АНОТАЦІЯ

Завражний Д.К. Розробка програмних додатків і використання об’єктно-орієнтованого підходу.

Курсова робота із дисципліни «Об’єктно-орієнтоване програмування» присвячена питанню закріплення та поглиблення знань, отриманих при вивченні дисципліни, отримання навичок при роботі із програмними додатками. Завданням курсової роботи є створення програмного забезпечення у сучасному середовищі розробки Microsoft Visual Studio 2010 мовою програмування високого рівня С++ із використанням об’єктно-орієнтованого підходу. В курсовій роботі було визначено класи та спрограмоване меню користувача (ієрархія класів програми: абстрактний клас – держава, класи-нащадки – республіка, монархія). Було створено об’єкти та використано контейнер (тип контейнера – однозв'язний кільцевий список). Також було використано збереження об’єктів у файл та їх завантаження з нього. Було виконано сортування контейнера та пошук (запит для пошуку – вивести площу заданої республіки).

**Структура та обсяг роботи.** Курсова робота складається із вступу, трьох розділів, висновків, списку використаної літератури із 3-х джерел і 2-х додатків. Загальний обсяг курсової роботи складає 58 сторінок, основного тексту (без додатків) – 46 сторінок, ілюстрацій – 29, таблиць – 31.

АННОТАЦИЯ

Завражный Д.К. Разработка программных приложений и использование объектно-ориентированного подхода.

Курсовая работа по дисциплине «Объектно-ориентированное программирование» посвящена вопросу закрепления и углубления знаний, полученных при изучении дисциплины, получения навыков при работе с приложениями. Задачей курсовой работы является создание программного обеспечения в современной среде разработки Microsoft Visual Studio 2010 на языке программирования высокого уровня С ++ с использованием объектно-ориентированного подхода. В курсовой работе были определены классы и спрограмоване меню пользователя (иерархия классов программы: абстрактный класс - государство, классы-потомки - республика, монархия). Было создано объекты и использовано контейнер (тип контейнера - односвязный кольцевой список). Также были использованы сохранения объектов в файл и их загрузки с него. Было выполнено сортировки контейнера и поиск (запрос поиска - вывести площадь заданной республики).

**Структура и объем работы.** Курсовая работа состоит из введения, трех глав, заключения, списка использованной литературы из 3-х источников и 2-х приложений. Общий объем курсовой работы составляет 58 страниц основного текста (без приложений) - 46 страниц, иллюстраций - 29, таблиц - 31.

ANNOTATION

Zavrazhnyy D.K. Development of software applications and their use of object oriented approach.

Coursework the discipline of "Object Oriented Programming" is devoted to the issue of deepening the knowledge gained in the study of subjects receiving skills when working with software applications. The objective of the course is the creation of software in today's development environment Microsoft Visual Studio 2010 programming language of high level C ++ using object-oriented approach. In the course work was defined classes and developed menus (class hierarchy of the program: an abstract class - a state-class descendants - republic, monarchy). It was created and used container objects (type of container – one connected annular list). It was also used in the preservation of their file and boot from it. There have been sorting container and Search (search query - display a given area of ​​the country).

The structure and scope of work. Coursework consists of an introduction, three chapters, conclusions, list of references from 3 sources and 2 applications. The total amount of course work is 58 pages of main text (no attachments) - 46 pages of illustrations - 29 tables - 31.

ЗМІСТ

Вступ 8

Розділ 1. Специфікація вимог програмного додатку 9

1.1 Функціональні 9

1.2 Не функціональні 12

Розділ 2. Структура та логіка роботи програмного додатку 13

2.1 Логічна структура 13

2.2 Фізична структура 24

2.3 Логіка роботи програмного додатку 29

Розділ 3. Тестування програмного додатку 30

3.1 Сценарій тестування 30

3.2 Тестові дані 35

3.3 Поетапні результати роботи 39

Висновки 43

Список використаної літератури 44

Додаток А 45

Додаток Б 60

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНТ І ТЕРМІНІВ

Скорочення

ПД – програмний додаток

КР – курсова робота

ООП – об’єктно-орієнтоване програмування

ОС – операційна система

Рис. – рисунок

Табл. – таблиця

UML – уніфікована мова моделювання

Терміни

UML (англ. Unified Modeling Language) — уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю.

Поліморфізм — концепція в програмуванні, відповідно до якої використовується спільний інтерфейс для обробки даних різних спеціалізованих типів.

Інкапсуляція — один з трьох основних механізмів об'єктно-орієнтованого програмування. Йдеться про те, що об’єкт вміщує не тільки дані, але і правила їх обробки, оформлені в вигляді виконуваних фрагментів (методів). А також про те, що доступ до стану об'єкта напряму заборонено, і ззовні з ним можна взаємодіяти виключно через заданий інтерфейс (відкриті поля та методи).

Успадкування — механізм утворення нових класів на основі використання вже існуючих. При цьому властивості і функціональність батьківського класу переходять до класу нащадка (дочірнього).

Об’єкт – екземпляр класу.

Клас – це користувацький тип даних, в якому дані (поля) і функції для їх обробки (методи) утворюють єдине ціле і відображають властивості і поведінку об’єкта.

Конструктор класу — спеціальний метод класу, який автоматично викликається при створенні об'єкта.

Деструктор — спеціальний метод класу, який викликається автоматично при знищенні об'єкта і призначений для його деініціалізації (наприклад, звільнення пам'яті).

Microsoft Visual Studio – середовище розробки програмних додатків.

ВСТУП

Метою курсової роботи є закріплення та поглиблення знань, набутих при вивченні дисципліни «Об’єктно-орієнтоване програмування» в ході лекційних і лабораторних занять, а також отримання практичних навичок створення програмних додатків, використовуючи об’єктно-орієнтований підхід та написання коду мовою програмування С++ в сучасному середовищі розробки програмних додатків Microsoft Visual Studion 2010.

Об’єктно-орієнтоване програмування - це метод програмування, оснований на поданні програми у вигляді сукупності взаємодіючих об’єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування. Таким чином, програма являє собою набір об’єктів, що мають стан та поведінку. Об’єкти взаємодіють використовуючи повідомлення. Будується ієрархія об’єктів: програма в цілому — це об’єкт, для виконання своїх функцій вона звертається до об’єктів що містяться у ньому, які у свою чергу виконують запит шляхом звернення до інших об’єктів програми.

РОЗДІЛ 1

СПЕЦИФІКАЦІЯ ВИМОГ ПД

* 1. Функціональні вимоги до ПД

Варіанти використання ПД характеризуються у таблицях 1.1-1.8 та рисунках 1.1.-1.8.

1.1.1. Варіант використання ПД – створення колекції

*Таблиця 1.1*

Варіант використання ПД – створення колекції

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Іденти-  фікатор | Назва | Мета | Дійова  особа | Тип | Опис |
| 001 | Створення колекції | Створення колекції(контейнер – однозв’язний кільцевий список) для зберігання об’єктів(держав) | Користувач | Основний | Можливість у процесі виконання ПД створення об’єкту колекції за допомогою запиту. Атрибути об’єкту вводяться користувачем |

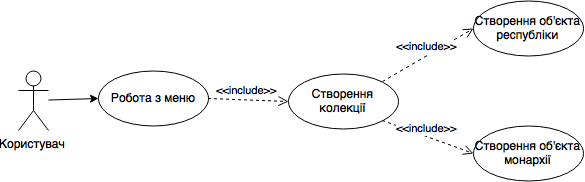


Рис. 1.1. Діаграма варіанту використання в нотації UML:створення колекції

1.1.2. Варіант використання ПД – вивід колекції на екран

*Таблиця 1.2*

**Варіант використання ПД – виведення колекції на екран**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Іденти-**  **фікатор** | **Назва** | **Мета** | **Дійова**  **особа** | **Тип** | **Опис** |
| V\_03 | Виведення колекції на екран | Виведення всіх елементів колекції на екран | Користувач | Основний | Можливість у процесі виконання ПД |
| *Продовж. табл. 1.2* | | | | | |
| **Іденти-**  **фікатор** | **Назва** | **Мета** | **Дійова**  **особа** | **Тип** | **Опис** |
|  |  |  |  |  | вивести всі елементи колекції на екран |

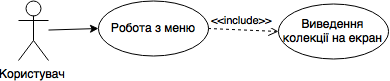


Рис. 1.2. Діаграма варіанту використання в нотації UML:виведення колекції на екран

1.1.3. Варіант використання ПД – очищення колекції

*Таблиця 1.3*

**Варіант використання ПД – очищення колекції**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Іденти-**  **фікатор** | **Назва** | **Мета** | **Дійова**  **особа** | **Тип** | **Опис** |
| V\_04 | Очищення колекції | Очищення колекції | Користувач | Основний | Можливість у процесі виконання ПД очищення об’єктів з колекції |

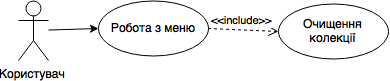


Рис. 1.3. Діаграма варіанту використання в нотації UML:очищення колекції

1.1.4. Варіант використання ПД – збереження колекції

*Таблиця 1.4*

**Варіант використання ПД – збереження колекції**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Іденти-**  **фікатор** | **Назва** | **Мета** | **Дійова**  **особа** | **Тип** | **Опис** |
| V\_05 | Збереження колекції | Збереження колекції до файлу | Користувач | Основний | Можливість у процесі виконання ПД зберегти у файл колекцію |

