rr

**МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

**Лабораторна робота №5**

**з дисципліни “ Основи програмування”**

**тема “С++ STL класи”**

|  |  |  |
| --- | --- | --- |
| **Виконав**  **студент I курсу**  **групи КП-83**  **Мортіков Владислав Євгенович**  **варіант №16** |  | **Перевірив**  **“\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_”  2019  р.**  **викладач**  **Гадиняк Руслан Анатолійович**  **(прізвище, ім’я, по батькові)** |

**Київ 2019**

### **Мета**

Навчитись використовувати стандартні шаблонні контейнери C++ STL.

### **Загальні вимоги до завдання**

Визначення та позначки:

1. **MainEntity** - позначка основної сутності з [лабораторної роботи №2](https://progbase.herokuapp.com/modules/progbase2/tasks/lab2), в коді має називатись як англійська версія назви типу даних, наприклад, Student.
2. **ExtraEntity** - позначка додаткової сутності, тип сутності отримати [з цього завдання](http://progbase.herokuapp.com/modules/progbase3/tasks/variant), в коді має називатись як англійська версія назви типу даних, наприклад, Course.
3. **CUI** - абревіатура від Console User Interface - Консольний Інтерфейс Користувача.

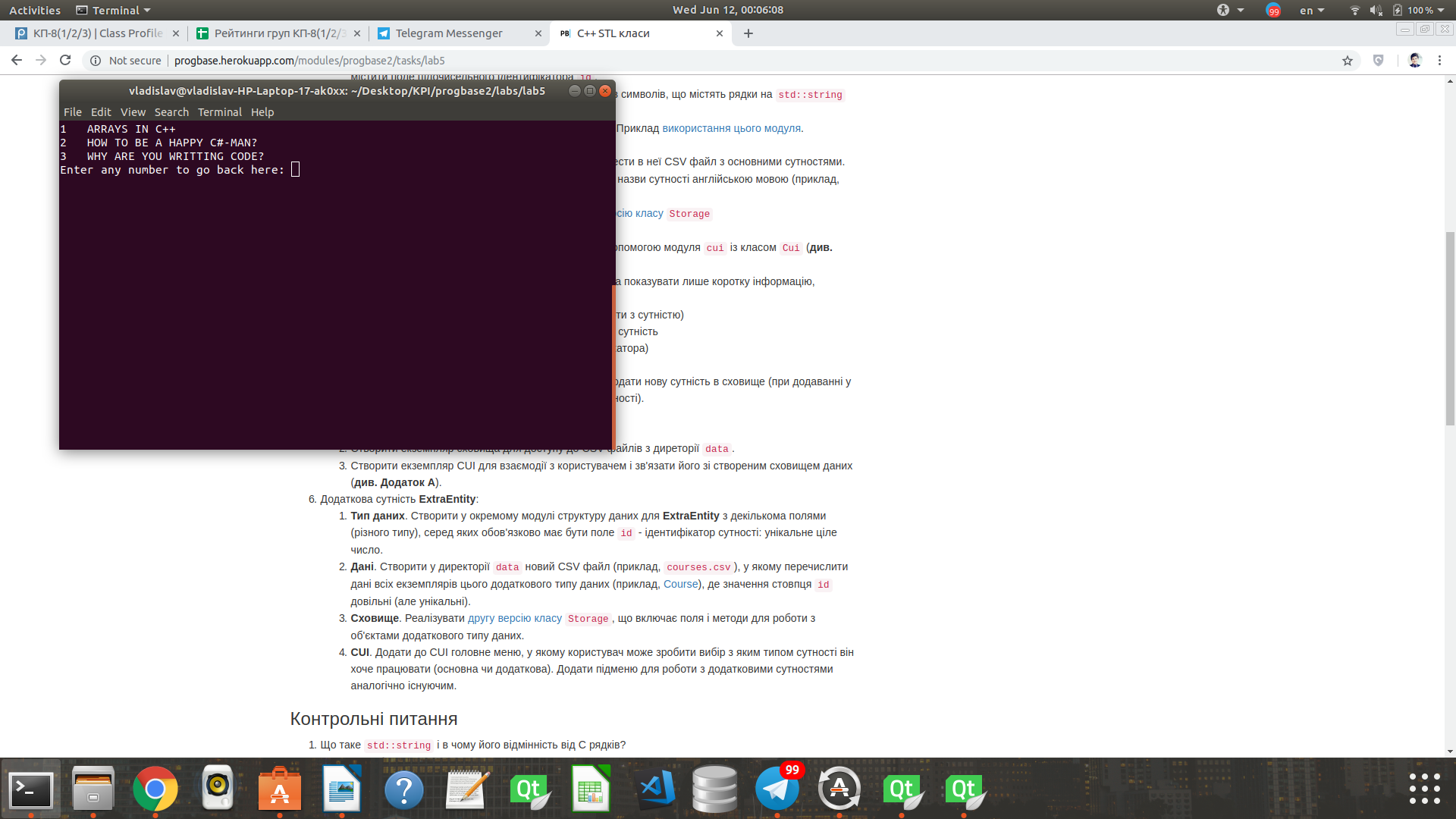
Кроки виконання:

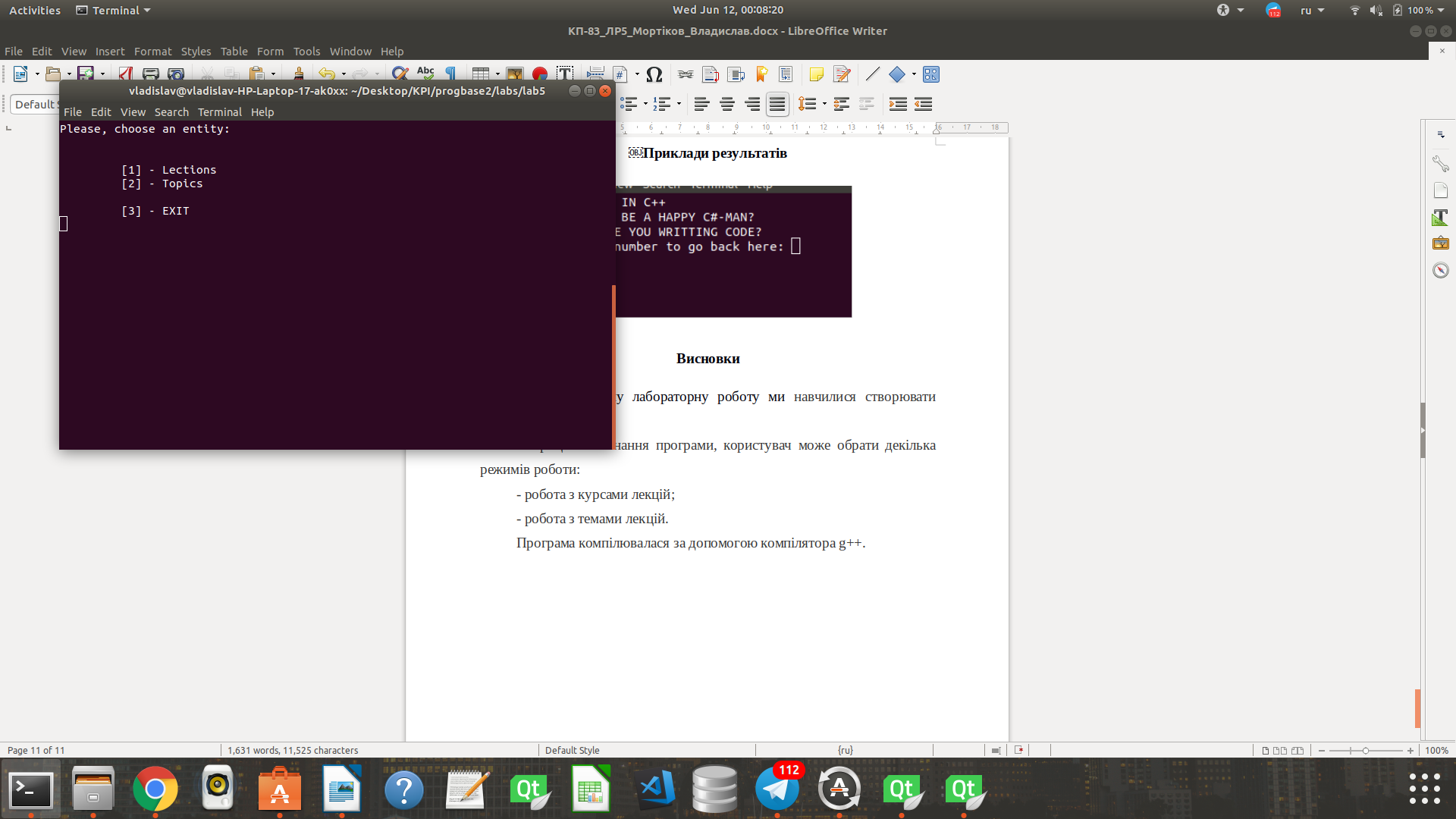
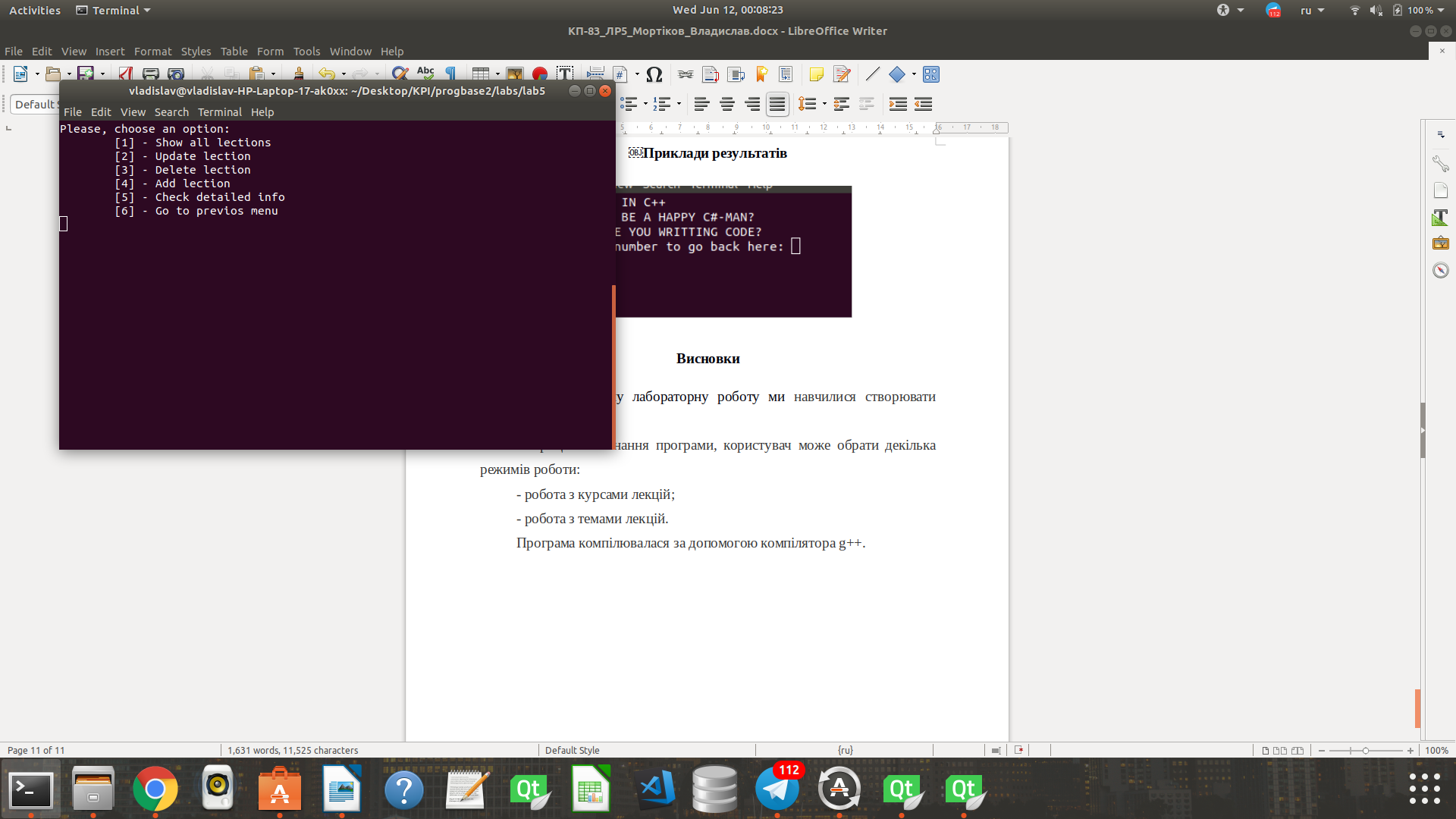
1. Скопіювати у проект даного завдання ваше рішення [лабораторної роботи №2](https://progbase.herokuapp.com/modules/progbase2/tasks/lab2).
2. Виконати міграцію коду з С на С++:
   1. НЕ переписувати АТД на власні класи, замість АТД **List** використати стандартний клас контейнер std::vector<T> із заголовку <vector>.
   2. Структура сутності за варіантом має залишитися структурою. Також вона повинна обов'язково містити поле цілочисельного ідентифікатора id.
   3. Замінити всі використання const char \* та масивів символів, що містять рядки на std::stringтам, де це має сенс.
   4. Переписати модуль **csv** на С++ ([інтерфейс модуля](https://docs.google.com/presentation/d/1L_4QHZRVX38KXFV_WjPoHp5qdSm_c0HoUSkgvZeQViA/edit" \l "slide=id.g55253d8ed8_2_0)). Приклад [використання цього модуля](https://docs.google.com/presentation/d/1L_4QHZRVX38KXFV_WjPoHp5qdSm_c0HoUSkgvZeQViA/edit" \l "slide=id.g554081724d_7_0).
3. Сховище даних (версія №1):
   1. Створити у корені проекту директорію data і перенести в неї CSV файл з основними сутностями.
