

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 № 4**

з дисципліни “Основи програмування”

тема “Стандартна бібліотека класів C++”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент I курсу  групи КП-92  Мовчан Максим  варіант №9 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

Розділити код консольної програми на модулі інтерфейсу користувача та файлового сховища даних.  
Використати стандартні класи потоків C++.  
Використати стандартні шаблонні контейнери C++ STL.

**Загальні вимоги до завдання**

Створити програму з Консольним Інтерфейсом Користувача, що взаємодіє з Файловим Сховищем Даних (CSV).

Реалізувати файлове сховище даних (FileStorage) з використанням текстового формату CSV:

1. Назва файлу з даними сутностей має бути множиною назви сутності англійською мовою (приклад, students.csv).
2. Розмістити файл з даними у директорії data (відносно кореня проекта).
3. В реалізації сховища обов'язково використати стандартні класи потоків C++ та C++ STL контейнери.

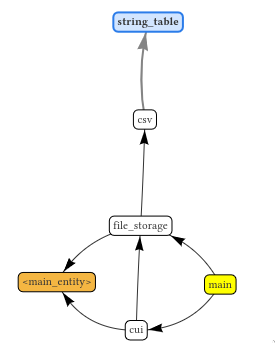
Реалізувати консольний інтерфейс користувача (CUI), що дозволятиме користувачу:

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).

**Методичні вказівки до виконання**

1. Кроки виконання:
2. Скопіювати у проект даного завдання ваше рішення лабораторної роботи №3 (можна №2).
   1. Видалити модуль options і обробку аргументів командного рядка.
   2. Видалити bintree, bstree та їх використання.
3. Виконати міграцію з власних шаблонних класів на стандартні шаблонні класи С++ STL:
   1. Видалити dynamic\_array та list.
   2. У коді, де використовувався клас List<T>, використати стандартний клас-контейнер std::vector<T> із заголовку <vector>.
4. Сховище даних:
   1. Передумова: структура сутності за варіантом обов'язково повинна містити поле цілочисельного ідентифікатора id.
   2. Створити у корені проекту директорію data і перенести в неї CSV файл з основними сутностями.
   3. Переназвати CSV файл так, щоби він був множиною назви сутності англійською мовою (приклад, students.csv).
   4. Створити модуль storage, що міститиме першу версію класу FileStorage (детальний опис за посиланням).
5. Реалізувати консольний інтерфейс користувача (CUI) за допомогою модуля cui із класом Cui (**див. Додаток А**) відповідно до завдання.
6. Переписати основну логіку роботи програми (в main(), **див. Додаток B**):
   1. Створити екземпляр сховища для доступу до CSV файлів з диреторії data.
   2. Створити екземпляр CUI для взаємодії з користувачем і зв'язати його зі створеним сховищем даних.

**Діаграма залежностей**

****

**Інтерфейси модулів**

|  |
| --- |
| **csv.hpp** |
| **#ifndef CSV\_H**  **#define CSV\_H**  **#include "string\_table.hpp"**  **using namespace std;**  **StringTable Csv\_parse(string& csvStr);**  **string Csv\_toString(StringTable& table);**  **#endif // CSV\_H** |

|  |
| --- |
| **cui.hpp** |
| **#ifndef CUI\_H**  **#define CUI\_H**  **#include "storage.hpp"**  **#include <termios.h>**  **#include <unistd.h>**  **#include <vector>**  **#include <iomanip>**  **#define blu\_f "\033[32m"**  **#define yel\_f "\033[33m"**  **#define cya\_f "\033[36m"**  **#define red\_f "\033[31m"**  **#define mag\_f "\033[35m"**  **#define res "\033[0m"**  **#define cont "\n\033[32mPress any key to continue..\033[0m"**  **using namespace std;**  **class Cui {**  **FileStorage\* const storage\_;**  **void OrgsMainMenu();**  **void OrgsMenu(int org\_id);**  **void OrgsUpdateMenu(int org\_id);**  **void OrgsDeleteMenu(int org\_id);**  **void OrgsCreateMenu();**  **public:**  **Cui(FileStorage\* storage): storage\_ {storage} {}**  **void show();**  **};**  **char getch();**  **#endif // CUI\_H** |

|  |
| --- |
| **storage.hpp** |
| **#ifndef STORAGE\_H**  **#define STORAGE\_H**  **#include "organisation.hpp"**  **#include "string\_table.hpp"**  **#include <fstream>**  **#include <optional>**  **#include <string>**  **#include <vector>**  **using namespace std;**  **class FileStorage {**  **string dir\_name\_;**  **fstream orgs\_file\_;**  **fstream fndrs\_file\_;**  **// students**  **vector<Organisation> loadOrgs();**  **void saveOrgs(const vector<Organisation>& orgs);**  **int getNewOrgId();**  **public:**  **explicit FileStorage(const string& dir\_name = "");**  **void setName(const string& dir\_name);**  **string name() const;**  **bool isOpen() const;**  **bool open();**  **void close();**  **vector<Organisation> getAllOrgs();**  **optional<Organisation> getOrgById(int org\_id);**  **bool updateOrg(const Organisation& org);**  **bool removeOrg(int org\_id);**  **int insertOrg(const Organisation& org);**  **};**  **#endif // STORAGE\_H** |