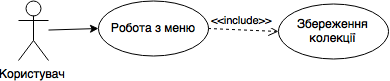


Рис. 1.4. Діаграма варіанту використання в нотації UML:збереження колекції

1.1.5. Варіант використання ПД – завантаження колекції

*Таблиця 1.5*

**Варіант використання ПД – завантаження колекції**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Іденти-**  **фікатор** | **Назва** | **Мета** | **Дійова**  **особа** | **Тип** | **Опис** |
| V\_06 | Завантаження колекції | Завантаження колекції з файлу | Користувач | Основний | Можливість у процесі виконання ПД завантажити колекції з файлу |

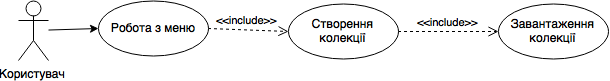


Рис. 1.5. Діаграма варіанту використання в нотації UML: завантаження колекції

1.1.6. Варіант використання ПД – пошук

*Таблиця 1.6*

**Варіант використання ПД – пошук**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Іденти-**  **фікатор** | **Назва** | **Мета** | **Дійова**  **особа** | **Тип** | **Опис** |
| V\_07 | Пошук | Пошук об’єкту в колекції за запитом | Користувач | Основний | Можливість у процесі виконання ПД виконати пошук об’єкта класа-нащадка(республіка) за запитом(назва), та виведення результату пошуку(площа республіки) |

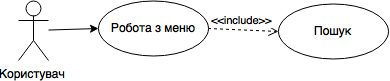


Рис. 1.6. Діаграма варіанту використання в нотації UML: пошук

1.1.7. Варіант використання ПД – завершення програми

*Таблиця 1.7*

**Варіант використання ПД – завершення програми**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Іденти-**  **фікатор** | **Назва** | **Мета** | **Дійова**  **особа** | **Тип** | **Опис** |
| V\_08 | Завершення програми | Вихід із програми | Користувач | Основний | Можливість у процесі виконання ПД виконати вихід із програми |

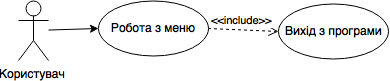


Рис. 1.7. Діаграма варіанту використання в нотації UML: завершення програми

* 1. Нефункціональні вимоги

Нефункціональні вимоги – широкий термін, який охоплює всі вимоги користувача, які не є в явному вигляді функціональними.

Нефункціональними вимогами до користувача у даній КР являються обмеження зі сторони вводу для користувача. У ПД передбачена можливість некоректного вводу за допомогою перевірок на правильний ввід.

РОЗДІЛ 2

СТРУКТУРА ТА ЛОГІКА РОБОТИ ПД

2.1. Логічна структура ПД

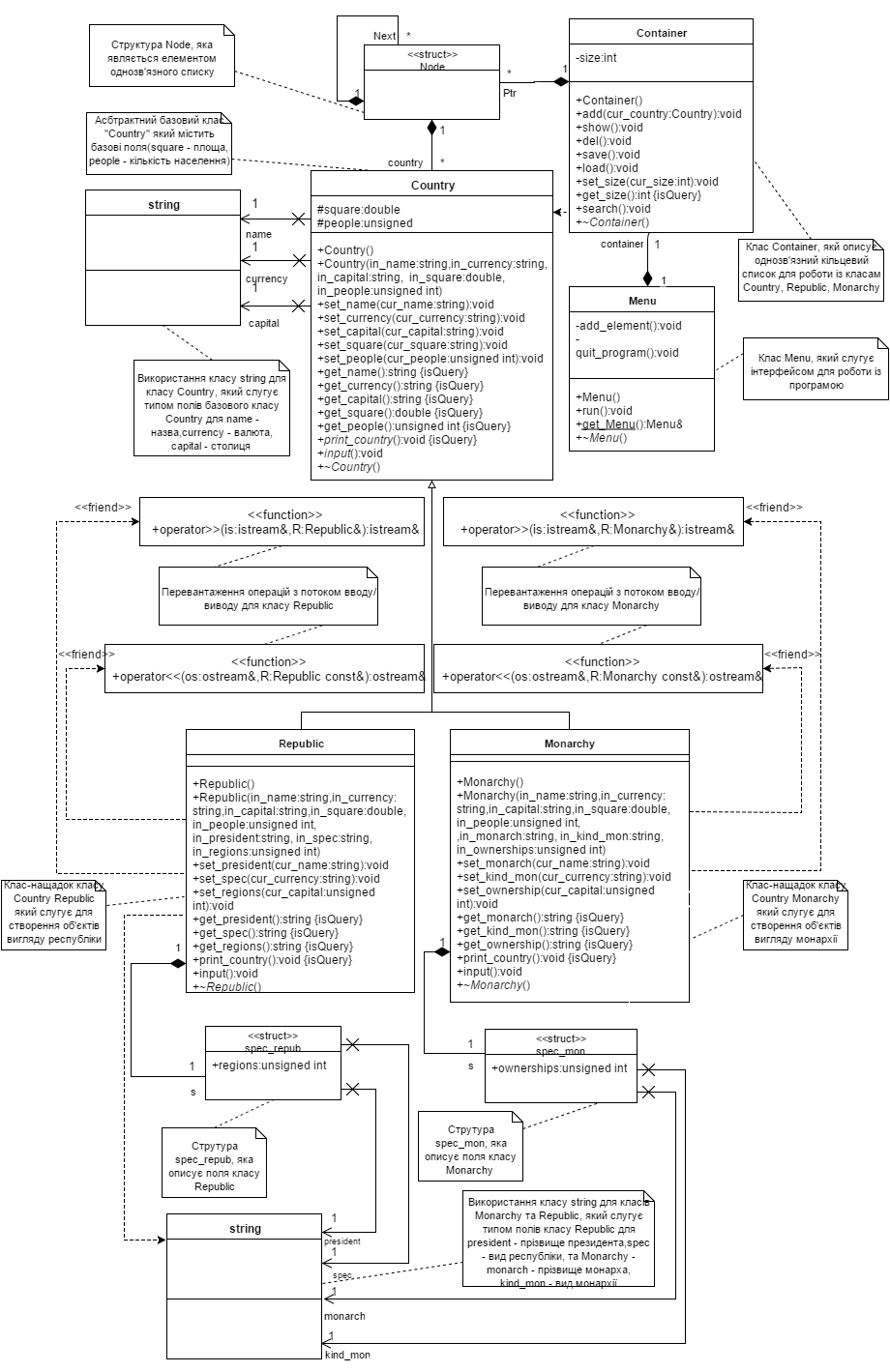
Діаграма класів у нотації UML зображена на рисунку 2.1. Опис класів ПД приведено нижче. (Більша версія у Додатку Б)

Рис. 2.1. Діаграма класів у нотації UML

2.1.1. Клас Country

* 1. Клас Country призначений для визначення сутності «Держава» та є абстрактним базовим класом для класів Republic та Monarchy.
  2. Для класу Country опис захищеного інтерфейсу наведено в таблиці 2.1, відкритого – в таблиці 2.2.

*Таблиця 2.1*

**Поля захищеного інтерфейсу класу Country**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ з/п** | **Назва** | **Тип** | **Призначення** |
| 1 | name | string | Назва держави |
| 2 | currency | string | Валюта |
| 3 | capital | string | Столиця |
| 4 | square | double | Площа |
| 5 | people | unsigned int | Кількість населення |

*Таблиця 2.2*

**Методи відкритого інтерфейсу класу Country**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ з/п** | **Сігнатура** | **Вхідні параметри** | **Повернення значення** | **Призначення** |
| 1 | Country() | - | - | Конструктор за замовчуванням |
| 2 | Country(string in\_name, string in\_currency, string in\_capital, double in\_square, unsigned in\_people) | in\_name - назва держави, in\_currency – валюта, in\_capital – столиця, in\_square – площа, in\_people - населення | - | Конструктор з параметрами |
| 3 | void set\_name(string cur\_name) | cur\_name – назва держави | - | Присвоєння значення про назву держави |
|  |  |  |  | *Продовж. табл. 2.2* |
| **№ з/п** | **Сігнатура** | **Вхідні параметри** | **Повернення значення** | **Призначення** |
| 4 | void set\_currency(string cur\_currency) | cur\_currency - валюта | - | Присвоєння значення про валюту |
| 5 | void set\_capital(string cur\_capital) | cur\_capital - столиця | - | Присвоєння значення про cтолицю |
| 6 | void set\_square(string cur\_square) | cur\_square - площа | - | Присвоєння значення про площу |
| 7 | void set\_people(string cur\_people) | cur\_people – населення | - | Присвоєння значення про населення |
| 8 | string get\_name() const | - | Повернення name | Отримання даних про назву держави |
| 9 | string get\_currency() const | - | Повернення currency | Отримання даних про валюту |
| 10 | string get\_capital() const | - | Повернення capital | Отримання даних про столицю |
| 11 | double get\_square() const | - | Повернення square | Отримання даних про площу |
| 12 | unsigned int get\_people() const | - | Повернення people | Отримання даних про кількість населення |
| 13 | virtual void print\_country() const | - | - | Віртуальна функція виведення даних |
| 14 | virtual void input() | - | - | Віртуальна функція введення даних |
| 15 | virual ~Country=0 | - | - | Віртуальних деструктор |

2.1.2. Клас «Republic»

1. Клас Republic призначений для визначення сутності «Республіка» та є класом-нащадком Country.
2. Для класу Republic опис закритого інтерфейсу наведено в таблиці 2.3, відкритого – в таблиці 2.4.

