МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**КУРСОВА РОБОТА**

з дисципліни «Об’єктно-орієнтоване програмування»

Варіант №15

|  |  |  |
| --- | --- | --- |
| **Керівник :**  доц. каф. БМК,  к.т.н. Алхімова С.М. |  | **Виконав:**  студент гр. БС-81, ФБМІ  Сєров О.В.  залікова книжка № БС-8122 |
| Допущено до захисту |  |
| І\_\_\_І \_\_\_\_\_\_\_\_\_ 2020 \_\_\_\_\_\_\_\_\_\_\_  підпис |  |
| Захищено з оцінкою  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_  оцінка підпис  І\_\_\_І \_\_\_\_\_\_\_\_\_ 2020 |  |

Київ-2020

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Факультет | ***біомедичної інженерії*** | | | | |
|  | (назва факультету, інституту) | | | | |
| Кафедра | ***біомедичної кібернетики*** | | | | |
|  | (назва кафедри) | | | | |
| Дисципліна | ***«Об’єктно-орієнтоване програмування»*** | | | | |
|  | (назва) | | | | |
| Курс | ***2*** | Група | ***БС-81*** | Семестр | ***4*** |

**ЗАВДАННЯ**

**на курсовий проект (роботу) студента**

|  |
| --- |
| ***Сєрова Олександра Владиславовича*** |
| (прізвище, ім’я, по батькові) |

.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. Тема роботи: | | ***Розробка програмного забезпечення з використанням об’єктно-*** | | | |
| ***орієнтованого підходу.*** | | | | | |
|  | | | | | |
|  | |  | | | |
| 2. Строк здачі студентом закінченого проекту (роботи) | | | | | ***15.05.2020*** |
|  | |  | | | |
| 3. Вихідні дані до проекту (роботи): | | | | ***Варіант №15*** | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | |  | | | |
| 4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають | | | | | |
| розробці): | ***1. Визначення класів та програмування меню користувача (ієрархія*** | | | | |
| ***класів програми: абстрактний клас – цех, класи-нащадки – цех пошиття одягу,*** | | | | | |
| ***меблевий цех). 2. Створення об’єктів та використання контейнерів (тип*** | | | | | |
| ***контейнера –***  ***двозв’язний кільцевий список). 3. Організація роботи з даними*** | | | | | |
| ***через файл. 4. Пошук даних у контейнері (запит для пошуку –***  ***сумарна*** | | | | | |
| ***потужність всіх верстатів в цехах заданого типу).*** | | | | | |
|  | |  | | | |
| 5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): | | | | | |
| ***діаграми класів, послідовності та об’єктів в нотації UML*** | | | | | |
|  | | | | | |
|  | |  | | | |
| 6. Дата видачі завдання: | | | ***18.02.2020*** | | |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| **№п/п** | **Назва етапів курсового проекту (роботи) та питань, які мають бути розроблені відповідно до завдання** | **Термін виконання етапу** | **Позначки керівника про виконання завдань** |
| 1. | Отримання завдання на курсову роботу | 18.02.2020 |  |
| 2. | Огляд технічної літератури за темою роботи | 21.02.2020 |  |
| 3. | Розробка першої частини курсової роботи (визначення класів та програмування меню користувача) | 06.03.2020 |  |
| 4. | Перший контроль за процесом виконання курсової роботи, консультація у викладача | 06.03.2020 |  |
| 5. | Розробка другої частини курсової роботи (створення об’єктів та використання контейнерів) | 20.03.2020 |  |
| 6. | Розробка третьої частини курсової роботи (робота з даними через файл) | 10.04.2020 |  |
| 7. | Другий контроль за процесом виконання курсової роботи, консультація у викладача | 10.04.2020 |  |
| 8. | Розробка четвертої частини курсової роботи (пошук даних у контейнері) | 24.04.2020 |  |
| 9. | Оформлення пояснювальної записки | 15.05.2020 |  |
| 10. | Захист курсової роботи | 26.05.2020 |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(підпис)

**Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Алхімова Світлана Миколаївна

(підпис) (прізвище ім'я, по батькові)

**«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 р.**

**Анотація**

Сєров О.В. Розробка програмного забезпечення з використанням об’єктно-орієнтованого підходу.

Курсова робота з дисципліни «Об’єктно-орієнтоване програмування» присвячена питанню створення ієрархії класів, застосування спадкування та поліморфізму, розробки контейнера для зберігання даних множини об’єктів, які створюються користувачем. У курсовій роботі було виконано визначення класів (ієрархія класів програми: базовий клас – Цех, класи-нащадки – Цех пошиття одягу, меблевий цех), розроблено меню користувача, створено множину об’єктів та розроблено контейнер для її зберігання (тип контейнера – двозв’язний кільцевий список), виконано серіалізацію даних елементів контейнера у файл та створення вмісту контейнера через десеріалізацію даних файлу, реалізовано пошук даних у контейнері (запит для пошуку – Обчислити сумарну потужність всіх верстатів в цехах заданого типу).

**Структура і обсяг роботи.** Курсова робота складається із вступу, трьох розділів, висновків, списку використаної літератури з 4 джерел і 5 додатків. Загальний обсяг курсової роботи становить 84 сторінки, основного тексту (без додатків) – 49 сторінок, ілюстрацій – 35, таблиць – 41.

**Annotation**

A. Serov Software development using object oriented approach.

Coursework on the Object Oriented Programming course is devoted to the issue of creating a hierarchy of classes, applying inheritance and polymorphism, developing storage container for the collection of objects created by the user. In the coursework class definitions were performed (hierarchy of program classes: base class – workshop, derived classes – sewing workshop, furniture workshop), user menu was designed, set of objects was created and its storage container was developed (type of container – circular doubly linked list), serialization of container data was done and container elements were created via deserialization of file data, data search in the container was performed (search query – Calculate the total power of all machines in the specified workshop type).

**The structure and the amount of work.** Coursework consists of an introduction, three partitions, conclusions, list of used literature with 4 references, and 5 applications. The total volume of coursework is 84 pages, main text (without applications) – 49 pages, illustrations – 35, tables – 41.

**Аннотация**

Серов А.В. Разработка программного обеспечения с использованием объектно-ориентированного подхода.

Курсовая работа по дисциплине «Объектно-ориентированное программирование» посвящена вопросу создания иерархии классов, применения наследования и полиморфизма, разработки контейнера для хранения данных коллекции объектов, которые создаются пользователем. В курсовой работе было выполнено определение классов (иерархия классов программы: базовый класс – цех, классы-потомки – цех пошива одежды, мебельный цех), разработано меню пользователя, создано множество объектов и разработан контейнер для их хранения (тип контейнера – двусвязный кольцевой список), выполнены сериализация данных элементов контейнера в файл и создание содержимого контейнера путем десериализации данных файла, реализован поиск данных в контейнере (запрос поиска – Вычислить суммарную мощность всех станков в цехах заданного типа).

**Структура и объем работы.** Курсовая работа состоит из введения, трех разделов, выводов, списка использованной литературы из 4 источников и 5 приложения. Общий объем курсовой работы составляет 84 страниц, основного текста (без приложений) – 49 страниц, иллюстраций – 35, таблиц – 41.

**ЗМІСТ**

[**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ** 10](#_Toc40462991)

[Скорочення: 10](#_Toc40462992)

[Терміни: 10](#_Toc40462993)

[**ВСТУП** 11](#_Toc40462994)

[Основні завдання: 11](#_Toc40462995)

[**ПЕРШИЙ РОЗДІЛ. СПЕЦИФІКАЦІЯ ВИМОГ** 12](#_Toc40462996)

[1.1 Функціональні вимоги 12](#_Toc40462997)

[*1.1.1* *Створення об’єкту «цех пошиття одягу» та додавання його у контейнер* 12](#_Toc40462998)

[*1.1.2* *Створення об’єкту «меблевий цех» та додавання його у контейнер* 12](#_Toc40462999)

[*1.1.3* *Переглянути усі цеха в контейнері* 13](#_Toc40463000)

[*1.1.4* *Видалення цеху за його назвою* 13](#_Toc40463001)

[*1.1.5* *Збереження даних у файлі* 14](#_Toc40463002)

[*1.1.6* *Зчитування даних з файлу* 15](#_Toc40463003)

[*1.1.7* *Сортування елементів контейнера* 15](#_Toc40463004)

[*1.1.8* *Сумарна потужність усіх верстатів в цехах заданого типу* 16](#_Toc40463005)

[*1.1.9* *Видалення усіх елементів з контейнера* 16](#_Toc40463006)

[*1.1.10* *Вихід із ПД* 17](#_Toc40463007)

[1.2 Нефункціональні вимоги 18](#_Toc40463008)

[**РОЗДІЛ 2. СТРУКТЕРА ТО ЛОГІКА РОБОТИ ПД** 19](#_Toc40463009)

[2.1 Логічна структура ПД 19](#_Toc40463010)

[*2.1.1 Клас «Workshop»* 19](#_Toc40463011)

[*2.1.2 Клас «Sewing\_workshop»* 20](#_Toc40463012)

[*2.1.3 Клас «Furniture\_workshop»* 22](#_Toc40463013)

[*2.1.3 Структура «Node»* 24](#_Toc40463014)

[*2.1.4 Клас «Сontainer»* 24](#_Toc40463015)

[*2.1.5 Клас «Menu»* 26](#_Toc40463016)

[2.2 Фізична структура 28](#_Toc40463017)

[2.3 Логіка роботи 32](#_Toc40463018)

[**РОЗДІЛ 3. ТЕСТУВАННЯ** 33](#_Toc40463019)

[3.1 Сценарії тестування функціональних вимог 33](#_Toc40463020)

[*3.1.1 Тест «Створення об’єкту Цеху пошиття одягу»* 33](#_Toc40463021)

[*3.1.2 Тест «Створення об’єкту Меблевого цеху»* 33](#_Toc40463022)

[*3.1.3 Вивід усіх цехів на екран* 34](#_Toc40463023)

[*3.1.4* *Видалення цеху з контейнера* 34](#_Toc40463024)

[*3.1.5 Збереження даних у файл* 35](#_Toc40463025)

[*3.1.6 Введення даних з файлу* 35](#_Toc40463026)

[*3.1.7 Сортування цехів у контейнері* 36](#_Toc40463027)

[*3.1.8 Сумарна потужність усіх верстатів в цехах заданого типу* 36](#_Toc40463028)

[*3.1.9 Видалення всіх цехів з контейнера* 37](#_Toc40463029)

[*3.1.10 Завершення роботи ПД* 38](#_Toc40463030)

[3.2 Сценарії тестування нефункціональних вимог 38](#_Toc40463031)

[*3.2.1 Запобігання повторного створення меню* 38](#_Toc40463032)

[*3.2.2 Некоректний вибір в меню* 39](#_Toc40463033)

[*3.2.3 Збереження пустого контейнера* 39](#_Toc40463034)

[*3.2.4 Видалення цеху за назвою, якої не має в контейнері* 40](#_Toc40463035)

[*3.2.5 Видалення елементу, коли контейнер вже пустий* 40](#_Toc40463036)

[*3.2.6 Виведення елементів на екран, коли контейнер пустий* 41](#_Toc40463037)

[*3.2.7 Сортування елементів, коли контейнер пустий* 41](#_Toc40463038)

[*3.2.8 Підрахунок потужності верстатів, коли контейнер пустий* 42](#_Toc40463039)

[*3.2.9 Підрахунок потужності верстатів у цехах заданого типу, коли такого типу в контейнері не існує* 42](#_Toc40463040)

[3.3 Тестові дані 43](#_Toc40463041)

[*3.3.1 Діаграма об’єктів* 43](#_Toc40463042)

[*3.3.2 Вміст файлу з даними* 43](#_Toc40463043)

[3.4 Поетапні результати роботи 44](#_Toc40463044)

[3.5 Модульне тестування ПД за допомоги тестової платформи Boost::Test 56](#_Toc40463045)

[**Висновки** 57](#_Toc40463046)

[**Список використаних джерел** 58](#_Toc40463047)

[**Додаток Б ДІАГРАМА ПОСЛІДОВНОСТЕЙ** 60](#_Toc40463048)

[**Додаток В ДІАГРАМА ОБ’ЄКТІВ** 61](#_Toc40463049)

[**Додаток Г КОД МОДУЛЬНИХ ТЕСТІВ** 62](#_Toc40463050)

[**Додаток Д КОД ПД** 66](#_Toc40463051)

[Workshop.h 66](#_Toc40463052)

[Workshop.cpp 66](#_Toc40463053)

[Sewing\_workshop.h 67](#_Toc40463054)

[Sewing\_workshop.cpp 68](#_Toc40463055)

[Furniture\_workshop.h 69](#_Toc40463056)

[Furniture\_workshop.cpp 70](#_Toc40463057)

[Container.h 71](#_Toc40463058)

[Container.cpp 72](#_Toc40463059)

[Menu.h 78](#_Toc40463060)

[Menu.cpp 79](#_Toc40463061)

[Main.cpp 84](#_Toc40463062)

# **ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ**

## Скорочення:

КР – курсова робота

ПД – програмний додаток

Рис. – рисунок

Табл. – таблиця

UML – (*Unified Modeling Language*) — уніфікована мова моделювання

## Терміни:

ООП (об'єктно-орієнтоване програмування) – це парадигма розробки програмних систем, в якій додатки складаються з об'єктів.

Клас – деяка конструкція для групування пов’язаних змінних та функцій.

Поля – змінні в деякому класі, де зберігаються дані.

Методи - спеціальні блоки коду, які можна викликати з різних частин програми, які забезпечують функціонал класів.

Об'єкти - це змінні класу, у яких є властивості цього класу.

Успадкування – процес створення нових класів, які називаються похідними класами і створені з вже існуючих (базових) класів.

Singleton (одинак) – це твірний патерн, який гарантує існування тільки одного об'єкта певного класу, а також дозволяє отримати доступ до цього об'єкта з будь-якого місця програми.

# **ВСТУП**

Метою курсової роботи являється закріплення теорії та набуття практичних навичок з дисципліни «Об’єктно-орієнтоване програмування» у створенні ПД на високорівневій мові програмування C++ у парадигмі ООП. Також метою КР є засвоєння знань аналізу та проектування програми на UML, поглиблення знань мови C++ та її принципів роботи, як мови ООП та на практиці навчитись особливостям написання ПД в інтегрованому середовищі розробки Microsoft Visual Studio.

Основною задачею курсової роботи полягає у підготовці до самостійної практичної діяльності з виконання всього комплексу задач розробки сучасного ПД.

## Основні завдання:

1. Визначення поставленої індивідуальної задачі
2. Проектування майбутнього ПД на UML
3. Реалізація ПД
   1. Створення абстрактного класу
   2. Створення класів-нащадків
   3. Створення класу меню користувача
   4. Створення класу контейнера з відповідними методами обробки даних
   5. Організація роботи ПД з даними через файл
   6. Об’єднання усіх елементів ПД
4. Тестування ПД

# **ПЕРШИЙ РОЗДІЛ. СПЕЦИФІКАЦІЯ ВИМОГ**

## Функціональні вимоги

### *Створення об’єкту «цех пошиття одягу» та додавання його у контейнер*

Описання варіанту виконання функціональної вимоги «Створення об’єкту «цех пошиття одягу» та додавання його у контейнер» наведено у Табл. 1.1, UML діаграма варіанту використання наведена на Рис. 1.1.

