Forms::Form

{

public:

/\*\*

\* @param hide if need to hide window

\* @param entrance if need to start entrance tasks

\*/

MainForm(bool hide, bool entrance);

void refresh();

Task\_Manager\* Get\_TaskManager() { return m\_TaskManager; }

static MainForm^ Get() { return m\_this; }

protected:

~MainForm();

private: System::Windows::Forms::Label^ ListHeader;

private: System::Windows::Forms::DataGridView^ TaskList;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ TaskId;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ TaskName;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ TaskDescription;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ TimeLeft;

private: System::Windows::Forms::DataGridViewButtonColumn^ DeleteTask;

private: System::Windows::Forms::DataGridViewButtonColumn^ TaskExport;

private: System::Windows::Forms::Button^ RefreshButton;

private: System::ComponentModel::Container ^components;

private: System::Windows::Forms::MenuStrip^ menuStrip1;

private: System::Windows::Forms::ToolStripMenuItem^ toolStripMenuItem1;

private: System::Windows::Forms::ToolStripMenuItem^ createTaskToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ exitToolStripMenuItem;

private: System::Windows::Forms::FolderBrowserDialog^ ExportTask;

private: System::Windows::Forms::ToolStripMenuItem^ importTaskToolStripMenuItem;

private: System::Windows::Forms::OpenFileDialog^ ImportTask;

private:

Task\_Manager \*m\_TaskManager;

CreateTaskForm^ m\_CreateForm;

Threading::Thread^ m\_check\_show\_thread;

bool m\_check\_show\_thread\_exit;

bool m\_exit;

bool m\_hide;

static MainForm^ m\_this;

#pragma region Windows Form Designer generated code

void InitializeComponent(void)

{

this->ListHeader = (gcnew System::Windows::Forms::Label());

this->TaskList = (gcnew System::Windows::Forms::DataGridView());

this->TaskId = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->TaskName = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->TaskDescription = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->TimeLeft = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->DeleteTask = (gcnew System::Windows::Forms::DataGridViewButtonColumn());

this->TaskExport = (gcnew System::Windows::Forms::DataGridViewButtonColumn());

this->RefreshButton = (gcnew System::Windows::Forms::Button());

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->toolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->createTaskToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->importTaskToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->exitToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->ExportTask = (gcnew System::Windows::Forms::FolderBrowserDialog());

this->ImportTask = (gcnew System::Windows::Forms::OpenFileDialog());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->TaskList))->BeginInit();

this->menuStrip1->SuspendLayout();

this->SuspendLayout();

//

// ListHeader

//

this->ListHeader->AutoSize = true;

this->ListHeader->Font = (gcnew System::Drawing::Font(L"Open Sans", 14));

this->ListHeader->Location = System::Drawing::Point(12, 33);

this->ListHeader->Name = L"ListHeader";

this->ListHeader->Size = System::Drawing::Size(149, 33);

this->ListHeader->TabIndex = 1;

this->ListHeader->Text = L"List of tasks:";

//

// TaskList

//

this->TaskList->AllowUserToAddRows = false;

this->TaskList->AllowUserToDeleteRows = false;

this->TaskList->ColumnHeadersHeight = 30;

this->TaskList->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::DisableResizing;

this->TaskList->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(6) {

this->TaskId, this->TaskName,

this->TaskDescription, this->TimeLeft, this->DeleteTask, this->TaskExport

});

this->TaskList->Location = System::Drawing::Point(18, 69);

this->TaskList->Name = L"TaskList";

this->TaskList->RowTemplate->Height = 24;

this->TaskList->Size = System::Drawing::Size(653, 150);

this->TaskList->TabIndex = 2;

this->TaskList->CellClick += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &MainForm::TaskList\_CellClick);

//

// TaskId

//

this->TaskId->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::ColumnHeader;

this->TaskId->HeaderText = L"id";

this->TaskId->Name = L"TaskId";

this->TaskId->ReadOnly = true;

this->TaskId->Width = 48;

//

// TaskName

//

this->TaskName->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::ColumnHeader;

this->TaskName->HeaderText = L"Name";

this->TaskName->Name = L"TaskName";

this->TaskName->ReadOnly = true;

this->TaskName->Width = 74;

//

// TaskDescription

//

this->TaskDescription->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::ColumnHeader;

this->TaskDescription->HeaderText = L"Description";

this->TaskDescription->Name = L"TaskDescription";

this->TaskDescription->ReadOnly = true;

this->TaskDescription->Width = 108;

//

// TimeLeft

//

this->TimeLeft->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::ColumnHeader;

this->TimeLeft->HeaderText = L"Time left from last refresh";

this->TimeLeft->Name = L"TimeLeft";

this->TimeLeft->ReadOnly = true;

this->TimeLeft->Width = 198;

//

// DeleteTask

//

this->DeleteTask->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::AllCells;

this->DeleteTask->HeaderText = L"Delete";

this->DeleteTask->Name = L"DeleteTask";

this->DeleteTask->Width = 55;

//

// TaskExport

//

this->TaskExport->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::AllCells;

this->TaskExport->HeaderText = L"Export";

this->TaskExport->Name = L"TaskExport";

this->TaskExport->Width = 54;

//

// RefreshButton

//

this->RefreshButton->Cursor = System::Windows::Forms::Cursors::Hand;

this->RefreshButton->Location = System::Drawing::Point(518, 40);

this->RefreshButton->Name = L"RefreshButton";

this->RefreshButton->Size = System::Drawing::Size(75, 23);

this->RefreshButton->TabIndex = 3;

this->RefreshButton->Text = L"Refresh";

this->RefreshButton->UseVisualStyleBackColor = true;

this->RefreshButton->Click += gcnew System::EventHandler(this, &MainForm::RefreshButton\_Click);

//

// menuStrip1

//

this->menuStrip1->ImageScalingSize = System::Drawing::Size(20, 20);

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this->toolStripMenuItem1 });

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(767, 28);

this->menuStrip1->TabIndex = 4;

this->menuStrip1->Text = L"menuStrip1";

//

// toolStripMenuItem1

//

this->toolStripMenuItem1->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {

this->createTaskToolStripMenuItem,

this->importTaskToolStripMenuItem, this->exitToolStripMenuItem

});

this->toolStripMenuItem1->Name = L"toolStripMenuItem1";

this->toolStripMenuItem1->Size = System::Drawing::Size(44, 24);

this->toolStripMenuItem1->Text = L"File";

//

// createTaskToolStripMenuItem

//

this->createTaskToolStripMenuItem->Name = L"createTaskToolStripMenuItem";

this->createTaskToolStripMenuItem->Size = System::Drawing::Size(159, 26);

this->createTaskToolStripMenuItem->Text = L"Create task";

this->createTaskToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainForm::createTaskToolStripMenuItem\_Click);

//

// importTaskToolStripMenuItem

//

this->importTaskToolStripMenuItem->Name = L"importTaskToolStripMenuItem";

this->importTaskToolStripMenuItem->Size = System::Drawing::Size(159, 26);

this->importTaskToolStripMenuItem->Text = L"Import task";

this->importTaskToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainForm::importTaskToolStripMenuItem\_Click);

//

// exitToolStripMenuItem

//

this->exitToolStripMenuItem->Name = L"exitToolStripMenuItem";

this->exitToolStripMenuItem->Size = System::Drawing::Size(159, 26);

this->exitToolStripMenuItem->Text = L"Exit";

this->exitToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainForm::exitToolStripMenuItem\_Click);

//

// ImportTask

//

this->ImportTask->Filter = L"Task files (.tm)|\*.tm";

//

// MainForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(767, 464);

this->Controls->Add(this->RefreshButton);

this->Controls->Add(this->TaskList);

this->Controls->Add(this->ListHeader);

this->Controls->Add(this->menuStrip1);

this->MainMenuStrip = this->menuStrip1;

this->Margin = System::Windows::Forms::Padding(4);

this->Name = L"MainForm";

this->Text = L"Task Manager";

this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this, &MainForm::MainForm\_FormClosing);

this->VisibleChanged += gcnew System::EventHandler(this, &MainForm::MainForm\_VisibleChanged);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->TaskList))->EndInit();

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private:

// Check if we need to show window

void check\_show\_cycle();

System::Void RefreshButton\_Click(System::Object^ sender, System::EventArgs^ e);

System::Void createTaskToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

System::Void TaskList\_CellClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e);

System::Void importTaskToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

System::Void MainForm\_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e);

System::Void exitToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

System::Void MainForm\_VisibleChanged(System::Object^ sender, System::EventArgs^ e);

};

}

1. MainForm.cpp

#include "stdafx.h"

#include "MainForm.h"

#include "CreateTaskForm.h"

namespace TaskManagerForm

{

MainForm::MainForm(bool hide, bool entrance)

:m\_hide(hide), m\_exit(false)

{

if (m\_this != nullptr)

exit(EXIT\_FAILURE);

m\_this = this;

m\_check\_show\_thread\_exit = false;

m\_check\_show\_thread = gcnew Threading::Thread(gcnew Threading::ThreadStart(this, &MainForm::check\_show\_cycle));

m\_check\_show\_thread->Start();

InitializeComponent();

try

{

m\_TaskManager = new Task\_Manager(CONFIG\_FILE\_NAME, entrance);

refresh();

}

catch (Task\_Exception &e)

{

MessageBox::Show(gcnew String(e.Get\_as\_string\_with\_solution().c\_str()), "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

MainForm::~MainForm()

{

if (components)

{

delete components;

}

m\_exit = false;

m\_check\_show\_thread\_exit = true;

m\_check\_show\_thread->Join();

if (m\_TaskManager != nullptr)

delete m\_TaskManager;

}

void MainForm::refresh()

{

this->Cursor = Cursors::WaitCursor;

std::string tmp;

std::vector<Task\_Info\_t> tasks\_info = m\_TaskManager->Get\_task\_info();

while (TaskList->RowCount != 0)

TaskList->Rows->RemoveAt(0);

unsigned int n = tasks\_info.size();

for (unsigned int i = 0; i < n; ++i)

{

TaskList->Rows->Add();

TaskList->Rows[i]->Cells[0]->Value = tasks\_info[i].header.id;

TaskList->Rows[i]->Cells[1]->Value = gcnew System::String(tasks\_info[i].header.name.c\_str());

TaskList->Rows[i]->Cells[2]->Value = gcnew System::String(tasks\_info[i].header.desc.c\_str());

if (tasks\_info[i].time\_left == INF)

TaskList->Rows[i]->Cells[3]->Value = gcnew System::String("More then hour...");

else

TaskList->Rows[i]->Cells[3]->Value = tasks\_info[i].time\_left;

TaskList->Rows[i]->Cells[4]->Value = gcnew System::String("Delete");

TaskList->Rows[i]->Cells[5]->Value = gcnew System::String("Export");

}

this->Cursor = Cursors::Default;

}

void MainForm::check\_show\_cycle()

{

boost::interprocess::shared\_memory\_object obj;

try

{

obj = boost::interprocess::shared\_memory\_object(boost::interprocess::create\_only, "ShowTaskManagerBool", boost::interprocess::read\_write);

obj.truncate(sizeof(bool));

}

catch (boost::interprocess::interprocess\_exception &e)

{

if (e.get\_error\_code() == boost::interprocess::already\_exists\_error)

obj = boost::interprocess::shared\_memory\_object(boost::interprocess::open\_only, "ShowTaskManagerBool", boost::interprocess::read\_write);

else

throw;

}

boost::interprocess::mapped\_region region(obj, boost::interprocess::read\_write);

bool \*show\_window = static\_cast<bool\*>(region.get\_address());

\*show\_window = false;

while (!m\_check\_show\_thread\_exit)

{

if (\*show\_window)

{

m\_hide = false;

this->Show();

\*show\_window = false;

}

Sleep(1000);

}

boost::interprocess::shared\_memory\_object::remove("ShowTaskManagerBool");

}

System::Void MainForm::RefreshButton\_Click(System::Object^ sender, System::EventArgs^ e)

{

refresh();

}

System::Void MainForm::createTaskToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

m\_CreateForm = gcnew TaskManagerForm::CreateTaskForm;

m\_CreateForm->Show();

}

System::Void MainForm::exitToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

m\_exit = true;

this->Close();

}

System::Void MainForm::TaskList\_CellClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e)

{

try

{

unsigned int id;

if (e->ColumnIndex == 4)

{

id = atoi(msclr::interop::marshal\_as<std::string>(this->TaskList->Rows[e->RowIndex]->Cells[0]->Value->ToString()).c\_str());

m\_TaskManager->delete\_task(id);

refresh();

MessageBox::Show(L"Task was successfully deleted!", L"Info", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

else if (e->ColumnIndex == 5)

{

id = atoi(msclr::interop::marshal\_as<std::string>(this->TaskList->Rows[e->RowIndex]->Cells[0]->Value->ToString()).c\_str());

System::Windows::Forms::DialogResult result = ExportTask->ShowDialog();

if (result == System::Windows::Forms::DialogResult::OK)

{

std::string file\_name = msclr::interop::marshal\_as<std::string>(ExportTask->SelectedPath) + "/Task.tm";

m\_TaskManager->export\_task(file\_name.c\_str(), id);

MessageBox::Show(L"Task was successfully exported!", L"Info", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

}

}

catch (Task\_Exception &e)

{

MessageBox::Show(gcnew String(e.Get\_as\_string\_with\_solution().c\_str()), "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

System::Void MainForm::importTaskToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

try

{

System::Windows::Forms::DialogResult result = ImportTask->ShowDialog();

if (result == System::Windows::Forms::DialogResult::OK)

{

std::string file\_name = msclr::interop::marshal\_as<std::string>(ImportTask->FileName);

m\_TaskManager->import\_task(file\_name.c\_str());

refresh();

MessageBox::Show(L"Task was successfully imported!", L"Info", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

}

catch (Task\_Exception &e)

{

MessageBox::Show(gcnew String(e.Get\_as\_string\_with\_solution().c\_str()), "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

System::Void MainForm::MainForm\_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e)

{

m\_hide = true;

this->Hide();

if(!m\_exit)

e->Cancel = true;

}

System::Void MainForm::MainForm\_VisibleChanged(System::Object^ sender, System::EventArgs^ e)

{

if (m\_hide)

this->Visible = false;

else

this->Visible = true;

}

}

1. CreateTaskForm.h

#pragma once

#include "stdafx.h"

#include "Form\_Exception.h"

namespace TaskManagerForm {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

public ref class CreateTaskForm : public System::Windows::Forms::Form

{

public:

CreateTaskForm(void)

{

InitializeComponent();

this->TrigType->SelectedIndex = 0;

this->ActType->SelectedIndex = 0;

m\_step = 0;

m\_wday = boost::date\_time::Monday;

}

protected:

~CreateTaskForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ PreviousStepBtn;

private: System::Windows::Forms::Button^ NextStepBtn;

private: System::Windows::Forms::Label^ TaskNameLabel;

private: System::Windows::Forms::Label^ TaskDescLabel;

private: System::Windows::Forms::TextBox^ TaskName;

private: System::Windows::Forms::RichTextBox^ TaskDesc;

private: System::Windows::Forms::Label^ MainHeaderLabel;

private: System::Windows::Forms::GroupBox^ StepOneBox;

private: System::Windows::Forms::GroupBox^ StepTwoBox;

private: System::Windows::Forms::Label^ TrigTypeLabel;

private: System::Windows::Forms::ComboBox^ TrigType;

private: System::Windows::Forms::Label^ TrigTimeLabel;

private: System::Windows::Forms::MaskedTextBox^ TrigTime;

private: System::Windows::Forms::CheckedListBox^ MonthlyDays;

private: System::Windows::Forms::Label^ MonthlyDaysLabel;

private: System::Windows::Forms::GroupBox^ MonthlyBox;

private: System::Windows::Forms::CheckedListBox^ MonthlyMonths;

private: System::Windows::Forms::Label^ MonthlyMonthsLabel;

private: System::Windows::Forms::GroupBox^ WeeklyBox;

private: System::Windows::Forms::Label^ WeeklyEveryNWeekLabel;

private: System::Windows::Forms::NumericUpDown^ WeeklyEveryNWeek;

private: System::Windows::Forms::GroupBox^ StepThreeBox;

private: System::Windows::Forms::GroupBox^ ProgramBox;

private: System::Windows::Forms::TextBox^ ProgramParams;

private: System::Windows::Forms::Label^ ProgramNameLabel;

private: System::Windows::Forms::Label^ ProgramParamsLabel;

private: System::Windows::Forms::TextBox^ ProgramName;

private: System::Windows::Forms::GroupBox^ MessageBox;

private: System::Windows::Forms::Label^ MessageNameLabel;

private: System::Windows::Forms::RichTextBox^ MessageText;

private: System::Windows::Forms::Label^ MessageTextLabel;

private: System::Windows::Forms::TextBox^ MessageName;

private: System::Windows::Forms::ComboBox^ ActType;

private: System::Windows::Forms::Label^ ActTypeLabel;

private: System::ComponentModel::Container ^components;

private: System::Windows::Forms::Button^ SendBtn;

private: System::Windows::Forms::GroupBox^ WeeklyWDayBox;

private: System::Windows::Forms::RadioButton^ WDaySun;

private: System::Windows::Forms::RadioButton^ WDaySat;

private: System::Windows::Forms::RadioButton^ WDayFri;

private: System::Windows::Forms::RadioButton^ WDayThu;

private: System::Windows::Forms::RadioButton^ WDayWed;

private: System::Windows::Forms::RadioButton^ WDayTue;

private: System::Windows::Forms::RadioButton^ WDayMon;

private:

unsigned int m\_step;

private: System::Windows::Forms::Label^ TrigPriorLabel;

private: System::Windows::Forms::NumericUpDown^ TrigPrior;

boost::date\_time::weekdays m\_wday;