   2. Переназвати CSV файл так, щоби він був множиною назви сутності англійською мовою (приклад, students.csv).
   3. Створити модуль storage, що міститиме [першу версію класу Storage](https://docs.google.com/document/d/1XI_8fQ7NscgXDIbTqmLGv-mxJRkc06Aznbte1L-IqkE/edit?usp=sharing)
4. Консольний інтерфейс користувача (версія №1).  
   Реалізувати консольний інтерфейс користувача (CUI) за допомогою модуля cui із класом Cui (**див. Додаток А**), що дозволятиме користувачу:
   1. Переглянути список всіх сутностей зі сховища (можна показувати лише коротку інформацію, наприклад, ідентифікатор + назву)
   2. Обрати сутність за її ідентифікатором (підменю роботи з сутністю)
      1. Переглянути детальну інформацію про обрану сутність
      2. Змінити поле обраної сутності (окрім ідентифікатора)
      3. Видалити обрану сутність
   3. Створити, заповнити поля (окрім ідентифікатора) і додати нову сутність в сховище (при додаванні у сховище отримати від нього ідентифікатор нової сутності).
5. Переписати основну логіку роботи програми:
   1. Видалити обробку аргументів командного рядка.
   2. Створити екземпляр сховища для доступу до CSV файлів з диреторії data.