Таблиця 1.1

*Створення об’єкту «цех пошиття одягу» та додавання у контейнер*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Назва | Мета | Дійові  особи | Тип варіанта використання | Опис |
| F\_01 | Створення «цеху пошиття одягу» та додавання у контейнер | Створення об’єкту «цех пошиття одягу» та додавання у контейнер | Користувач | Основний | Заповнення усіх полів «цеху пошиття одягу», створення об’єкту та додавання цього об’єкту у контейнер |



Рис. 1.1 Створення об’єкту «цех пошиття одягу» та додавання у контейнер

### *Створення об’єкту «меблевий цех» та додавання його у контейнер*

Описання варіанту виконання функціональної вимоги «Створення об’єкту «меблевий цех» та додавання його у контейнер» наведено у Табл. 1.2, UML діаграма варіанту використання наведена на Рис. 1.2.

Таблиця 1.2

*Створення об’єкту «цех пошиття одягу» та додавання у контейнер*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Назва | Мета | Дійові  особи | Тип варіанта використання | Опис |
| F\_02 | Створення «меблевого цеху» та додавання у контейнер | Створення об’єкту «меблевий цех» та додавання у контейнер | Користувач | Основний | Заповнення усіх полів «меблевого цеху», створення об’єкту та додавання цього об’єкту у контейнер |



Рис. 1.2 Створення об’єкту «меблевий цех» та додавання у контейнер

### *Переглянути усі цеха в контейнері*

Описання варіанту виконання функціональної вимоги «Переглянути усі цеха в контейнері» наведено у Табл. 1.3, UML діаграма варіанту використання наведена на Рис. 1.3.

Таблиця 1.3

*Переглянути усі цеха в контейнері*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Назва | Мета | Дійові  особи | Тип варіанта використання | Опис |
| F\_03 | Переглянути усі цеха | Вивід усіх цехів для перегляду | Користувач | Основний | Вивід усіх цехів у контейнері на екран |



Рис. 1.3 Переглянути усі цеха в контейнері

### *Видалення цеху за його назвою*

Описання варіанту виконання функціональної вимоги «Видалення цеху за його назвою» наведено у Табл. 1.4, UML діаграма варіанту використання наведена на Рис. 1.4.

Таблиця 1.4

*Видалення цеху за його назвою*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Назва | Мета | Дійові  особи | Тип варіанта використання | Опис |
| F\_04 | Видалення цеху | Видалення цеху за його назвою | Користувач | Основний | Видалення цеху за його назвою з контейнера |



Рис. 1.4 Видалення цеху за його назвою

### *Збереження даних у файлі*

Описання варіанту виконання функціональної вимоги «Збереження даних у файлі» наведено у Табл. 1.5, UML діаграма варіанту використання наведена на Рис. 1.5.

Таблиця 1.5

*Збереження даних у файл*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Назва | Мета | Дійові  особи | Тип варіанта використання | Опис |
| F\_05 | Збереження даних | Збереження даних про цеха у файлі | Користувач | Основний | Запис усіх даних у контейнері про цеха у файл «Workshops\_info.dat» |



Рис. 1.5 Збереження даних у файлі

### *Зчитування даних з файлу*

Описання варіанту виконання функціональної вимоги «Зчитування даних з файлу» наведено у Табл. 1.6, UML діаграма варіанту використання наведена на Рис. 1.6.

Таблиця 1.6

*Зчитування даних з файлу*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Назва | Мета | Дійові  особи | Тип варіанта використання | Опис |
| F\_06 | Зчитування даних | Зчитування даних про цеха з файлу | Користувач | Основний | Зчитування даних з файлу «Workshops\_info.dat» та додавання у контейнер |



Рис. 1.6 Зчитування даних з файлу

### *Сортування елементів контейнера*

Описання варіанту виконання функціональної вимоги «Сортування елементів контейнера» наведено у Табл. 1.7, UML діаграма варіанту використання наведена на Рис. 1.7.

Таблиця 1.7

*Сортування елементів контейнера*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Назва | Мета | Дійові  особи | Тип варіанта використання | Опис |
| F\_07 | Сортування | Сортування елементів контейнера | Користувач | Основний | Сортування цехів у контейнері за їх назвами |



Рис. 1.7 Сортування елементів контейнера

### *Сумарна потужність усіх верстатів в цехах заданого типу*

Описання варіанту виконання функціональної вимоги «Сумарна потужність усіх верстатів в цехах заданого типу» наведено у Табл. 1.8, UML діаграма варіанту використання наведена на Рис. 1.8.

Таблиця 1.8

*Сумарна потужність усіх верстатів в цехах заданого типу*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Назва | Мета | Дійові  особи | Тип варіанта використання | Опис |
| F\_08 | Потужність верстатів в цехах | Підрахувати потужність усіх верстатів в цехах за їх типом | Користувач | Основний | Підраховується сумарна потужність усіх верстатів (середня потужність верстата множиться на кількість верстатів на цеху) в цехах заданого типу у контейнері |



Рис. 1.8 Сумарна потужність усіх верстатів в цехах заданого типу

### *Видалення усіх елементів з контейнера*

Описання варіанту виконання функціональної вимоги «Видалення усіх елементів з контейнера» наведено у Табл. 1.9, UML діаграма варіанту використання наведена на Рис. 1.9.

Таблиця 1.9

*Видалення усіх елементів з контейнера*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Назва | Мета | Дійові  особи | Тип варіанта використання | Опис |
| F\_09 | Видалення усіх елементів | Видалення усіх елементів з контейнера | Користувач | Основний | Почергове видалення усіх цехів з контейнера |



Рис. 1.9 Видалення усіх елементів з контейнера

### *Вихід із ПД*

Описання варіанту виконання функціональної вимоги «Вихід із ПД» наведено у Табл. 1.10, UML діаграма варіанту використання наведена на Рис. 1.10.

Таблиця 1.10

*Вихід із ПД*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Назва | Мета | Дійові  особи | Тип варіанта використання | Опис |
| F\_10 | Вихід | Вихід із ПД | Користувач | Основний | Видалення усіх цехів з контейнера, видалення контейнера, завершення програми |



Рис. 1.10 Вихід із ПД

## Нефункціональні вимоги

ID: F\_11 – передбачене повторне створення меню, що являється не потрібною витратою пам’яті обчислювального пристрою. Клас меню створений за шаблоном ООП – Singleton, який не дозволяє створювати більше одного об’єкту меню.

ID: F\_12 – передбачене введення користувачем не відповідних запиту програми символів, цифр та букв при роботі з меню. Поки користувач не введе відповідні дані з клавіатури, ПД не буде продовжувати роботу. З’явиться відповідне повідомлення користувачу.

ID: F\_13 – передбачене збереження даних у файл, коли контейнер пустий. ПД в такому випадку не буде записувати у файл нічого. З’явиться відповідне повідомлення користувачу.

ID: F\_14 – передбачене видалення цеху за назвою, якої не має в контейнері. В такому випадку ПД не буде видаляти нічого. З’явиться відповідне повідомлення користувачу.

ID: F\_15 – передбачене видалення елементу, коли контейнер вже пустий. В такому випадку ПД виведе відповідне повідомлення користувачу.

ID: F\_16 – передбачене виведення на екран всіх елементів, коли контейнер пустий. В такому випадку ПД виведе відповідне повідомлення користувачу.

ID: F\_17 – передбачене сортування елементів, коли контейнер пустий. В такому випадку ПД виведе відповідне повідомлення користувачу.

ID: F\_18 – передбачений підрахунок потужності верстатів у цехах за їх типом, коли контейнер пустий. В такому випадку ПД виведе відповідне повідомлення користувачу.

ID: F\_19 – передбачений підрахунок потужності верстатів у цехах за їх типом, коли у контейнері немає введеного користувачем типу. В такому випадку ПД виведе відповідне повідомлення користувачу.

# **РОЗДІЛ 2. СТРУКТЕРА ТО ЛОГІКА РОБОТИ ПД**

## 2.1 Логічна структура ПД

Діаграма класів у нотації UML наведена окремо (Додаток А).

### *2.1.1 Клас «Workshop»*

Клас «Workshop» - абстрактний клас, необхідний для узагальненої інформації щодо цеху. Опис приватного інтерфейсу класу «Workshop» наведено в Табл. 2.1, доступного – в Табл. 2.2.

Таблиця 2.1

*Поля приватного інтерфейсу класу «Workshop»*

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва поля | Тип | Призначення |
| 1 | workshop\_title | string | Назва цеху |
| 2 | num\_of\_workbench | unsigned short | Кількість верстатів у цеху |
| 3 | average\_capacity | int | Середня потужність верстата у цеху |

Таблиця 2.2

*Поля доступного інтерфейсу класу «Workshop»*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Сигнатура | Вхідні параметри | Вихідні дані | Призначення |
| 1 | Workshop() | - | - | Конструктор без параметрів |
| 2 | Workshop(string title, unsigned short workbench, int capacity) | Назва цеху (workshop\_title), кількість верстатів (num\_of\_workbench), середня потужність (average\_capacity) | - | Конструктор, заповнює параметрами поля об’єкта |
| 3 | virtual ~Workshop() | - | - | Віртуальний деструктор, використовується у нащадках |
| 4 | string Get\_title() const | - | Назва цеху (workshop\_title) | Повертає назву цеху не змінюючи поля |
| 5 | unsigned short Get\_workbench() const | - | Кількість верстатів (num\_of\_workbench) | Повертає кількість верстатів не змінюючи поля |
| 6 | int Get\_capacity() const | - | Середня потужність (average\_capacity) | Повертає середню потужність верстатів не змінюючи поля |
| 7 | void Set\_title(string new\_title) | Нове значення назви цеху (workshop\_title) | - | Змінює поле назви цеху |
| 8 | void Set\_workbench(unsigned short new\_workbench) | Нове значення кількості верстатів (num\_of\_workbench) | - | Змінює поле кількості верстатів |
| 9 | void Set\_capacity(int new\_capacity) | Нове значення середньої потужності (average\_capacity) | - | Змінює поле середньої потужності верстатів |
| 10 | virtual void Print\_workshop() = 0 | - | - | Віртуальна функція виводу інформації щодо цеху, використовується лише нащадками |

### *2.1.2 Клас «Sewing\_workshop»*

Клас «Sewing\_workshop» - похідний клас від «Workshop», необхідний для визначення сутності цеху пошиття одягу. Опис приватного інтерфейсу класу «Sewing\_workshop» наведено в Табл. 2.3, доступного – в Табл. 2.4.

Таблиця 2.3

*Поля приватного інтерфейсу класу «Sewing\_workshop»*

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва поля | Тип | Призначення |
| 1 | fabric\_type | string | Тип матеріалу який цех виготовляє |
| 2 | rolls\_of\_fabric | unsigned | Кількість рулонів матеріалу на місяць |

Таблиця 2.4

*Поля відкритого інтерфейсу класу «Sewing\_workshop»*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Сигнатура | Вхідні параметри | Вихідні дані | Призначення |
| 1 | Sewing\_workshop() | - | - | Конструктор без параметрів |
| 2 | Sewing\_workshop(string title, unsigned short workbench, int capacity, string fabric, unsigned rolls) | Назва цеху (workshop\_title), кількість верстатів (num\_of\_workbench), середня потужність (average\_capacity), тип матеріалу (fabric\_type), кількість рулонів (rolls\_of\_fabric) | - | Конструктор, заповнює параметрами поля об’єкта |
| 3 | ~ Sewing\_workshop () | - | - | Деструктор |
| 4 | string Get\_fabric() const | - | Тип матеріалу (fabric\_type) | Повертає тип матеріалу не змінюючи поля |
| 5 | unsigned Get\_rolls() const | - | Кількість рулонів (rolls\_of\_fabric) | Повертає кількість рулонів не змінюючи поля |
| 6 | void Set\_fabric(string new\_fabric) | Нове значення типу матеріалу (fabric\_type) | - | Змінює поле типу матеріалу |
| 7 | void Set\_rolls(unsigned new\_rolls) | Нове значення кількості рулонів (rolls\_of\_fabric) | - | Змінює поле кількості рулонів |
| 8 | friend istream& operator >> (istream& is, Sewing\_workshop& sew\_is) | Адреса потоку (is), адреса переданого об’єкта класу цеху пошиття одягу (sew\_is) | Потік введення даних | Перевантажений оператор введення потоку даних необхідний для зчитування файлу |
| 9 | friend ostream& operator << (ostream& os, Sewing\_workshop& sew\_os) | Адреса потоку (os), класу цеху пошиття одягу (sew\_os) | Потік виведення даних | Перевантажений оператор виведення потоку даних необхідний для запису даних у файл |
| 10 | void Print\_workshop() | - | - | Функція виводу інформації щодо цеху пошиття одягу |

### *2.1.3 Клас «Furniture\_workshop»*

Клас «Furniture\_workshop» - похідний клас від «Workshop», необхідний для визначення сутності меблевого цеху. Опис приватного інтерфейсу класу «Furniture\_workshop» наведено в Табл. 2.5, доступного – в Табл. 2.6.

Таблиця 2.5

*Поля приватного інтерфейсу класу «Furniture\_workshop»*

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва поля | Тип | Призначення |
| 1 | wood\_type | string | Тип дерева який цех обробляє |
| 2 | amount\_of\_wood | unsigned | Кількість дерева на місяць |

Таблиця 2.6

*Поля відкритого інтерфейсу класу «Furniture\_workshop»*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Сигнатура | Вхідні параметри | Вихідні дані | Призначення |
| 1 | Furniture\_workshop () | - | - | Конструктор без параметрів |
| 2 | Furniture\_workshop(string title, unsigned short workbench, int capacity, string wood, unsigned amount) | Назва цеху (workshop\_title), кількість верстатів (num\_of\_workbench), середня потужність (average\_capacity), тип дерева (wood\_type), кількість дерева (amount\_of\_wood) | - | Конструктор, заповнює параметрами поля об’єкта |
| 3 | ~Furniture\_workshop () | - | - | Деструктор |
| 4 | string Get\_wood() const | - | Тип дерева (wood\_type) | Повертає тип дерева не змінюючи поля |
| 5 | unsigned Get\_amount() const | - | Кількість дерева (amount\_of\_wood) | Повертає кількість рулонів не змінюючи поля |
| 6 | void Set\_wood(string new\_wood) | Нове значення типу дерева (wood\_type) | - | Змінює поле типу дерева |
| 7 | void Set\_amount(unsigned new\_amount) | Нове значення кількості дерева (amount\_of\_wood) | - | Змінює поле кількості дерева |
| 8 | friend istream& operator >> (istream& is, Furniture\_workshop& sew\_is) | Адреса потоку (is), адреса переданого об’єкта класу меблевого цеху (sew\_is) | Потік введення даних | Перевантажений оператор введення потоку даних необхідний для зчитування файлу |
| 9 | friend ostream& operator << (ostream& os, Furniture\_workshop& sew\_os) | Адреса потоку (os), класу меблевого цеху (sew\_os) | Потік виведення даних | Перевантажений оператор виведення потоку даних необхідний для запису даних у файл |
| 10 | void Print\_workshop() | - | - | Функція виводу інформації щодо меблевого цеху |

### *2.1.3 Структура «Node»*

Структура «Node» - структура, яка необхідна для простої роботи вузла двозв’язного кільцевого списку та збереження даних цеху. Опис інтерфейсу структури «Node» наведено в Табл. 2.7

Таблиця 2.7

*Поля інтерфейсу структури «*Node»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва поля | Тип | Призначення |
| 1 | data | Workshop\* | Посилання на дані об’єкту класу Workshop |
| 2 | next\_node | Node\* | Посилання на наступний вузол списку |
| 3 | prev\_node | Node\* | Посилання на наступний вузол списку |

### *2.1.4 Клас «Сontainer»*

Клас «Сontainer» - клас-реалізація двозв’язного кільцевого списку, як контейнера для збереження та роботи з даними класів «Sewing\_workshop» та «Furniture\_workshop». Опис приватного інтерфейсу класу «Сontainer» наведено в Табл. 2.8, доступного – в Табл. 2.9.