#pragma region Windows Form Designer generated code

void InitializeComponent(void)

{

this->PreviousStepBtn = (gcnew System::Windows::Forms::Button());

this->NextStepBtn = (gcnew System::Windows::Forms::Button());

this->TaskNameLabel = (gcnew System::Windows::Forms::Label());

this->TaskDescLabel = (gcnew System::Windows::Forms::Label());

this->TaskName = (gcnew System::Windows::Forms::TextBox());

this->TaskDesc = (gcnew System::Windows::Forms::RichTextBox());

this->MainHeaderLabel = (gcnew System::Windows::Forms::Label());

this->StepOneBox = (gcnew System::Windows::Forms::GroupBox());

this->StepTwoBox = (gcnew System::Windows::Forms::GroupBox());

this->TrigPriorLabel = (gcnew System::Windows::Forms::Label());

this->TrigPrior = (gcnew System::Windows::Forms::NumericUpDown());

this->WeeklyBox = (gcnew System::Windows::Forms::GroupBox());

this->WeeklyWDayBox = (gcnew System::Windows::Forms::GroupBox());

this->WDaySun = (gcnew System::Windows::Forms::RadioButton());

this->WDaySat = (gcnew System::Windows::Forms::RadioButton());

this->WDayFri = (gcnew System::Windows::Forms::RadioButton());

this->WDayThu = (gcnew System::Windows::Forms::RadioButton());

this->WDayWed = (gcnew System::Windows::Forms::RadioButton());

this->WDayTue = (gcnew System::Windows::Forms::RadioButton());

this->WDayMon = (gcnew System::Windows::Forms::RadioButton());

this->WeeklyEveryNWeekLabel = (gcnew System::Windows::Forms::Label());

this->WeeklyEveryNWeek = (gcnew System::Windows::Forms::NumericUpDown());

this->MonthlyBox = (gcnew System::Windows::Forms::GroupBox());

this->MonthlyMonths = (gcnew System::Windows::Forms::CheckedListBox());

this->MonthlyDays = (gcnew System::Windows::Forms::CheckedListBox());

this->MonthlyMonthsLabel = (gcnew System::Windows::Forms::Label());

this->MonthlyDaysLabel = (gcnew System::Windows::Forms::Label());

this->TrigTimeLabel = (gcnew System::Windows::Forms::Label());

this->TrigTime = (gcnew System::Windows::Forms::MaskedTextBox());

this->TrigTypeLabel = (gcnew System::Windows::Forms::Label());

this->TrigType = (gcnew System::Windows::Forms::ComboBox());

this->StepThreeBox = (gcnew System::Windows::Forms::GroupBox());

this->ProgramBox = (gcnew System::Windows::Forms::GroupBox());

this->ProgramParams = (gcnew System::Windows::Forms::TextBox());

this->ProgramNameLabel = (gcnew System::Windows::Forms::Label());

this->ProgramParamsLabel = (gcnew System::Windows::Forms::Label());

this->ProgramName = (gcnew System::Windows::Forms::TextBox());

this->MessageBox = (gcnew System::Windows::Forms::GroupBox());

this->MessageNameLabel = (gcnew System::Windows::Forms::Label());

this->MessageText = (gcnew System::Windows::Forms::RichTextBox());

this->MessageTextLabel = (gcnew System::Windows::Forms::Label());

this->MessageName = (gcnew System::Windows::Forms::TextBox());

this->ActType = (gcnew System::Windows::Forms::ComboBox());

this->ActTypeLabel = (gcnew System::Windows::Forms::Label());

this->SendBtn = (gcnew System::Windows::Forms::Button());

this->StepOneBox->SuspendLayout();

this->StepTwoBox->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->TrigPrior))->BeginInit();

this->WeeklyBox->SuspendLayout();

this->WeeklyWDayBox->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->WeeklyEveryNWeek))->BeginInit();

this->MonthlyBox->SuspendLayout();

this->StepThreeBox->SuspendLayout();

this->ProgramBox->SuspendLayout();

this->MessageBox->SuspendLayout();

this->SuspendLayout();

//

// PreviousStepBtn

//

this->PreviousStepBtn->Enabled = false;

this->PreviousStepBtn->Location = System::Drawing::Point(19, 527);

this->PreviousStepBtn->Name = L"PreviousStepBtn";

this->PreviousStepBtn->Size = System::Drawing::Size(75, 23);

this->PreviousStepBtn->TabIndex = 8;

this->PreviousStepBtn->Text = L"Previous";

this->PreviousStepBtn->UseVisualStyleBackColor = true;

this->PreviousStepBtn->Click += gcnew System::EventHandler(this, &CreateTaskForm::PreviousStepBtn\_Click);

//

// NextStepBtn

//

this->NextStepBtn->Location = System::Drawing::Point(121, 527);

this->NextStepBtn->Name = L"NextStepBtn";

this->NextStepBtn->Size = System::Drawing::Size(75, 23);

this->NextStepBtn->TabIndex = 5;

this->NextStepBtn->Text = L"Next";

this->NextStepBtn->UseVisualStyleBackColor = true;

this->NextStepBtn->Click += gcnew System::EventHandler(this, &CreateTaskForm::NextStepBtn\_Click);

//

// TaskNameLabel

//

this->TaskNameLabel->AutoSize = true;

this->TaskNameLabel->Location = System::Drawing::Point(15, 28);

this->TaskNameLabel->Name = L"TaskNameLabel";

this->TaskNameLabel->Size = System::Drawing::Size(95, 17);

this->TaskNameLabel->TabIndex = 0;

this->TaskNameLabel->Text = L"Name of task:";

//

// TaskDescLabel

//

this->TaskDescLabel->AutoSize = true;

this->TaskDescLabel->Location = System::Drawing::Point(15, 80);

this->TaskDescLabel->Name = L"TaskDescLabel";

this->TaskDescLabel->Size = System::Drawing::Size(165, 17);

this->TaskDescLabel->TabIndex = 1;

this->TaskDescLabel->Text = L"Short description of task:";

//

// TaskName

//

this->TaskName->Location = System::Drawing::Point(16, 48);

this->TaskName->Name = L"TaskName";

this->TaskName->Size = System::Drawing::Size(100, 22);

this->TaskName->TabIndex = 2;

//

// TaskDesc

//

this->TaskDesc->Location = System::Drawing::Point(16, 100);

this->TaskDesc->Name = L"TaskDesc";

this->TaskDesc->Size = System::Drawing::Size(267, 67);

this->TaskDesc->TabIndex = 3;

this->TaskDesc->Text = L"";

//

// MainHeaderLabel

//

this->MainHeaderLabel->AutoSize = true;

this->MainHeaderLabel->Font = (gcnew System::Drawing::Font(L"Open Sans", 12));

this->MainHeaderLabel->Location = System::Drawing::Point(16, 16);

this->MainHeaderLabel->Name = L"MainHeaderLabel";

this->MainHeaderLabel->Size = System::Drawing::Size(181, 27);

this->MainHeaderLabel->TabIndex = 4;

this->MainHeaderLabel->Text = L"Fill info about task";

//

// StepOneBox

//

this->StepOneBox->Controls->Add(this->TaskName);

this->StepOneBox->Controls->Add(this->TaskNameLabel);

this->StepOneBox->Controls->Add(this->TaskDescLabel);

this->StepOneBox->Controls->Add(this->TaskDesc);

this->StepOneBox->Location = System::Drawing::Point(20, 46);

this->StepOneBox->Name = L"StepOneBox";

this->StepOneBox->Size = System::Drawing::Size(296, 184);

this->StepOneBox->TabIndex = 6;

this->StepOneBox->TabStop = false;

this->StepOneBox->Text = L"Step 1/3";

//

// StepTwoBox

//

this->StepTwoBox->Controls->Add(this->TrigPriorLabel);

this->StepTwoBox->Controls->Add(this->TrigPrior);

this->StepTwoBox->Controls->Add(this->WeeklyBox);

this->StepTwoBox->Controls->Add(this->MonthlyBox);

this->StepTwoBox->Controls->Add(this->TrigTimeLabel);

this->StepTwoBox->Controls->Add(this->TrigTime);

this->StepTwoBox->Controls->Add(this->TrigTypeLabel);

this->StepTwoBox->Controls->Add(this->TrigType);

this->StepTwoBox->Location = System::Drawing::Point(19, 46);

this->StepTwoBox->Name = L"StepTwoBox";

this->StepTwoBox->Size = System::Drawing::Size(367, 475);

this->StepTwoBox->TabIndex = 7;

this->StepTwoBox->TabStop = false;

this->StepTwoBox->Text = L"Step 2/3";

this->StepTwoBox->Visible = false;

//

// TrigPriorLabel

//

this->TrigPriorLabel->AutoSize = true;

this->TrigPriorLabel->Location = System::Drawing::Point(13, 134);

this->TrigPriorLabel->Name = L"TrigPriorLabel";

this->TrigPriorLabel->Size = System::Drawing::Size(117, 17);

this->TrigPriorLabel->TabIndex = 13;

this->TrigPriorLabel->Text = L"Priority of trigger:";

//

// TrigPrior

//

this->TrigPrior->Location = System::Drawing::Point(13, 154);

this->TrigPrior->Name = L"TrigPrior";

this->TrigPrior->Size = System::Drawing::Size(108, 22);

this->TrigPrior->TabIndex = 12;

this->TrigPrior->Maximum = 100;

this->TrigPrior->Minimum = 1;

//

// WeeklyBox

//

this->WeeklyBox->Controls->Add(this->WeeklyWDayBox);

this->WeeklyBox->Controls->Add(this->WeeklyEveryNWeekLabel);

this->WeeklyBox->Controls->Add(this->WeeklyEveryNWeek);

this->WeeklyBox->Location = System::Drawing::Point(13, 190);

this->WeeklyBox->Name = L"WeeklyBox";

this->WeeklyBox->Size = System::Drawing::Size(233, 273);

this->WeeklyBox->TabIndex = 8;

this->WeeklyBox->TabStop = false;

this->WeeklyBox->Text = L"Select info for weekly trigger";

this->WeeklyBox->Visible = false;

//

// WeeklyWDayBox

//

this->WeeklyWDayBox->Controls->Add(this->WDaySun);

this->WeeklyWDayBox->Controls->Add(this->WDaySat);

this->WeeklyWDayBox->Controls->Add(this->WDayFri);

this->WeeklyWDayBox->Controls->Add(this->WDayThu);

this->WeeklyWDayBox->Controls->Add(this->WDayWed);

this->WeeklyWDayBox->Controls->Add(this->WDayTue);

this->WeeklyWDayBox->Controls->Add(this->WDayMon);

this->WeeklyWDayBox->Location = System::Drawing::Point(10, 72);

this->WeeklyWDayBox->Name = L"WeeklyWDayBox";

this->WeeklyWDayBox->Size = System::Drawing::Size(157, 191);

this->WeeklyWDayBox->TabIndex = 3;

this->WeeklyWDayBox->TabStop = false;

this->WeeklyWDayBox->Text = L"Choose day of week";

//

// WDaySun

//

this->WDaySun->AutoSize = true;

this->WDaySun->Location = System::Drawing::Point(8, 161);

this->WDaySun->Name = L"WDaySun";

this->WDaySun->Size = System::Drawing::Size(54, 21);

this->WDaySun->TabIndex = 6;

this->WDaySun->Text = L"Sun";

this->WDaySun->UseVisualStyleBackColor = true;

this->WDaySun->Click += gcnew System::EventHandler(this, &CreateTaskForm::WDaySun\_Click);

//

// WDaySat

//

this->WDaySat->AutoSize = true;

this->WDaySat->Location = System::Drawing::Point(8, 138);

this->WDaySat->Name = L"WDaySat";

this->WDaySat->Size = System::Drawing::Size(50, 21);

this->WDaySat->TabIndex = 5;

this->WDaySat->Text = L"Sat";

this->WDaySat->UseVisualStyleBackColor = true;

this->WDaySat->Click += gcnew System::EventHandler(this, &CreateTaskForm::WDaySat\_Click);

//

// WDayFri

//

this->WDayFri->AutoSize = true;

this->WDayFri->Location = System::Drawing::Point(8, 115);

this->WDayFri->Name = L"WDayFri";

this->WDayFri->Size = System::Drawing::Size(45, 21);

this->WDayFri->TabIndex = 4;

this->WDayFri->Text = L"Fri";

this->WDayFri->UseVisualStyleBackColor = true;

this->WDayFri->Click += gcnew System::EventHandler(this, &CreateTaskForm::WDayFri\_Click);

//

// WDayThu

//

this->WDayThu->AutoSize = true;

this->WDayThu->Location = System::Drawing::Point(8, 92);

this->WDayThu->Name = L"WDayThu";

this->WDayThu->Size = System::Drawing::Size(54, 21);

this->WDayThu->TabIndex = 3;

this->WDayThu->Text = L"Thu";

this->WDayThu->UseVisualStyleBackColor = true;

this->WDayThu->Click += gcnew System::EventHandler(this, &CreateTaskForm::WDayThu\_Click);

//

// WDayWed

//

this->WDayWed->AutoSize = true;

this->WDayWed->Location = System::Drawing::Point(8, 69);

this->WDayWed->Name = L"WDayWed";

this->WDayWed->Size = System::Drawing::Size(58, 21);

this->WDayWed->TabIndex = 2;

this->WDayWed->Text = L"Wed";

this->WDayWed->UseVisualStyleBackColor = true;

this->WDayWed->Click += gcnew System::EventHandler(this, &CreateTaskForm::WDayWed\_Click);

//

// WDayTue

//

this->WDayTue->AutoSize = true;

this->WDayTue->Location = System::Drawing::Point(8, 45);

this->WDayTue->Name = L"WDayTue";

this->WDayTue->Size = System::Drawing::Size(54, 21);

this->WDayTue->TabIndex = 1;

this->WDayTue->Text = L"Tue";

this->WDayTue->UseVisualStyleBackColor = true;

this->WDayTue->Click += gcnew System::EventHandler(this, &CreateTaskForm::WDayTue\_Click);

//

// WDayMon

//

this->WDayMon->AutoSize = true;

this->WDayMon->Checked = true;

this->WDayMon->Location = System::Drawing::Point(8, 21);

this->WDayMon->Name = L"WDayMon";

this->WDayMon->Size = System::Drawing::Size(56, 21);

this->WDayMon->TabIndex = 0;

this->WDayMon->TabStop = true;

this->WDayMon->Text = L"Mon";

this->WDayMon->UseVisualStyleBackColor = true;

this->WDayMon->Click += gcnew System::EventHandler(this, &CreateTaskForm::WDayMon\_Click);

//

// WeeklyEveryNWeekLabel

//

this->WeeklyEveryNWeekLabel->AutoSize = true;

this->WeeklyEveryNWeekLabel->Location = System::Drawing::Point(7, 23);

this->WeeklyEveryNWeekLabel->Name = L"WeeklyEveryNWeekLabel";

this->WeeklyEveryNWeekLabel->Size = System::Drawing::Size(96, 17);

this->WeeklyEveryNWeekLabel->TabIndex = 1;

this->WeeklyEveryNWeekLabel->Text = L"Every n week:";

//

// WeeklyEveryNWeek

//

this->WeeklyEveryNWeek->Location = System::Drawing::Point(10, 44);

this->WeeklyEveryNWeek->Name = L"WeeklyEveryNWeek";

this->WeeklyEveryNWeek->Size = System::Drawing::Size(108, 22);

this->WeeklyEveryNWeek->TabIndex = 0;

this->WeeklyEveryNWeek->Minimum = 1;

this->WeeklyEveryNWeek->Maximum = 100;

//

// MonthlyBox

//

this->MonthlyBox->Controls->Add(this->MonthlyMonths);

this->MonthlyBox->Controls->Add(this->MonthlyDays);

this->MonthlyBox->Controls->Add(this->MonthlyMonthsLabel);

this->MonthlyBox->Controls->Add(this->MonthlyDaysLabel);

this->MonthlyBox->Location = System::Drawing::Point(13, 190);

this->MonthlyBox->Name = L"MonthlyBox";

this->MonthlyBox->Size = System::Drawing::Size(309, 171);

this->MonthlyBox->TabIndex = 11;

this->MonthlyBox->TabStop = false;

this->MonthlyBox->Text = L"Select info for monthly trigger";

this->MonthlyBox->Visible = false;