   3. Створити екземпляр CUI для взаємодії з користувачем і зв'язати його зі створеним сховищем даних (**див. Додаток А**).
6. Додаткова сутність **ExtraEntity**:
   1. **Тип даних**. Створити у окремому модулі структуру даних для **ExtraEntity** з декількома полями (різного типу), серед яких обов'язково має бути поле id - ідентифікатор сутності: унікальне ціле число.
   2. **Дані**. Створити у директорії data новий CSV файл (приклад, courses.csv), у якому перечислити дані всіх екземплярів цього додаткового типу даних (приклад, [Course](https://docs.google.com/spreadsheets/d/1RmpAUL6cHRgzPtQAQiGB-qYRsvp5wxcYzn-ptAjec-Y/edit" \l "gid=1040180202)), де значення стовпця idдовільні (але унікальні).
   3. **Сховище**. Реалізувати [другу версію класу](https://docs.google.com/document/d/1XI_8fQ7NscgXDIbTqmLGv-mxJRkc06Aznbte1L-IqkE/edit" \l "heading=h.uyv3ggv1zfpi)Storage, що включає поля і методи для роботи з об'єктами додаткового типу даних.
   4. **CUI**. Додати до CUI головне меню, у якому користувач може зробити вибір з яким типом сутності він хоче працювати (основна чи додаткова). Додати підменю для роботи з додатковими сутностями аналогічно існуючим.