Таблиця 2.8

*Поля приватного інтерфейсу класу «Сontainer»*

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва поля | Тип | Призначення |
| 1 | current\_node | Node\* | Посилання на поточний вузол |
| 2 | start\_node | Node\* | Посилання на стартовий вузол списку |

Таблиця 2.9

*Поля відкритого інтерфейсу класу «Сontainer»*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Сигнатура | Вхідні параметри | Вихідні дані | Призначення |
| 1 | Сontainer() | - | - | Конструктор без параметрів |
| 2 | ~Сontainer() | - | - | Деструктор, видаляє всі об’єкти списку |
| 3 | void Push(Workshop\* new\_item) | Посилання на новий цех (Workshop\*) | - | Додає у список новий цех |
| 4 | void Pop() | - | - | Видаляє вузол з контейнера |
| 5 | unsigned Count\_elements() const | - | Кількість елементів списку | Повертає кількість елементів списку |
| 6 | bool Print() const | - | Бульова змінна, що показує наявність хоча б одного елемента в контейнері | Виводить на екран дані усіх цехів в контейнері |
| 7 | bool Delete\_element(string title\_to\_delete) | Назва цеху, який необхідно видалити | Бульова змінна, що показує наявність хоча б одного елемента в контейнері | Видаляє елемент – цех, за назвою за допомогою функції Pop() |
| 8 | bool Save() const | - | Бульова змінна, що показує наявність хоча б одного елемента в контейнері | Збереження даних про цеха у файл |
| 9 | void Load() | - | - | Зчитування файлу та додавання елементів у контейнер |
| 10 | bool Sort\_elements\_by\_title() | - | Бульова змінна, що показує наявність хоча б одного елемента в контейнері | Сортує цеха у списку за назвою у алфавітному порядку |
| 11 | bool Count\_capacity() const | - | Бульова змінна, що показує наявність хоча б одного елемента в контейнері | Рахує сумарну потужність верстатів усіх цехів за їх типом |
| 12 | bool Delete\_all() | - | Бульова змінна, що показує наявність хоча б одного елемента в контейнері | Видаляє усі елементи з контейнера, залишаючи його пустим |

### *2.1.5 Клас «Menu»*

Клас «Menu» - клас-реалізація меню, для роботи інтерфейсу ПД та взаємодії користувача з ним. Опис приватного інтерфейсу класу «Menu» наведено в Табл. 2.10, доступного – в Табл. 2.11.

Таблиця 2.10

*Поля приватного інтерфейсу класу «Menu»*

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва поля/методу | Тип | Призначення |
| 1 | Menu() {} | [метод] | Так як клас Меню виконаний за шаблоном Singleton, конструктор класу повинен бути приватним, для запобігання створення більш ніж одного об’єкта |
| 2 | single\_menu | static Menu\* | Статичне поле, визначене як глобальна змінна, необхідна для коректної роботи меню за шаблоном Singleton |
| 3 | all\_workshops | Сontainer | Поле контейнера, в якому обробляються дані цехів |
| 4 | choice | int | Поточний вибір дії користувача |

Таблиця 2.11

*Поля відкритого інтерфейсу класу «Menu»*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Сигнатура | Вхідні параметри | Вихідні дані | Призначення |
| 1 | Menu(Menu& other) = delete | Передбачувані нові об’єкти меню | - | Конструктор, що не дозволяє створювати об’єкти з параметром класу меню |
| 2 | void operator = (const Menu&) = delete | Константна адреса класу меню | - | Перевантажений оператор присвоєння, що не дозволяє копіювати об’єкт класу меню |
| 3 | static Menu\* GetInstance() | - | Адреса об’єкту класу меню | Статичний метод, що керує доступом до об’єкту класу меню. При першому запуску, він створює меню і поміщає його в статичне поле. При наступних запусках, він повертає користувачу об'єкт, що зберігається в статичному полі. |
| 4 | void Start() | - | - | Доступ до функцій головного меню ПД |
| 5 | void Create\_sewing\_workshop (Сontainer\* Data) | Дані про цех пошиття одягу (class Sewing workshop) | - | Користувач вводить дані з клавіатури про цех пошиття одягу, створюється відповідний об’єкт класу та додається у контейнер |
| 6 | void Create\_furniture\_workshop (Сontainer\* Data) | Дані про меблевий цех (class Furniture workshop) | - | Користувач вводить дані з клавіатури про меблевий цех, створюється відповідний об’єкт класу та додається у контейнер |
| 7 | void Show() const | - | - | Виводить усі цеха на екран, не змінюючи даних за допомогою Print() |
| 8 | void Delete() | - | - | Видаляє цех за назвою, якщо така є в контейнері за допомогою Delete\_element() |
| 9 | void Save() const | - | - | Зберігає усі цеха, якщо вони є у контейнері, у файл, не змінюючи даних за допомогою Container::Save() |
| 10 | void Load() | - | - | Додає у контейнер цеха, попередньо зчитавши дані про них з файлу за допомогою Container::Load() |
| 11 | void Sort() | - | - | Сортує усі цеха за їх назвою, якщо вони є у контейнері за допомогою Sort\_elements\_by\_title() |
| 12 | void Capacity() const | - | - | Обчислює сумарну потужність всіх верстатів в цехах заданого типу, якщо такий тип існує не змінюючи даних за допомогою Count\_capacity(), не змінюючи |
| 13 | void Full\_delete() | - | - | Видаляє усі елементи з контейнера |
| 14 | void Exit() |  | - | Видаляє усі елементи контейнера та виконує завершення ПД |

## 2.2 Фізична структура

Розподілення класів, змінних, функцій та зовнішніх бібліотек, які підключаються по файлах наведено в Табл. 2.12.

Таблиця 2.12

*Розподілення фізичних структур по файлах*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Назва файлу | Компоненти | | Зовнішні компоненти | |
| Назва | Опис | Назва | Опис |
| 1 | Main.cpp | main | Точка початку роботи ПД, запуск меню за шаблоном Singleton | Menu.h | Меню користувача |
| 2 | Menu.h | Menu | Визначення сутності класу меню | Сontainer.h | Робота з двозв’язним кільцевим списком |
| 3 | Menu.cpp | GetInstance | Функції реалізації пунктів меню ПД | Menu.h | Клас меню користувача |
| Start |
| Create\_sewing\_ workshop |
| Create\_furniture\_ workshop |
| Show |
| Delete |
| Save |
| Load |
| Sort |
| Capacity |
| Full\_delete |
| Exit |
| 4 | Container.h | Node | Структура вузла двозв’язноко кільцевого списку з даними цеху | Sewing\_ workshop.h | Клас цеху пошиття одягу |
| Furniture\_ workshop.h | Клас меблевого цеху |
| Сontainer | Визначення класу сутності двозв’язноко кільцевого списку | fstream | Файл роботи з файловим вводом та виводом |
| 5 | Container. cpp | Сontainer | Функції реалізації контейнеру для даних всіх цехів | Сontainer.h | Клас двозв’язного кільцевого списку |
| ~Сontainer |
| Push |
| Pop |
| Count\_elements |
| Print |
| Delete\_element |
| Save |
| Load |
| Sort\_elements\_ by\_title |
| Count\_capacity |
| Delete\_all |
| 6 | Sewing\_ workshop.h | Sewing\_ workshop | Визначення класу сутності цеху пошиття одягу | Workshop.h | Абстрактний клас цеху |
| 7 | Sewing\_ workshop. cpp | Sewing\_ workshop | Функції реалізації сутності цеху пошиття одягу | Sewing\_workshop.h | Клас цеху пошиття одягу |
| Sewing\_ workshop з параметрами |
| ~Sewing\_ workshop |
| Get\_fabric |
| Get\_rolls |
| Set\_fabric |
| Set\_rolls |
| operator >> |
| operator << |
| Print\_workshop |
| 8 | Furniture\_workshop.h | Furniture\_ workshop | Визначення класу сутності меблевого цеху | Workshop.h | Абстрактний клас цеху |
| 9 | Furniture\_workshop. cpp | Furniture\_ workshop | Функції реалізації сутності меблевого цеху | Furniture\_ workshop.h | Клас меблевого цеху |
| Furniture\_ workshop з параметрами |
| ~Furniture\_ workshop |
| Get\_wood |
| Get\_amount |
| Set\_wood |
| Set\_amount |
| operator >> |
| operator << |
| Print\_workshop |
| 10 | Workshop.h | Workshop | Визначення класу сутності абстрактного цеху | string | Файл для роботи з рядками |
| iostream | Файл для роботи з потоками |
| 11 | Workshop. cpp | Workshop | Функції реалізації сутності класу абстрактного цеху | Workshop.h | Абстрактний клас цеху |
| Workshop з параметрами |
| ~Workshop |
| Get\_title |
| Get\_workbench |
| Get\_capacity |
| Set\_title |
| Set\_workbench |
| Set\_capacity |
| Print\_workshop |

## 2.3 Логіка роботи

Діаграма послідовності у нотації UML наведена окремо (Додаток Б).

# **РОЗДІЛ 3. ТЕСТУВАННЯ**

## 3.1 Сценарії тестування функціональних вимог

### *3.1.1 Тест «Створення об’єкту Цеху пошиття одягу»*

Сценарій тестування функціональної вимоги створення об’єкту Цех пошиття одягу наведено у Табл. 3.1, а також на Рис. 3.4.1.

Таблиця 3.1

*Створення об’єкту Цех пошиття одягу*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_01 | Cтворення об’єкту Цех пошиття одягу | F\_01 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Вибір «1» 3. Введення полів цеху | 1. Запит вибору в меню 2. Запит на введення полів 3. Додавання об’єкту в контейнер | Меню |

### *3.1.2 Тест «Створення об’єкту Меблевого цеху»*

Сценарій тестування функціональної вимоги створення об’єкту Меблевого цеху наведено у Табл. 3.2, а також на Рис. 3.4.2.

Таблиця 3.2

*Створення об’єкту Меблевого цеху*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_02 | Cтворення об’єкту Меблевого цеху | F\_02 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Вибір «2» 3. Введення полів цеху | 1. Запит вибору в меню 2. Запит на введення полів 3. Додавання об’єкту в контейнер | Меню |

### *3.1.3* *Вивід усіх цехів на екран*

Сценарій тестування функціональної вимоги виводу усіх цехів на екран наведено у Табл. 3.3, а також на Рис. 3.4.3.

Таблиця 3.3

*Вивід усіх цехів на екран*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_03 | Вивід цехів | F\_03 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Вибір «3» 3. Виведення цехів на екран | 1. Запит вибору в меню 2. Вивід цехів, якщо контейнер не пустий | Меню |

### *Видалення цеху з контейнера*

Сценарій тестування функціональної вимоги видалення цеху з контейнера наведено у Табл. 3.4, а також на Рис. 3.4.4.

Таблиця 3.4

*Видалення цеху з контейнера*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_04 | Видалення цеху | F\_04 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Вибір «4» 3. Введення назви цеху 4. Видалення цеху | 1. Запит вибору в меню 2. Введення назви цеху для видалення 3. Видалення цеху, якщо контейнер не пустий і такий цех існує | Меню |

### *3.1.5 Збереження даних у файл*

Сценарій тестування функціональної вимоги збереження даних у файл наведено у Табл. 3.5, а також на Рис. 3.4.5.

Таблиця 3.5

*Видалення цеху з контейнера*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_05 | Збереження | F\_05 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Вибір «5» 3. Збереження цехів у файл | 1. Запит вибору в меню 2. Збереження усіх цехів, якщо контейнер не пустий | Меню |

### *3.1.6* *Введення даних з файлу*

Сценарій тестування функціональної вимоги введення даних з файлу наведено у Табл. 3.6, а також на Рис. 3.4.6.

Таблиця 3.6

*Введення даних з файлу*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_06 | Введення з файлу | F\_06 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Вибір «6» 3. Введення даних з файлу | 1. Запит вибору в меню 2. Введення даних з файлу та додавання цехів у контейнер | Меню |

### *3.1.7 Сортування цехів у контейнері*

Сценарій тестування функціональної вимоги сортування цехів у контейнері за назвою наведено у Табл. 3.7, а також на Рис. 3.4.7.

Таблиця 3.7

*Сортування цехів у контейнері*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_07 | Сортування | F\_07 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Вибір «7» 3. Сортування елементів контейнера | 1. Запит вибору в меню 2. Перевірка наявності цехів у контейнері 3. Сортування цехів за назвою | Меню |

### *3.1.8* *Сумарна потужність усіх верстатів в цехах заданого типу*

Сценарій тестування функціональної вимоги обчислення сумарної потужності усіх верстатів в цехах заданого типу наведено у Табл. 3.8, а також на Рис. 3.4.8.

Таблиця 3.8

*Сумарна потужність усіх верстатів в цехах заданого типу*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_08 | Сумарна потужність | F\_08 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Вибір «8» 3. Введення типу цеху | 1. Запит вибору в меню 2. Перевірка наявності цехів у контейнері 3. Вивід сумарної потужності усіх верстатів в цехах заданого типу | Меню |

### *3.1.9 Видалення всіх цехів з контейнера*

Сценарій тестування функціональної вимоги видалення всіх цехів з контейнера наведено у Табл. 3.9 а також на Рис. 3.4.9.

Таблиця 3.9

*Видалення всіх цехів з контейнера*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_09 | Видалення всіх цехів | F\_09 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Вибір «9» 3. Видалення всіх елементів з контейнера | 1. Запит вибору в меню 2. Перевірка наявності цехів у контейнері 3. Видалення всіх цехів з контейнера | Меню |

### *3.1.10* *Завершення роботи ПД*

Сценарій тестування функціональної вимоги завершення роботи ПД наведено у Табл. 3.10, а також на Рис. 3.4.10.