//

// MonthlyMonths

//

this->MonthlyMonths->FormattingEnabled = true;

this->MonthlyMonths->Items->AddRange(gcnew cli::array< System::Object^ >(12) {

L"1", L"2", L"3", L"4", L"5", L"6", L"7", L"8",

L"9", L"10", L"11", L"12"

});

this->MonthlyMonths->Location = System::Drawing::Point(17, 46);

this->MonthlyMonths->Name = L"MonthlyMonths";

this->MonthlyMonths->Size = System::Drawing::Size(123, 106);

this->MonthlyMonths->TabIndex = 10;

//

// MonthlyDays

//

this->MonthlyDays->FormattingEnabled = true;

this->MonthlyDays->Items->AddRange(gcnew cli::array< System::Object^ >(31) {

L"1", L"2", L"3", L"4", L"5", L"6", L"7", L"8",

L"9", L"10", L"11", L"12", L"13", L"14", L"15", L"16", L"17", L"18", L"19", L"20", L"21", L"22", L"23", L"24", L"25", L"26",

L"27", L"28", L"29", L"30", L"31"

});

this->MonthlyDays->Location = System::Drawing::Point(167, 46);

this->MonthlyDays->Name = L"MonthlyDays";

this->MonthlyDays->Size = System::Drawing::Size(124, 106);

this->MonthlyDays->TabIndex = 7;

//

// MonthlyMonthsLabel

//

this->MonthlyMonthsLabel->AutoSize = true;

this->MonthlyMonthsLabel->Location = System::Drawing::Point(14, 26);

this->MonthlyMonthsLabel->Name = L"MonthlyMonthsLabel";

this->MonthlyMonthsLabel->Size = System::Drawing::Size(101, 17);

this->MonthlyMonthsLabel->TabIndex = 9;

this->MonthlyMonthsLabel->Text = L"Select months:";

//

// MonthlyDaysLabel

//

this->MonthlyDaysLabel->AutoSize = true;

this->MonthlyDaysLabel->Location = System::Drawing::Point(165, 26);

this->MonthlyDaysLabel->Name = L"MonthlyDaysLabel";

this->MonthlyDaysLabel->Size = System::Drawing::Size(85, 17);

this->MonthlyDaysLabel->TabIndex = 8;

this->MonthlyDaysLabel->Text = L"Select days:";

//

// TrigTimeLabel

//

this->TrigTimeLabel->AutoSize = true;

this->TrigTimeLabel->Location = System::Drawing::Point(10, 79);

this->TrigTimeLabel->Name = L"TrigTimeLabel";

this->TrigTimeLabel->Size = System::Drawing::Size(153, 17);

this->TrigTimeLabel->TabIndex = 4;

this->TrigTimeLabel->Text = L"Time from which begin:";

//

// TrigTime

//

this->TrigTime->Location = System::Drawing::Point(13, 99);

this->TrigTime->Mask = L"00/00/0000 90:00";

this->TrigTime->Name = L"TrigTime";

this->TrigTime->Size = System::Drawing::Size(125, 22);

this->TrigTime->TabIndex = 3;

this->TrigTime->TextMaskFormat = System::Windows::Forms::MaskFormat::IncludePrompt;

this->TrigTime->ValidatingType = System::DateTime::typeid;

//

// TrigTypeLabel

//

this->TrigTypeLabel->AutoSize = true;

this->TrigTypeLabel->Location = System::Drawing::Point(10, 25);

this->TrigTypeLabel->Name = L"TrigTypeLabel";

this->TrigTypeLabel->Size = System::Drawing::Size(105, 17);

this->TrigTypeLabel->TabIndex = 2;

this->TrigTypeLabel->Text = L"Type of trigger:";

//

// TrigType

//

this->TrigType->FormattingEnabled = true;

this->TrigType->Items->AddRange(gcnew cli::array< System::Object^ >(5) { L"Dayly", L"Weekly", L"Monthly", L"Once", L"Entrance" });

this->TrigType->Location = System::Drawing::Point(13, 45);

this->TrigType->Name = L"TrigType";

this->TrigType->Size = System::Drawing::Size(121, 24);

this->TrigType->TabIndex = 1;

this->TrigType->SelectedIndexChanged += gcnew System::EventHandler(this, &CreateTaskForm::TrigType\_SelectedIndexChanged);

//

// StepThreeBox

//

this->StepThreeBox->Controls->Add(this->ProgramBox);

this->StepThreeBox->Controls->Add(this->MessageBox);

this->StepThreeBox->Controls->Add(this->ActType);

this->StepThreeBox->Controls->Add(this->ActTypeLabel);

this->StepThreeBox->Location = System::Drawing::Point(19, 46);

this->StepThreeBox->Name = L"StepThreeBox";

this->StepThreeBox->Size = System::Drawing::Size(320, 263);

this->StepThreeBox->TabIndex = 9;

this->StepThreeBox->TabStop = false;

this->StepThreeBox->Text = L"Step 3/3";

this->StepThreeBox->Visible = false;

//

// ProgramBox

//

this->ProgramBox->Controls->Add(this->ProgramParams);

this->ProgramBox->Controls->Add(this->ProgramNameLabel);

this->ProgramBox->Controls->Add(this->ProgramParamsLabel);

this->ProgramBox->Controls->Add(this->ProgramName);

this->ProgramBox->Location = System::Drawing::Point(17, 79);

this->ProgramBox->Name = L"ProgramBox";

this->ProgramBox->Size = System::Drawing::Size(263, 126);

this->ProgramBox->TabIndex = 6;

this->ProgramBox->TabStop = false;

this->ProgramBox->Text = L"Input info for program action";

this->ProgramBox->Visible = false;

//

// ProgramParams

//

this->ProgramParams->Location = System::Drawing::Point(6, 93);

this->ProgramParams->Name = L"ProgramParams";

this->ProgramParams->Size = System::Drawing::Size(136, 22);

this->ProgramParams->TabIndex = 5;

//

// ProgramNameLabel

//

this->ProgramNameLabel->AutoSize = true;

this->ProgramNameLabel->Location = System::Drawing::Point(6, 24);

this->ProgramNameLabel->Name = L"ProgramNameLabel";

this->ProgramNameLabel->Size = System::Drawing::Size(122, 17);

this->ProgramNameLabel->TabIndex = 4;

this->ProgramNameLabel->Text = L"Name of program:";

//

// ProgramParamsLabel

//

this->ProgramParamsLabel->AutoSize = true;

this->ProgramParamsLabel->Location = System::Drawing::Point(5, 73);

this->ProgramParamsLabel->Name = L"ProgramParamsLabel";

this->ProgramParamsLabel->Size = System::Drawing::Size(203, 17);

this->ProgramParamsLabel->TabIndex = 1;

this->ProgramParamsLabel->Text = L"Start program with parameters:";

//

// ProgramName

//

this->ProgramName->Location = System::Drawing::Point(6, 44);

this->ProgramName->Name = L"ProgramName";

this->ProgramName->Size = System::Drawing::Size(136, 22);

this->ProgramName->TabIndex = 2;

//

// MessageBox

//

this->MessageBox->Controls->Add(this->MessageNameLabel);

this->MessageBox->Controls->Add(this->MessageText);

this->MessageBox->Controls->Add(this->MessageTextLabel);

this->MessageBox->Controls->Add(this->MessageName);

this->MessageBox->Location = System::Drawing::Point(17, 78);

this->MessageBox->Name = L"MessageBox";

this->MessageBox->Size = System::Drawing::Size(289, 172);

this->MessageBox->TabIndex = 5;

this->MessageBox->TabStop = false;

this->MessageBox->Text = L"Input info for message action";

//

// MessageNameLabel

//

this->MessageNameLabel->AutoSize = true;

this->MessageNameLabel->Location = System::Drawing::Point(6, 24);

this->MessageNameLabel->Name = L"MessageNameLabel";

this->MessageNameLabel->Size = System::Drawing::Size(136, 17);

this->MessageNameLabel->TabIndex = 4;

this->MessageNameLabel->Text = L"Header of message:";

//

// MessageText

//

this->MessageText->Location = System::Drawing::Point(6, 93);

this->MessageText->Name = L"MessageText";

this->MessageText->Size = System::Drawing::Size(267, 67);

this->MessageText->TabIndex = 3;

this->MessageText->Text = L"";

//

// MessageTextLabel

//

this->MessageTextLabel->AutoSize = true;

this->MessageTextLabel->Location = System::Drawing::Point(5, 73);

this->MessageTextLabel->Name = L"MessageTextLabel";

this->MessageTextLabel->Size = System::Drawing::Size(116, 17);

this->MessageTextLabel->TabIndex = 1;

this->MessageTextLabel->Text = L"Text of message:";

//

// MessageName

//

this->MessageName->Location = System::Drawing::Point(6, 44);

this->MessageName->Name = L"MessageName";

this->MessageName->Size = System::Drawing::Size(136, 22);

this->MessageName->TabIndex = 2;

//

// ActType

//

this->ActType->FormattingEnabled = true;

this->ActType->Items->AddRange(gcnew cli::array< System::Object^ >(2) { L"Start program", L"Show message" });

this->ActType->Location = System::Drawing::Point(17, 48);

this->ActType->Name = L"ActType";

this->ActType->Size = System::Drawing::Size(128, 24);

this->ActType->TabIndex = 4;

this->ActType->SelectedIndexChanged += gcnew System::EventHandler(this, &CreateTaskForm::ActType\_SelectedIndexChanged);

//

// ActTypeLabel

//

this->ActTypeLabel->AutoSize = true;

this->ActTypeLabel->Location = System::Drawing::Point(15, 28);

this->ActTypeLabel->Name = L"ActTypeLabel";

this->ActTypeLabel->Size = System::Drawing::Size(102, 17);

this->ActTypeLabel->TabIndex = 0;

this->ActTypeLabel->Text = L"Type of action:";

//

// SendBtn

//

this->SendBtn->Location = System::Drawing::Point(310, 527);

this->SendBtn->Name = L"SendBtn";

this->SendBtn->Size = System::Drawing::Size(75, 23);

this->SendBtn->TabIndex = 10;

this->SendBtn->Text = L"OK";

this->SendBtn->UseVisualStyleBackColor = true;

this->SendBtn->Click += gcnew System::EventHandler(this, &CreateTaskForm::SendBtn\_Click);

//

// CreateTaskForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(441, 609);

this->Controls->Add(this->SendBtn);

this->Controls->Add(this->StepThreeBox);

this->Controls->Add(this->PreviousStepBtn);

this->Controls->Add(this->StepTwoBox);

this->Controls->Add(this->StepOneBox);

this->Controls->Add(this->NextStepBtn);

this->Controls->Add(this->MainHeaderLabel);

this->Name = L"CreateTaskForm";

this->Text = L"Create task";

this->StepOneBox->ResumeLayout(false);

this->StepOneBox->PerformLayout();

this->StepTwoBox->ResumeLayout(false);

this->StepTwoBox->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->TrigPrior))->EndInit();

this->WeeklyBox->ResumeLayout(false);

this->WeeklyBox->PerformLayout();

this->WeeklyWDayBox->ResumeLayout(false);

this->WeeklyWDayBox->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->WeeklyEveryNWeek))->EndInit();

this->MonthlyBox->ResumeLayout(false);

this->MonthlyBox->PerformLayout();

this->StepThreeBox->ResumeLayout(false);

this->StepThreeBox->PerformLayout();

this->ProgramBox->ResumeLayout(false);

this->ProgramBox->PerformLayout();

this->MessageBox->ResumeLayout(false);

this->MessageBox->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

void ShowStep();

Task\_header\_t InputHeader();

Task\_trigger\* InputTrigger();

Task\_act\* InputAct();

private: System::Void NextStepBtn\_Click(System::Object^ sender, System::EventArgs^ e) {

if (++m\_step == 2)

this->NextStepBtn->Enabled = false;

this->PreviousStepBtn->Enabled = true;

ShowStep();

}

private: System::Void PreviousStepBtn\_Click(System::Object^ sender, System::EventArgs^ e) {

if (--m\_step == 0)

this->PreviousStepBtn->Enabled = false;

this->NextStepBtn->Enabled = true;

ShowStep();

}

private: System::Void ActType\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

this->MessageBox->Visible = false;

this->ProgramBox->Visible = false;

if (this->ActType->SelectedIndex == 0)

this->ProgramBox->Visible = true;

else

this->MessageBox->Visible = true;

}

private: System::Void TrigType\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

this->MonthlyBox->Visible = false;

this->WeeklyBox->Visible = false;

if (this->TrigType->SelectedIndex == 1)

this->WeeklyBox->Visible = true;

else if (this->TrigType->SelectedIndex == 2)

this->MonthlyBox->Visible = true;

}

private: System::Void SendBtn\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void WDayMon\_Click(System::Object^ sender, System::EventArgs^ e) {

m\_wday = boost::date\_time::Monday;

}

private: System::Void WDayTue\_Click(System::Object^ sender, System::EventArgs^ e) {

m\_wday = boost::date\_time::Tuesday;

}

private: System::Void WDayWed\_Click(System::Object^ sender, System::EventArgs^ e) {

m\_wday = boost::date\_time::Wednesday;

}

private: System::Void WDayThu\_Click(System::Object^ sender, System::EventArgs^ e) {

m\_wday = boost::date\_time::Thursday;

}

private: System::Void WDayFri\_Click(System::Object^ sender, System::EventArgs^ e) {

m\_wday = boost::date\_time::Friday;

}

private: System::Void WDaySat\_Click(System::Object^ sender, System::EventArgs^ e) {

m\_wday = boost::date\_time::Saturday;

}

private: System::Void WDaySun\_Click(System::Object^ sender, System::EventArgs^ e) {

m\_wday = boost::date\_time::Sunday;

}

};

}

1. CreateTaskForm.cpp

#include "stdafx.h"

#include "CreateTaskForm.h"

#include "MainForm.h"

namespace TaskManagerForm

{

void CreateTaskForm::ShowStep()

{

this->StepOneBox->Visible = false;

this->StepTwoBox->Visible = false;

this->StepThreeBox->Visible = false;

switch (m\_step)

{

case 0:

this->StepOneBox->Visible = true;

break;

case 1:

this->StepTwoBox->Visible = true;

break;

case 2:

this->StepThreeBox->Visible = true;

break;

default:

this->StepOneBox->Visible = true;

m\_step = 0;

break;

}

}

System::Void CreateTaskForm::SendBtn\_Click(System::Object^ sender, System::EventArgs^ e)

{

Task\_header\_t header;

Task\_trigger \*trigger = nullptr;

Task\_act \*act = nullptr;

try

{

header = InputHeader();

trigger = InputTrigger();

act = InputAct();

TaskManagerForm::MainForm::Get()->Get\_TaskManager()->create\_task(header, trigger, act);

}

catch (Task\_Exception &e)

{

MessageBox::Show(gcnew String(e.Get\_as\_string\_with\_solution().c\_str()), L"Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

catch (Form\_Exception &e)

{

MessageBox::Show(gcnew String(e.Get\_as\_string\_with\_solution().c\_str()), L"Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

MainForm::Get()->refresh();

MessageBox::Show(L"Task was successfully created!", L"Info", MessageBoxButtons::OK, MessageBoxIcon::Information);

this->~CreateTaskForm();

}

Task\_header\_t CreateTaskForm::InputHeader()

{

Task\_header\_t tmp;

tmp.name = msclr::interop::marshal\_as<std::string>(this->TaskName->Text);

if (tmp.name == "")

throw TaskManagerForm::TaskHeaderNotFound\_ex();

tmp.desc = msclr::interop::marshal\_as<std::string>(this->TaskDesc->Text);

if (tmp.desc == "")

throw TaskManagerForm::TaskDescNotFound\_ex();

return tmp;

}

Task\_trigger\* CreateTaskForm::InputTrigger()

{

Task\_trigger \*tmp = nullptr;

std::string tmp\_str;

unsigned int priority = atoi(msclr::interop::marshal\_as<std::string>(this->TrigPrior->Text).c\_str());

Trigger\_type\_t type = static\_cast<Trigger\_type\_t>(this->TrigType->SelectedIndex);

Time time;

try

{

tmp\_str = msclr::interop::marshal\_as<std::string>(this->TrigTime->Text);

time = Time(atoi(tmp\_str.substr(4, 4).c\_str()),

atoi(tmp\_str.substr(2, 2).c\_str()),

atoi(tmp\_str.substr(0, 2).c\_str()),

atoi(tmp\_str.substr(8, 2).c\_str()),

atoi(tmp\_str.substr(10, 2).c\_str()));

}

catch (WrongTimeFormat\_ex &e)

{

throw TrigTimeWrongFormat\_ex();

}

unsigned int n;

std::vector<boost::date\_time::months\_of\_year> months;

std::vector<unsigned int> days;

if (type == WEEKLY)

{

n = atoi(msclr::interop::marshal\_as<std::string>(this->WeeklyEveryNWeek->Text).c\_str());

if (n <= 0)

throw WrongWeeklyTriggerEveryNWeek\_ex();

}

else if (type == MONTHLY)