*Таблиця 2.3*

**Поля закритого інтерфейсу класу Republic**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ з/п** | **Назва** | **Тип** | **Призначення** |
| 1 | s | spec\_repub | Структура з полями класу Republic |

*Таблиця 2.4*

**Методи відкритого інтерфейсу класу Republic**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ п/п** | **Сігнатура** | **Вхідні параметри** | **Повернення значення** | **Призначення** |
| 1 | Republic() | - | - | Конструктор за замовчуванням |
| 2 | Republic(string in\_name, string in\_currency, string in\_capital, double in\_square, unsigned in\_people, string in\_president, string in\_spec, unsigned regions) | in\_name - назва держави, in\_currency – валюта, in\_capital – столиця, in\_square – площа, in\_people – населення, in\_president – президент, in\_spec – вид республіки, in\_regions – кількість областей | - | Конструктор із параметрами |
| 3 | void set\_president(string cur\_president) | cur\_president – президент | - | Присвоєння значення про президента |
|  |  |  | *Продовж. табл. 2.4* | |
| **№ п/п** | **Сігнатура** | **Вхідні параметри** | **Повернення значення** | **Призначення** |
| 4 | void set\_spec(string cur\_spec) | cur\_spec – вид республіки | - | Присвоєння значення про вид республіки |
| 5 | void set\_regions(unsigned int cur\_regions) | cur\_regions – кількість областей | - | Присвоєння значення про кількість областей |
| 6 | string get\_president() const | - | Повернення s.president | Повернення значення про президента |
| 7 | string get\_spec() const | - | Повернення s.spec | Повернення значення про вид республіки |
| 8 | unsigned int get\_regions() const | - | Повернення s.regions | Повернення значення про кількість областей |
| 9 | friend ostream &operator <<(ostream &os, const Republic &R) | Вказівник на потік ofstream та вказівник на один з наслідуваних класів | Вхідний потік | Дружня функція для введення даних |
| 10 | friend istream &operator >>(istream &is, Republic &R) | Вказівник на потік istream та вказівник на один з наслідуваних класів | Вихідний потік | Дружня функція для виведення даних |
| 11 | void print\_country() const | - | - | Виведення даних |
| 12 | void input() | - | - | Введення даних |
| 13 | virtual ~Republic |  |  | Віртуальний деструктор |

2.1.3. Структура «spec\_repub»

1. Структура «spec\_repub» слугує для класу «Republic», призначений для запису даних – президент, вид та кількість областей республіки.
2. Для структури «spec\_repub» опис відкритого інтерфейсу наведено у таблиці 2.5.

*Таблиця 2.5*

**Поля відкритого інтерфейсу структури spec\_repub**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ з/п** | **Назва** | **Тип** | **Призначення** |
| 1 | president | string | Президент |
| 2 | spec | string | Вид республіки |
| 3 | regions | insigned int | Кількість областей |

* + 1. Клас «Monarchy»
  1. Клас Monarchy призначений для визначення сутності «Монархія» та є класом-нащадком Country.
  2. Для класу Monarchy опис закритого інтерфейсу наведено в таблиці 2.6, відкритого – в таблиці 2.1.7.

*Таблиця 2.6*

**Поля закритого інтерфейсу класу Monarchy**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ з/п** | **Назва** | **Тип** | **Призначення** |
| 1 | s | spec\_mon | Структура з полями класу Monarchy |

*Таблиця 2.7*

**Методи відкритого інтерфейсу класу Monarchy**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ з/п** | **Сігнатура** | **Вхідні параметри** | **Повернення значення** | **Призначення** |
| 1 | Monarchy() | - | - | Конструктор за замовчуванням |
|  |  |  | *Продовж. табл. 2.7* | |
| **№ з/п** | **Сігнатура** | **Вхідні параметри** | **Повернення значення** | **Призначення** |
| 2 | Monarchy(string in\_name, string in\_currency, string in\_capital, double in\_square, unsigned in\_people, string in\_monarch, string in\_kind\_mon, unsigned ownerships) | in\_name - назва держави, in\_currency – валюта, in\_capital – столиця, in\_square – площа, in\_people – населення, in\_monarch– монарх, in\_kind\_mon – вид монархії, in\_ownerships– кількість володінь | - | Конструктор із параметрами |
| 3 | void set\_monarch(string cur\_monarch) | cur\_monarch – монарх | - | Присвоєння значення про монарха |
| 4 | void set\_kind\_mon(string cur\_kind\_mon) | cur\_kind\_mon – вид монархії | - | Присвоєння значення про вид монархії |
| 5 | void set\_ownership(unsigned int cur\_ownership) | cur\_ownership – кількість володінь | - | Присвоєння значення про кількість володінь |
| 6 | string get\_monarch() const | - | Повернення s.monarch | Повернення значення про монарха |
| 7 | string get\_kind\_spec() const | - | Повернення s.kind\_mon | Повернення значення про вид монархії |
| 8 | unsigned int get\_ownership() const | - | Повернення s.ownerships | Повернення значення про |
|  |  |  | *Продовж. табл. 2.7* | |
| **№ з/п** | **Сігнатура** | **Вхідні параметри** | **Повернення значення** | **Призначення** |
|  |  |  |  | кількість володінь |
| 9 | friend ostream &operator <<(ostream  &os, const Monarchy &M) | Вказівник на потік ofstream та вказівник на  один з наслідуваних класів | Вхідний потік | Дружня функція для введення даних |
| 10 | friend istream &operator >>(istream &is, Monarchy &M) | Вказівник на потік istream та вказівник на один з наслідуваних класів | Вихідний потік | Дружня функція для виведення даних |
| 11 | void print\_country() const | - | - | Виведення даних |
| 12 | void input() | - | - | Введення даних |
| 13 | virtual ~Monarchy |  |  | Віртуальний деструктор |

* + 1. Структура «spec\_mon»

1. Структура «spec\_mon» слугує для класу «Monarchy», призначений для запису даних – монарха, вид та кількість областей монархії.
2. Для структури «spec\_mon» опис відкритого інтерфейсу наведено у таблиці 2.8.

*Таблиця 2.8*

**Поля відкритого інтерфейсу структури spec\_repub**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ з/п** | **Назва** | **Тип** | **Призначення** |
| 1 | monarch | string | Монарх |
| 2 | kind\_mon | string | Вид монархії |
|  |  |  | *Продовж. табл. 2.8* |
| **№ з/п** | **Назва** | **Тип** | **Призначення** |
| 3 | ownerships | insigned int | Кількість володінь |

* + 1. Структура «Node»

1. Структура «Node» слугує для класу «Container», призначений для визначення контейнеру та містить у собі вказівник на наступний елемент та на державу.
2. Для структури «Node» опис відкритого інтерфейсу наведено у таблиці 2.9.

*Таблиця 2.9*

**Поля відкритого інтерфейсу структури Node**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ з/п** | **Назва** | **Тип** | **Призначення** |
| 1 | country | Country\* | Вказівник на країну |
| 2 | Next | Node\* | Вказівник на наступний елемент |

* + 1. Клас «Container»

1. Клас «Container» слугує як контейнер для зберігання та опрацювання всіх даних.
   1. Для класу «Container» опис закритого інтерфейсу наведено у таблиці 2.10, відкритого – у таблиці 2.11.

*Таблиця 2.10*

**Поля відкритого інтерфейсу класу Container**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Назва** | **Тип** | **Призначення** |
| 1 | Ptr | Node\* | Вказівник на поточний елемент в контейнеру |
| 2 | size | int | Розмір контейнеру |

*Таблиця 2.11*

**Методи закритого інтерфейсу класу Container**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ з/п** | **Сігнатура** | **Вхідні параметри** | **Повернення значення** | **Призначення** |
| 1 | Container() | - | - | Конструктор за замовчуванням |
| 2 | void add(Country \*cur\_country) | \*cur\_country – вказівник на поточну державу | - | Функція додавання обїекту до контейнеру |
| 3 | void show() | - | - | Функція виведення контейнеру |
| 4 | void del() | - | - | Функція видалення контейнеру |
| 5 | void save() | - | - | Функція збереження контейнеру до файлу |
| 6 | void load() | - | - | Функція завантаження контейнеру з файлу |
| 7 | void set\_size(int cur\_size) | cur\_size – розмір контейнеру | - | Функція призначення розміру контейнеру |
| 8 | int get\_size() const | - | - | Функція визначення розміру контейнеру |
| 9 | void search() | - | - | Функція пошуку |
| 10 | virtual ~Container() | - | - | Віртуальний деструктор |

* + 1. Клас «Menu»

1. Клас «Menu» слугує як меню користувача для роботи із програмою.
2. Для класу «Menu» опис закритого інтерфейсу наведено у таблиці 2.12, 2.13, відкритого – у таблиці 2.14.