Таблиця 3.10

*Завершення роботи ПД*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_10 | Завершення | F\_10 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Вибір «0» 3. Завершення роботи ПД | 1. Запит вибору в меню 2. Видалення всіх цехів з контейнера та завершення роботи ПД | Меню |

## 3.2 Сценарії тестування нефункціональних вимог

### *3.2.1 Запобігання повторного створення меню*

Сценарій тестування нефункціональної вимоги запобігання повторного створення меню наведено у Табл. 3.11, а також на Рис. 3.4.11-13.

Таблиця 3.11

*Запобігання повторного створення меню*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_11 | Singleton меню | F\_11 | Початок роботи ПД | 1. Додати цех 2. Повторно визвати GetInstance() 3. Вивести на екран цехи 4. Перевірити співпадіння елементів | 1. Повернення початкового меню 2. Співпадіння цехів | Меню |

### *3.2.2 Некоректний вибір в меню*

Сценарій тестування нефункціональної вимоги некоректного вибору в меню наведено у Табл. 3.12, а також на Рис. 3.4.14.

Таблиця 3.12

*Некоректний вибір в меню*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_12 | Некоректне введення в меню | F\_12 | Меню ПД  Вибір користувача ПД | 1. Меню 2. Ввести не відповідний меню запит | 1. Запит вибору в меню 2. Запит на повторне введення цифри від 0 до 9 | Меню |

### *3.2.3 Збереження пустого контейнера*

Сценарій тестування нефункціональної вимоги збереження даних з пустого контейнера наведено у Табл. 3.13, а також на Рис. 3.4.15.

Таблиця 3.13

*Збереження пустого контейнера*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_13 | Збереження пустого контейнера | F\_13 | Меню ПД  Пустий контейнер  Вибір користувача ПД | 1. Меню 2. Видалити всі цеха 3. Спробувати зберегти дані | 1. Запит вибору в меню 2. Відповідне повідомлення, щодо відсутності цехів | Меню |

### *3.2.4 Видалення цеху за назвою, якої не має в контейнері*

Сценарій тестування нефункціональної вимоги видалення цеху за назвою, якої не має в контейнері наведено у Табл. 3.14, а також на Рис. 3.4.16.

Таблиця 3.14

*Видалення цеху за назвою, якої не має в контейнері*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_14 | Видалення неіснуючого цеху | F\_14 | Меню ПД  Вибір користувача ПД  Введення назви цеху, якого не має в контейнері | 1. Меню 2. Видалити цех за назвою 3. Ввести назву, якої не має в контейнері | 1. Запит вибору в меню 2. Запит на введення назви цеху 3. Відповідне повідомлення, щодо відсутності цеху | Меню |

### *3.2.5 Видалення елементу, коли контейнер вже пустий*

Сценарій тестування нефункціональної вимоги видалення цеху за назвою, якої не має в контейнері наведено у Табл. 3.15, а також на Рис. 3.4.17.

Таблиця 3.15

*Видалення елементу, коли контейнер вже пустий*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_15 | Видалення цеху при пустому контейнері | F\_15 | Меню ПД  Пустий контейнер  Вибір користувача ПД | 1. Меню 2. Видалити всі цеха 3. Спробувати видалити цех | 1. Запит вибору в меню 2. Відповідне повідомлення, щодо відсутності цехів | Меню |

### *3.2.6 Виведення елементів на екран, коли контейнер пустий*

Сценарій тестування нефункціональної вимоги виведення елементів на екран, коли контейнер пустий наведено у Табл. 3.16, а також на Рис. 3.4.18.

Таблиця 3.16

*Виведення елементів на екран, коли контейнер пустий*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_16 | Виведення цехів при пустому контейнері | F\_16 | Меню ПД  Пустий контейнер  Вибір користувача ПД | 1. Меню 2. Видалити всі цеха 3. Спробувати вивести цеха на екран | 1. Запит вибору в меню 2. Відповідне повідомлення, щодо відсутності цехів | Меню |

### *3.2.7 Сортування елементів, коли контейнер пустий*

Сценарій тестування нефункціональної вимоги сортування елементів на екран, коли контейнер пустий наведено у Табл. 3.17, а також на Рис. 3.4.19.

Таблиця 3.17

*Сортування елементів, коли контейнер пустий*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_17 | Сортування цехів при пустому контейнері | F\_17 | Меню ПД  Пустий контейнер  Вибір користувача ПД | 1. Меню 2. Видалити всі цеха 3. Спробувати вивести цеха на екран | 1. Запит вибору в меню 2. Відповідне повідомлення, щодо відсутності цехів | Меню |

### *3.2.8 Підрахунок потужності верстатів, коли контейнер пустий*

Сценарій тестування нефункціональної вимоги підрахунок потужності верстатів, коли контейнер пустий наведено у Табл. 3.18, а також на Рис. 3.4.20.

Таблиця 3.18

*Сортування елементів, коли контейнер пустий*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_18 | Підрахунок потужності при пустому контейнері | F\_18 | Меню ПД  Пустий контейнер  Вибір користувача ПД | 1. Меню 2. Видалити всі цеха 3. Спробувати підрахувати потужність контейнера | 1. Запит вибору в меню 2. Відповідне повідомлення, щодо відсутності цехів | Меню |

### *3.2.9 Підрахунок потужності верстатів у цехах заданого типу, коли такого типу в контейнері не існує*

Сценарій тестування нефункціональної вимоги підрахунок потужності верстатів у цехах заданого типу, коли такого типу в контейнері не існує наведено у Табл. 3.19, а також на Рис. 3.4.21.

Таблиця 3.19

*Підрахунок потужності верстатів у цехах заданого типу, коли такого типу в контейнері не існує*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Назва | ID вимоги | Передумова | Тестові кроки | Очікуваний результат | Післяумова |
| S\_19 | Потужність цехів за типом, якого не існує в контейнері | F\_19 | Меню ПД  Вибір користувача ПД  Введення типу цехів, яких не має в контейнері | 1. Меню 2. Потужність верстатів за типом цехів 3. Ввести тип, якого не має в контейнері | 1. Запит вибору в меню 2. Запит на введення типу цехів 3. Відповідне повідомлення, щодо відсутності цехів | Меню |

## 3.3 Тестові дані

### *3.3.1 Діаграма об’єктів*

Діаграма об’єктів у нотації UML наведена окремо (Додаток В).

### *3.3.2* *Вміст файлу з даними*

Вміст файлу з даними наведено на Рис. 3.3.1

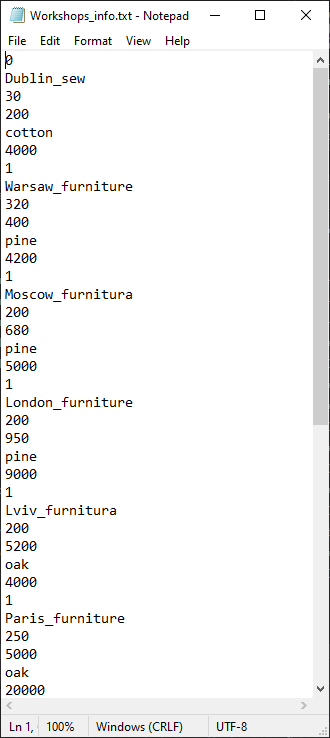
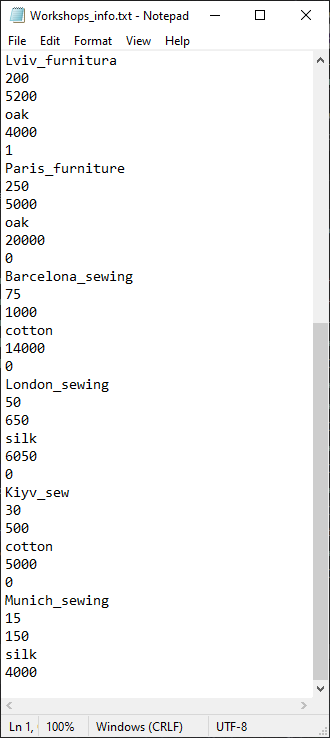


Рис. 3.3.1 Вміст файлу з даними

## 3.4 Поетапні результати роботи

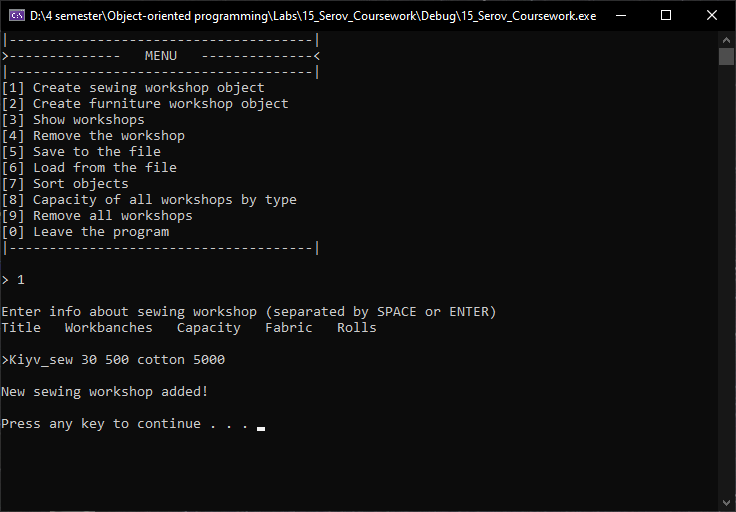


Рис. 3.4.1 Створення Цеху пошиття одягу

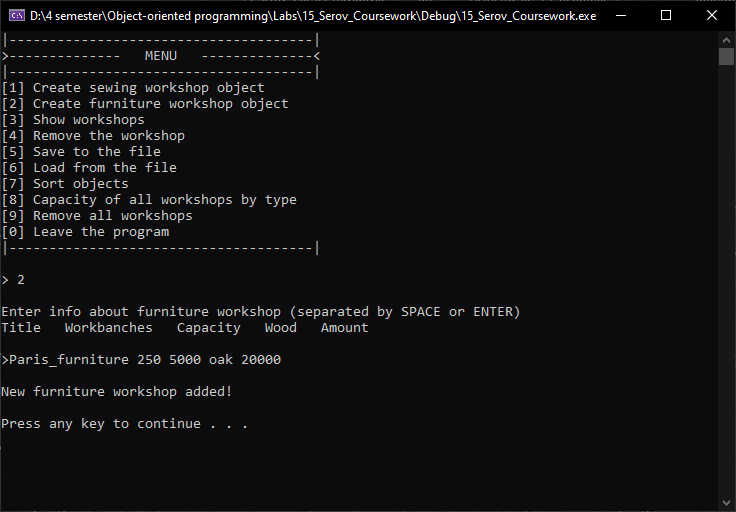
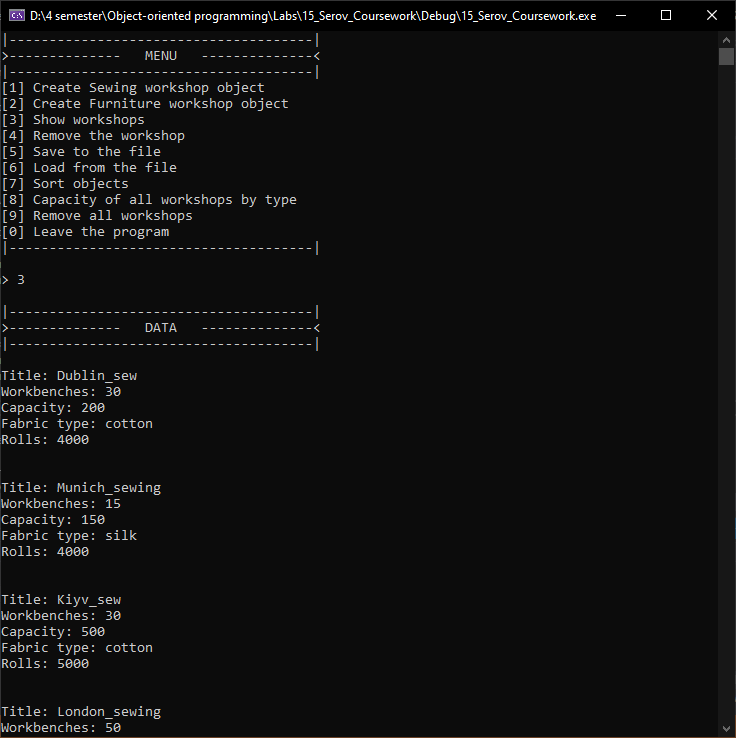
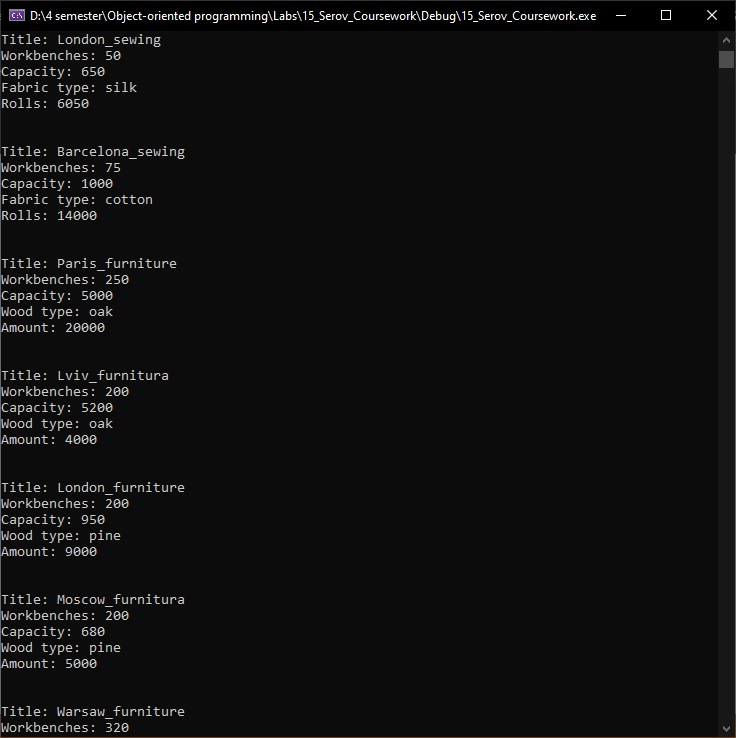