{

IEnumerator^ DaysEnum = this->MonthlyDays->CheckedIndices->GetEnumerator();

while (DaysEnum->MoveNext())

{

days.push\_back(atoi(msclr::interop::marshal\_as<std::string>(DaysEnum->Current->ToString()).c\_str()) + 1);

}

if (days.size() == 0)

throw WrongMonthlyTriggerDaysVecSize\_ex();

IEnumerator^ MonthsEnum = this->MonthlyMonths->CheckedIndices->GetEnumerator();

while (MonthsEnum->MoveNext())

{

months.push\_back(static\_cast<boost::date\_time::months\_of\_year>(atoi(msclr::interop::marshal\_as<std::string>(MonthsEnum->Current->ToString()).c\_str()) + 1));

}

if (months.size() == 0)

throw WrongMonthlyTriggerMonthsVecSize\_ex();

}

switch (type)

{

case DAYLY:

tmp = new Task\_trigger\_dayly(time, priority);

break;

case WEEKLY:

tmp = new Task\_trigger\_weekly(time, priority, m\_wday, n);

break;

case MONTHLY:

tmp = new Task\_trigger\_monthly(time, priority, months, days);

break;

case ONCE:

tmp = new Task\_trigger\_once(time, priority);

break;

case ENTRANCE:

tmp = new Task\_trigger\_entrance(time, priority);

break;

default:

throw WrongTriggerType\_ex();

break;

}

return tmp;

}

Task\_act\* CreateTaskForm::InputAct()

{

Task\_act \*tmp = nullptr;

Task\_act\_type\_t type = static\_cast<Task\_act\_type\_t>(this->ActType->SelectedIndex);

std::string name, text;

switch (type)

{

case PROG:

name = msclr::interop::marshal\_as<std::string>(this->ProgramName->Text);

if (name == "")

throw TaskManagerForm::ProgActNameNotFound\_ex();

text = msclr::interop::marshal\_as<std::string>(this->ProgramParams->Text);

tmp = new Task\_act\_prog(name, text);

break;

case ALERT:

name = msclr::interop::marshal\_as<std::string>(this->MessageName->Text);

if (name == "")

throw TaskManagerForm::MessageActHeaderNotFound\_ex();

text = msclr::interop::marshal\_as<std::string>(this->MessageText->Text);

if (text == "")

throw TaskManagerForm::MessageActTextNotFound\_ex();

tmp = new Task\_act\_alert(name, text);

break;

default:

throw WrongActType\_ex();

break;

}

return tmp;

}

}

1. Form\_Exception.h

#pragma once

#include "stdafx.h"

namespace TaskManagerForm

{

enum Form\_Exception\_error\_code\_t

{

TaskHeaderNotFound,

TaskDescNotFound,

TrigTimeNotFound,

TrigTimeWrongFormat,

WeeklyTrigEveryNWeekNotFound,

WeeklyTrigWDaysNotFound,

MonthlyTrigMonthsNotFound,

MonthlyTrigDaysNotFound,

MessageActHeaderNotFound,

MessageActTextNotFound,

ProgActNameNotFound,

WrongExceptionErrorCode

};

class Form\_Exception

{

public:

Form\_Exception(Form\_Exception\_error\_code\_t error\_code\_);

~Form\_Exception();

static void delete\_(Form\_Exception \*&ex\_);

operator std::string() const;

std::string Get\_as\_string\_with\_solution() const;

Form\_Exception\_error\_code\_t Get\_error\_code() const { return error\_code; }

protected:

Form\_Exception\_error\_code\_t error\_code;

};

class TaskHeaderNotFound\_ex : public Form\_Exception

{

public:

TaskHeaderNotFound\_ex() : Form\_Exception(TaskHeaderNotFound) {}

};

class TaskDescNotFound\_ex : public Form\_Exception

{

public:

TaskDescNotFound\_ex() : Form\_Exception(TaskDescNotFound) {}

};

class TrigTimeNotFound\_ex : public Form\_Exception

{

public:

TrigTimeNotFound\_ex() : Form\_Exception(TrigTimeNotFound) {}

};

class TrigTimeWrongFormat\_ex : public Form\_Exception

{

public:

TrigTimeWrongFormat\_ex() : Form\_Exception(TrigTimeWrongFormat) {}

};

class WeeklyTrigEveryNWeekNotFound\_ex : public Form\_Exception

{

public:

WeeklyTrigEveryNWeekNotFound\_ex() : Form\_Exception(WeeklyTrigEveryNWeekNotFound) {}

};

class WeeklyTrigWDaysNotFound\_ex : public Form\_Exception

{

public:

WeeklyTrigWDaysNotFound\_ex() : Form\_Exception(WeeklyTrigWDaysNotFound) {}

};

class MonthlyTrigMonthsNotFound\_ex : public Form\_Exception

{

public:

MonthlyTrigMonthsNotFound\_ex() : Form\_Exception(MonthlyTrigMonthsNotFound) {}

};

class MonthlyTrigDaysNotFound\_ex : public Form\_Exception

{

public:

MonthlyTrigDaysNotFound\_ex() : Form\_Exception(MonthlyTrigDaysNotFound) {}

};

class MessageActHeaderNotFound\_ex : public Form\_Exception

{

public:

MessageActHeaderNotFound\_ex() : Form\_Exception(MessageActHeaderNotFound) {}

};

class MessageActTextNotFound\_ex : public Form\_Exception

{

public:

MessageActTextNotFound\_ex() : Form\_Exception(MessageActTextNotFound) {}

};

class ProgActNameNotFound\_ex : public Form\_Exception

{

public:

ProgActNameNotFound\_ex() : Form\_Exception(ProgActNameNotFound) {}

};

class WrongExceptionErrorCode\_ex : public Form\_Exception

{

public:

WrongExceptionErrorCode\_ex() : Form\_Exception(WrongExceptionErrorCode) {}

};

}

1. Form\_Exception.cpp

#include "stdafx.h"

#include "Form\_Exception.h"

namespace TaskManagerForm

{

Form\_Exception::Form\_Exception(Form\_Exception\_error\_code\_t error\_code\_)

: error\_code(error\_code\_)

{

}

Form\_Exception::~Form\_Exception()

{

}

void Form\_Exception::delete\_(Form\_Exception \*&ex\_)

{

exit(EXIT\_FAILURE); // Function not supported

//WeeklyTrigEveryNWeekWrong

switch (ex\_->Get\_error\_code())

{

case TaskManagerForm::TaskHeaderNotFound:

delete (TaskManagerForm::TaskHeaderNotFound\_ex\*)ex\_;

break;

case TaskManagerForm::TaskDescNotFound:

delete (TaskManagerForm::TaskDescNotFound\_ex\*)ex\_;

break;

case TaskManagerForm::TrigTimeNotFound:

delete (TaskManagerForm::TrigTimeNotFound\_ex\*)ex\_;

break;

case TaskManagerForm::TrigTimeWrongFormat:

delete (TaskManagerForm::TrigTimeWrongFormat\_ex\*)ex\_;

break;

case TaskManagerForm::WeeklyTrigEveryNWeekNotFound:

delete (TaskManagerForm::WeeklyTrigEveryNWeekNotFound\_ex\*)ex\_;

break;

case TaskManagerForm::WeeklyTrigWDaysNotFound:

delete (TaskManagerForm::WeeklyTrigWDaysNotFound\_ex\*)ex\_;

break;

case TaskManagerForm::MonthlyTrigMonthsNotFound:

delete (TaskManagerForm::MonthlyTrigMonthsNotFound\_ex\*)ex\_;

break;

case TaskManagerForm::MonthlyTrigDaysNotFound:

delete (TaskManagerForm::MonthlyTrigDaysNotFound\_ex\*)ex\_;

break;

case TaskManagerForm::MessageActHeaderNotFound:

delete (TaskManagerForm::MessageActHeaderNotFound\_ex\*)ex\_;

break;

case TaskManagerForm::MessageActTextNotFound:

delete (TaskManagerForm::MessageActTextNotFound\_ex\*)ex\_;

break;

case TaskManagerForm::ProgActNameNotFound:

delete (TaskManagerForm::ProgActNameNotFound\_ex\*)ex\_;

break;

case TaskManagerForm::WrongExceptionErrorCode:

delete (TaskManagerForm::WrongExceptionErrorCode\_ex\*)ex\_;

break;

default:

throw TaskManagerForm::WrongExceptionErrorCode\_ex();

break;

}

}

Form\_Exception::operator std::string() const

{

switch (error\_code)

{

case TaskManagerForm::TaskHeaderNotFound:

return std::string("Task name not found");

break;

case TaskManagerForm::TaskDescNotFound:

return std::string("Task description not found");

break;

case TaskManagerForm::TrigTimeNotFound:

return std::string("Task trigger time not found");

break;

case TaskManagerForm::TrigTimeWrongFormat:

return std::string("Wrong trigger time format");

break;

case TaskManagerForm::WeeklyTrigEveryNWeekNotFound:

return std::string("\"Every N week\" not found");

break;

case TaskManagerForm::WeeklyTrigWDaysNotFound:

return std::string("Trigger weekdays not found");

break;

case TaskManagerForm::MonthlyTrigMonthsNotFound:

return std::string("Trigger months not found");

break;

case TaskManagerForm::MonthlyTrigDaysNotFound:

return std::string("Trigger days not found");

break;

case TaskManagerForm::MessageActHeaderNotFound:

return std::string("Message header not found");

break;

case TaskManagerForm::MessageActTextNotFound:

return std::string("Message text not found");

break;

case TaskManagerForm::ProgActNameNotFound:

return std::string("Program name not found");

break;

case TaskManagerForm::WrongExceptionErrorCode:

return std::string("Wrong exception error code");

break;

default:

throw TaskManagerForm::WrongExceptionErrorCode\_ex();

break;

}

}

std::string Form\_Exception::Get\_as\_string\_with\_solution() const

{

switch (error\_code)

{

case TaskManagerForm::TaskHeaderNotFound:

return std::string("Error: Task name not found\nSolution: Input task name");

break;

case TaskManagerForm::TaskDescNotFound:

return std::string("Error: Task description not found\nSolution: Input task description");

break;

case TaskManagerForm::TrigTimeNotFound:

return std::string("Error: Trigger time not found\nSolution: Input trigger time");

break;

case TaskManagerForm::TrigTimeWrongFormat:

return std::string("Error: Wrong trigger time format\nSolution: Input time in format dd.mm.yyyy HH:MM");

break;

case TaskManagerForm::WeeklyTrigEveryNWeekNotFound:

return std::string("Error: \"Every N week\" not found\nSolution: Input \"Every N week\"");

break;

case TaskManagerForm::WeeklyTrigWDaysNotFound:

return std::string("Error: Trigger weekdays not found\nSolution: Choose at least one weekday");

break;

case TaskManagerForm::MonthlyTrigMonthsNotFound:

return std::string("Error: Trigger months not found\nSolution: Choose at least one month");

break;

case TaskManagerForm::MonthlyTrigDaysNotFound:

return std::string("Error: Trigger days not found\nSolution: Choose at least one day");

break;

case TaskManagerForm::MessageActHeaderNotFound:

return std::string("Error: Message header not found\nSolution: Input message header");

break;

case TaskManagerForm::MessageActTextNotFound:

return std::string("Error: Message text not found\nSolution: Input message text");

break;

case TaskManagerForm::ProgActNameNotFound:

return std::string("Error: Program name not found\nSolution: Input program name");

break;

case TaskManagerForm::WrongExceptionErrorCode:

return std::string("Error: Wrong exception error code\nSolution: Restart program");

break;

default:

throw TaskManagerForm::WrongExceptionErrorCode\_ex();

break;

}

}

}

1. AssemblyInfo.cpp

#include "stdafx.h"

using namespace System;

using namespace System::Reflection;

using namespace System::Runtime::CompilerServices;

using namespace System::Runtime::InteropServices;

using namespace System::Security::Permissions;

[assembly:AssemblyTitleAttribute(L"CppCLR\_WinformsProjekt")];

[assembly:AssemblyDescriptionAttribute(L"")];

[assembly:AssemblyConfigurationAttribute(L"")];

[assembly:AssemblyCompanyAttribute(L"")];

[assembly:AssemblyProductAttribute(L"CppCLR\_WinformsProjekt")];

[assembly:AssemblyCopyrightAttribute(L"Copyright (c) 2017")];

[assembly:AssemblyTrademarkAttribute(L"")];

[assembly:AssemblyCultureAttribute(L"")];

[assembly:AssemblyVersionAttribute("1.0.0.1")];

[assembly:ComVisible(false)];

[assembly:CLSCompliantAttribute(true)];

1. resource.h

//{{NO\_DEPENDENCIES}}

// Microsoft Visual C++ generated include file.

// Used by app.rc

1. stdafx.h

#pragma once

#pragma unmanaged

#include <windows.h>

#include <Task\_Manager.h>

#include <boost/interprocess/managed\_shared\_memory.hpp>

#include <algorithm>

#pragma managed

#include <msclr/marshal\_cppstd.h>

1. stdafx.cpp

#include "stdafx.h"

1. Task\_Manager.h

#pragma once

#include "stdafx.h"

#include "Task.h"

#include "Alert\_Manager.h"

struct Task\_Info\_t

{

Task\_header\_t header;

int time\_left;

};

class Task\_Manager

{

public:

Task\_Manager(const char\* file\_name\_, bool entrance);

~Task\_Manager() throw();

void create\_task(Task\_header\_t header, Task\_trigger \*&trigger, Task\_act \*&act);

void delete\_task(unsigned int id\_);

void import\_task(const char \*import\_file\_name\_);

void export\_task(const char \*export\_file\_name\_, unsigned int id\_);

std::vector<Task\_Info\_t> Get\_task\_info();

#ifdef DEBUG

void output();

#endif // DEBUG

private:

/\*\*

\* Create task. This function can be called only from create\_task function

\* @param header - header of task

\* @param trigger - trigger of task

\* @param act - action of task

\* @exceptions WrongTriggerType, WrongActType, WrongTimeFormat, WrongTime

\*/

void create\_task\_private(Task\_header\_t header, Task\_trigger \*&trigger, Task\_act \*&act);

/\*\*

\* Delete task by id. This function can be called only from delete\_task and delete\_task\_waiter functions

\* @param id\_ - id of task to delete

\* @param from\_waiter - if function was called from delete\_task\_waiter function

\* @exceptions TaskIdDoesNotExist, ConfigFileCorrupted

\*/

void delete\_task\_private(unsigned int id\_);

//void delete\_task\_waiter(unsigned int id\_);

/\*\*

\* Safely continue program after exception

\*/

//void safely\_continue();

/\*\*

\* Read from file @n bytes and check on errors

\* @param s - pointer to char buffer

\* @param n - number of bytes to read

\* @exceptions EndOfFileWasReached

\*/

void read\_(char \*s, std::streamsize n);

/\*\*

\* Open file with path m\_file\_name

\* @param mode\_ mode of opening

\* @exceptions ConfigFileAlreadyOpened, CanNotOpenConfigFile

\*/

void open\_(unsigned int mode\_);

/\*\*

\* Check if configuration file is corrupted

\* @return true if file corrupted

\*/

bool is\_corrupted();

/\*

read\_task:

read\_header

read\_trigger

read\_act

\*/

Task\* read\_task(unsigned int id);

Task\_header\_t read\_header(unsigned int id);

Task\_trigger\* read\_trigger();

Task\_act\* read\_act();

void write\_header(Task\_header\_t header);

void write\_trigger(Task\_trigger \*&trigger);

void write\_act(Task\_act \*&act);

/\*\*

\* Find task by id in m\_Tasks vector

\* @param id - id of task

\* @return iterrator, which pointing on the task

\*/

std::vector<Task\*>::iterator find\_task\_by\_id(unsigned int id);

void skeep\_task();

#ifdef DEBUG

public:

#endif // DEBUG

void waiter\_cycle();

void refresh();

#ifdef DEBUG

private:

#endif // DEBUG

void waiter();

std::string m\_file\_name;

/\*\*

\* Struct of .task file:

unsigned int n // Number of tasks

\*Task struct:

{

\*header struct:

{

unsigned int n; // Size of name

char name[n];

unsigned int n; // Size of desc

char desc[n];

}

\*trigger struct:

{

Trigger\_type

Time

priority

// If weekly:

boost::date\_time::weekdays week\_day

unsigned int every\_n\_week

// If monthly

unsigned int n; // Size of month\_list

boost::date\_time::months\_of\_year month\_list[n]

unsigned int n; // Size of days\_list

unsigned int days\_list[n]

}

\*act struct:

{

act\_type

unsigned int n; // Size of name

char name[n];

unsigned int n; // Size of text

char text[n];

}

}

\*/

std::fstream \*m\_file;

boost::mutex m\_file\_mutex;

std::vector<Task\*> m\_Tasks;

boost::mutex m\_Tasks\_mutex;

int m\_last\_id;

boost::mutex m\_last\_id\_mutex;

Alert\_Manager \*m\_alert\_manager;

// Waiter:

bool m\_stop\_waiting;

bool m\_exit;

boost::thread \*m\_waiter\_cycle\_thread;

};