*Таблиця 2.12*

**Поля відкритого інтерфейсу класу Menu**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ з/п** | **Назва** | **Тип** | **Призначення** |
| 1 | container | Container | Вказівник на поточний елемент в контейнеру |
| 2 | size | int | Розмір контейнеру |

*Таблиця 2.13*

**Методи відкритого інтерфейсу класу Menu**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ з/п** | **Сігнатура** | **Вхідні параметри** | **Повернення значення** | **Призначення** |
| 1 | void add\_element() | - | - | Функція додавання елементу |
| 2 | void quit\_program() | - | - | Функція виходу з програми |

*Таблиця 2.14*

**Методи закритого інтерфейсу класу Menu**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ з/п** | **Сігнатура** | **Вхідні параметри** | **Повернення значення** | **Призначення** |
| 1 | Menu() | - | - | Конструктор за замовчуванням |
| 2 | static Menu &get\_Menu() | - | - | Функція роботи із меню |
| 3 | void run() | - | - | Функція роботи з програмою |
| 4 | virtual ~Menu() | - | - | Віртуальний деструктор |

* 1. Фізична структура ПД

Розподіл класів, змінних, функцій та звонішніх бібліотек, що підключаються по файлах наведено у таблиці 2.15.

*Таблиця 2.15*

**Розподіл класів та інших компонентних структур по файлах**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№**  **з/п** | **Назва файлу** | **Компоненти** | | **Зовнішні компоненти** | |
| **Назва** | **Опис** | **Назва** | **Опис** |
| 1 | Country.h | Country | Визначення абстрактного класу для опису сутності «Країна» | string | Робота зі строками |
| iostream | Робота з потоком |
| fstream | Робота з файлами |
| 2 | Republic.h | Republic | Визначення похідного класу, який є нащадком абстрактного класу «Country» для опису сутності «Республіка» | Country.h | Робота з абстрактним класом Country |
| spec\_repub | Визначення структури, що описує сутність «Параметри республіки» |
| 3 | Monarchy.h | Monarchy | Визначення похідного класу, який є нащадком абстрактного класу | Country.h | Робота з абстрактним класом Country |
|  |  |  |  | *Продовж. табл. 2.15* | |
| **№**  **з/п** | **Назва файлу** | **Компоненти** | | **Зовнішні компоненти** | |
| **Назва** | **Опис** | **Назва** | **Опис** |
|  |  |  | «Country» для опису сутності «Монархія» |  |  |
| spec\_mon | Визначення структури, що описує сутність «Параметри монархії» |
| 4 | Container.h | Container | Визначення класу для опису сутності контейнера за принципом однозв’язного кільцевого списку | Republic.h | Підключення классу Республіка |
| Node | Визначення струк., що описує сутність елемента однозв'язного кільц. списку | Monarchy.h | Підключення классу Монархія |
| 5 | Menu.h | Menu | Визначення класу для опису сутності «Меню користувача», завдяки патерну проектування | Container.h | Робота з контейнером |
| *Продовж. табл. 2.15* | | | | | |
| **№**  **з/п** | **Назва файлу** | **Компоненти** | | **Зовнішні компоненти** | |
| **Назва** | **Опис** | **Назва** | **Опис** |
|  |  |  | Singleton можливе створення лише одного екземпляру даного класу. |  |  |
| 6 | Country.cpp | Country | Реалізація методів класу Country, що визначає сутність Країни | Country.h | Підключення заголовочного файлу |
| Country |
| set\_name |
| set\_currency |
| set\_capital |
| set\_square |
| set\_people |
| get\_name |
| get\_currency |
| get\_capital |
| get\_square |
| get\_people |
| print\_country |
| input |
| ~Country |
| 7 | Republic.cpp | Republic | Реалізація методів класу Republic, що визначає сутність Республіка | Republic.h | Підключення заголовочного файлу |
| Republic |
| set\_president |
| set\_spec |
| set\_regions |
| get\_president |
| get\_spec |
| get\_regions |
| operator>> |
| operator<< |
| print\_country |
| *Продовж. табл. 2.15* | | | | | |
| **№**  **з/п** | **Назва файлу** | **Компоненти** | | **Зовнішні компоненти** | |
| **Назва** | **Опис** | **Назва** | **Опис** |
|  |  | input |  |  |  |
| ~Republic |
| 8 | Monarchy.cpp | Monarchy | Реалізація методів класу Monarchy, що визначає сутність Монархія | Monarchy.h | Підключення заголовочного файлу |
| Monarchy |
| set\_monarch |
| set\_kind\_mon |
| set\_ownership |
| get\_monarch |
| get\_kind\_mon |
| get\_ownership |
| operator>> |
| operator<< |
| print\_country |
| input |
| ~Monarchy |
| 9 | Container.cpp | Container | Реалізація методів класу Container, що визначає контейнер для роботи з програмою | Сontainer.h | Підключення заголовочного файлу |
| add |
| show |
| del |
| save |
| load |
| set\_size |
| get\_size |
| search |
| ~Container |
| 10 | Menu.cpp | Menu | Реалізація методів класу Menu, що визначає інтерфейс | Menu.h | Підключення заголовочного файлу |
| add\_element |
| quit\_program |
| get\_Menu |
|  |  |  |  | *Продовж. табл. 2.15* | |
| **№**  **з/п** | **Назва файлу** | **Компоненти** | | **Зовнішні компоненти** | |
| **Назва** | **Опис** | **Назва** | **Опис** |
|  |  | run | для роботи з програмою |  |  |
| ~Menu |
| 11 | Main.cpp | Main | Головна функція програмного додатку | Menu.h | Підключення заголовочного файлу Menu.h |

* 1. Логіка роботи ПД

Визначення часової послідовності подій ПД наведено у рис. 2.2.

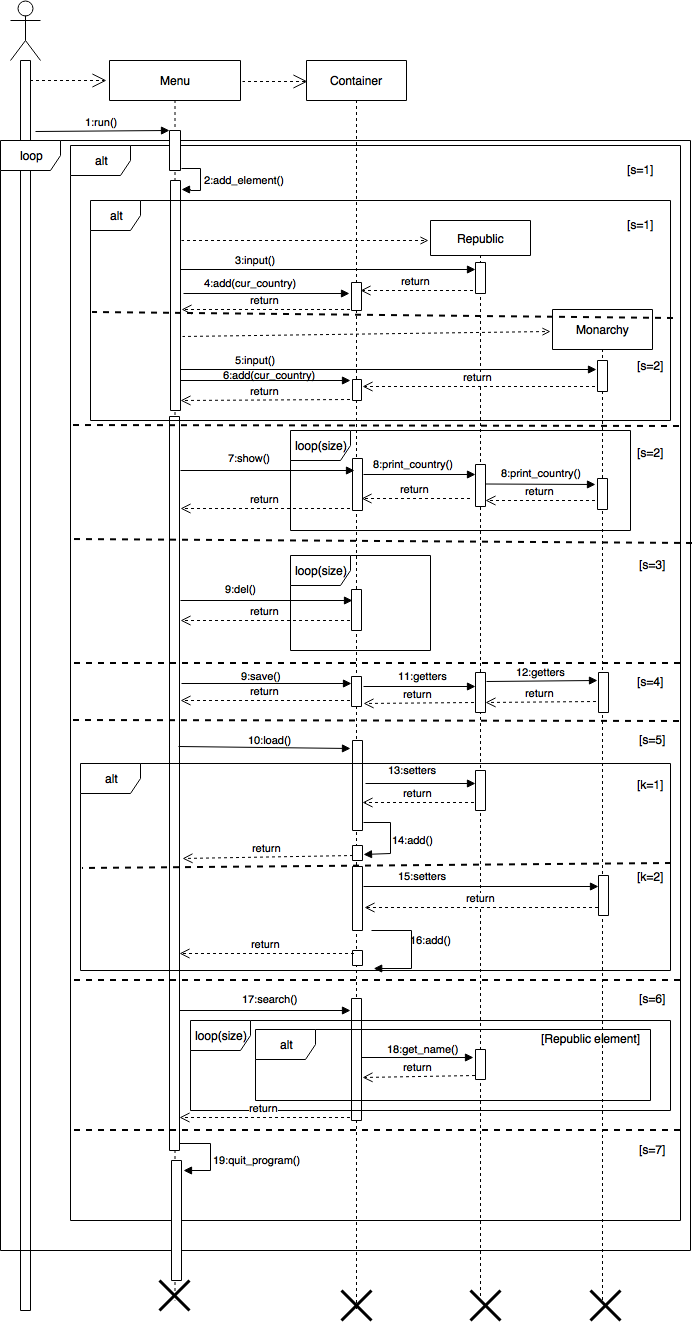


Рис. 2.2. Визначення часової послідовності подій у нотації UML:

1 – створення об’єкта;

2 – виведення даних на екран;

3 - видалення всіх об’єктів;

4 – збереження колекції у файл;

5 – Завантаження колекції з файлу;

6 – Пошук об’єкта за запитом;

7 – Вихід із программ

РОЗДІЛ 3

ТЕСТУВАННЯ ПД

3.1. Сценарій тестування ПД

Таблицями 3.1-3.8 представлено функціональне тестування. Таблицею 3.9 - нефункціональне тестування.