Рис. 3.4.2 Створення меблевого цеху





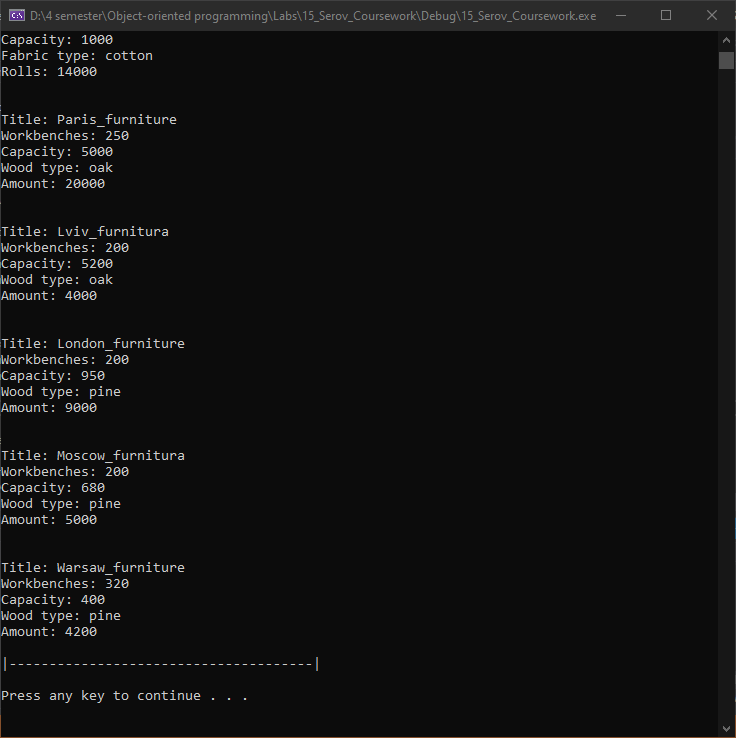


Рис. 3.4.3 Виведення цехів

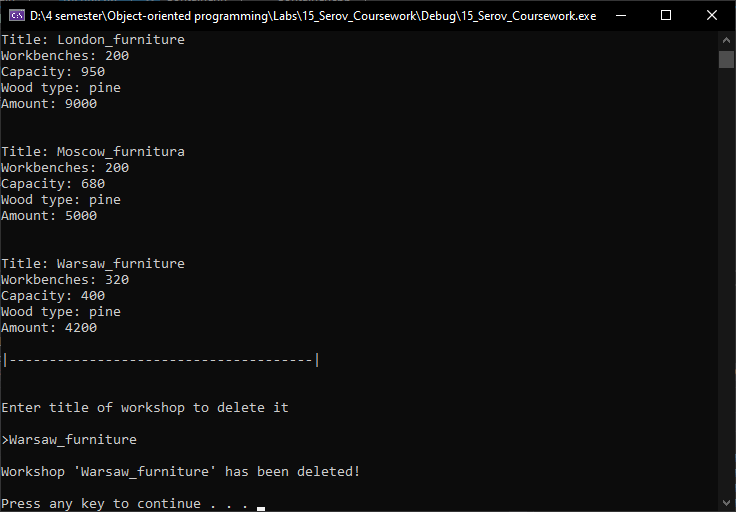


Рис. 3.4.4 Видалення цеху за назвою

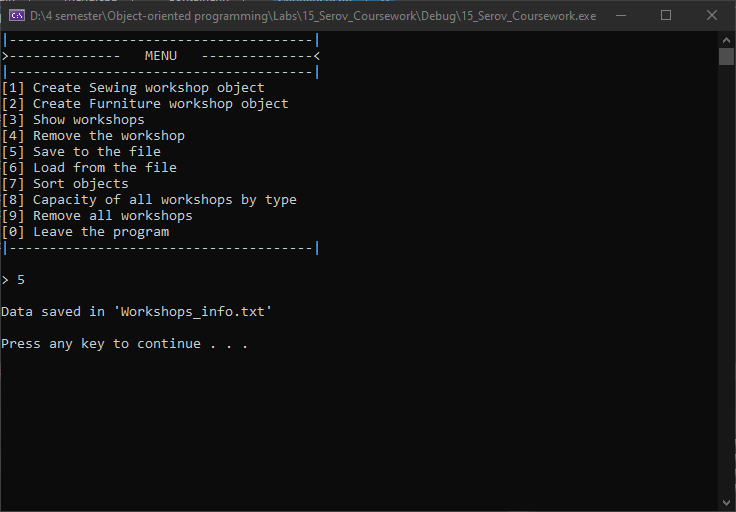


Рис. 3.4.5 Збереження даних у файл

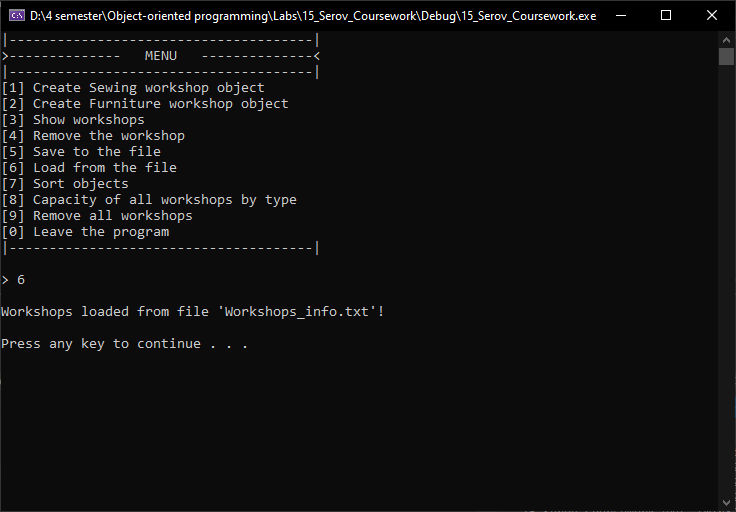


Рис. 3.4.6 Введення даних з файлу

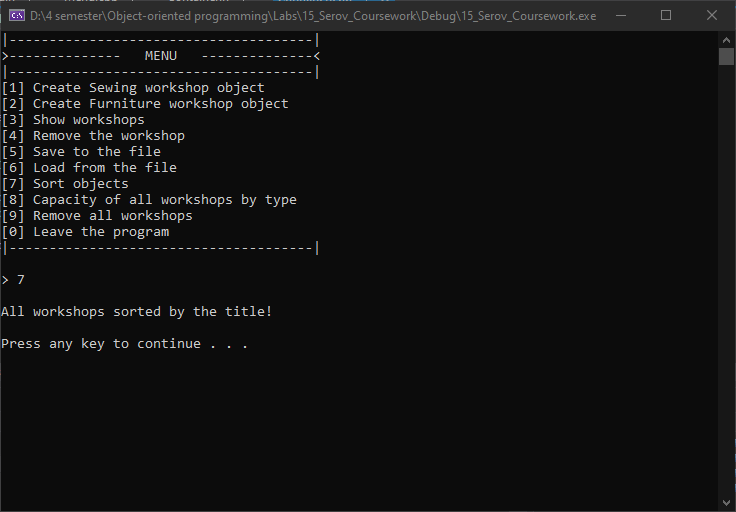


Рис. 3.4.7 Сортування цехів за назвою

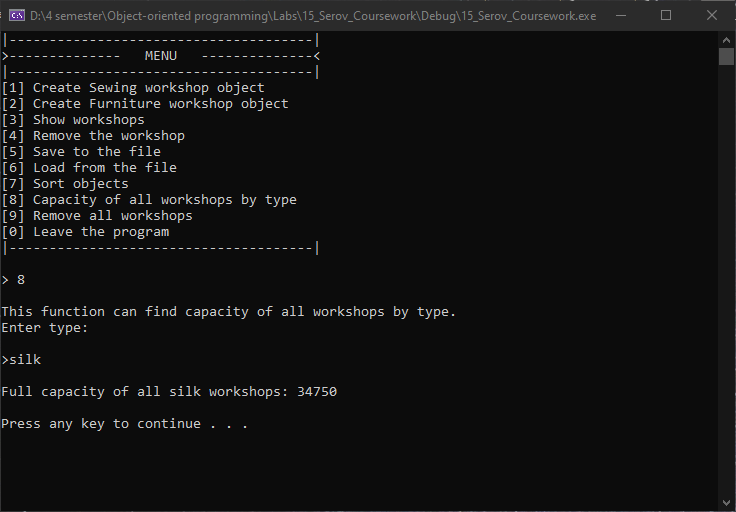


Рис. 3.4.8 Обчислення сумарної потужності усіх верстатів в цехах за типом

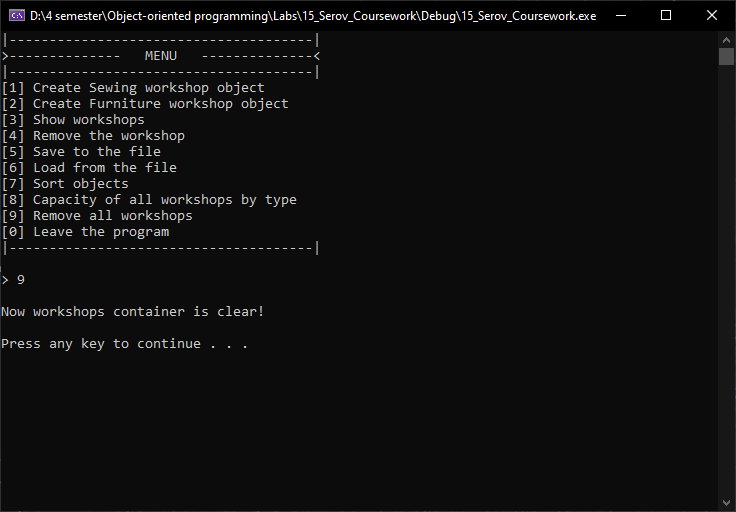


Рис. 3.4.9 Видалення всіх цехів

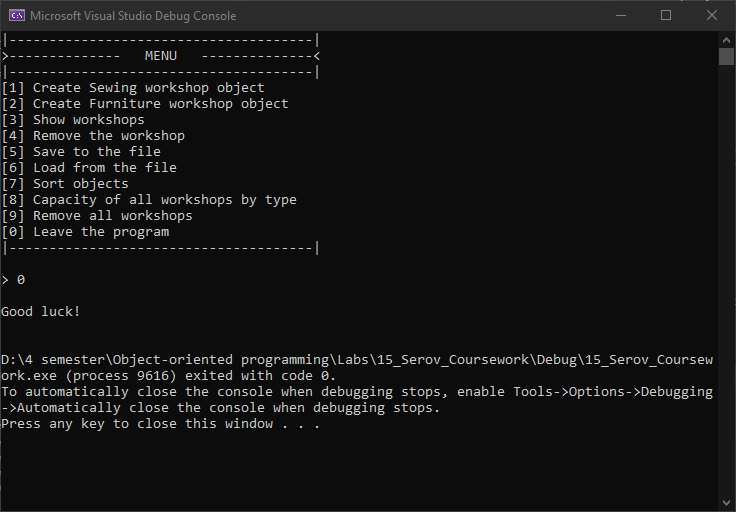


Рис. 3.4.10 Завершення роботи ПД

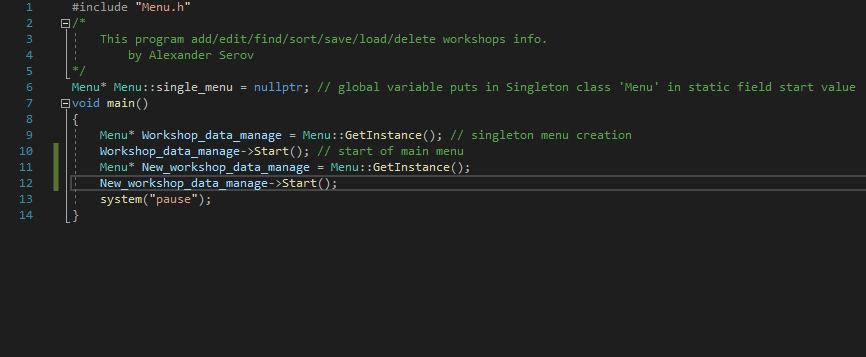


Рис. 3.4.11 Код ілюструє спробу створення другого об’єкту Singleton-класу меню

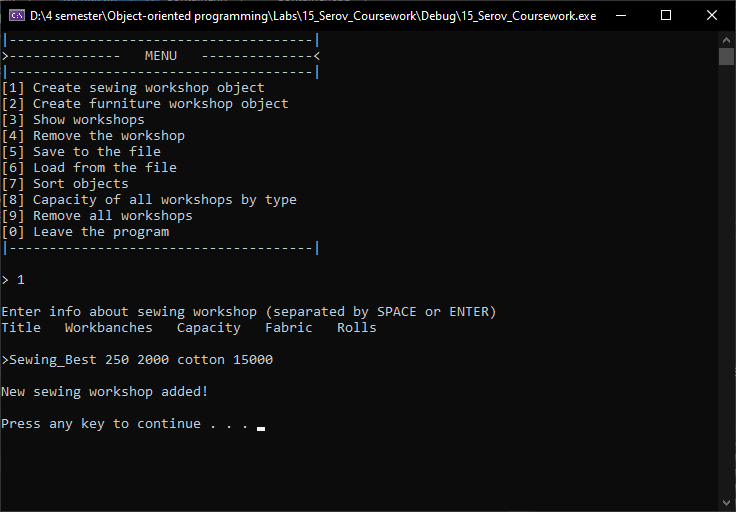


Рис. 3.4.12 Додавання цеху пошиття одягу в контейнер

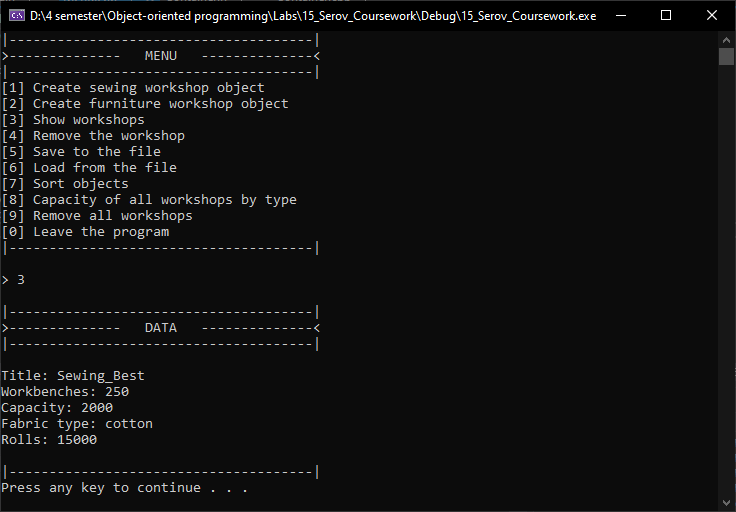


Рис. 3.4.13 Виведення тих самих даних, з «іншого» об’єкту

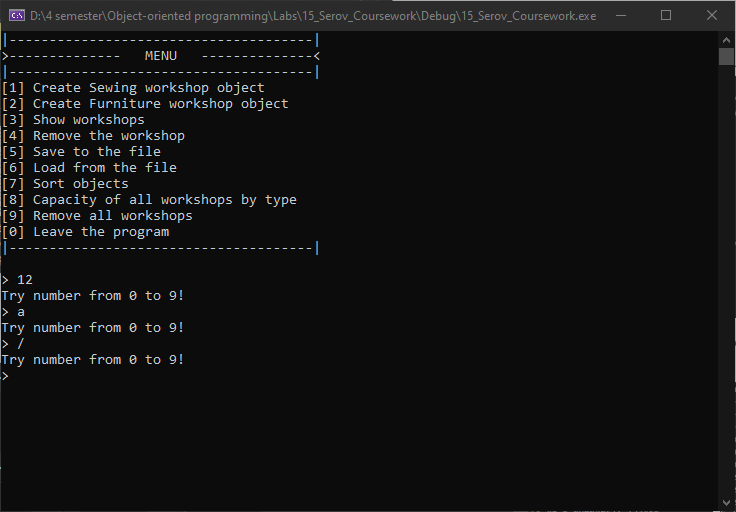


Рис. 3.4.14 Некоректне введення в меню

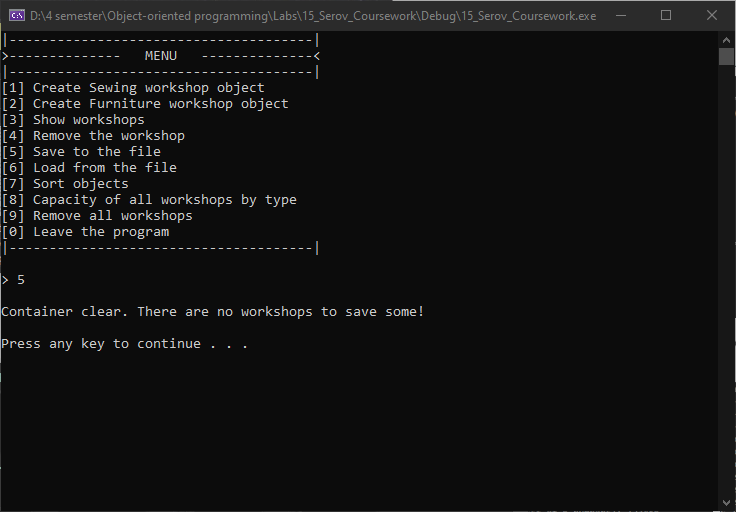


Рис. 3.4.15 Збереження файлу, коли контейнер пустий

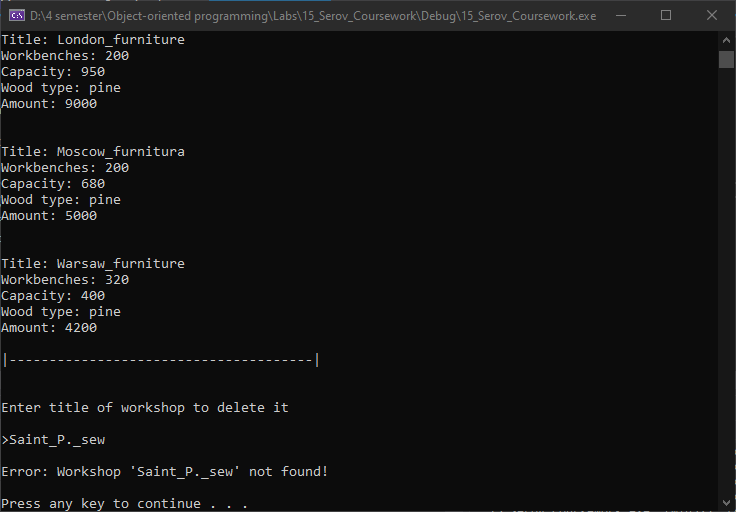


Рис. 3.4.16 Видалення цеху, якого немає в контейнері

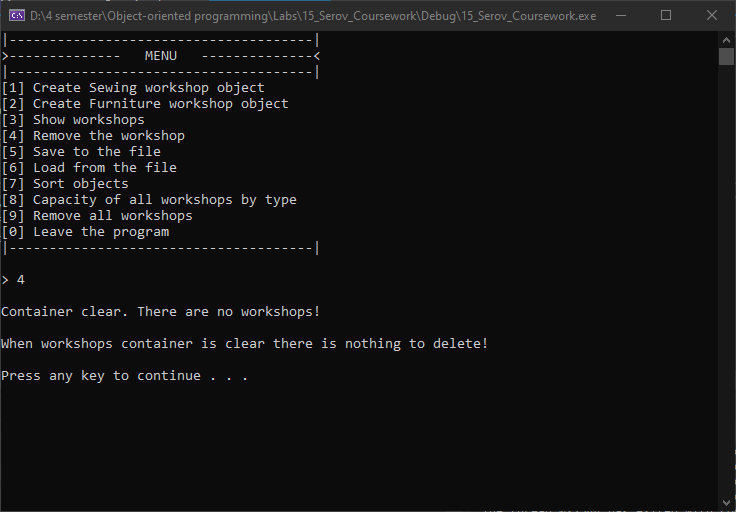


Рис. 3.4.17 Видалення цеху, коли контейнер вже пустий

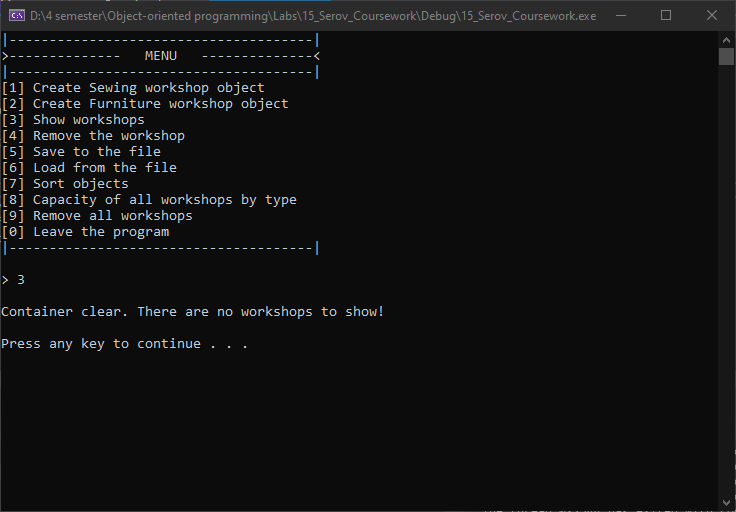


Рис. 3.4.18 Виведення цехів на екран, коли контейнер пустий

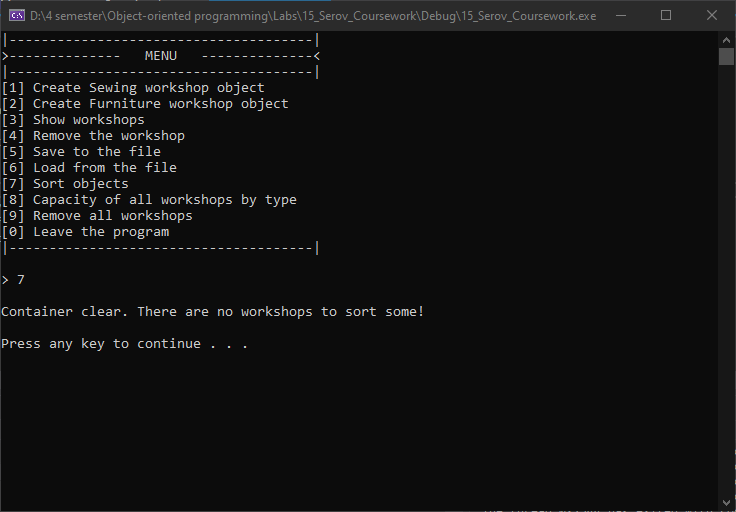


Рис. 3.4.19 Сортування елементів, коли контейнер пустий

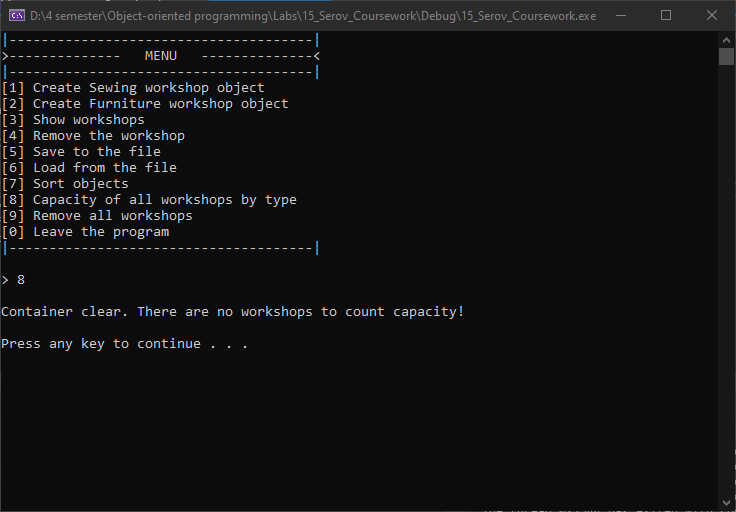


Рис. 3.4.20 Обчислення сумарної потужності, коли контейнер пустий

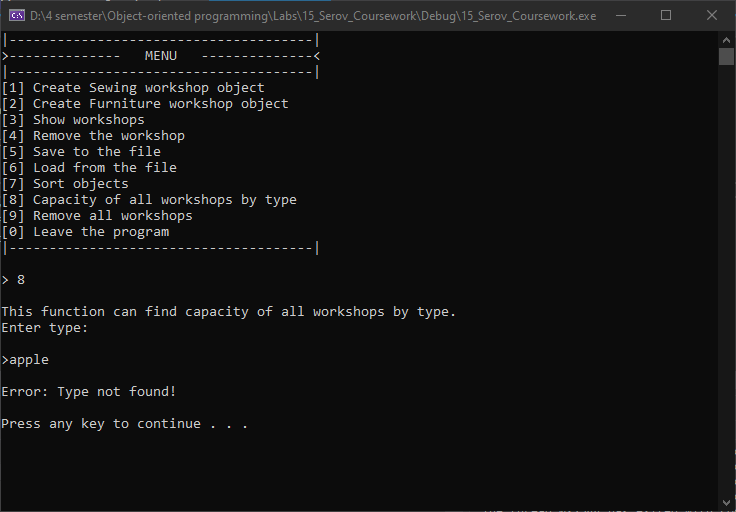


Рис. 3.4.20 Обчислення сумарної потужності цеху, якого немає в контейнері

## 3.5 Модульне тестування ПД за допомоги тестової платформи Boost::Test

Код тесту наведений окремо (Додаток Г). Результати тесту наведено на Рис. 3.5.1

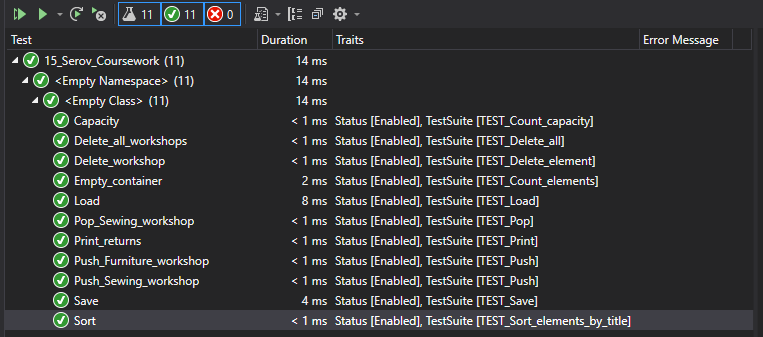


Рис. 3.5.1 Результати Boost тесту