1. Task\_Manager.cpp

#include "Task\_Manager.h"

Task\_Manager::Task\_Manager(const char\* file\_name\_, bool entrance)

{

#ifdef DEBUG

printf("\nReading tasks...");

#endif // DEBUG

m\_last\_id = -1;

m\_file\_name = file\_name\_;

m\_file = new std::fstream;

unsigned int n = 0; // Number of tasks

if (is\_corrupted())

{

// Trunc file

open\_(std::ios::out | std::ios::trunc | std::ios::binary);

m\_file->write((char\*)&n, sizeof(unsigned int));

m\_file->close();

}

open\_(std::ios::in | std::ios::binary);

m\_file->seekg(0, std::ios::beg);

read\_((char\*)&n, sizeof(unsigned int));

// Read tasks

Task \*tmp;

for (unsigned int i = 0; i < n; i++)

{

tmp = read\_task(i);

m\_Tasks.push\_back(tmp);

if (!entrance && m\_Tasks[i]->Get\_trigger\_type() == ENTRANCE)

{

m\_Tasks[i]->Set\_was\_maked(true);

}

}

m\_last\_id = n - 1;

m\_file->close();

m\_alert\_manager = new Alert\_Manager;

#ifdef DEBUG

printf("\n%d tasks was read.\n", n);

#endif // DEBUG

m\_exit = false;

m\_waiter\_cycle\_thread = new boost::thread(boost::bind(&Task\_Manager::waiter\_cycle, this));

}

Task\_Manager::~Task\_Manager()

{

m\_exit = true;

if (m\_waiter\_cycle\_thread == nullptr)

throw WaiterThreadCycleAlreadyDeleted\_ex();

m\_waiter\_cycle\_thread->join();

delete m\_waiter\_cycle\_thread;

m\_waiter\_cycle\_thread = nullptr;

m\_file\_mutex.lock();

if (m\_file == nullptr)

{

throw ConfigFileAlreadyDeleted\_ex();

}

delete m\_file;

m\_file = nullptr;

m\_Tasks\_mutex.lock();

for (unsigned int i = 0; i < m\_Tasks.size(); i++)

{

if (m\_Tasks[i] == nullptr)

{

throw TaskAlreadyDeleted\_ex();

}

delete m\_Tasks[i];

}

if (m\_alert\_manager != nullptr)

{

delete m\_alert\_manager;

m\_alert\_manager = nullptr;

}

}

void Task\_Manager::create\_task(Task\_header\_t header, Task\_trigger \*&trigger, Task\_act \*&act)

{

try

{

create\_task\_private(header, trigger, act);

}

catch (Task\_Exception &e)

{

throw e;

}

}

void Task\_Manager::delete\_task(unsigned int id\_)

{

Task\_Exception\_error\_code\_t e\_code;

try

{

delete\_task\_private(id\_);

}

catch (Task\_Exception &e)

{

e\_code = e.Get\_error\_code();

if(e\_code == EndOfFileWasReached)

throw ConfigFileCorrupted\_ex();

if(e\_code != TaskIdDoesNotFound)

throw e;

}

}

void Task\_Manager::import\_task(const char \*import\_file\_name\_)

{

#ifdef DEBUG

printf("\nImporting task...");

#endif // DEBUG

m\_file\_mutex.lock();

try

{

if (m\_file->is\_open())

throw ConfigFileAlreadyOpened\_ex();

m\_file->open(import\_file\_name\_, std::ios::in | std::ios::binary);

if (!m\_file->is\_open())

throw CanNotOpenFile\_ex();

skeep\_task();

// Save file in memory

std::streampos file\_size = m\_file->tellg();

m\_file->seekg(0, std::ios::beg);

char \*file = new char[file\_size];

read\_(file, file\_size);

m\_file->seekg(0, std::ios::beg);

m\_last\_id\_mutex.lock();

Task \*tmp = nullptr;

try

{

tmp = read\_task(++m\_last\_id);

}

catch (Task\_Exception &e)

{

m\_last\_id\_mutex.unlock();

throw e;

}

m\_last\_id\_mutex.unlock();

m\_file->close();

m\_Tasks\_mutex.lock();

m\_stop\_waiting = true;

m\_Tasks.push\_back(tmp);

m\_Tasks\_mutex.unlock();

// Write task in config file

open\_(std::ios::in | std::ios::out | std::ios::binary);

// Write number of tasks

m\_file->seekp(0, std::ios::beg);

m\_last\_id\_mutex.lock();

unsigned int n = m\_last\_id + 1;

m\_last\_id\_mutex.unlock();

m\_file->write((char\*)&n, sizeof(unsigned int));

m\_file->seekp(0, std::ios::end);

m\_file->write(file, file\_size);

m\_file->close();

}

catch (EndOfFileWasReached\_ex &e)

{

if (m\_file->is\_open())

m\_file->close();

m\_file\_mutex.unlock();

throw FileCorrupted\_ex();

}

catch (Task\_Exception &e)

{

if (m\_file->is\_open())

m\_file->close();

m\_file\_mutex.unlock();

throw e;

}

m\_file\_mutex.unlock();

#ifdef DEBUG

printf("\nTask was imported.\n");

#endif // DEBUG

}

void Task\_Manager::export\_task(const char \*export\_file\_name\_, unsigned int id\_)

{

#ifdef DEBUG

printf("\nExporting task...");

#endif // DEBUG

char \*file = nullptr;

m\_last\_id\_mutex.lock();

if ((int)id\_ > m\_last\_id)

{

m\_last\_id\_mutex.unlock();

throw TaskIdDoesNotExist\_ex();

}

m\_last\_id\_mutex.unlock();

m\_file\_mutex.lock();

try

{

open\_(std::ios::in | std::ios::binary);

m\_file->seekg(sizeof(unsigned int));

for (unsigned int i = 0; i < id\_; ++i)

skeep\_task();

std::streampos task\_begin = m\_file->tellg();

skeep\_task();

std::streampos task\_end = m\_file->tellg();

unsigned int file\_size = task\_end - task\_begin;

file = new char[file\_size];

m\_file->seekg(task\_begin);

read\_(file, file\_size);

m\_file->close();

m\_file->open(export\_file\_name\_, std::ios::out | std::ios::binary);

if (!m\_file->is\_open())

{

throw CanNotOpenFile\_ex();

}

m\_file->write(file, file\_size);

m\_file->close();

}

catch (EndOfFileWasReached\_ex &e)

{

if (m\_file->is\_open())

m\_file->close();

m\_file\_mutex.unlock();

if (is\_corrupted())

throw ConfigFileCorrupted\_ex();

}

catch (Task\_Exception &e)

{

if (m\_file->is\_open())

m\_file->close();

m\_file\_mutex.unlock();

throw e;

}

m\_file\_mutex.unlock();

#ifdef DEBUG

printf("\nTask was exported.\n");

#endif // DEBUG

}

std::vector<Task\_Info\_t> Task\_Manager::Get\_task\_info()

{

std::vector<Task\_Info\_t> tasks\_info;

Task\_Info\_t tmp;

m\_stop\_waiting = true;

while (!m\_stop\_waiting);

m\_Tasks\_mutex.lock();

unsigned int n = m\_Tasks.size();

for (unsigned int i = 0; i < n; ++i)

{

tmp.header = m\_Tasks[i]->Get\_header();

tmp.time\_left = m\_Tasks[i]->Get\_time\_left();

tasks\_info.push\_back(tmp);

}

m\_Tasks\_mutex.unlock();

return tasks\_info;

}

#ifdef DEBUG

void Task\_Manager::output()

{

for (unsigned int i = 0; i < m\_Tasks.size(); i++)

{

printf("\n==========\nTask #%d", i);

m\_Tasks[i]->output();

}

}

#endif //DEBUG

//

// private:

//

void Task\_Manager::create\_task\_private(Task\_header\_t header, Task\_trigger \*&trigger, Task\_act \*&act)

{

#ifdef DEBUG

printf("\nCreating task...");

#endif // DEBUG

if (!trigger->calculate\_time\_left(Time::current\_time()))

{

throw WrongTime\_ex();

}

m\_last\_id\_mutex.lock();

header.id = ++m\_last\_id;

m\_last\_id\_mutex.unlock();

m\_file\_mutex.lock();

try

{

open\_(std::ios::in | std::ios::out | std::ios::binary);

// Write number of tasks

m\_file->seekp(0, std::ios::beg);

unsigned int n = m\_last\_id + 1;

m\_file->write((char\*)&n, sizeof(unsigned int));

// Write in end of file

m\_file->seekp(0, std::ios::end);

write\_header(header);

write\_trigger(trigger);

write\_act(act);

m\_file->close();

}

catch (Task\_Exception &e)

{

if (m\_file->is\_open())

m\_file->close();

m\_file\_mutex.unlock();

throw e;

}

m\_file\_mutex.unlock();

// Add task in list

Task \*tmp = new Task(header, trigger, act);

if (tmp->Get\_trigger\_type() == ENTRANCE)

tmp->Set\_was\_maked(true);

m\_Tasks\_mutex.lock();

m\_stop\_waiting = true;

m\_Tasks.push\_back(tmp);

m\_Tasks\_mutex.unlock();

#ifdef DEBUG

printf("\nTask was create.\n");

#endif // DEBUG

}

void Task\_Manager::delete\_task\_private(unsigned int id\_)

{

#ifdef DEBUG

printf("\nDeleting task...");

#endif // DEBUG

m\_last\_id\_mutex.lock();

unsigned int n = m\_last\_id; // Number of tasks

m\_last\_id\_mutex.unlock();

if (id\_ > n)

throw TaskIdDoesNotExist\_ex();

m\_file\_mutex.lock();

try

{

open\_(std::ios::in | std::ios::out | std::ios::binary);

// Change number of tasks

m\_file->seekp(0, std::ios::beg);

m\_file->write((char\*)&n, sizeof(unsigned int));

// Skeep tasks before id\_ task

m\_file->seekg(sizeof(unsigned int), std::ios::beg);

for (unsigned int i = 0; i < id\_; i++)

skeep\_task();

std::streamoff begin\_of\_task = m\_file->tellg();

skeep\_task();

std::streamoff end\_of\_task = m\_file->tellg();

m\_file->seekg(0, std::ios::end);

std::streamoff file\_size = m\_file->tellg();

std::streamoff rest\_file\_size = file\_size - end\_of\_task;

m\_file->seekg(end\_of\_task);

char \*rest\_file = new char[rest\_file\_size];

read\_(rest\_file, rest\_file\_size);

m\_file->seekp(begin\_of\_task, std::ios::beg);

m\_file->write(rest\_file, rest\_file\_size);

delete[] rest\_file;

m\_file->seekg(0, std::ios::beg);

file\_size -= end\_of\_task - begin\_of\_task;

char \*all\_file = new char[file\_size];

read\_(all\_file, file\_size);

m\_file->close();

// Trunc file

open\_(std::ios::out | std::ios::binary);

m\_file->seekp(0, std::ios::beg);

m\_file->write(all\_file, file\_size);

delete[] all\_file;

m\_file->close();

}

catch (Task\_Exception &e)

{

if (m\_file->is\_open())

m\_file->close();

m\_file\_mutex.unlock();

throw e;

}

m\_file\_mutex.unlock();

m\_Tasks\_mutex.lock();

m\_stop\_waiting = true;

try

{

m\_stop\_waiting = true;

m\_last\_id\_mutex.lock();

m\_last\_id--;

m\_last\_id\_mutex.unlock();

std::vector<Task\*>::iterator elem = find\_task\_by\_id(id\_);

if (\*elem != nullptr)

{

delete \*elem;

\*elem = nullptr;

m\_Tasks.erase(elem);

}

else

throw TaskIdDoesNotFound\_ex();

for (unsigned int i = 0; i < m\_Tasks.size(); ++i)

{

if (m\_Tasks[i]->Get\_id() > id\_)

--\*m\_Tasks[i];

}

}

catch (Task\_Exception &e)

{

m\_Tasks\_mutex.unlock();

throw e;

}

m\_Tasks\_mutex.unlock();

#ifdef DEBUG

printf("\nTask was deleted.\n");

#endif // DEBUG

}

void Task\_Manager::read\_(char \*s, std::streamsize n)

{

m\_file->read(s, n);

if ((m\_file->rdstate() & std::ios::eofbit) != 0)

{

throw EndOfFileWasReached\_ex();

}

}

void Task\_Manager::open\_(unsigned int mode\_)

{

if (m\_file->is\_open())

{

throw ConfigFileAlreadyOpened\_ex();

}

m\_file->open(m\_file\_name, mode\_);

if (!m\_file->is\_open())

{

throw CanNotOpenConfigFile\_ex();

}

}

bool Task\_Manager::is\_corrupted()

{

m\_file\_mutex.lock();

try

{

open\_(std::ios::in | std::ios::binary);

unsigned int n;

read\_((char\*)&n, sizeof(unsigned int));

if (n == 0)

{

m\_file->seekg(0, std::ios::end);

if (((int)m\_file->tellg()) != sizeof(unsigned int))

throw EndOfFileWasReached\_ex();

}

for (unsigned int i = 0; i < n; ++i)

skeep\_task();

m\_file->close();

}

catch (Task\_Exception &e)

{

Task\_Exception\_error\_code\_t e\_code = e.Get\_error\_code();

m\_file\_mutex.unlock();

if (m\_file->is\_open())

m\_file->close();

if (e\_code == EndOfFileWasReached || e\_code == CanNotOpenConfigFile)

return true;

throw e;

}

m\_file\_mutex.unlock();

return false;

}

Task\* Task\_Manager::read\_task(unsigned int id)

{

Task\_header\_t header;

header = read\_header(id);

Task\_trigger \*trigger = nullptr;

trigger = read\_trigger();

Task\_act \*act = nullptr;

act = read\_act();

return new Task(header, trigger, act);

}

Task\_header\_t Task\_Manager::read\_header(unsigned int id)

{

// Fill task header

Task\_header\_t header;

header.id = id;

unsigned int n;

char \*tmp\_char;

read\_((char\*)&n, sizeof(unsigned int));

tmp\_char = new char[n + 1];

read\_(tmp\_char, sizeof(char)\*n);

tmp\_char[n] = '\0';

header.name = tmp\_char;

delete[] tmp\_char;

read\_((char\*)&n, sizeof(unsigned int));

tmp\_char = new char[n + 1];

read\_(tmp\_char, sizeof(char)\*n);

tmp\_char[n] = '\0';

header.desc = tmp\_char;

delete[] tmp\_char;

return header;

}

Task\_trigger\* Task\_Manager::read\_trigger()

{

// Fill trigger

Trigger\_type\_t trigger\_type;

Time time;

unsigned int priority;

// For weekly trigger:

boost::date\_time::weekdays wday;

unsigned int every\_n\_week;

// For monthly trigger:

std::vector<boost::date\_time::months\_of\_year> m\_vec;

std::vector<unsigned int> d\_vec;

read\_((char\*)&trigger\_type, sizeof(Trigger\_type\_t));

read\_((char\*)&time, sizeof(Time));

read\_((char\*)&priority, sizeof(unsigned int));

if (trigger\_type == WEEKLY)

{

read\_((char\*)&wday, sizeof(boost::date\_time::weekdays));

read\_((char\*)&every\_n\_week, sizeof(unsigned int));

}

else if (trigger\_type == MONTHLY)

{

unsigned int n;

read\_((char\*)&n, sizeof(unsigned int));

boost::date\_time::months\_of\_year \*months = new boost::date\_time::months\_of\_year[n];

read\_((char\*)months, n \* sizeof(boost::date\_time::months\_of\_year));

if (n == 1)

m\_vec.push\_back(months[0]);

else

m\_vec = std::vector<boost::date\_time::months\_of\_year>(months, months + n);

delete[] months;

read\_((char\*)&n, sizeof(unsigned int));

unsigned int \*days = new unsigned int[n];

read\_((char\*)days, n \* sizeof(unsigned int));

if (n == 1)

d\_vec.push\_back(days[0]);

else

d\_vec = std::vector<unsigned int>(days, days + n);

delete[] days;

}

switch (trigger\_type)

{

case DAYLY:

return new Task\_trigger\_dayly(time, priority);

break;

case WEEKLY:

return new Task\_trigger\_weekly(time, priority, wday, every\_n\_week);

break;

case MONTHLY:

return new Task\_trigger\_monthly(time, priority, m\_vec, d\_vec);

break;

case ONCE:

return new Task\_trigger\_once(time, priority);

break;

case ENTRANCE:

return new Task\_trigger\_entrance(time, priority);

break;

default:

throw WrongTriggerType\_ex();

}

return nullptr;

}

Task\_act\* Task\_Manager::read\_act()

{

Task\_act\_type\_t type;

read\_((char\*)&type, sizeof(Task\_act\_type\_t));

unsigned int n;

char \*buf;

read\_((char\*)&n, sizeof(unsigned int));

buf = new char[n + 1];

read\_(buf, sizeof(char)\*n);

buf[n] = '\0';