*Таблиця 3.1*

**Сценарій функціонального тестування меню програми**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
| TF\_1 | Меню програми | V\_01 | Запустіть програму | 1) Виберіть один з пунктів меню від 0 до 7  2) Введіть не коректні дані. Наприклад, букву заміть числового значення | 1) Вибраний пункт меню запускається  2) Виведено повідомлення про некоректність вводу даних | Виконайте тести TF\_2-TF\_8 |

*Таблиця 3.2*

**Сценарій функціонального тестування створення об’єктів програми**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
| TF\_2 | Створення об’єктів | V\_02 | Запустіть програму | 1) З меню натисніть 1  2) Виберіть 1 або 2 (1 – створити об’єкт Республіка, 2 – Монархія)  3) Введіть дані  4) Поверніться до меню, | 1) Виводиться запит вибору між об’єктами  2) Виводиться запит для вводу даних  3) Виводиться затримка екрану  4) Аналогічно 1-3 | Виконайте тест TF\_3 |
|  |  |  |  |  | *Продовж. табл. 3.2* | |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
|  |  |  |  | натисніть 1, далі кроки 1-3  5) При створенні об’єкта введіть не корректні дані | 5) Повідомлення про неправильне введення даних |  |

*Таблиця 3.3*

**Сценарій функціонального тестування перегляду колекції програми**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
| TF\_3 | Перегляд колекції | V\_03 | Запустіть програму | 1. Натисніть 2 у меню | Дані, що вводилися у попередніх пунктах, виведені повність та правильно. Якщо колекція пуста то виведено відповідне повідомлення | Виконайте тест TF\_5 для збереження об’єктів у файл, або TF\_4 для видалення колекції |

*Таблиця 3.4*

**Сценарій функціонального тестування очищення колекції**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
| TF\_4 | Очищення колекції | V\_04 | Запустіть програму | 1) Виконайте тест TF\_2  2) Натисніть 3 у меню | Виведено повідомлення про успішне видалення елементів. Для перевірки | Виконайте тест TF\_6 для завантаженняоб’єктів із файлу, або TF\_2 для |
|  |  |  |  |  | *Продовж. табл. 3.4* | |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
|  |  |  |  |  | виконайте тест TF\_3 у меню | повторного заповнення |

*Таблиця 3.5*

**Сценарій функціонального тестування збереження до файлу колекції**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
| TF\_5 | Збереження до файлу | V\_05 | Запустіть програму | 1) Виконайте тест TF\_2  2) Виберіть 4 | Виведено повідомлення про успішний запис до файлу | Виконайте тест TF\_4. Цим очиститься колекція |

*Таблиця 3.6*

**Сценарій функціонального тестування завантаження колекції з файлу**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
| TF\_6 | Завантаження з файлу | V\_06 | Запустіть програму | 1) Виконайте тести TF\_2, TF\_4, TF\_5  2) З меню оберіть 5 | Дані з файлу внесені до колекції. Для перевірки виконайте тест TF\_3 меню | Виконайте тест TF\_4. Цим очиститься колекція |

*Таблиця 3.7*

**Сценарій функціонального тестування пошуку**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
| TF\_7 | Виконання запиту для пошуку | V\_07 | Запустіть програму | 1) Виконайте тест TF\_2, TF\_3  2) З меню натисніть 6  3) Введіть назву республіки  4) Повторити 1-2, ввести не коректні дані | 1) Заповнення колекції  2) Запит для вводу Республіки  3) Успішне введення, вивід площі республіки  4) Вивід повідомлення про помилку | Виконайте тест TF\_4. Цим очиститься колекція |

*Таблиця 3.8*

**Сценарій функціонального тестування виходу з програми**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
| TF\_8 | Вихід з програми | V\_08 | Запустіть програму | 1) Виконайте тест TF\_1  2) З меню оберіть 7  3) Надається можливість повернутись до роботи, ввести Y або N  4) На кроці 3 ввести не коректні дані | 1) Заповнення колекції  2)Виведено запит для вибору повернутись до роботи чи ні  3) Успішний ввід. Продовження або завершення роботи  4)Повідомлення про помилку | - |

*Таблиця 3.9*

**Сценарій нефункціонального тестування**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ідент.** | **Назва** | **Ід. пов’яз.вимоги** | **Передумова** | **Тестові кроки** | **Очікуваний результат** | **Післяумова виконання** |
| NT\_01 | Запуск з різних місць | NV\_01 | Скопіюйте exe-файл  програми | 1) Збережіть даний файл на диску D у будь-якій існуючій папці або створіть нову. Запустіть збережений файл  2) Виконайте аналогічні дії, зберігаючи файл в інших місцях (Мої документи, диск С,…) | 1) exe-файл програми, що зберігається на будь-якому диску (розташовується за довільною адресою), відкривається успішно.  2) Програма працює коректно | Видаліть збережені exe-файли з дисків, де вони не потрібні |

3.2.Тестові дані

Тестові дані представлено діаграмою об’єктів в нотації UML на рисунку 3.1. та скріншотом файлу на рисунках 3.2.,3.3.,3.4.