# **Висновки**

Курсова робота складалася з багатьох послідовних етапів розробки та тестування ПД, що забезпечило повне розуміння процесу розробки програмного забезпечення на мові C++ у парадигмі ООП в інтегрованому середовищі розробки Microsoft Visual Studio.

Була виконана реалізація успадкування класу «Цех» нащадками «Цех пошиття одягу» та «Меблевий цех», що дозволило гнучко налаштувати методи для подальшої роботи. У нащадках використані перевантажені оператори задля комфортної роботи з введенням/виведенням даних. У класі контейнера було реалізовано роботу двозв’язного кільцевого списку для зберігання даних цехів та методів для роботи з ним. У КР також було виконано реалізацію роботи з файлом, в який записується та зчитується дані цехів, що зберігаються в контейнері. Клас меню виконаний за шаблоном Singleton, який запобігає створенню декількох об’єктів меню, що може стати проблемою для кінцевого користувача. У класі меню існує широкий функціонал для роботи з цехами, зрозумілий інтерфейс та обробка помилок при неправильній роботі з ПД.

Програмний додаток був повністю протестований вручну та за допомогою тестової платформи Boost, що забезпечує впевненість у правильній подальшій роботі ПД та безпечності даних користувача.

# **Список використаних джерел**

1. Алгоритмізація та програмування. Програмно-алгоритмічна реалізація наближеного обчислення значення функції через її розклад в степеневий ряд : метод. вказівки до викон. розрахунково-граф. роботи для студ. спец. 122 «Комп’ютерні науки та інформаційні технології» спеціалізації «Інформаційні технології в біології та медицині» / Уклад. С. М. Алхімова. – Київ: КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2018. – 151 с.
2. Страуструп Б. Язык программирования С++ / Б. Страуструп. – [3-e изд.] – Москва: Издательство Бином, 2000. – 544 с.
3. Об’єктно-орієнтовне програмування : підручник. У 2-х ч. Ч. 2. Обєктно-орієнтовний підхід до розробки програмного забезпечення / С.М. Алхімова. – Київ: КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2019. – 89с.
4. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования = Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development. ­– 3-е изд. – М.: Вильямс, 2006. – 472 с. – ISBN 0-13-148906-2.