/\*

str1:

name of prog for Tasc\_act\_prog

header of alert for Task\_act\_alert

\*/

std::string str1 = buf;

delete[] buf;

read\_((char\*)&n, sizeof(unsigned int));

buf = new char[n + 1];

read\_(buf, sizeof(char)\*n);

buf[n] = '\0';

/\*

str2:

params of prog for Tasc\_act\_prog

text of alert for Task\_act\_alert

\*/

std::string str2 = buf;

delete[] buf;

switch (type)

{

case PROG:

return new Task\_act\_prog(str1, str2);

break;

case ALERT:

return new Task\_act\_alert(str1, str2);

break;

default:

throw WrongActType\_ex();

break;

}

return nullptr;

}

void Task\_Manager::write\_header(Task\_header\_t header)

{

unsigned int n;

const char\* tmp;

n = header.name.size();

m\_file->write((char\*)&n, sizeof(unsigned int));

tmp = header.name.c\_str();

m\_file->write((char\*)tmp, sizeof(char)\*n);

n = header.desc.size();

m\_file->write((char\*)&n, sizeof(unsigned int));

tmp = header.desc.c\_str();

m\_file->write((char\*)tmp, sizeof(char)\*n);

}

void Task\_Manager::write\_trigger(Task\_trigger \*&trigger)

{

Trigger\_type\_t type = trigger->Get\_type();

m\_file->write((char\*)&type, sizeof(Trigger\_type\_t));

Time time = trigger->Get\_time();

m\_file->write((char\*)&time, sizeof(Time));

unsigned int priority = trigger->Get\_priority();

m\_file->write((char\*)&priority, sizeof(unsigned int));

if (type == WEEKLY)

{

boost::date\_time::weekdays wd = ((Task\_trigger\_weekly\*)trigger)->Get\_week\_day();

m\_file->write((char\*)&wd, sizeof(boost::date\_time::weekdays));

unsigned int every\_n = ((Task\_trigger\_weekly\*)trigger)->Get\_every\_n\_week();

m\_file->write((char\*)&every\_n, sizeof(unsigned int));

}

else if (type == MONTHLY)

{

std::vector<boost::date\_time::months\_of\_year> m\_list = ((Task\_trigger\_monthly\*)trigger)->Get\_month\_list();

unsigned int n = m\_list.size();

m\_file->write((char\*)&n, sizeof(unsigned int));

m\_file->write((char\*)&m\_list[0], sizeof(boost::date\_time::months\_of\_year)\*n);

std::vector<unsigned int> d\_list = ((Task\_trigger\_monthly\*)trigger)->Get\_days\_list();

n = d\_list.size();

m\_file->write((char\*)&n, sizeof(unsigned int));

m\_file->write((char\*)&d\_list[0], sizeof(unsigned int)\*n);

}

}

void Task\_Manager::write\_act(Task\_act \*&act)

{

Task\_act\_type\_t type = act->Get\_type();

m\_file->write((char\*)&type, sizeof(Task\_act\_type\_t));

std::string name, text;

if (type == ALERT)

{

name = ((Task\_act\_alert\*)act)->Get\_alert\_name();

text = ((Task\_act\_alert\*)act)->Get\_alert\_text();

}

else if (type == PROG)

{

name = ((Task\_act\_prog\*)act)->Get\_prog\_name();

text = ((Task\_act\_prog\*)act)->Get\_prog\_params();

}

unsigned int n;

const char \*tmp;

n = name.size();

m\_file->write((char\*)&n, sizeof(unsigned int));

tmp = name.c\_str();

m\_file->write(tmp, sizeof(char)\*n);

n = text.size();

m\_file->write((char\*)&n, sizeof(unsigned int));

tmp = text.c\_str();

m\_file->write(tmp, sizeof(char)\*n);

}

std::vector<Task\*>::iterator Task\_Manager::find\_task\_by\_id(unsigned int id)

{

std::vector<Task\*>::iterator curr, end;

curr = m\_Tasks.begin();

end = m\_Tasks.end();

while (curr != end)

{

if ((\*curr)->Get\_id() == id)

return curr;

++curr;

}

return end;

}

void Task\_Manager::skeep\_task()

{

Trigger\_type\_t trig\_type;

Task\_act\_type\_t act\_type;

unsigned int n;

// Skip header

read\_((char\*)&n, sizeof(unsigned int));

m\_file->seekg(sizeof(char)\*n, std::ios::cur);

read\_((char\*)&n, sizeof(unsigned int));

m\_file->seekg(sizeof(char)\*n, std::ios::cur);

// Skip trigger

read\_((char\*)&trig\_type, sizeof(Trigger\_type\_t));

m\_file->seekg(sizeof(Time) + sizeof(unsigned int), std::ios::cur);

switch (trig\_type)

{

case WEEKLY:

m\_file->seekg(sizeof(boost::date\_time::weekdays) + sizeof(unsigned int), std::ios::cur);

break;

case MONTHLY:

read\_((char\*)&n, sizeof(unsigned int));

m\_file->seekg(sizeof(boost::date\_time::months\_of\_year)\*n, std::ios::cur);

read\_((char\*)&n, sizeof(unsigned int));

m\_file->seekg(sizeof(unsigned int)\*n, std::ios::cur);

break;

}

// Skip act

read\_((char\*)&act\_type, sizeof(Task\_act\_type\_t));

read\_((char\*)&n, sizeof(unsigned int));

m\_file->seekg(sizeof(char)\*n, std::ios::cur);

read\_((char\*)&n, sizeof(unsigned int));

m\_file->seekg(sizeof(char)\*n, std::ios::cur);

}

void Task\_Manager::waiter\_cycle()

{

#ifdef DEBUG

printf("\nWaiter cycle was started.\n");

#endif // DEBUG

while (!m\_exit)

{

refresh();

m\_stop\_waiting = false;

waiter();

}

#ifdef DEBUG

printf("\nExit from waiter cycle.\n");

#endif // DEBUG

}

void Task\_Manager::refresh()

{

#ifdef DEBUG

printf("\nRefresh task list...");

#endif // DEBUG

Time c\_time = Time::current\_time();

m\_last\_refresh = c\_time;

m\_Tasks\_mutex.lock();

int n = m\_Tasks.size();

for (int i = 0; i < n; ++i)

{

if (!m\_Tasks[i]->calculate\_time\_left(c\_time))

{

unsigned int c\_task\_id = m\_Tasks[i]->Get\_id();

m\_Tasks\_mutex.unlock();

delete\_task(c\_task\_id);

m\_Tasks\_mutex.lock();

--i;

--n;

}

}

m\_Tasks\_mutex.unlock();

if (n <= 1)

{

#ifdef DEBUG

printf("\nTasks list has been refreshed.\n");

#endif // DEBUG

return;

}

m\_Tasks\_mutex.lock();

std::sort(m\_Tasks.begin(), m\_Tasks.end(), Task::compare);

m\_Tasks\_mutex.unlock();

#ifdef DEBUG

printf("\nTasks list has been refreshed.\n");

#endif // DEBUG

}

void Task\_Manager::waiter()

{

#ifdef DEBUG

printf("\nWaiter was started:\n");

#endif // DEBUG

unsigned int to\_wait;

m\_Tasks\_mutex.lock();

if (m\_Tasks.size() != 0)

to\_wait = m\_Tasks[0]->Get\_time\_left();

else

to\_wait = INF;

m\_Tasks\_mutex.unlock();

unsigned int seconds;

if (to\_wait == INF)

{

#ifdef DEBUG

printf("To wait: INF");

#endif // DEBUG

seconds = Time::to\_hour\_left();

if (seconds == 0 && m\_last\_refresh == Time::current\_time())

seconds = 3600;

for (unsigned int i = 0; i < seconds; ++i)

{

if (m\_stop\_waiting || m\_exit)

return;

Sleep(1000);

}

return;

}

to\_wait \*= 60;

if (to\_wait)

{

seconds = Time::to\_minute\_left();

if (seconds != 0)

{

to\_wait -= 60;

to\_wait += seconds;

}

}

#ifdef DEBUG

printf("To wait: %d seconds.\n", to\_wait);

#endif // DEBUG

for (unsigned int i = 0; i < to\_wait; i++)

{

if (m\_stop\_waiting || m\_exit)

return;

Sleep(1000);

}

m\_Tasks\_mutex.lock();

if (m\_stop\_waiting)

{

m\_Tasks\_mutex.unlock();

return;

}

m\_Tasks[0]->Set\_last\_time(Time::current\_time());

m\_Tasks[0]->make\_act();

m\_Tasks\_mutex.unlock();

}

1. Alert\_Manager.h

#pragma once

#include "stdafx.h"

struct alert\_t

{

alert\_t() {}

alert\_t(std::string header\_, std::string message\_)

: header(header\_), message(message\_)

{}

std::string header;

std::string message;

};

class Alert\_Manager

{

public:

Alert\_Manager();

~Alert\_Manager();

void add\_alert(std::string header, std::string message);

static Alert\_Manager\* Get();

protected:

void show\_cycle();

static Alert\_Manager \*m\_this;

boost::thread \*m\_show\_cycle\_thread;

bool m\_show\_cycle\_thread\_exit;

boost::mutex m\_alerts\_mutex;

std::vector<alert\_t> m\_alerts;

};

1. Alert\_Manager.cpp

#include "Alert\_Manager.h"

#include "Task\_Exception.h"

Alert\_Manager \*Alert\_Manager::m\_this = nullptr;

Alert\_Manager::Alert\_Manager()

:m\_show\_cycle\_thread(nullptr), m\_show\_cycle\_thread\_exit(false)

{

if (m\_this != nullptr)

throw AlertManagerAlreadyCreated\_ex();

m\_this = this;

m\_show\_cycle\_thread = new boost::thread(boost::bind(&Alert\_Manager::show\_cycle, this));

}

Alert\_Manager::~Alert\_Manager()

{

if (m\_show\_cycle\_thread != nullptr)

{

m\_show\_cycle\_thread\_exit = true;

m\_show\_cycle\_thread->join();

delete m\_show\_cycle\_thread;

m\_show\_cycle\_thread = nullptr;

}

m\_this = nullptr;

}

void Alert\_Manager::add\_alert(std::string header, std::string message)

{

m\_alerts\_mutex.lock();

m\_alerts.push\_back(alert\_t(header, message));

m\_alerts\_mutex.unlock();

}

Alert\_Manager\* Alert\_Manager::Get()

{

if (m\_this == nullptr)

throw AlertManagerDoesNotExist\_ex();

return m\_this;

}

void Alert\_Manager::show\_cycle()

{

while (!m\_show\_cycle\_thread\_exit)

{

alert\_t c\_alert;

m\_alerts\_mutex.lock();

while (m\_alerts.size() && !m\_show\_cycle\_thread\_exit)

{

c\_alert = m\_alerts[0];

m\_alerts\_mutex.unlock();

MessageBoxA(NULL, c\_alert.message.c\_str(), c\_alert.header.c\_str(), MB\_OK | MB\_ICONINFORMATION | MB\_SYSTEMMODAL);

m\_alerts\_mutex.lock();

m\_alerts.erase(m\_alerts.begin());

}

m\_alerts\_mutex.unlock();

}

}

1. Task.h

#pragma once

// Task triggers:

#include "Task\_trigger\_dayly.h"

#include "Task\_trigger\_weekly.h"

#include "Task\_trigger\_monthly.h"

#include "Task\_trigger\_once.h"

#include "Task\_trigger\_entrance.h"

// Task acts:

#include "Task\_act\_prog.h"

#include "Task\_act\_alert.h"

struct Task\_header\_t

{

unsigned int id;

std::string name;

std::string desc;

};

class Task

{

public:

Task(Task\_header\_t header\_, Task\_trigger \*&trigger\_, Task\_act \*&act\_);

Task(const Task &right);

~Task();

/\*\*

\* Caclulate time left for trigger

\* @param c\_time - time for which calculations must be done

\* @return false if task must be deleted(because it would never be done)

\*/

bool calculate\_time\_left(Time c\_time);

void make\_act();

static const bool compare(const Task \*left, const Task \*right);

//

// Getters

//

unsigned int Get\_id() const { return header.id; }

int Get\_time\_left() const { return trigger->Get\_time\_left(); }

Trigger\_type\_t Get\_trigger\_type() const { return trigger->Get\_type(); }

Task\_header\_t Get\_header() const { return header; }

#ifdef DEBUG

Task\_trigger\* Get\_trigger() const { return trigger; }

#endif // DEBUG

//

// Setters

//

void Set\_last\_time(Time last\_time\_) { last\_time = last\_time\_; }

void Set\_was\_maked(bool was\_maked) { m\_was\_maked = was\_maked; }

const Task& operator--();

#ifdef DEBUG

void output();

#endif // DEBUG

private:

Task\_header\_t header;

Task\_trigger \*trigger;

Task\_act \*act;

Time last\_time; // Last time when task has been done

bool m\_was\_maked;

};

1. Task.cpp

#include "Task.h"

Task::Task(Task\_header\_t header\_, Task\_trigger \*&trigger\_, Task\_act \*&act\_)

: header(header\_), m\_was\_maked(false)

{

trigger = trigger\_;

trigger\_ = nullptr;

act = act\_;

act\_ = nullptr;

}

Task::Task(const Task &right)

:m\_was\_maked(false)

{

header = right.header;

switch (right.trigger->Get\_type())

{

case DAYLY:

trigger = new Task\_trigger\_dayly(\*((Task\_trigger\_dayly\*)right.trigger));

break;

case WEEKLY:

trigger = new Task\_trigger\_weekly(\*((Task\_trigger\_weekly\*)right.trigger));

break;

case MONTHLY:

trigger = new Task\_trigger\_monthly(\*((Task\_trigger\_monthly\*)right.trigger));

break;

case ONCE:

trigger = new Task\_trigger\_once(\*((Task\_trigger\_once\*)right.trigger));

break;

case ENTRANCE:

trigger = new Task\_trigger\_entrance(\*((Task\_trigger\_entrance\*)right.trigger));

break;

default:

throw WrongTriggerType\_ex();

break;

}

switch (right.act->Get\_type())

{

case PROG:

act = new Task\_act\_prog(\*((Task\_act\_prog\*)right.act));

break;

case ALERT:

act = new Task\_act\_prog(\*((Task\_act\_prog\*)right.trigger));

break;

default:

throw WrongActType\_ex();

break;

}

}

Task::~Task()

{

if (trigger != nullptr)

{

delete trigger;

trigger = nullptr;

}

if (act != nullptr)

{

delete act;

act = nullptr;

}

}

bool Task::calculate\_time\_left(Time c\_time)

{

if (m\_was\_maked && trigger->Get\_type() == ENTRANCE)

{

trigger->Set\_time\_left(INF);

return true;

}

if (c\_time == last\_time)

{

trigger->Set\_time\_left(INF);

if (trigger->Get\_type() == ONCE)

return false;

}

else

{

trigger->calculate\_time\_left(c\_time);

}

return true;

}

void Task::make\_act()

{

m\_was\_maked = true;

act->make\_act();

}

const bool Task::compare(const Task \*left, const Task \*right)

{

return (\*(left->trigger) < \*(right->trigger));

}

const Task& Task::operator--()

{

this->header.id--;

return \*this;

}

#ifdef DEBUG

void Task::output()

{

printf("Task #%u\nName: %s\nDesc: %s", header.id, header.name.c\_str(), header.desc.c\_str());

printf("\nTrigger:\n");

trigger->output();

printf("\nAct:\n");

act->output();

std::string s1, s2;

switch (act->Get\_type())

{

case ALERT:

s1 = ((Task\_act\_alert\*)act)->Get\_alert\_name();

s2 = ((Task\_act\_alert\*)act)->Get\_alert\_text();

break;

case PROG:

s1 = ((Task\_act\_prog\*)act)->Get\_prog\_name();

s2 = ((Task\_act\_prog\*)act)->Get\_prog\_params();

break;

default:

throw WrongActType\_ex();

break;

}

printf("\nName:\n%s\nText:\n%s\n", s1.c\_str(), s2.c\_str());

}

#endif // DEBUG

1. \_Time.h

#pragma once

#include "stdafx.h"

#include "globals.h"

struct Time\_t

{

Time\_t()

: dat(boost::gregorian::date(1400, 1, 1)), hours(0), minutes(0)

{}

Time\_t(boost::gregorian::date dat\_, int hours\_, int minutes\_)

: dat(dat\_), hours(hours\_), minutes(minutes\_)

{}

boost::gregorian::date dat;

int hours;

int minutes;

};

class Time

{

public:

Time();

Time(Time\_t time\_);

Time(unsigned int year\_, unsigned int month\_, unsigned int day\_, unsigned int hours\_, unsigned int minutes\_);

Time(boost::gregorian::date dat\_, unsigned int hours\_, unsigned int minutes\_);

Time(const Time& right);

static Time current\_time();

/\*\*

\* Get number of seconds left to nearly new minute

\* @return number of seconds

\*/

static unsigned int to\_minute\_left();

/\*\*

\* Get number of minutes left to nearly new hour

\* @return number of minutes in SECONDS

\*/

static unsigned int to\_hour\_left();

//

// Setters

//

void Set\_dat(boost::gregorian::date dat\_);

