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 3 джерел і 2 додатків. Загальний обсяг курсової роботи становить 45 сторінок, основного тексту (без додатків) – 25 сторінок, ілюстрацій – 34, таблиць – 38.

**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 3 references, and 2 applications. The total volume of coursework is 45 pages, main text (without applications) – 25 pages, illustrations – 34, tables – 38.

**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 3 источников и 1 приложения. Общий объем курсовой работы составляет 45 страниц, основного текста (без приложений) – 25 страниц, иллюстраций – 34, таблиц – 38.

# 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.

Конструктор – метод класу, який викликається автоматично, під час створення об’єкту класу і призначений для ініціалізації полів об’єктів.

Деструктор – викликається автоматично під час знищення об’єкта і необхідний для збереження ресурсів обчислювальної техніки.

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 у 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 |
| 7 | void Set\_fabric(string new\_fabric) | Нове значення типу матеріалу (fabric\_type) | - | Змінює поле типу матеріалу |
| 8 | void Set\_rolls(unsigned new\_rolls) | Нове значення кількості рулонів (rolls\_of\_fabric) | - | Змінює поле кількості рулонів |
| 9 | 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 |
| 10 | 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 |
| 11 | 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 |
| 7 | void Set\_wood(string new\_wood) | Нове значення типу дерева (wood\_type) | - | Змінює поле типу дерева |
| 8 | void Set\_amount(unsigned new\_amount) | Нове значення кількості дерева (amount\_of\_wood) | - | Змінює поле кількості дерева |
| 9 | 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 |
| 10 | 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 |
| 11 | 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 |
| 7 | bool Print() const | - | Бульова змінна, що показує наявність хоча б одного елемента в контейнері | Виводить на екран дані усіх цехів в контейнері |
| 8 | 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() |
| 9 | bool Save() const | - | Бульова змінна, що показує наявність хоча б одного елемента в контейнері | Збереження даних про цеха у файл |
| 10 | void Load() | - | - | Зчитування файлу та додавання елементів у контейнер |
| 11 | 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 |
| 12 | bool Count\_capacity() const | - | Бульова змінна, що показує наявність хоча б одного елемента в контейнері | Рахує сумарну потужність верстатів усіх цехів за їх типом |
| 13 | bool Delete\_all() | - | Бульова змінна, що показує наявність хоча б одного елемента в контейнері | Видаляє усі елементи з контейнера, залишаючи його пустим |