**Додаток А ДІАГРАМА КЛАСІВ**



# **Додаток Б ДІАГРАМА ПОСЛІДОВНОСТЕЙ**



# **Додаток В ДІАГРАМА ОБ’ЄКТІВ**



# **Додаток Г КОД МОДУЛЬНИХ ТЕСТІВ**

#define BOOST\_TEST\_MODULE CW\_Test

#include <boost/test/included/unit\_test.hpp>

#include "Сontainer.h"

BOOST\_AUTO\_TEST\_SUITE(TEST\_Count\_elements)

BOOST\_AUTO\_TEST\_CASE(Empty\_container) // check if Count\_elements() work properly

{

Сontainer test\_all\_workshops;

BOOST\_REQUIRE\_EQUAL(test\_all\_workshops.Count\_elements(), 0);

}

BOOST\_AUTO\_TEST\_SUITE\_END()

//------------------------------------------------------------------

BOOST\_AUTO\_TEST\_SUITE(TEST\_Push)

BOOST\_AUTO\_TEST\_CASE(Push\_Sewing\_workshop) // push 10 sewing workshops and count them

{

Сontainer test\_container;

Sewing\_workshop\* Sewing = new Sewing\_workshop("Kiyv\_sew", 1010, 1010, "cotton", 1010);

for (char i = 0; i < 10; i++)

test\_container.Push(Sewing);

BOOST\_REQUIRE\_EQUAL(test\_container.Count\_elements(), 10);

}

BOOST\_AUTO\_TEST\_CASE(Push\_Furniture\_workshop) // push 10 furniture workshops and count them

{

Сontainer test\_container;

Furniture\_workshop\* Furniture = new Furniture\_workshop("Kiyv\_furniture", 1010, 1010, "oak", 1010);

for (char i = 0; i < 10; i++)

test\_container.Push(Furniture);

BOOST\_REQUIRE\_EQUAL(test\_container.Count\_elements(), 10);

}

BOOST\_AUTO\_TEST\_SUITE\_END()

//------------------------------------------------------------------

BOOST\_AUTO\_TEST\_SUITE(TEST\_Pop)

BOOST\_AUTO\_TEST\_CASE(Pop\_Sewing\_workshop) // add 10 workshops and pop 5 = 5

{

Сontainer test\_container;

Sewing\_workshop\* Sewing = new Sewing\_workshop("Kiyv\_sew", 1010, 1010, "cotton", 1010);

for (char i = 0; i < 10; i++)

test\_container.Push(Sewing);

BOOST\_REQUIRE\_EQUAL(test\_container.Count\_elements(), 10);

for (char i = 0; i < 5; i++)

test\_container.Pop();

BOOST\_REQUIRE\_EQUAL(test\_container.Count\_elements(), 5);

}

BOOST\_AUTO\_TEST\_SUITE\_END()

//------------------------------------------------------------------

BOOST\_AUTO\_TEST\_SUITE(TEST\_Print)

BOOST\_AUTO\_TEST\_CASE(Print\_returns) // check false/true returns from Print()

{

Сontainer test\_container;

BOOST\_REQUIRE\_EQUAL(test\_container.Print(), false);

Sewing\_workshop\* Sewing = new Sewing\_workshop("Kiyv\_sew", 1010, 1010, "cotton", 1010);

test\_container.Push(Sewing);

BOOST\_REQUIRE\_EQUAL(test\_container.Print(), true);

}

BOOST\_AUTO\_TEST\_SUITE\_END()

//------------------------------------------------------------------

BOOST\_AUTO\_TEST\_SUITE(TEST\_Delete\_element)

BOOST\_AUTO\_TEST\_CASE(Delete\_workshop) // add 10 workshops and delete the one that in the middle

{

Сontainer test\_container;

Sewing\_workshop\* Sewing = new Sewing\_workshop("Kiyv\_sew", 1010, 1010, "cotton", 1010);

Furniture\_workshop\* Furniture = new Furniture\_workshop("Kiyv\_furniture", 1010, 1010, "oak", 1010);

for (char i = 0; i < 4; i++)

test\_container.Push(Sewing);

test\_container.Push(Furniture);

for (char i = 0; i < 5; i++)

test\_container.Push(Sewing);

test\_container.Delete\_element("Kiyv\_furniture");

BOOST\_REQUIRE\_EQUAL(test\_container.Count\_elements(), 9);

}

BOOST\_AUTO\_TEST\_SUITE\_END()

//------------------------------------------------------------------

BOOST\_AUTO\_TEST\_SUITE(TEST\_Save)

BOOST\_AUTO\_TEST\_CASE(Save) // add 10 workshops and save it, Save() will return true if everything well

{

Сontainer test\_container;

Sewing\_workshop\* Sewing = new Sewing\_workshop("Kiyv\_sew", 1010, 1010, "cotton", 1010);

Furniture\_workshop\* Furniture = new Furniture\_workshop("Kiyv\_furniture", 1010, 1010, "oak", 1010);

for (char i = 0; i < 5; i++)

{

test\_container.Push(Sewing);

test\_container.Push(Furniture);

}

BOOST\_REQUIRE\_EQUAL(test\_container.Save(), true);

}

BOOST\_AUTO\_TEST\_SUITE\_END()

//------------------------------------------------------------------

BOOST\_AUTO\_TEST\_SUITE(TEST\_Delete\_all)

BOOST\_AUTO\_TEST\_CASE(Delete\_all\_workshops) // delete return false, add 10 workshops and it return true

{

Сontainer test\_container;

BOOST\_REQUIRE\_EQUAL(test\_container.Delete\_all(), false);

Sewing\_workshop\* Sewing = new Sewing\_workshop("Kiyv\_sew", 1010, 1010, "cotton", 1010);

Furniture\_workshop\* Furniture = new Furniture\_workshop("Kiyv\_furniture", 1010, 1010, "oak", 1010);

for (char i = 0; i < 5; i++)

{

test\_container.Push(Sewing);

test\_container.Push(Furniture);

}

BOOST\_REQUIRE\_EQUAL(test\_container.Delete\_all(), true);

}

BOOST\_AUTO\_TEST\_SUITE\_END()

//------------------------------------------------------------------

BOOST\_AUTO\_TEST\_SUITE(TEST\_Load)

BOOST\_AUTO\_TEST\_CASE(Load) // create, delete, load 10 workshops and count them

{

Сontainer test\_container;

Sewing\_workshop\* Sewing = new Sewing\_workshop("Kiyv\_sew", 1010, 1010, "cotton", 1010);

Furniture\_workshop\* Furniture = new Furniture\_workshop("Kiyv\_furniture", 1010, 1010, "oak", 1010);

for (char i = 0; i < 5; i++)

{

test\_container.Push(Sewing);

test\_container.Push(Furniture);

}

test\_container.Save();

test\_container.Delete\_all();

test\_container.Load();

BOOST\_REQUIRE\_EQUAL(test\_container.Save(), true);

}

BOOST\_AUTO\_TEST\_SUITE\_END()

//------------------------------------------------------------------

BOOST\_AUTO\_TEST\_SUITE(TEST\_Sort\_elements\_by\_title)

BOOST\_AUTO\_TEST\_CASE(Sort) // method do not returns value, so check false/true returns from Sort\_elements\_by\_title()

{

Сontainer test\_container;

BOOST\_REQUIRE\_EQUAL(test\_container.Sort\_elements\_by\_title(), false);

Sewing\_workshop\* Sewing = new Sewing\_workshop("Kiyv\_sew", 1010, 1010, "cotton", 1010);

Furniture\_workshop\* Furniture = new Furniture\_workshop("Kiyv\_furniture", 1010, 1010, "oak", 1010);

test\_container.Push(Sewing);

test\_container.Push(Furniture);

test\_container.Push(Sewing);

/\*

for real, test doesn't work because of inaccessibility of start\_node. But it works, I swear

test\_container.Sort\_elements\_by\_title();

BOOST\_REQUIRE\_EQUAL((test\_container.start\_node)->data->Get\_title(),

(test\_container.start\_node)->next\_node->data->Get\_title());

\*/

BOOST\_REQUIRE\_EQUAL(test\_container.Sort\_elements\_by\_title(), true);

}

BOOST\_AUTO\_TEST\_SUITE\_END()

//------------------------------------------------------------------

BOOST\_AUTO\_TEST\_SUITE(TEST\_Count\_capacity)

BOOST\_AUTO\_TEST\_CASE(Capacity) // method do not returns value, so check false/true returns from Count\_capacity()

{

Сontainer test\_container;

BOOST\_REQUIRE\_EQUAL(test\_container.Count\_capacity(), false);

Sewing\_workshop\* Sewing = new Sewing\_workshop("Kiyv\_sew", 10, 10, "cotton", 1010);

Furniture\_workshop\* Furniture = new Furniture\_workshop("Kiyv\_furniture", 1010, 1010, "oak", 1010);

test\_container.Push(Sewing);

test\_container.Push(Furniture);

test\_container.Push(Sewing);

// string type\_to\_count = "cotton"; cin >> type\_to\_count; CONTAINER.CPP->Count\_capacity() MODIFIED

BOOST\_REQUIRE\_EQUAL(test\_container.Count\_capacity(), true); // 10\*10 + 10\*10 = 200

}

BOOST\_AUTO\_TEST\_SUITE\_END()

# **Додаток Д КОД ПД**

## Workshop.h

#pragma once

#include <string>

#include <iostream>

using namespace std;

/\*

Base class with basic workshop info

\*/

class Workshop

{

protected:

string workshop\_title;

unsigned short num\_of\_workbench;

int average\_capacity;

public:

Workshop();

Workshop(string title, unsigned short workbench, int capacity);

virtual ~Workshop();

string Get\_title() const;

unsigned short Get\_workbench() const;

int Get\_capacity() const;

void Set\_title(string new\_title);

void Set\_workbench(unsigned short new\_workbench);

void Set\_capacity(int new\_capacity);

virtual void Print\_workshop() = 0;

};

## Workshop.cpp

#include "Workshop.h"

Workshop::Workshop() : workshop\_title("Noname"), num\_of\_workbench(0), average\_capacity(0)

{}

Workshop::Workshop(string title, unsigned short workbench, int capacity) :

workshop\_title(title), num\_of\_workbench(workbench), average\_capacity(capacity)

{}

Workshop::~Workshop()

{}

string Workshop::Get\_title() const

{

return workshop\_title;

}

unsigned short Workshop::Get\_workbench() const

{

return num\_of\_workbench;

}

int Workshop::Get\_capacity() const

{

return average\_capacity;

}

void Workshop::Set\_title(string new\_title)

{

workshop\_title = new\_title;

}

void Workshop::Set\_workbench(unsigned short new\_workbench)

{

num\_of\_workbench = new\_workbench;

}

void Workshop::Set\_capacity(int new\_capacity)

{

average\_capacity = new\_capacity;

}

void Workshop::Print\_workshop()

{

cout << "\nTitle: " << Workshop::Get\_title()

<< "\nWorkbenches: " << Workshop::Get\_workbench()

<< "\nCapacity: " << Workshop::Get\_capacity() << endl;

}

## Sewing\_workshop.h

#pragma once

#include "Workshop.h"

/\*

Derived class with basic workshop info and sewing info

\*/

class Sewing\_workshop : public Workshop

{

private:

string fabric\_type;

unsigned rolls\_of\_fabric;

public:

Sewing\_workshop();

Sewing\_workshop(string title, unsigned short workbench, int capacity, string fabric, unsigned rolls);

~Sewing\_workshop();

string Get\_fabric() const;

unsigned Get\_rolls() const;

void Set\_fabric(string new\_fabric);

void Set\_rolls(unsigned new\_rolls);

friend istream& operator >> (istream& is, Sewing\_workshop& sew\_is); // overloaded input operator

friend ostream& operator << (ostream& os, Sewing\_workshop& sew\_os); // overloaded output operator

void Print\_workshop();

};

## Sewing\_workshop.cpp

#include "Sewing\_workshop.h"

Sewing\_workshop::Sewing\_workshop() : fabric\_type("None"), rolls\_of\_fabric(0)

{}

Sewing\_workshop::Sewing\_workshop(string title, unsigned short workbench, int capacity, string fabric, unsigned rolls) :

Workshop(title, workbench, capacity), fabric\_type(fabric), rolls\_of\_fabric(rolls)

{}

Sewing\_workshop::~Sewing\_workshop()

{}

string Sewing\_workshop::Get\_fabric() const

{

return fabric\_type;

}

unsigned Sewing\_workshop::Get\_rolls() const

{

return rolls\_of\_fabric;

}

void Sewing\_workshop::Set\_fabric(string new\_fabric)

{

fabric\_type = new\_fabric;

}

void Sewing\_workshop::Set\_rolls(unsigned new\_rolls)

{

rolls\_of\_fabric = new\_rolls;

}

istream& operator >> (istream& is, Sewing\_workshop& sew\_is)

{

is >> sew\_is.workshop\_title >> sew\_is.num\_of\_workbench >> sew\_is.average\_capacity >>

sew\_is.fabric\_type >> sew\_is.rolls\_of\_fabric;

return is;

}

ostream& operator << (ostream& os, Sewing\_workshop& sew\_os)

{

os << sew\_os.workshop\_title << '\n' << sew\_os.num\_of\_workbench << '\n' << sew\_os.average\_capacity <<

'\n' << sew\_os.fabric\_type << '\n' << sew\_os.rolls\_of\_fabric << '\n';

return os;

}

void Sewing\_workshop::Print\_workshop()

{

Workshop::Print\_workshop();

cout << "Fabric type: " << Sewing\_workshop::Get\_fabric()

<< "\nRolls: " << Sewing\_workshop::Get\_rolls() << '\n' << endl;

}

## Furniture\_workshop.h

#pragma once

#include "Workshop.h"

/\*

Derived class with basic workshop info and furniture info

\*/

class Furniture\_workshop : public Workshop

{

private:

string wood\_type;

unsigned amount\_of\_wood;

public:

Furniture\_workshop();

Furniture\_workshop(string title, unsigned short workbench, int capacity, string wood, unsigned amount);

~Furniture\_workshop();

string Get\_wood() const;

unsigned Get\_amount() const;

void Set\_wood(string new\_wood);

void Set\_amount(unsigned new\_amount);

friend istream& operator >> (istream& is, Furniture\_workshop& fur\_is); // overloaded input operator

friend ostream& operator << (ostream& os, Furniture\_workshop& fur\_os); // overloaded output operator

void Print\_workshop();

};

## Furniture\_workshop.cpp

#include "Furniture\_workshop.h"

Furniture\_workshop::Furniture\_workshop() : wood\_type("None"), amount\_of\_wood(0)

{}

Furniture\_workshop::Furniture\_workshop(string title, unsigned short workbench, int capacity, string wood, unsigned amount) :

Workshop(title, workbench, capacity), wood\_type(wood), amount\_of\_wood(amount)

{}

Furniture\_workshop::~Furniture\_workshop()

{}

string Furniture\_workshop::Get\_wood() const

{

return wood\_type;

}

unsigned Furniture\_workshop::Get\_amount() const

{

return amount\_of\_wood;

}

void Furniture\_workshop::Set\_wood(string new\_wood)

{

wood\_type = new\_wood;

}

void Furniture\_workshop::Set\_amount(unsigned new\_amount)

{

amount\_of\_wood = new\_amount;

}

istream& operator >> (istream& is, Furniture\_workshop& fur\_is)

{

is >> fur\_is.workshop\_title >> fur\_is.num\_of\_workbench >> fur\_is.average\_capacity >>

fur\_is.wood\_type >> fur\_is.amount\_of\_wood;

return is;

}

ostream& operator << (ostream& os, Furniture\_workshop& fur\_os)

{

os << fur\_os.workshop\_title << '\n' << fur\_os.num\_of\_workbench << '\n' << fur\_os.average\_capacity <<

'\n' << fur\_os.wood\_type << '\n' << fur\_os.amount\_of\_wood << '\n';

return os;

}

void Furniture\_workshop::Print\_workshop()

{

Workshop::Print\_workshop();

cout << "Wood type: " << Furniture\_workshop::Get\_wood()

<< "\nAmount: " << Furniture\_workshop::Get\_amount() << '\n' << endl;

}

## Container.h

#pragma once

#include "Sewing\_workshop.h"

#include "Furniture\_workshop.h"

#include <fstream>

//Doubly Circular Linked List

struct Node

{

Workshop\* data; // workshop data

Node\* next\_node;

Node\* prev\_node;

};

/\*

Circular Doubly Linked List has properties of both doubly linked list

and circular linked list in which two consecutive elements are linked

or connected by previous and next pointer and the last node points to

first node by next pointer and also the first node points to last node

by previous pointer.