/\*\*

\* Increase current date, so current weekday = wday\_

\* @param wday\_ weekday

\*/

void Set\_next\_weekday(const boost::date\_time::weekdays wday\_);

/\*\*

\* Increase current date to next nearly date dependce on month\_list and days\_list

\* @param month\_list sorted, not empty vector of months from which choose must be done

\* @param days\_list sorted, not empty vector of days from which choose must be done

Example:

input:

\* this: 30.01.2017 22:50

\* month\_list = { 1, 2, 12 }

\* days\_list = { 29 }

output:

\* this: 29.12.2017 22:50

\*/

void Set\_next\_month\_day(const std::vector<boost::gregorian::months\_of\_year> month\_list, const std::vector<unsigned int> days\_list);

//

// Getters

//

boost::gregorian::date Get\_dat() const;

Time\_t Get\_time() const;

//

// Operators

//

const Time operator+(const int num\_of\_days) const;

const Time operator+=(const int num\_of\_days);

/\*\*

\* Calculate difference between two Time's

\* @return number of minutes, if all except minutes in @this time is equal to @right time

\* \* Otherwise: return INF

\*/

const int operator-(const Time& right) const;

/\*\*

\* Calculate how many weeks must be added to @right, so @righth would be > @this

\* @param right Time dependce on which calculation must be done

\* @return number of weeks

\*/

const int operator%(const Time& right) const;

const bool operator>(const Time& right) const;

const bool operator>=(const Time& right) const;

const bool operator<(const Time& right) const;

const bool operator<=(const Time& right) const;

const bool operator==(const Time& right) const;

const bool operator!=(const Time& right) const;

#ifdef DEBUG

void output();

#endif // DEBUG

private:

Time\_t time;

};

1. \_Time.cpp

#include "\_Time.h"

Time::Time()

{

}

Time::Time(Time\_t time\_)

: time(time\_)

{

}

Time::Time(unsigned int year\_, unsigned int month\_, unsigned int day\_, unsigned int hours\_, unsigned int minutes\_)

{

if (year\_ < 1400 || month\_ <= 0 || month\_ > 12 || day\_ > boost::gregorian::date(year\_, month\_, 1).end\_of\_month().day() || hours\_ >= 24 || minutes\_ >= 60)

{

throw WrongTimeFormat\_ex();

}

time.dat = boost::gregorian::date(year\_, month\_, day\_);

time.hours = hours\_;

time.minutes = minutes\_;

}

Time::Time(boost::gregorian::date dat\_, unsigned int hours\_, unsigned int minutes\_)

: time(dat\_, hours\_, minutes\_)

{

}

Time::Time(const Time& right)

{

time = right.time;

}

Time Time::current\_time()

{

time\_t c\_time = std::time(NULL);

tm \*tm\_c\_time;

tm\_c\_time = std::localtime(&c\_time);

return Time(tm\_c\_time->tm\_year + 1900, tm\_c\_time->tm\_mon + 1, tm\_c\_time->tm\_mday, tm\_c\_time->tm\_hour, tm\_c\_time->tm\_min);

}

unsigned int Time::to\_minute\_left()

{

time\_t c\_time = std::time(NULL);

tm \*tm\_c\_time;

tm\_c\_time = std::localtime(&c\_time);

unsigned int n;

if (tm\_c\_time->tm\_sec == 0)

n = 0;

else

n = 60 - tm\_c\_time->tm\_sec;

return n;

}

unsigned int Time::to\_hour\_left()

{

time\_t c\_time = std::time(NULL);

tm \*tm\_c\_time;

tm\_c\_time = std::localtime(&c\_time);

unsigned int n\_sec = 0, n\_min = 0, n;

if (tm\_c\_time->tm\_min == 0 && tm\_c\_time->tm\_sec == 0)

n = 0;

else

{

n\_min = (60 - tm\_c\_time->tm\_min);

if (tm\_c\_time->tm\_sec != 0)

{

--n\_min;

n\_sec = 60 - tm\_c\_time->tm\_sec;

}

n = n\_min \* 60 + n\_sec;

}

return n;

}

//

// Setters

//

void Time::Set\_dat(boost::gregorian::date dat\_)

{

time.dat = dat\_;

}

void Time::Set\_next\_weekday(const boost::date\_time::weekdays wday\_)

{

int wday = time.dat.day\_of\_week();

int inc = wday\_ - wday;

if (inc == 0)

return;

if (inc < 0)

inc += 7;

time.dat += boost::gregorian::date\_duration(inc);

}

void Time::Set\_next\_month\_day(const std::vector<boost::gregorian::months\_of\_year> month\_list, const std::vector<unsigned int> days\_list)

{

if (month\_list.size() == 0 || days\_list.size() == 0)

return;

std::vector<boost::gregorian::months\_of\_year>::const\_iterator month = std::lower\_bound(month\_list.cbegin(), month\_list.cend(), time.dat.month().as\_enum());//std::find\_if(month\_list.begin(), month\_list.end(), std::bind<bool>(std::greater\_equal<boost::date\_time::months\_of\_year>(), \_1, time.dat.month()));

std::vector<unsigned int>::const\_iterator day;

unsigned int year = time.dat.year();

if (month == month\_list.cend())

{

month = month\_list.begin();

++year;

day = days\_list.begin();

}

else

{

if (\*month == time.dat.month())

{

day = std::lower\_bound(days\_list.begin(), days\_list.end(), time.dat.day().as\_number());//std::find\_if(days\_list.begin(), days\_list.end(), std::bind<bool>(std::greater\_equal<unsigned int>(), \_1, time.dat.day()));

if (day == days\_list.end())

{

day = days\_list.begin();

++month;

if (month == month\_list.end())

{

month = month\_list.begin();

++year;

}

}

}

else

day = days\_list.begin();

}

while (boost::gregorian::date(year, \*month, 1).end\_of\_month().day() < \*day)

{

day = days\_list.begin();

++month;

if (month == month\_list.end())

{

month = month\_list.begin();

++year;

}

}

time.dat = boost::gregorian::date(year, \*month, \*day);

}

//

// Getters

//

boost::gregorian::date Time::Get\_dat() const

{

return time.dat;

}

Time\_t Time::Get\_time() const

{

return time;

}

const Time Time::operator+(const int num\_of\_days) const

{

Time tmp = \*this;

tmp.time.dat += boost::gregorian::date\_duration(num\_of\_days);

return tmp;

}

const Time Time::operator+=(const int num\_of\_days)

{

time.dat += boost::gregorian::date\_duration(num\_of\_days);

return \*this;

}

const int Time::operator-(const Time& right) const

{

if (time.dat == right.time.dat)

{

int time\_left = time.minutes - right.time.minutes + (time.hours - right.time.hours)\*60;

if (time\_left >= 0)

return time\_left;

}

return INF;

}

const int Time::operator%(const Time& right) const

{

return (time.dat - right.time.dat).days() / 7;

}

const bool Time::operator>(const Time& right) const

{

if (time.dat > right.time.dat)

{

return true;

}

else if (time.dat == right.time.dat)

{

if (time.hours > right.time.hours)

{

return true;

}

else if (time.hours == right.time.hours)

{

if (time.minutes > right.time.minutes)

{

return true;

}

}

}

/\*if (time.year > right.time.year)

{

return true;

}

else if (time.year == right.time.year)

{

if (time.month > right.time.month)

{

return true;

}

else if (time.month == right.time.month)

{

if (time.mday > right.time.mday)

{

return true;

}

else if (time.mday == right.time.mday)

{

if (time.hours > right.time.hours)

{

return true;

}

else if (time.hours == right.time.hours)

{

if (time.minutes > right.time.minutes)

{

return true;

}

}

}

}

}

\*/

return false;

}

const bool Time::operator>=(const Time& right) const

{

if (\*this > right || \*this == right)

return true;

return false;

}

const bool Time::operator<(const Time& right) const

{

if (\*this != right)

return !(\*this > right);

return false;

}

const bool Time::operator<=(const Time& right) const

{

if (\*this < right || \*this == right)

return true;

return false;

}

const bool Time::operator==(const Time& right) const

{

if (time.dat == right.time.dat && time.hours == right.time.hours && time.minutes == right.time.minutes)

return true;

return false;

}

const bool Time::operator!=(const Time& right) const

{

return !(\*this == right);

}

#ifdef DEBUG

void Time::output()

{

printf("%d.%d.%d %d:%d\n", time.dat.year(), time.dat.month(), time.dat.day(), time.hours, time.minutes);

}

#endif // DEBUG

1. Task\_Exception.h

#pragma once

#include "stdafx.h"

enum Task\_Exception\_error\_code\_t

{

ConfigFileNotFound,

ConfigFileAlreadyOpened,

ConfigFileAlreadyDeleted,

ConfigFileCorrupted,

CanNotOpenConfigFile,

CanNotOpenFile,

FileCorrupted,

TaskAlreadyDeleted,

WaiterThreadCycleAlreadyDeleted,

TaskIdDoesNotExist,

TaskIdDoesNotFound,

EndOfFileWasReached,

TryAgain,

WrongTriggerType,

WrongActType,

WrongTimeFormat,

WrongTime,

WrongWeeklyTriggerEveryNWeek,

WrongMonthlyTriggerDaysVecSize,

WrongMonthlyTriggerMonthsVecSize,

AlertManagerDoesNotExist,

AlertManagerAlreadyCreated,

WrongExceptionErrorCode

};

class Task\_Exception

{

public:

Task\_Exception(Task\_Exception\_error\_code\_t error\_code\_);

virtual ~Task\_Exception();

static void delete\_(Task\_Exception \*&ex\_);

operator std::string() const;

std::string Get\_as\_string\_with\_solution() const;

Task\_Exception\_error\_code\_t Get\_error\_code() const { return error\_code; }

protected:

Task\_Exception\_error\_code\_t error\_code;

};

class ConfigFileNotFound\_ex : public Task\_Exception

{

public:

ConfigFileNotFound\_ex() : Task\_Exception(ConfigFileNotFound) {}

};

class ConfigFileAlreadyOpened\_ex : public Task\_Exception

{

public:

ConfigFileAlreadyOpened\_ex() : Task\_Exception(ConfigFileAlreadyOpened) {}

};

class ConfigFileAlreadyDeleted\_ex : public Task\_Exception

{

public:

ConfigFileAlreadyDeleted\_ex() : Task\_Exception(ConfigFileAlreadyDeleted) {}

};

class ConfigFileCorrupted\_ex : public Task\_Exception

{

public:

ConfigFileCorrupted\_ex() : Task\_Exception(ConfigFileCorrupted) {}

};

class CanNotOpenConfigFile\_ex : public Task\_Exception

{

public:

CanNotOpenConfigFile\_ex() : Task\_Exception(CanNotOpenConfigFile) {}

};

class CanNotOpenFile\_ex : public Task\_Exception

{

public:

CanNotOpenFile\_ex() : Task\_Exception(CanNotOpenFile) {}

};

class FileCorrupted\_ex : public Task\_Exception

{

public:

FileCorrupted\_ex() : Task\_Exception(FileCorrupted) {}

};

class TaskAlreadyDeleted\_ex : public Task\_Exception

{

public:

TaskAlreadyDeleted\_ex() : Task\_Exception(TaskAlreadyDeleted) {}

};

class WaiterThreadCycleAlreadyDeleted\_ex : public Task\_Exception

{

public:

WaiterThreadCycleAlreadyDeleted\_ex() : Task\_Exception(WaiterThreadCycleAlreadyDeleted) {}

};

class TaskIdDoesNotExist\_ex : public Task\_Exception

{

public:

TaskIdDoesNotExist\_ex() : Task\_Exception(TaskIdDoesNotExist) {}

};

class TaskIdDoesNotFound\_ex : public Task\_Exception

{

public:

TaskIdDoesNotFound\_ex() : Task\_Exception(TaskIdDoesNotFound) {}

};

class EndOfFileWasReached\_ex : public Task\_Exception

{

public:

EndOfFileWasReached\_ex() : Task\_Exception(EndOfFileWasReached) {}

};

class TryAgain\_ex : public Task\_Exception

{

public:

TryAgain\_ex() : Task\_Exception(TryAgain) {}

};

class WrongTriggerType\_ex : public Task\_Exception

{

public:

WrongTriggerType\_ex() : Task\_Exception(WrongTriggerType) {}

};

class WrongActType\_ex : public Task\_Exception

{

public:

WrongActType\_ex() : Task\_Exception(WrongActType) {}

};

class WrongTimeFormat\_ex : public Task\_Exception

{

public:

WrongTimeFormat\_ex() : Task\_Exception(WrongTimeFormat) {}

};

class WrongTime\_ex : public Task\_Exception

{

public:

WrongTime\_ex() : Task\_Exception(WrongTime) {}

};

class WrongWeeklyTriggerEveryNWeek\_ex : public Task\_Exception

{

public:

WrongWeeklyTriggerEveryNWeek\_ex() : Task\_Exception(WrongWeeklyTriggerEveryNWeek) {}

};

class WrongMonthlyTriggerDaysVecSize\_ex : public Task\_Exception

{

public:

WrongMonthlyTriggerDaysVecSize\_ex() : Task\_Exception(WrongMonthlyTriggerDaysVecSize) {}

};

class WrongMonthlyTriggerMonthsVecSize\_ex : public Task\_Exception

{

public:

WrongMonthlyTriggerMonthsVecSize\_ex() : Task\_Exception(WrongMonthlyTriggerMonthsVecSize) {}

};

class AlertManagerDoesNotExist\_ex : public Task\_Exception

{

public:

AlertManagerDoesNotExist\_ex() : Task\_Exception(AlertManagerDoesNotExist) {}

};

class AlertManagerAlreadyCreated\_ex : public Task\_Exception

{

public:

AlertManagerAlreadyCreated\_ex() : Task\_Exception(AlertManagerAlreadyCreated) {}

};

class WrongExceptionErrorCode\_ex : public Task\_Exception

{

public:

WrongExceptionErrorCode\_ex() : Task\_Exception(WrongExceptionErrorCode) {}

};

1. Task\_Exeption.cpp

#include "Task\_Exception.h"

Task\_Exception::Task\_Exception(Task\_Exception\_error\_code\_t error\_code\_)

: error\_code(error\_code\_)

{

}

Task\_Exception::~Task\_Exception()

{

}

void Task\_Exception::delete\_(Task\_Exception \*&ex\_)

{

exit(EXIT\_FAILURE); // TODO: Not supported function

//WrongWeeklyTriggerEveryNWeek

switch (ex\_->Get\_error\_code())

{

case ConfigFileNotFound:

delete ((ConfigFileNotFound\_ex\*)ex\_);

break;

case ConfigFileAlreadyOpened:

delete ((ConfigFileAlreadyOpened\_ex\*)ex\_);

break;

case ConfigFileAlreadyDeleted:

delete ((ConfigFileAlreadyDeleted\_ex\*)ex\_);

break;

case ConfigFileCorrupted:

delete ((ConfigFileCorrupted\_ex\*)ex\_);

break;

case CanNotOpenConfigFile:

delete ((CanNotOpenConfigFile\_ex\*)ex\_);

break;

case CanNotOpenFile:

delete ((CanNotOpenFile\_ex\*)ex\_);

break;

case FileCorrupted:

delete ((FileCorrupted\_ex\*)ex\_);

break;

case TaskAlreadyDeleted:

delete ((TaskAlreadyDeleted\_ex\*)ex\_);

break;

case WaiterThreadCycleAlreadyDeleted:

delete ((WaiterThreadCycleAlreadyDeleted\_ex\*)ex\_);

break;

case TaskIdDoesNotExist:

delete ((TaskIdDoesNotExist\_ex\*)ex\_);

break;

case TaskIdDoesNotFound:

delete ((TaskIdDoesNotFound\_ex\*)ex\_);

break;

case EndOfFileWasReached:

delete ((EndOfFileWasReached\_ex\*)ex\_);

break;

case TryAgain:

delete ((TryAgain\_ex\*)ex\_);

break;

case WrongTriggerType:

delete ((WrongTriggerType\_ex\*)ex\_);

break;

case WrongActType:

delete ((WrongActType\_ex\*)ex\_);

break;

case WrongTimeFormat:

delete ((WrongTimeFormat\_ex\*)ex\_);

break;

case WrongTime:

delete ((WrongTime\_ex\*)ex\_);

break;

case AlertManagerDoesNotExist:

delete ((AlertManagerDoesNotExist\_ex\*)ex\_);

break;

case AlertManagerAlreadyCreated:

delete ((AlertManagerAlreadyCreated\_ex\*)ex\_);

break;

case WrongExceptionErrorCode:

delete ((WrongExceptionErrorCode\_ex\*)ex\_);

break;

default:

break;

}

}

Task\_Exception::operator std::string() const

{

switch (error\_code)