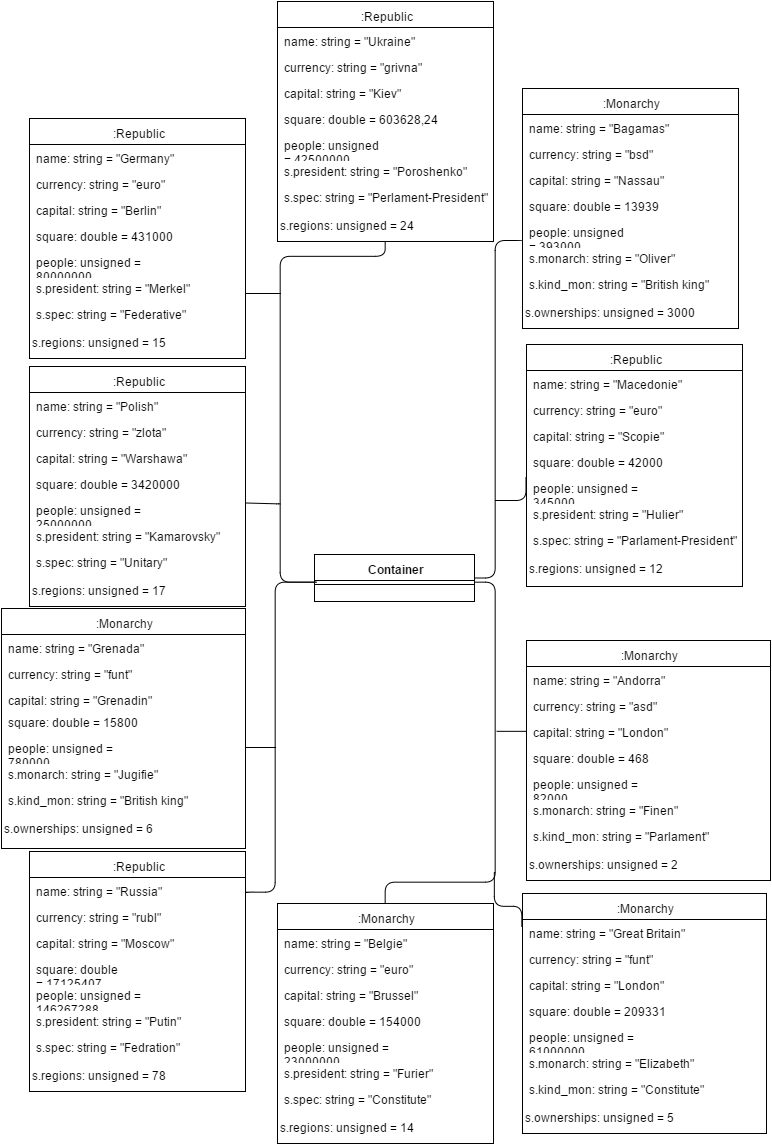


Рис. 3.1. Діаграма об’єктів у нотації UML

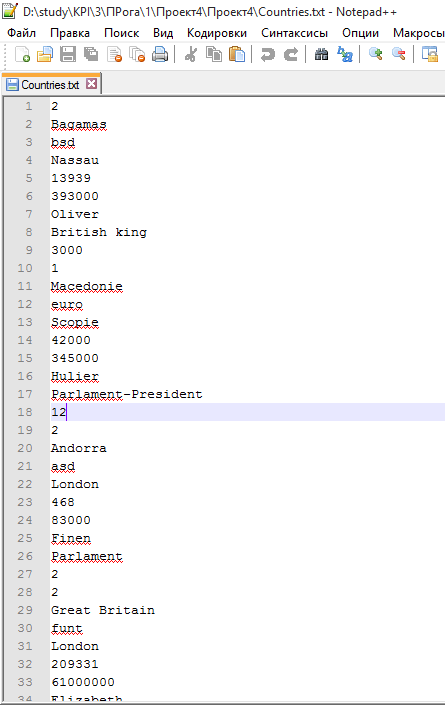


Рис. 3.2. Дані тестування ПД

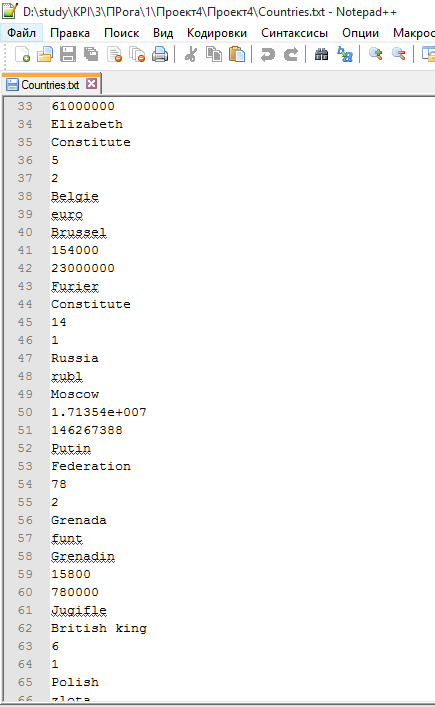


Рис. 3.3. Дані тестування ПД

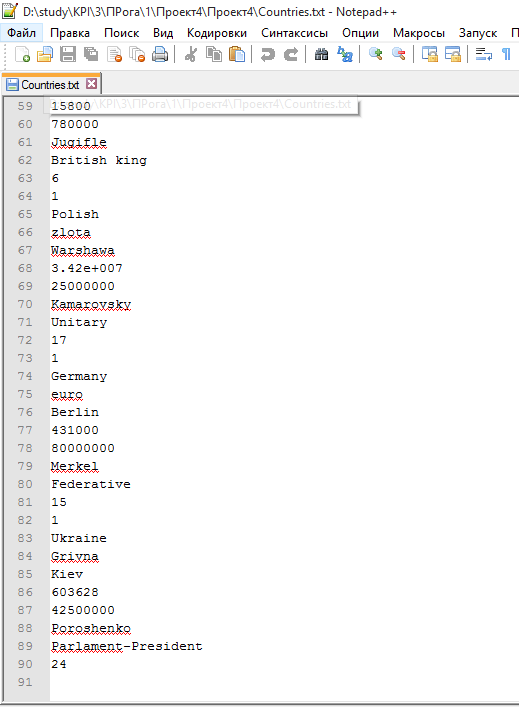


Рис. 3.4. Дані тестування ПД

3.3. Поєтапні результати роботи ПД

Програма на початку виводить Меню користувача для роботи із програмою (рис. 3.5.). На рисунку 3.6. показаний не коректний ввід даних. (Сценарій використання ПД TF\_1)



Рис. 3.5. Головне меню користувача Рис. 3.6. Не коректний ввід

Далі показаний сценарій роботи програми при заповненні елементів колекції. На рисунку 3.7. показано вибір між об’єктами для подальшого заповнення даними (рис. 3.8., 3.9.). Рисунки 3.10., 3.11. показують не коректний ввід даних. (Сценарій використання ПД TF\_2)



Рис. 3.7. Вибір типу об’єкта

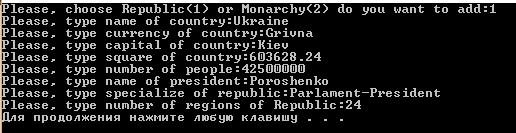


Рис. 3.8. Введення даних про Республіку



Рис. 3.9. Введення даних про монархію



Рис. 3.10. Не коректний ввід при виборі

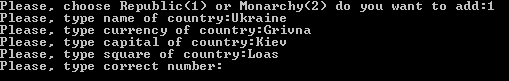


Рис. 3.11. Не коректний ввід при заповненні

На рисунку 3.12. показано використання пункту Меню користувача – Вивід даних на екран. (Сценарій використання ПД TF\_3)

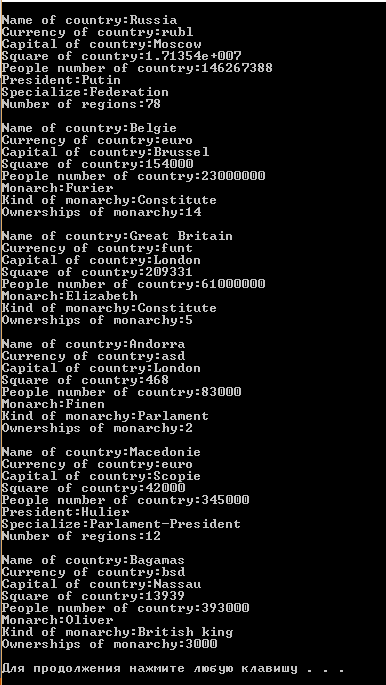


Рис. 3.12. Вивід даних на екран

На рисунку 3.13. показано варіант використання Меню користувача – Очищення колекції. (Сценарій використання ПД TF\_4)



Рис. 3.13. Очищення колекції

На рисунку 3.14. показано результат збереження колекції до файлу. (Сценарій використання ПД TF\_5)



Рис. 3.14. Збереження до файлу

На рисунку 3.15. показано результат завантаження колекції із файлу. (Сценарій використання ПД TF\_6)



Рис. 3.15. Завантаження з файлу

На рисунку 3.16. показано результат виконання пошуку площі заданої республіки. (Сценарій використання ПД TF\_7)



Рис. 3.16. Виконання запиту

На рисунку 3.17. показано результат вибору пункту Меню користувача Вихід із програми: ми бачимо запит для повернення до меню користувача. На рисунку 3.18. зображено не коректний ввід даних. (Сценарій використання ПД TF\_8)



Рис. 3.17. Питання при виході про повернення до роботи

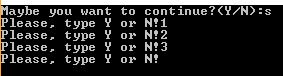


Рис. 3.18. Не коректний ввід

На рисунку 3.19. показано результат запуску файлу програми з розширенням .exe з диску С. На рисунку 3.20. – з диску D. (Сценарій використання ПД NT\_1)

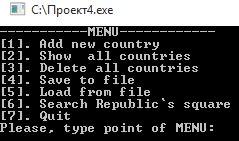
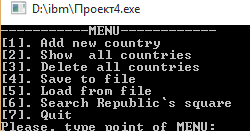
 

Рис. 3.19. Запуск з диску C Рис. 3.20. Запуск з диску D

ВИСНОВКИ

У даній курсовій роботі було проведено створення ПД на онові мови високого рівня C++. Було проведено проектування ПД, його реалізація та тестування.

У процесі постановки завдання чітко було сформульовано призначення ПД і визначені основні функціональні, експлуатаційні та технологічні вимоги до нього.

Результатом проектування є детальна модель ПД. Реалізація включає в себе написаний код, в якому було використано основні принципи ООП. Розроблений ПД поданий у вигляді у вигляді сукупності об’єктів, у процесі взаємодії яких за допомогою передачі повідомлень відбувається виконання необхідних функцій (обє’ктна декомпозиція). Так як розроблюване за допомогою об’єктного підходу ПЗ дуже складне, то для його опису було використано спеціальну мов – універсальну мову моделювання UML.

Під час виконання курсової роботи було виконано безліч задач та отримано конкретні результати. Було написано код програми для реалізації поставленого завдання, розроблено діаграми у нотації UML, таблиці.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Б. Страуструп. Язык программирования С++. Специальное издание. Пер. с англ. - М.: Издательство Бином, 2012 г. - 1136 с.: ил.
2. Джесс Либерти. Освой самостоятельно С++ за 21 день: 3-е изд., пер. С англ. : Уч. пос. – М. : Издательский дом «Вильямс», 2000. – 816 с.
3. Г. Шилдт. Полный справочник по С++. Пер. с англ. – М.: Издательский дом "Вильямс", 2006 – 800 с.

ДОДАТОК А

Лістинг програми

/\*

Country.h

Coursework

Zavrazhniy D.K. BS-41

Interface to abstract class "Country"

\*/

#ifndef KURSOVAYA\_COUNTRY\_H

#define KURSOVAYA\_COUNTRY\_H

#include <iostream> //підлючаємо бібліотеку для роботи з потоком

#include <string> //підлючаємо бібліотеку для роботи з рядкам

#include <fstream> //підключаємо бібліотеку для роботи із потоком

using namespace std;

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

//Абстрактний клас Country

class Country

{

protected:

string name;

string currency;

string capital;

double square;

unsigned people;

public:

Country();

Country(string, string, string, double, unsigned);

void set\_name(string);

void set\_currency(string);

void set\_capital(string);

void set\_square(double);

void set\_people(unsigned);

string get\_name() const;

string get\_currency() const;

string get\_capital() const;

double get\_square() const;

unsigned get\_people() const;

virtual void print\_country() const;

virtual void input();

virtual ~Country()=0; //визначення класу абстрактним

};

#endif //KURSOVAYA\_COUNTRY\_H

/\*

Country.cpp

Coursework

Zavrazhniy D.K. BS-41

Implementation to abstract class "Country"

#include "Country.h"

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

//Конструктор класу Country

Country::Country()

{

}

Country::Country(string in\_name, string in\_currency, string in\_capital, double in\_square, unsigned in\_people) : name(in\_name)

, currency(in\_currency)

, capital(in\_capital)

, square(in\_square)

, people(in\_people) //список ініціалізації

{

}

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

//Функція для встановлення name(назви країни)

void Country::set\_name(string cur\_name)

{

name = cur\_name;

}

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

//Функція для встановлення currency(валюти країни)

void Country::set\_currency(string cur\_currency)

{

currency = cur\_currency;

}

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

//Функція для встановлення capital(столиці країни)

void Country::set\_capital(string cur\_capital)

{

capital = cur\_capital;

}

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

//Функція для встановлення square(площі країни)

void Country::set\_square(double cur\_square)