\*/

class Сontainer

{

private:

Node\* current\_node;

Node\* start\_node; // main node

public:

Сontainer();

~Сontainer();

void Push(Workshop\* new\_item); // standart push method adding element and makes it previous to start\_node

void Pop();

unsigned Count\_elements() const; // return counted elements

bool Print() const; // print all elements on screen

bool Delete\_element(string title\_to\_delete); // pop element by title

bool Save() const; // save all elements to file

void Load(); // load all elements from file

bool Sort\_elements\_by\_title();

bool Count\_capacity() const; // avarage capacity \* workbanches in workshops

bool Delete\_all();

};

## Container.cpp

#include "Сontainer.h"

Сontainer::Сontainer() : current\_node(NULL)

{}

Сontainer::~Сontainer()

{

Delete\_all();

}

void Сontainer::Push(Workshop\* new\_item)

{

Node\* new\_node = new Node;

new\_node->data = new\_item; //make a node with parameter data

if (current\_node) // if container have at least 1 element - link them

{

new\_node->next\_node = current\_node;

current\_node->prev\_node = new\_node;

}

else

start\_node = new\_node;

// connecting first and last nodes

new\_node->prev\_node = start\_node;

start\_node->next\_node = new\_node;

current\_node = new\_node;

}

void Сontainer::Pop()

{

Node\* tmp\_node = start\_node;

if (tmp\_node->next\_node == tmp\_node) // if element point to itself - delete single element

start\_node = NULL;

else

{

if (tmp\_node == start\_node)

start\_node = start\_node->next\_node; // if pop main element - make next element new main

// connection exchange between nodes near popped element

(tmp\_node->next\_node)->prev\_node = tmp\_node->prev\_node;

(tmp\_node->prev\_node)->next\_node = tmp\_node->next\_node;

}

}

unsigned Сontainer::Count\_elements() const

{

if (start\_node)

{

unsigned num\_of\_elements = 0;

Node\* tmp\_node = start\_node;

do

{

tmp\_node = tmp\_node->prev\_node;

num\_of\_elements++;

}

while (tmp\_node != start\_node);

return num\_of\_elements;

}

else

{

cout << "\nContainer clear. There are no workshops!\n";

return 0;

}

}

bool Сontainer::Print() const

{

bool print\_flag = false; // flag show success of printing

Node\* tmp\_node = start\_node;

if (start\_node)

{

cout << "\n|--------------------------------------|\n"

<< ">-------------- DATA --------------<\n"

<< "|--------------------------------------|\n";

do

{

tmp\_node->data->Print\_workshop();

tmp\_node = tmp\_node->next\_node;

} while (tmp\_node != start\_node);

cout << "|--------------------------------------|\n\n";

print\_flag = true;

}

return print\_flag;

}

bool Сontainer::Delete\_element(string title\_to\_delete)

{

bool pop\_flag = false; // flag show success of finding and deleting the node

Node\* tmp\_node = start\_node;

do // cicle working before it find needed title or make a loop in container

{

if (tmp\_node->data->Get\_title() == title\_to\_delete)

{

pop\_flag = true;

Pop();

break;

}

tmp\_node = tmp\_node->next\_node;

}

while (tmp\_node != start\_node);

return pop\_flag;

}

bool Сontainer::Save() const

{

bool save\_flag = false; // flag show that container have at least one element

if (start\_node)

{

save\_flag = true;

ofstream save\_file;

save\_file.open("Workshops\_info.txt"); // make or remake file

if (save\_file.is\_open())

{

Node\* tmp\_node = start\_node;

unsigned containers\_length = Count\_elements();

for (unsigned i = 0; i < containers\_length; i++)

{

if (typeid(\*(tmp\_node->data)) == typeid(Sewing\_workshop))

{

save\_file << 1 << '\n'; // for easier data processing added label to sewing workshops...

save\_file << \*((Sewing\_workshop\*)(tmp\_node->data)); // print method declarated in Sewing\_workshop.h

}

if (typeid(\*(tmp\_node->data)) == typeid(Furniture\_workshop))

{

save\_file << 2 << '\n'; // and furniture workshops

save\_file << \*((Furniture\_workshop\*)(tmp\_node->data)); // print method declarated in Furniture\_workshop.h

}

tmp\_node = tmp\_node->next\_node;

}

save\_file.close();

cout << "\nData saved in 'Workshops\_info.txt'\n\n";

}

else

cout << "\nError: data may not saved properly!\n\n";

}

return save\_flag;

}

void Сontainer::Load()

{

ifstream read\_file;

read\_file.open("Workshops\_info.txt"); // open file

if (read\_file.is\_open())

{

char one\_two\_string[2]; // char that will read labels of workshops

read\_file.getline(one\_two\_string, 2); // reading first label

unsigned char one\_two\_num = atoi(one\_two\_string); // transform char to number

Sewing\_workshop\* sewing\_1; // temp workshops

Furniture\_workshop\* furniture\_2;

while (!read\_file.eof()) // while file not end yet

{

if (one\_two\_num == 1)

{

sewing\_1 = new Sewing\_workshop();

read\_file >> (\*sewing\_1); // read method declarated in Sewing\_workshop.h

Push(sewing\_1); // add workshop to container

}

else if (one\_two\_num == 2)

{

furniture\_2 = new Furniture\_workshop();

read\_file >> (\*furniture\_2);// read method declarated in Furniture\_workshop.h

Push(furniture\_2); // add workshop to container

}

read\_file.getline(one\_two\_string, 2); // reading next workshop label

one\_two\_num = atoi(one\_two\_string); // and transform char to number

}

read\_file.close();

}

else

cout << "\nError: poor data!\n\n";

}

/\*

Sort\_elements\_by\_title() will sort nodes of the list in ascending order.

1. Define a node current which will point to head.

2. Define another node index which will point to node next to current.

3. Compare data of current and index node. If the current's data is greater than the index's data, then swap the data between them.

4. Current will point to current->next\_node and index will point to index->next\_node.

5. Continue this process till the entire list is sorted.

\*/

bool Сontainer::Sort\_elements\_by\_title()

{

bool sort\_flag = false; // flag show that container have at least one element

if (start\_node)

{

sort\_flag = true;

unsigned containers\_length = Count\_elements() - 1; // length of container without 1 used temporary node

Node\* current = start\_node;

Node\* index;

Workshop\* tmp; // temp workshop

for (int i = 0; i < containers\_length; i++)

{

index = current->next\_node;

for (int j = 0; j < containers\_length; j++)

{

if (current->data->Get\_title() < index->data->Get\_title())

{

tmp = index->data;

index->data = current->data;

current->data = tmp;

}

index = index->next\_node;

}

current = current->next\_node;

}

start\_node = current;

}

return sort\_flag;

}

bool Сontainer::Count\_capacity() const

{

if (start\_node)

{

cout << "\nThis function can find capacity of all workshops by type.\nEnter type:\n\n>";

string type\_to\_count = "cotton";

//cin >> type\_to\_count;

bool type\_flag = false; // flag show success of founding type

float full\_capacity = 0;

Node\* tmp\_node = start\_node;

do

{

if (typeid(\*(tmp\_node->data)) == typeid(Sewing\_workshop)) // type is Sewing\_workshop..

{

if ( (((Sewing\_workshop\*)(tmp\_node->data))->Get\_fabric()) == type\_to\_count ) // and titles match up

{

full\_capacity += (tmp\_node->data->Get\_capacity()) \* (tmp\_node->data->Get\_workbench());

type\_flag = true; // adding multiply of benches and average capacity and flag turn up

}

}

else if (typeid(\*(tmp\_node->data)) == typeid(Furniture\_workshop)) // type is Furniture\_workshop..

{

if ( (((Furniture\_workshop\*)(tmp\_node->data))->Get\_wood()) == type\_to\_count ) // and titles match up

{

full\_capacity += tmp\_node->data->Get\_capacity() \* tmp\_node->data->Get\_workbench();

type\_flag = true; // adding multiply of benches and average capacity and flag turn up

}

}

tmp\_node = tmp\_node->next\_node;

}

while (tmp\_node != start\_node);

if (type\_flag)

cout << "\nFull capacity of all " << type\_to\_count << " workshops: " << full\_capacity << "\n\n";

else

cout << "\nError: Type not found!\n\n";

return true; // there is at least one node and finding end with success

}

return false; // there is no nodes in container

}

bool Сontainer::Delete\_all()

{

if (start\_node)

{

while (start\_node) // end - start\_node is NULL

{

Pop();

}

current\_node = NULL;

return true;

}

return false;

}

## Menu.h

#pragma once

#include "Сontainer.h"

/\*

The Singleton menu class provides the 'Getinstance' method,

which acts as an alternative constructor and allows customers

to receive the same class instance at each call.

\*/

class Menu

{

private:

/\*

The Singlton constructor must always be hidden

to prevent the creation of an object through the new operator.

\*/

Menu() {}

static Menu\* single\_menu;

Сontainer all\_workshops;

int choice;

public:

//Menu should not be assignable.

Menu(Menu& other) = delete;

void operator = (const Menu&) = delete;

/\*

It’s a static method that controls access to a Singleton.

At the first launch, it creates a single instance and places

it in a static field. In subsequent runs, it returns an object

stored in a static field to the client.

\*/

static Menu\* GetInstance();

void Start(); // menu function that gives a choice to user (UI)

void Create\_sewing\_workshop(Сontainer\* Data);

void Create\_furniture\_workshop(Сontainer\* Data);

void Show() const; // show all elements

void Delete(); // delete one element by the title

void Save() const; // save all elements to file

void Load(); // save all elements from file

void Sort(); // sort elements by titles

void Capacity() const; // count capacity of all workshops by type

void Full\_delete(); // delete all elements

void Exit(); // exit from the program

};

## Menu.cpp

#include "Menu.h"

Menu\* Menu::single\_menu = nullptr; // global variable puts in Singleton class 'Menu' in static field start value

Menu\* Menu::GetInstance() //This is a safe way to create an instance

{

if (single\_menu == nullptr) {

single\_menu = new Menu();

}

return single\_menu;

}

void Menu::Start()

{

cout << "|--------------------------------------|\n\n" // start window

<< "This program works with worshops data\n"

<< "and can help you to manage your info\n"

<< "about different workshops: Sewing or\n"

<< "Furniture. In any time you can save\n"

<< "or load data to file and use it for\n"

<< "your purposes.\tLet's start!\n\n"

<< "|--------------------------------------|\n\n";

system("pause");

do // menu loop

{

system("cls");

cout << "|--------------------------------------|\n"

<< ">-------------- MENU --------------<\n"

<< "|--------------------------------------|\n"

<< "[1] Create Sewing workshop object\n"

<< "[2] Create Furniture workshop object\n"

<< "[3] Show workshops\n"

<< "[4] Remove the workshop\n"

<< "[5] Save to the file\n"

<< "[6] Load from the file\n"

<< "[7] Sort objects\n"

<< "[8] Capacity of all workshops by type\n"

<< "[9] Remove all workshops\n"

<< "[0] Leave the program\n"

<< "|--------------------------------------|\n"

<< "\n> ";

while (!(cin >> choice) || choice < 0 || choice > 9)

{

cin.clear();

cin.ignore();

fflush(stdin);

cout << "Try number from 0 to 9!\n> ";

}

switch (choice)

{

case 1:

{

Create\_sewing\_workshop(&all\_workshops);

break;

}

case 2:

{

Create\_furniture\_workshop(&all\_workshops);

break;

}

case 3:

{

Show();

break;

}

case 4:

{

Delete();

break;

}

case 5:

{

Save();

break;

}

case 6:

{

Load();

break;

}

case 7:

{

Sort();

break;

}

case 8:

{

Capacity();

break;

}

case 9:

{

Full\_delete();

break;

}

case 0:

{

Exit();

break;

}

}

} while (true);

}

void Menu::Create\_sewing\_workshop(Сontainer\* Data)

{

Sewing\_workshop\* new\_sewing\_workshop = new Sewing\_workshop();

cout << "\nEnter info about sewing workshop (separated by SPACE or ENTER)\n"

<< "Title Workbanches Capacity Fabric Rolls\n\n>";

cin >> \*new\_sewing\_workshop;

Data->Push(new\_sewing\_workshop);

cout << "\nNew sewing workshop added!\n\n";

system("pause");

system("cls");

}

void Menu::Create\_furniture\_workshop(Сontainer\* Data)

{

Furniture\_workshop\* new\_furniture\_workshop = new Furniture\_workshop();

cout << "\nEnter info about furniture workshop (separated by SPACE or ENTER)\n"

<< "Title Workbanches Capacity Wood Amount\n\n>";

cin >> \*new\_furniture\_workshop;

Data->Push(new\_furniture\_workshop);

cout << "\nNew furniture workshop added!\n\n";

system("pause");

system("cls");

}

void Menu::Show() const

{

if (all\_workshops.Print()) // print workshops and pause if container have at least one element

{

system("pause");

system("cls");

}

else

{

cout << "\nContainer clear. There are no workshops to show!\n\n";

system("pause");

system("cls");

}

}

void Menu::Delete()

{

if (all\_workshops.Count\_elements() == 0) // delete works only if container is not empty

{

cout << "\nWhen workshops container is clear there is nothing to delete!\n\n";

system("pause");

system("cls");

}

else

{

all\_workshops.Print(); // show elements before choose for beter UX

string title\_to\_delete;

cout << "\nEnter title of workshop to delete it\n\n>";

cin >> title\_to\_delete;

bool pop\_flag = all\_workshops.Delete\_element(title\_to\_delete); // deleting element return bool

if (pop\_flag)

cout << "\nWorkshop '" << title\_to\_delete << "' has been deleted!\n\n";

else

cout << "\nError: Workshop '" << title\_to\_delete << "' not found!\n\n";

system("pause");

system("cls");

}

}

void Menu::Save() const

{

if (all\_workshops.Save()) // save and pause if container have at least one element

{

system("pause");

system("cls");

}

else

{

cout << "\nContainer clear. There are no workshops to save some!\n\n";

system("pause");

system("cls");

}

}

void Menu::Load()

{

all\_workshops.Load();

cout << "\nWorkshops loaded from file 'Workshops\_info.txt'!\n\n";

system("pause");

system("cls");

}

void Menu::Sort()

{

if (all\_workshops.Sort\_elements\_by\_title()) // sort and pause if container have at least one element

{

cout << "\nAll workshops sorted by the title!\n\n";

system("pause");

system("cls");

}

else

{

cout << "\nContainer clear. There are no workshops to sort some!\n\n";

system("pause");

system("cls");

}

}

void Menu::Capacity() const

{

if (all\_workshops.Count\_capacity()) // count capacity and pause if container have at least one element

{

system("pause");

system("cls");

}

else

{

cout << "\nContainer clear. There are no workshops to count capacity!\n\n";

system("pause");

system("cls");

}

}

void Menu::Full\_delete() // delete all elements and pause if container have at least one element

{

if (all\_workshops.Delete\_all())

{

cout << "\nNow workshops container is clear!\n\n";

system("pause");

system("cls");

}

else

{

cout << "\nWorkshops container is clear already!\n\n";

system("pause");

system("cls");

}

}

void Menu::Exit()

{

all\_workshops.Delete\_all();

cout << "\nGood luck!\n\n";

exit(0);

}

## Main.cpp

#include "Menu.h"

/\*

This program add/edit/find/sort/save/load/delete workshops info.

by Alexander Serov

void main()

{

Menu\* Workshop\_data\_manage = Menu::GetInstance(); // singleton menu creation

Workshop\_data\_manage->Start(); // start of main menu

system("pause");

}

\*/