{

case ConfigFileNotFound:

return std::string("Configuration file does not found");

break;

case ConfigFileAlreadyOpened:

return std::string("Configuration file was already opened");

break;

case ConfigFileAlreadyDeleted:

return std::string("Configuration file was already deleted");

break;

case ConfigFileCorrupted:

return std::string("Configuration file was corrupted");

break;

case CanNotOpenConfigFile:

return std::string("Can't open configuration file");

break;

case CanNotOpenFile:

return std::string("Can't open file");

break;

case FileCorrupted:

return std::string("File was corrupted");

break;

case TaskAlreadyDeleted:

return std::string("Task was already deleted");

break;

case WaiterThreadCycleAlreadyDeleted:

return std::string("Waiter cycle thread was already deleted");

break;

case TaskIdDoesNotExist:

return std::string("Task with current id does not exist");

break;

case TaskIdDoesNotFound:

return std::string("Can't found task with current id");

break;

case EndOfFileWasReached:

return std::string("End of file was reached");

break;

case TryAgain:

return std::string("Something went wrong");

break;

case WrongTriggerType:

return std::string("Wrong trigger type");

break;

case WrongActType:

return std::string("Wrong act type");

break;

case WrongTimeFormat:

return std::string("Wrong time format");

break;

case WrongTime:

return std::string("Wrong time was specified");

break;

case WrongWeeklyTriggerEveryNWeek:

return std::string("Wrong \"Every n week\" was specified");

break;

case WrongMonthlyTriggerDaysVecSize:

return std::string("Wrong number of days was specified for monthly trigger");

break;

case WrongMonthlyTriggerMonthsVecSize:

return std::string("Wrong number of months was specified for monthly trigger");

break;

case AlertManagerDoesNotExist:

return std::string("Alert Manager does not exist");

break;

case AlertManagerAlreadyCreated:

return std::string("Alert Manager was already created");

break;

case WrongExceptionErrorCode:

return std::string("Wrong exception error code");

break;

default:

throw WrongExceptionErrorCode\_ex();

break;

}

}

std::string Task\_Exception::Get\_as\_string\_with\_solution() const

{

switch (error\_code)

{

case ConfigFileNotFound:

return std::string("Error: Configuration file does not found\nSolution: Restart task manager");

break;

case ConfigFileAlreadyOpened:

return std::string("Error: Configuration file was already opened\nSolution: Restart task manager");

break;

case ConfigFileAlreadyDeleted:

return std::string("Error: Config file was already deleted\nSolution: Restart task manager");

break;

case ConfigFileCorrupted:

return std::string("Error: Configuration file was corrupted\nSolution: Restart task manager");

break;

case CanNotOpenConfigFile:

return std::string("Error: Can't open configuration file\nSolution: Restart task manager");

break;

case CanNotOpenFile:

return std::string("Error: Can't open file\nSolution: Close file and retry");

break;

case FileCorrupted:

return std::string("Error: File was corrupted\nSolution: Choose another file");

break;

case TaskAlreadyDeleted:

return std::string("Error: Task was already deleted\nSolution: Restart task manager");

break;

case WaiterThreadCycleAlreadyDeleted:

return std::string("Error: Waiter cycle thread was already deleted\nSolution: Restart task manager");

break;

case TaskIdDoesNotExist:

return std::string("Error: Task with current id does not exist\nSolution: Restart task manager");

break;

case TaskIdDoesNotFound:

return std::string("Error: Can't found task with current id\nSolution: Restart task manager");

break;

case EndOfFileWasReached:

return std::string("Error: End of file was reached\nSolution: Restart task manager");

break;

case TryAgain:

return std::string("Error: Something went wrong\nSolution: Restart task manager");

break;

case WrongTriggerType:

return std::string("Error: Wrong trigger type\nSolution: Restart task manager");

break;

case WrongActType:

return std::string("Error: Wrong act type\nSolution: Restart task manager");

break;

case WrongTimeFormat:

return std::string("Error: Wrong time format\nSolution: Input another time");

break;

case WrongTime:

return std::string("Error: Wrong time was specified\nSolution: Input another time");

break;

case WrongWeeklyTriggerEveryNWeek:

return std::string("Error: Wrong \"Every n week\" was specified\nSolution: Input number which is more then zero");

break;

case WrongMonthlyTriggerDaysVecSize:

return std::string("Error: Wrong number of days was specified for monthly trigger\nSolution: input at least one day");

break;

case WrongMonthlyTriggerMonthsVecSize:

return std::string("Error: Wrong number of months was specified for monthly trigger\nSolution: input at least one month");

break;

case AlertManagerDoesNotExist:

return std::string("Error: Alert Manager does not exist\nSolution:\n\tFor user: Restart task manager\n\tFor developer: Create Alert\_Manager before calling this function");

break;

case AlertManagerAlreadyCreated:

return std::string("Error: Alert Manager was already created\nSolution:\n\tFor user: Restrart task manager\n\tFor developer: Be sure you don't create a few Alert\_Manager objects");

break;

case WrongExceptionErrorCode:

return std::string("Error: Wrong exception error code\nSolution: Restart task manager");

break;

default:

throw WrongExceptionErrorCode\_ex();

break;

}

}

1. Task\_trigger.h

#pragma once

#include "stdafx.h"

#include "globals.h"

#include "\_Time.h"

enum Trigger\_type\_t

{

DAYLY,

WEEKLY,

MONTHLY,

ONCE,

ENTRANCE

};

class Task\_trigger

{

public:

Task\_trigger();

virtual ~Task\_trigger();

Task\_trigger(Trigger\_type\_t type\_, Time time\_, unsigned int priority\_);

const bool operator<(const Task\_trigger &right) const;

/\*\*

\* Caclulate time left

\* @param c\_time - time for which calculations must be done

\* @return false if trigger must be deleted(because it would never work)

\*/

virtual bool calculate\_time\_left(Time c\_time) = 0;

//

// Getters

//

unsigned int Get\_time\_left() const { return time\_left; }

unsigned int Get\_priority() const { return priority; }

Trigger\_type\_t Get\_type() const { return type; }

Time Get\_time() const { return time; }

//

// Setters

//

void Set\_time\_left(int time\_left\_) { time\_left = time\_left\_; }

#ifdef DEBUG

void output();

#endif // DEBUG

protected:

Trigger\_type\_t type;

Time time; // Time of next triggering

Time time\_begin; // Time from which begin triggering

unsigned int priority;

unsigned int time\_left; // In minutes

bool is\_not\_calculated;

};

1. Task\_trigger.h

#include "Task\_trigger.h"

Task\_trigger::Task\_trigger(Trigger\_type\_t type\_, Time time\_, unsigned int priority\_)

: type(type\_), time(time\_), time\_begin(time\_), priority(priority\_), time\_left(INF),

is\_not\_calculated(true)

{

}

Task\_trigger::Task\_trigger()

{

}

Task\_trigger::~Task\_trigger()

{

}

const bool Task\_trigger::operator<(const Task\_trigger &right) const

{

if (time\_left < right.time\_left || (time\_left == right.time\_left && priority < right.priority))

return true;

return false;

}

#ifdef DEBUG

void Task\_trigger::output()

{

printf("Trigger type: ");

switch (type)

{

case DAYLY:

printf("DAYLY");

break;

case WEEKLY:

printf("WEEKLY");

break;

case MONTHLY:

printf("MONTHLY");

break;

case ONCE:

printf("ONCE");

break;

case ENTRANCE:

printf("ENTRANCE");

break;

default:

printf("ERROR");

break;

}

printf("\nTime:");

time.output();

printf("\nPriotiry: %u\nTime left: %d", priority, time\_left);

}

#endif // DEBUG

1. Task\_trigger\_dayly.h

#pragma once

#include "Task\_trigger.h"

class Task\_trigger\_dayly :

public Task\_trigger

{

public:

Task\_trigger\_dayly(Time time\_, unsigned int priority\_);

/\*\*

\* Caclulate time left

\* @param c\_time - time for which calculations must be done

\* @return false if trigger must be deleted(because it would never work)

\*/

bool calculate\_time\_left(Time c\_time);

};

1. Task\_trigger\_dayly.cpp

#include "Task\_trigger\_dayly.h"

Task\_trigger\_dayly::Task\_trigger\_dayly(Time time\_, unsigned int priority\_)

: Task\_trigger(DAYLY, time\_, priority\_)

{

}

bool Task\_trigger\_dayly::calculate\_time\_left(Time c\_time)

{

if (time < c\_time || is\_not\_calculated)

{

is\_not\_calculated = false;

time.Set\_dat(c\_time.Get\_dat());

}

time\_left = time - c\_time;

return true;

}

1. Task\_trigger\_entrace.h

#pragma once

#include "Task\_trigger.h"

class Task\_trigger\_entrance :

public Task\_trigger

{

public:

Task\_trigger\_entrance(Time time\_, unsigned int priority\_);

/\*\*

\* Caclulate time left

\* @param c\_time - time for which calculations must be done

\* @return false if trigger must be deleted(because it would never work)

\*/

bool calculate\_time\_left(Time c\_time);

};

1. Task\_trigger\_entrace.cpp

#include "Task\_trigger\_entrance.h"

Task\_trigger\_entrance::Task\_trigger\_entrance(Time time\_, unsigned int priority\_)

: Task\_trigger(ENTRANCE, time\_, priority\_)

{

}

bool Task\_trigger\_entrance::calculate\_time\_left(Time c\_time)

{

if (c\_time >= time)

time\_left = 0;

return true;

}

1. Task\_trigger\_monthly.h

#pragma once

#include "Task\_trigger.h"

class Task\_trigger\_monthly :

public Task\_trigger

{

public:

Task\_trigger\_monthly(Time time\_, unsigned int priority\_, const std::vector<boost::date\_time::months\_of\_year> &month\_list\_, const std::vector<unsigned int> &days\_list\_);

/\*\*

\* Caclulate time left

\* @param c\_time - time for which calculations must be done

\* @return false if trigger must be deleted(because it would never work)

\*/

bool calculate\_time\_left(Time c\_time);

const std::vector<boost::date\_time::months\_of\_year> Get\_month\_list() const { return month\_list; }

const std::vector<unsigned int> Get\_days\_list() const { return days\_list; }

private:

std::vector<boost::date\_time::months\_of\_year> month\_list;

std::vector<unsigned int> days\_list;

};

1. Task\_trigger\_monthly.cpp

#include "Task\_trigger\_monthly.h"

Task\_trigger\_monthly::Task\_trigger\_monthly(Time time\_, unsigned int priority\_, const std::vector<boost::date\_time::months\_of\_year> &month\_list\_, const std::vector<unsigned int> &days\_list\_)

: Task\_trigger(MONTHLY, time\_, priority\_), month\_list(month\_list\_), days\_list(days\_list\_)

{

if (month\_list.size() == 0)

throw WrongMonthlyTriggerMonthsVecSize\_ex();

if (days\_list.size() == 0)

throw WrongMonthlyTriggerDaysVecSize\_ex();

std::sort(month\_list.begin(), month\_list.end());

std::unique(month\_list.begin(), month\_list.end());

std::sort(days\_list.begin(), days\_list.end());

std::unique(days\_list.begin(), days\_list.end());

}

bool Task\_trigger\_monthly::calculate\_time\_left(Time c\_time)

{

if (time < c\_time || is\_not\_calculated)

{

is\_not\_calculated = false;

Time\_t tmp = time.Get\_time();

time = Time(c\_time.Get\_dat(), tmp.hours, tmp.minutes);

if (time < c\_time)

time += 1;

time.Set\_next\_month\_day(month\_list, days\_list);

}

time\_left = time - c\_time;

return true;

}

1. Task\_trigger\_once.h

#pragma once

#include "Task\_trigger.h"

class Task\_trigger\_once :

public Task\_trigger

{

public:

Task\_trigger\_once(Time time\_, unsigned int priority\_);

/\*\*

\* Caclulate time left

\* @param c\_time - time for which calculations must be done

\* @return false if trigger must be deleted(because it would never work)

\*/

bool calculate\_time\_left(Time c\_time);

};

1. Task\_trigger\_once.cpp

#include "Task\_trigger\_once.h"

Task\_trigger\_once::Task\_trigger\_once(Time time\_, unsigned int priority\_)

: Task\_trigger(ONCE, time\_, priority\_)

{

}

bool Task\_trigger\_once::calculate\_time\_left(Time c\_time)

{

if (c\_time <= time)

time\_left = time - c\_time;

else

return false;

return true;

}

1. Task\_trigger\_weekly.h

#pragma once

#include "Task\_trigger.h"

class Task\_trigger\_weekly :

public Task\_trigger

{

public:

Task\_trigger\_weekly(Time time\_, unsigned int priority\_, boost::date\_time::weekdays week\_day\_, unsigned int every\_n\_week\_);

/\*\*

\* Caclulate time left

\* @param c\_time - time for which calculations must be done

\* @return false if trigger must be deleted(because it would never work)

\*/

bool calculate\_time\_left(Time c\_time);

//

// Getters

//

boost::date\_time::weekdays Get\_week\_day() const { return week\_day; }

unsigned int Get\_every\_n\_week() const { return every\_n\_week; }

private:

boost::date\_time::weekdays week\_day;

unsigned int every\_n\_week;

};

1. Task\_trigger\_weekly.cpp

#include "Task\_trigger\_weekly.h"

Task\_trigger\_weekly::Task\_trigger\_weekly(Time time\_, unsigned int priority\_, boost::date\_time::weekdays week\_day\_, unsigned int every\_n\_week\_)

: Task\_trigger(WEEKLY, time\_, priority\_), week\_day(week\_day\_), every\_n\_week(every\_n\_week\_)

{

if (every\_n\_week <= 0)

throw WrongWeeklyTriggerEveryNWeek\_ex();

}

bool Task\_trigger\_weekly::calculate\_time\_left(Time c\_time)

{

if (time < c\_time || is\_not\_calculated)

{

is\_not\_calculated = false;

time.Set\_next\_weekday(week\_day);

if (time < c\_time)

{

int num\_of\_weeks = c\_time%time;

int num\_of\_weeks\_add = (num\_of\_weeks / every\_n\_week)\*every\_n\_week;

time += num\_of\_weeks\_add \* 7;

}

if (time < c\_time)

time += every\_n\_week \* 7;

}

time\_left = time - c\_time;

return true;

}

1. Task\_act.h

#pragma once

#include "stdafx.h"

#include "globals.h"

enum Task\_act\_type\_t

{

PROG,

ALERT

};

class Task\_act

{

public:

Task\_act(Task\_act\_type\_t type\_);

virtual void make\_act() = 0;

Task\_act\_type\_t Get\_type() const { return type; }

#ifdef DEBUG

void output();

#endif // DEBUG

protected:

Task\_act\_type\_t type;

};

1. Task\_act.cpp

#include "Task\_act.h"

Task\_act::Task\_act(Task\_act\_type\_t type\_)

: type(type\_)

{

}

#ifdef DEBUG

void Task\_act::output()

{

printf("Act type: ");

switch (type)

{

case PROG:

printf("PROG");

break;

case ALERT:

printf("ALERT");

break;

default:

printf("ERROR");

break;

}

}

#endif // DEBUG

1. Task\_act\_alert.h

#pragma once

#include "Task\_act.h"

class Task\_act\_alert :

public Task\_act

{

public:

Task\_act\_alert(std::string alert\_name\_, std::string alert\_text\_);

void make\_act();

std::string Get\_alert\_name() const { return alert\_name; }

std::string Get\_alert\_text() const { return alert\_text; }

private:

std::string alert\_name;

std::string alert\_text;

};

1. Task\_act\_alert.cpp

#include "Task\_act\_alert.h"

#include "Alert\_Manager.h"

Task\_act\_alert::Task\_act\_alert(std::string alert\_name\_, std::string alert\_text\_)

: Task\_act(ALERT), alert\_name(alert\_name\_), alert\_text(alert\_text\_)

{

}

void Task\_act\_alert::make\_act()

{

#ifdef DEBUG

printf("Act was maked:\nName:%s\nText:\n%s\n", alert\_name.c\_str(), alert\_text.c\_str());

#endif // DEBUG

Alert\_Manager::Get()->add\_alert(alert\_name, alert\_text);

}

1. Task\_act\_prog.h

#pragma once

#include "Task\_act.h"

class Task\_act\_prog :

public Task\_act

{

public:

Task\_act\_prog(std::string prog\_name\_, std::string prog\_params\_);

void make\_act();

std::string Get\_prog\_name() const { return prog\_name; }

std::string Get\_prog\_params() const { return prog\_params; }

private:

std::string prog\_name;

std::string prog\_params;

};

1. Task\_act\_prog.cpp

#include "Task\_act\_prog.h"

Task\_act\_prog::Task\_act\_prog(std::string prog\_name\_, std::string prog\_params\_)

: Task\_act(PROG), prog\_name(prog\_name\_), prog\_params(prog\_params\_)

{

}

void Task\_act\_prog::make\_act()

{

system(("start " + prog\_name + " " + prog\_params).c\_str());

}

1. globals.h

#pragma once

#define INF 999999

#ifdef \_DEBUG

#define DEBUG

#endif // \_DEBUG

#include "Task\_Exception.h"

1. stdafx.h

#pragma once

#include <vector>

#include <functional>

#include <algorithm>

#include <windows.h>

#include <boost/date\_time/gregorian/gregorian.hpp>

#include <boost/thread.hpp>

#include <string>

#include <fstream>

#include <time.h>