**Тексти коду програм**

|  |
| --- |
| **cui.cpp** |
| #include "cui.hpp"  void Cui::**show**()  {  system("clear");  int choose;  cout << "Please, choose an entity:" << endl  << endl  << "\n\t [1] - Lections \n\t [2] - Topics \n\n\t [3] - EXIT" << endl;  cin >> choose;  if (choose == 1)  lectionsMainMenu();  else if (choose == 2)  topicsMainMenu();  else if (choose == 3)  {  system("clear");  int out;  cout << "ENTER 1 TO SAVE ALL CHANGES or any other number not to save changes" << endl;  cin >> out;  if (out == 1)  storage\_->save();  exit(0);  }  }  void Cui::**lectionsMainMenu**()  {  system("clear");  int choose;  cout << "Please, choose an option:"  << "\n\t[1] - Show all lections\n\t[2] - Update lection\n\t[3] - Delete lection\n\t[4] - Add lection\n\t[5] - Check detailed info\n\t[6] - Go to previos menu" << endl;  cin >> choose;  if (choose == 1)  {  system("clear");  for (Lection lec : storage\_->getAllLections())  {  cout << lec.id << " " << lec.title << endl;  }  cout << "Enter any number to go back here: ";  cin >> choose;  lectionsMainMenu();  }  else if (choose == 2)  {  cout << "\nEnter the lection's id:\n";  int id;  cin >> id;  lectionUpdateMenu(id);  }  else if (choose == 3)  {  cout << "\nEnter the lection's id:\n";  int id;  cin >> id;  lectionDeleteMenu(id);  }  else if (choose == 4)  {  lectionCreateMenu();  }  else if (choose == 5)  {  cout << "\nEnter the lection's id:\n";  int id;  cin >> id;  lectionMenu(id);  }  else if (choose == 6)  {  show();  }  else  {  cout << "\nIncorrect input";  exit(1);  }  }  void Cui::**printLection**(vector<Lection> &lec)  {  for (Lection &lec : lec)  {  cout << lec.id << " ";  cout << lec.title.c\_str() << " ";  cout << lec.lectorer.c\_str() << " ";  cout << lec.students << endl;  }  }  void Cui::**lectionMenu**(int entity\_id)  {  system("clear");  if (!storage\_->getLectionById(entity\_id))  {  cout << "No lection with id" << endl;  }  else  {  optional<Lection> lec\_opt = storage\_->getLectionById(entity\_id);  Lection lec = lec\_opt.value();  cout << "The info for the lection with id [" << entity\_id << "]" << endl;  cout << "The title of course is " << lec.title << endl;  cout << "The lectorer`s name is " << lec.lectorer << endl;  cout << "The number of students on the course is " << lec.students << endl;  }  cout << endl  << "Enter any number to back to previous menu" << endl;  int out;  cin >> out;  lectionsMainMenu();  }  void Cui::**lectionUpdateMenu**(int entity\_id)  {  system("clear");  Lection lec;  lec.id = entity\_id;  cout << "\t\tLection update menu\n\n Enter new course`s title: ";  cin >> lec.title;  cout << "Enter lectorer`s name: ";  cin >> lec.lectorer;  cout << "Enter students on the course: ";  cin >> lec.students;  if (storage\_->updateLection(lec))  cout << "Succesfully updated! :)";  else  {  cout << "No lection with id " << entity\_id;  }  cout << "\n\nEnter any value to back to previous menu" << endl;  char c;  cin >> c;  lectionsMainMenu();  }  void Cui::**lectionDeleteMenu**(int entity\_id)  {  system("clear");  if (storage\_->removeLection(entity\_id))  cout << "\nLection with id " << entity\_id << "was deleted";  else  cout << "No lection with id " << entity\_id;  cout << "\n\nEnter any value to back to previous menu\n";  char c;  cin >> c;  lectionsMainMenu();  }  void Cui::**lectionCreateMenu**()  {  system("clear");  Lection lec;  cout << "\t\tLection create menu\n\nEnter new topic: ";  cin >> lec.title;  cout << "Enter a lectorer`s name: ";  cin >> lec.lectorer;  cout << "Enter number of students: ";  cin >> lec.students;  lec.id = storage\_->insertLection(lec);  