{

square = cur\_square;

}

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

//Функція для встановлення people(кількості населення країни)

void Country::set\_people(unsigned cur\_people)

{

people = cur\_people;

}

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

//Функція отримання name

string Country::get\_name() const

{

return name;

}

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

//Функція отримання currency

string Country::get\_currency() const

{

return currency;

}

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

//Функція отримання capital

string Country::get\_capital() const

{

return capital;

}

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

//Функція отримання square

double Country::get\_square() const

{

return square;

}

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

//Функція отримання people

unsigned Country::get\_people() const

{

return people;

}

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

//Функція виведення інформації про країну

void Country::print\_country() const

{

cout << "Name of country:" << name << endl;

cout << "Currency of country:" << currency << endl;

cout << "Capital of country:" << capital << endl;

cout << "Square of country:" << square << endl;

cout << "People number of country:" << people << endl;

}

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

//Функція введення інформації про країну

void Country::input()

{

cin.get();

cout << "Please, type name of country:";

getline(cin,name);

cout << "Please, type currency of country:";

getline(cin,currency);

cout << "Please, type capital of country:";

getline(cin,capital);

cout << "Please, type square of country:";

while (!(cin >> square) || (cin.peek() != '\n') || (square < 0)) /\*перевірка на

правильність вводу\*/

{

cin.clear();

while (cin.get() != '\n');

cout << "Please, type correct number:";

}

cout << "Please, type number of people:";

while (!(cin >> people) || (cin.peek() != '\n') || cin.fail()) /\*перевірка на

правильність вводу\*/

{

cin.clear();

while (cin.get() != '\n');

cout << "Please, type correct number:";

}

}

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

//Деструктор класу Country

Country::~Country()

{

}

/\*

Republic.h

Coursework

Zavrazhniy D.K. BS-41

Interface to class "Republic"

\*/

#pragma once

#include "Country.h"

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

//Структура spec\_repub для класу Republic

struct spec\_repub

{

string president;

string spec;

unsigned regions;

};

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

//Клас Republic - нащадок класу Country

class Republic: public Country

{

private:

spec\_repub s; //об'єкт структури spec\_repub

public:

Republic();

Republic(string, string, string, double, unsigned, string, string, unsigned);

void set\_president(string);

void set\_spec(string);

unsigned get\_regions() const;

void set\_regions(unsigned);

string get\_spec() const;

string get\_president() const;

void print\_country() const;

friend ostream &operator <<(ostream &os, const Republic &R);

friend istream &operator >>(istream &is, Republic &R);

void input();

virtual ~Republic();

};

/\*

Republic.cpp

Coursework

Zavrazhniy D.K. BS-41

Implementation to class "Republic"

\*/

using namespace std;

#include "Republic.h"

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

//Конструктор за замовчванням класу Republic

Republic::Republic()

{

}

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

//Конструктор з параметрамикласу Republic

Republic::Republic(string in\_name, string in\_currency, string in\_capital, double in\_square, unsigned in\_people

, string in\_president, string in\_spec, unsigned in\_regions) : Country(in\_name, in\_currency, in\_capital, in\_square, in\_people)

{

s.president = in\_president;

s.spec = in\_spec;

s.regions = in\_regions;

}

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

//Функція встановлення president(президента країни)

void Republic::set\_president(string cur\_president)

{

s.president = cur\_president;

}

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

//Функція встановлення s.spec(виду республіки)

void Republic::set\_spec(string cur\_spec)

{

s.spec = cur\_spec;

}

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

//Функція встановлення s.regions(кількості регіонів)

void Republic::set\_regions(unsigned cur\_reg)

{

s.regions = cur\_reg;

}

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

//Функція отримання s.regions(кількості регіонів)

unsigned Republic::get\_regions() const

{

return s.regions;

}

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

//Функція отримання s.spec(виду республіки)

string Republic::get\_spec() const

{

return s.spec;

}

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

//Функція отрмання president

string Republic::get\_president() const

{

return s.president;

}

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

//Функція виводу на екран інформації про країну

void Republic::print\_country() const

{

Country::print\_country();

cout << "President:" << s.president << endl;

cout << "Specialize:" << s.spec << endl;

cout << "Number of regions:" << s.regions << endl << endl;

}

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

//Функція введення інформації про країну

void Republic::input()

{

Country::input();

cin.get();

cout << "Please, type name of president:";

getline(cin,s.president);

cout << "Please, type specialize of republic:";

getline(cin,s.spec);

cout << "Please, type number of regions of Republic:";

while (!(cin >> s.regions) || (cin.peek() != '\n') || cin.fail()) /\*перевірка на

правильність вводу\*/

{

cin.clear();

while (cin.get() != '\n');

cout << "Please, type correct number:";

}

}

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

//Перегрузка оператора вводу

ostream &operator <<(ostream &os, const Republic &R)

{

os << 1 << endl << R.name << endl << R.currency << endl << R.capital << endl << R.square << endl << R.people << endl << R.s.president << endl << R.s.spec << endl << R.s.regions << endl;

return os;

}

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

//Перегрузка оператора виводу

istream &operator >>(istream &is, Republic &R)

{

getline(is, R.name);

getline(is, R.currency);

getline(is, R.capital);

is >> R.square;

is >> R.people;

is.get();

getline(is, R.s.president);

getline(is, R.s.spec);

is >> R.s.regions;

return is;

}

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

//Деструктор класу Republic

Republic::~Republic()

{

}

/\*

Monarchy.h

Coursework

Zavrazhniy D.K. BS-41

Interface to class "Monarchy"

\*/

#ifndef KURSOVAYA\_MONARCHY\_H

#define KURSOVAYA\_MONARCHY\_H

#include "Country.h"

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

//Структура spec\_mon для класу Monarchy

struct spec\_mon

{

string monarch;

string kind\_mon;

unsigned ownerships;

};

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

//Клас Monarchy - нащадок класу Country

class Monarchy: public Country

{

private:

spec\_mon s; //об'єкт структури spec\_mon

public:

Monarchy();

Monarchy(string, string, string, double, unsigned, string, string, unsigned);

void set\_monarch(string);

void set\_ownership(unsigned);

void set\_kind\_mon(string);

string get\_kind\_mon() const;

unsigned get\_ownership() const;

string get\_monarch() const;

void print\_country() const;

friend ostream &operator <<(ostream &os, const Monarchy &M);

friend istream &operator >>(istream &is, Monarchy &M);

void input();

virtual ~Monarchy();

};

#endif //KURSOVAYA\_MONARCHY\_H

/\*Monarchy.cpp

Coursework

Zavrazhniy D.K. BS-41

Implementation to class "Monarchy"

\*/

#include "Monarchy.h"

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

//Конструктор за замовчуванням класу Monarchy

Monarchy::Monarchy()

{

}

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

//Конструктор із параметрами класу Monarchy

Monarchy::Monarchy(string in\_name, string in\_currency, string in\_capital, double in\_square, unsigned in\_people

, string in\_monarch, string in\_kind\_mon, unsigned in\_ownerships): Country(in\_name, in\_currency, in\_capital, in\_square, in\_people)

{

s.monarch = in\_monarch;

s.kind\_mon = in\_kind\_mon;

s.ownerships = in\_ownerships;

}

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

//Функція встановлення monarch(ім’я монарха країни)

void Monarchy::set\_monarch(string cur\_monarch)

{

s.monarch = cur\_monarch;

}

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

//Функція встановлення ownerships(володіння)

void Monarchy::set\_ownership(unsigned cur\_owner)

{

s.ownerships = cur\_owner;

}

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

//Функція встановлення kind\_mon(виду монархії)

void Monarchy::set\_kind\_mon(string cur\_mon)

{

s.kind\_mon = cur\_mon;

}

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

//Функція отримання kind\_mon(виду монархії)

string Monarchy::get\_kind\_mon() const

{

return s.kind\_mon;

}

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

//Функція отримання ownerships(володіння)

unsigned Monarchy::get\_ownership() const

{

return s.ownerships;

}

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

//Функція отримання monarch

string Monarchy::get\_monarch() const

{

return s.monarch;

}

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

//Функція виводу на екран інформації про країну

void Monarchy::print\_country() const

{

Country::print\_country();

cout << "Monarch:" << s.monarch << endl;

cout << "Kind of monarchy:" << s.kind\_mon << endl;

cout << "Ownerships of monarchy:" << s.ownerships << endl << endl;

}

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

//Функція введення інформації про країну

void Monarchy::input()

{

Country::input();

cin.get();

cout << "Please, type name of monarch:";

getline(cin,s.monarch);

cout << "Please, type kind of monarch:";

getline(cin, s.kind\_mon);

cout << "Please, type number of ownerships: ";

while (!(cin >> s.ownerships) || (cin.peek() != '\n') || cin.fail()) /\*перевірка на

правильність вводу\*/

{

cin.clear();

while (cin.get() != '\n');

cout << "Please, type correct number:";

}

}

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

//Перегрузка операції вводу

ostream &operator <<(ostream &os, const Monarchy &M)

{

os << 2 << endl << M.name << endl << M.currency << endl << M.capital << endl << M.square << endl << M.people << endl << M.s.monarch << endl << M.s.kind\_mon << endl << M.s.ownerships << endl;

return os;

}

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

//Перегрузка операції виводу

istream &operator >>(istream &is, Monarchy &M)