cout << "\n Lection course created, id = " << lec.id << endl;  cout << "\n\nenter any value to back to previous menu\n";  char c;  cin >> c;  lectionsMainMenu();  }  void Cui::**topicsMainMenu**()  {  system("clear");  int choose;  cout << "Please, choose an option:"  << "\n\t[1] - Show all topics\n\t[2] - Update topic\n\t[3] - Delete topic\n\t[4] - Add topic\n\t[5] - Check detailed info\n\t[6] - Go to previos menu" << endl;  cin >> choose;  if (choose == 1)  {  system("clear");  for (Topic top : storage\_->getAllTopics())  {  cout << top.id << " " << top.topic << endl;  }  cout << "Enter any number to go back here: ";  cin >> choose;  topicsMainMenu();  }  else if (choose == 2)  {  cout << "\nEnter the topic's id:\n";  int id;  cin >> id;  topicUpdateMenu(id);  }  else if (choose == 3)  {  cout << "\nEnter the topic's id:\n";  int id;  cin >> id;  topicDeleteMenu(id);  }  else if (choose == 4)  {  topicCreateMenu();  }  else if (choose == 5)  {  cout << "\nEnter the topic's id:\n";  int id;  cin >> id;  topicMenu(id);  }  else if (choose == 6)  {  show();  }  else  {  cout << "\nIncorrect input";  exit(1);  }  }  void Cui::**printTopics**(vector<Topic> &top)  {  for (Topic &top : top)  {  cout << top.id << " ";  cout << top.data.c\_str() << " ";  cout << top.topic.c\_str() << " ";  cout << top.students << endl;  }  }  void Cui::**topicMenu**(int entity\_id)  {  system("clear");  if (!storage\_->getTopicById(entity\_id))  {  cout << "No topic with id" << endl;  }  else  {  optional<Topic> top\_opt = storage\_->getTopicById(entity\_id);  Topic top = top\_opt.value();  cout << "The info for the topic with id [" << entity\_id << "]" << endl;  cout << "The lection`s data is " << top.data << endl;  cout << "The lection`s topic is " << top.topic << endl;  cout << "The number of students on the lection was " << top.students << endl;  }  cout << endl  << "Enter any number to back to previous menu" << endl;  int out;  cin >> out;  topicsMainMenu();  }  void Cui::**topicUpdateMenu**(int entity\_id)  {  system("clear");  Topic top;  top.id = entity\_id;  cout << "\t\tTopic update menu\n\n Enter new lection`s data: ";  cin >> top.data;  cout << "Enter lection`s topic: ";  cin >> top.topic;  cout << "Enter students on the lections: ";  cin >> top.students;  if (storage\_->updateTopic(top))  cout << "Succesfully updated! :)";  else  {  cout << "No topic with id " << entity\_id;  }  cout << "\n\nEnter any value to back to previous menu" << endl;  char c;  cin >> c;  topicsMainMenu();  }  void Cui::**topicDeleteMenu**(int entity\_id)  {  system("clear");  if (storage\_->removeTopic(entity\_id))  cout << "\nTopic with id " << entity\_id << "was deleted";  else  cout << "No Topic with id " << entity\_id;  cout << "\n\nEnter any value to back to previous menu\n";  char c;  cin >> c;  topicsMainMenu();  }  void Cui::**topicCreateMenu**()  {  system("clear");  Topic top;  cout << "\t\tTopic create menu\n\nEnter new data: ";  cin >> top.data;  cout << "Enter a topic of lection: ";  cin >> top.topic;  cout << "Enter number of students on lection: ";  cin >> top.students;  top.id = storage\_->insertTopic(top);  cout << "\n Topic created, id = " << top.id << endl;  cout << "\n\nEnter any value to back to previous menu\n";  char c;  cin >> c;  topicsMainMenu();  } |

**￼Приклади результатів**

****

****

**Висновки**

Виконавши дану лабораторну роботу ми навчилися створювати консольні програми.

У процесі виконання програми, користувач може обрати декілька режимів роботи:

- робота з курсами лекцій;

- робота з темами лекцій.

Програма компілювалася за допомогою g++.