{

getline(is, M.name);

getline(is, M.currency);

getline(is, M.capital);

is >> M.square;

is >> M.people;

is.get();

getline(is, M.s.monarch);

getline(is, M.s.kind\_mon);

is >> M.s.ownerships;

return is;

}

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

//Деструктор класу Monarchy

Monarchy::~Monarchy()

{

}/\*

Container.h

Coursework

Zavrazhniy D.K. BS-41

Interface to class "Container"

\*/

#ifndef KURSOVAYA\_CONTAINER\_H

#define KURSOVAYA\_CONTAINER\_H

#include "Monarchy.h"

#include "Republic.h"

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

//Структура Node для роботи із однозв’язним кільцевим списком

struct Node

{

Country \*country;

Node \*Next;

};

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

//Клас container для роботи із однозв’язним кільцевим списком

class Container

{

private:

Node \*Ptr; //вказівник на поточний елемент

int size;

public:

Container();

void add(Country\*);

void show();

void del();

void save();

void load();

void set\_size(int);

int get\_size() const;

void search();

virtual ~Container();

};

#endif //KURSOVAYA\_CONTAINER\_H/\*

Container.cpp

Coursework

Zavrazhniy D.K. BS-41

Implementation to class "Container"

\*/

#include "Container.h"

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

//Конструктор класу container

Container::Container():Ptr(NULL)

,size(0)

{

}

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

//Функція додавання елементу до списку

void Container::add(Country \*cur\_country)

{

size++;

Node \*temp = new Node; //створення тимчасового вказівника

temp->country = cur\_country;

if(!Ptr)

temp->Next=temp;

else

{

temp->Next=Ptr->Next;

Ptr->Next = temp;

} //створення кільця

Ptr=temp;

}

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

//Функція для виведення всього списку на екран

void Container::show()

{

Node \*tempHead=Ptr;

int tsize=size;

if (tsize == 0)

cout << "There are not countries to show";

else

cout << "Countries:\n\n";

while (tsize!=0)

{

tempHead->country->print\_country();

tempHead=tempHead->Next;

tsize--;

}

}

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

//Функція видалення всього списку

void Container::del()

{

if (Ptr == NULL) // перевірка на порожність списку

cout << "There are not countries to remove";

else

{

cout << "The countries are removed:\n\n";

while (Ptr != NULL) //допоки поточний елемент існує

if (Ptr->Next == NULL) //якщо елемент один

Ptr = NULL;

else

{

Node \*temp; //створення тимчасового вказівника

while (size != 0)

{

temp = Ptr;

Ptr = Ptr->Next;

temp->country->print\_country();

cout << endl;

delete temp->country; //видалення

delete temp;

size--;

}

Ptr = NULL;

}

}

Node \*temp=new Node;

while (size != 0)

{

temp=Ptr;

Ptr=Ptr->Next;

cout<<"Deleted: \n";

temp->country->print\_country();

size--;

delete temp;

}

}

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

//Функція встановлення розміру списку

void Container::set\_size(int cur\_size)

{

size = cur\_size;

}

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

//Функція отримання розміру списку

int Container::get\_size() const

{

return size;

}

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

//Деструктор класу container

Container::~Container()

{

}

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

//Функція збереження контейнеру до файлу

void Container::save()

{

if(Ptr==NULL)

{

system("cls");

cout<<"Countries is not found!";;

}

else

{

ofstream Countries("Countries.txt"); //створення потоку до файлу Country.txt

Node \*k;

for(int i=0; i<size; i++)

{

if(Countries.is\_open()==0) //перевірка на працездатність файлу

{

cout<<"Failed to open file";

break;

}

k=Ptr;

for(int j=0; j<size-(i+1); j++) //заповнення елементами файл циклом

k=k->Next;

if(typeid(Republic)==typeid(\*(k->country)))

Countries<<(\*((Republic\*)(k->country)));

if (typeid(Monarchy) == typeid(\*(k->country)))

Countries<<(\*((Monarchy\*)(k->country)));

}

Countries.close();

}

cout << "Saved" << endl;

}

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

//Функція загрузки контейнеру з файлу

void Container::load()

{

int k;

ifstream Countries("Countries.txt"); //створення потоку до файлу Contries.txt

while (!Countries.eof())

{

if (Countries.is\_open() == 0) //перевірка на працездатність

{

cout << "failed to open file";

system("pause");

break;

}

k = 0;

Countries >> k;

Countries.get();

if (k == 1) //перевірка на відношення файлу до потрібного класу

{

Republic \*B = new Republic;

Countries >> (\*B);

add(B); //заповнення контейнеру файлами

}

if (k == 2)

{

Monarchy \*B = new Monarchy;

Countries >> (\*B);

add(B);

}

}

cout << "Loaded" << endl;

}

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

//Функція пошуку

void Container::search()

{

string cur\_name;

if (Ptr == NULL)

cout << "No countries found";

else

{

int temp = size, check = 0;

Node \*node\_temp = Ptr;

cout << "Please, enter Republic to find it square:";

cin.get();

getline(cin, cur\_name);

while (temp != 0)

{

if (typeid(\*(node\_temp->country)) == typeid(Republic)) /\*перевірка на

правильність вводу\*/

if (((node\_temp->country)->get\_name()) == (cur\_name))

{

cout << "Square of Republic " << cur\_name << " is " << node\_temp->country->get\_square() << endl;

check++;

}

temp--;

node\_temp = node\_temp->Next;

}

if (check == 0)

cout << "Country is not found!" << endl;

}

}

/\*

Menu.h

Coursework

Zavrazhniy D.K. BS-41

Interface to class "Menu"

\*/

#ifndef KURSOVAYA\_MENU\_H

#define KURSOVAYA\_MENU\_H

#include "Country.h"

#include "Container.h"

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

//Клас для роботи із програмою

class Menu

{

private:

Container container;

void add\_element();

void quit\_program();

public:

static Menu &get\_Menu(); //використання паттерну singleton

Menu();

void run();

virtual ~Menu();

};

#endif //KURSOVAYA\_MENU\_H

/\*

Menu.cpp

Coursework

Zavrazhniy D.K. BS-41

Implementation to class "Menu"

\*/

#include "Menu.h"

Menu &Menu::get\_Menu()/\*функція що використовується патерном проектування

Singletone\*/

{

static Menu m\_Menu;

return m\_Menu;

}

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

//Конструктор класу Menu

Menu::Menu()

{

}

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

//Функція для роботи з програмою

void Menu::run()

{

int s;

do {

cout << "-----------MENU------------\n";

cout << "[1]. Add new country\n";

cout << "[2]. Show all countries\n";

cout << "[3]. Delete all countries\n";

cout << "[4]. Save to file\n";

cout << "[5]. Load from file\n";

cout << "[6]. Search Republic`s square\n";

cout << "[7]. Quit\n";

cout << "Please, type point of MENU:";

while (!(cin >> s) || (cin.peek() != '\n')) {

cin.clear();

while (cin.get() != '\n');

cout << "Please, type number:";

}

switch (s)

{

case 1:

{

system("cls");

add\_element();

system("pause");

system("cls");

break;

}

case 2:

{

system("cls");

container.show();

system("pause");

system("cls");

break;

}

case 3:

{

system("cls");

container.del();

system("pause");

system("cls");

break;

}

case 4:

{

system("cls");

container.save();

system("pause");

system("cls");

break;

}

case 5:

{

system("cls");

container.load();

system("pause");

system("cls");

break;

}

case 6:

{

system("cls");

container.search();

system("pause");

system("cls");

break;

}

case 7:

{

system("cls");

quit\_program();

system("pause");

system("cls");

break;

}

default:

{

system("cls");

cout << "Please, type correct point of MENU (number from 1 to 6)!";

break;

}

}

}while(s);

}

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

//Функція вибору між формами державного устрою, і переходу до

//функції контейнера заповнення інформації про елемент

void Menu::add\_element()

{

Country \*cur\_country;

int s;

cout << "Please, choose Republic(1) or Monarchy(2) do you want to add:";

while (!(cin >> s) || (cin.peek() != '\n') || s<1 || s>2) //перевірка на правильність вводу

{

cin.clear();

while (cin.get() != '\n');

cout << "Please, type correct number(1 or 2):";

}

if(s == 1) //перевірка на бажаний об'єкт для додавання

{

cur\_country = new Republic;

cur\_country->input();

container.add(cur\_country);

}

else

{

cur\_country = new Monarchy;

cur\_country->input();

container.add(cur\_country);

}

}

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

//Функція виходу з програми

void Menu::quit\_program()

{

char a;

cout << "Maybe you want to continue?(Y/N):";

do

{

cin >> a;

if (a == 'N' || a == 'n')

break;

if (a == 'Y' || a == 'y')

run();

else

cout << "Please, type Y or N!";

} while (a != 'Y' || a != 'y' || a != 'N' || a != 'n');

system("cls");

exit(0);

}

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

//Деструктор класу Menu

Menu::~Menu()

{

}

/\*

Main.cpp

Coursework

Zavrazhniy D.K. BS-41

Implementation of main function"

\*/

#include "Menu.h"

using namespace std;

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

//Функція основної програми

int main()

{

Menu::get\_Menu().run();//створюємо об'єкт класу "Menu"

system("pause");

}