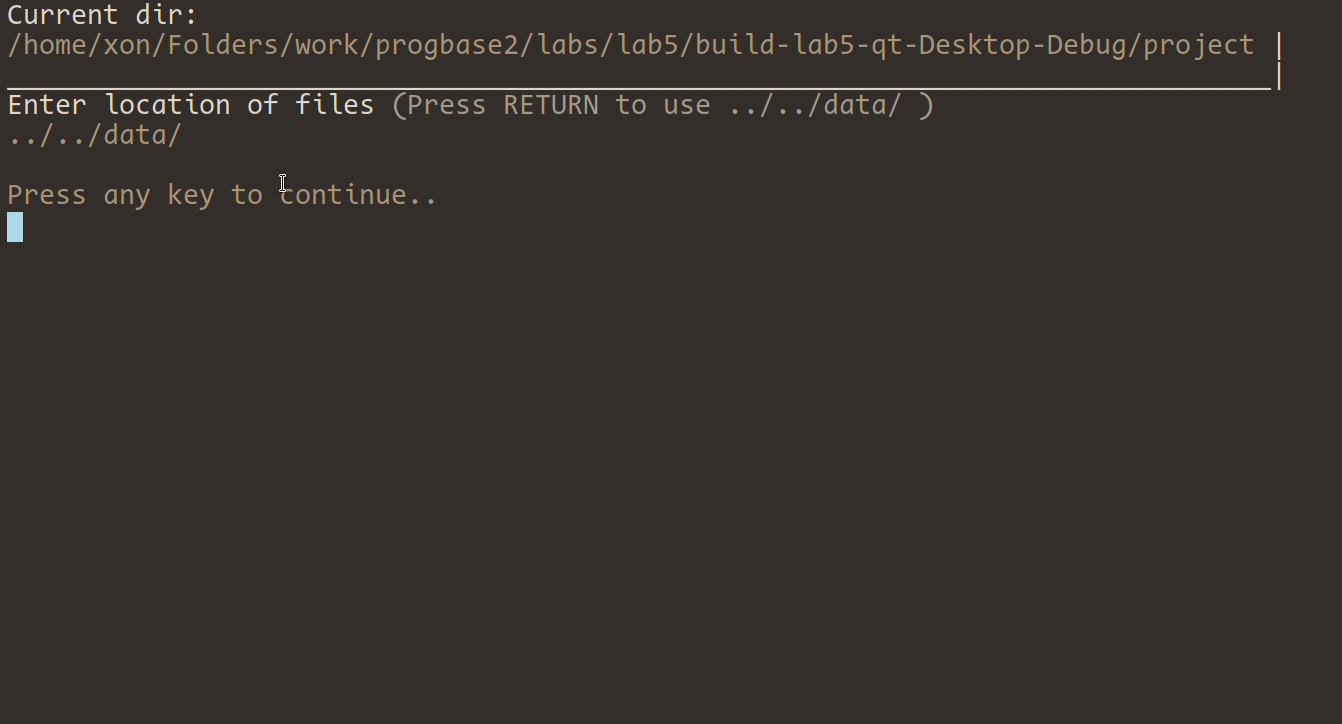
|  |
| --- |
| **string\_table.hpp** |
| **#ifndef STRING\_TABLE\_H**  **#define STRING\_TABLE\_H**  **#include <iostream>**  **#include <string>**  **using namespace std;**  **class StringTable {**  **string\* cells\_;**  **size\_t rows\_;**  **size\_t cols\_;**  **public:**  **StringTable(size\_t rows, size\_t cols);**  **~StringTable();**  **size\_t size\_rows();**  **size\_t size\_columns();**  **string& at(int rowIndex, int colIndex);**  **void print();**  **};**  **#endif // STRING\_TABLE\_H** |

**Текст коду програми**

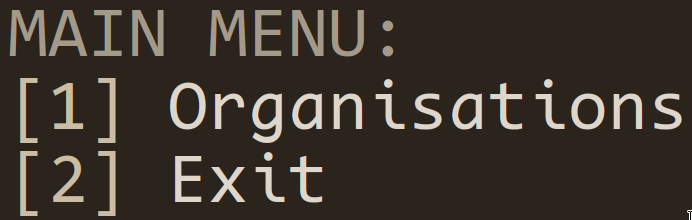
|  |
| --- |
| **main.cpp** |
| **#include "cui.hpp"**  #include <string>  #include <stdio.h> /\* defines FILENAME\_MAX \*/  #ifdef \_\_unix\_\_  # include <unistd.h>  # define GetCurrentDir getcwd  #elif defined(\_WIN32) || defined(WIN32)  # include <direct.h>  # define GetCurrentDir \_getcwd  #endif  #include<iostream>  using namespace std;  auto **main**() -> int {  char buff[FILENAME\_MAX];  getcwd( buff, FILENAME\_MAX );  string wdir(buff);  cout << "Current dir: " << endl;  cout << blu\_f + wdir + res " |" << endl;  for (int i = 0; i < (int)wdir.length() + 1; i++) cout << '\_';  cout << '|' << endl;  cout << "Enter location of files " yel\_f "(Press RETURN to use ../../data/ )" res << endl << blu\_f;  string str = "";  getline(*cin*, *str*);  cout << res;  if (str == "") str.append("../../data/");  cout << cont << endl;  getch();  system("clear");  FileStorage fs {str};  Cui cui{&fs};  cui.show();  } |

**Console**

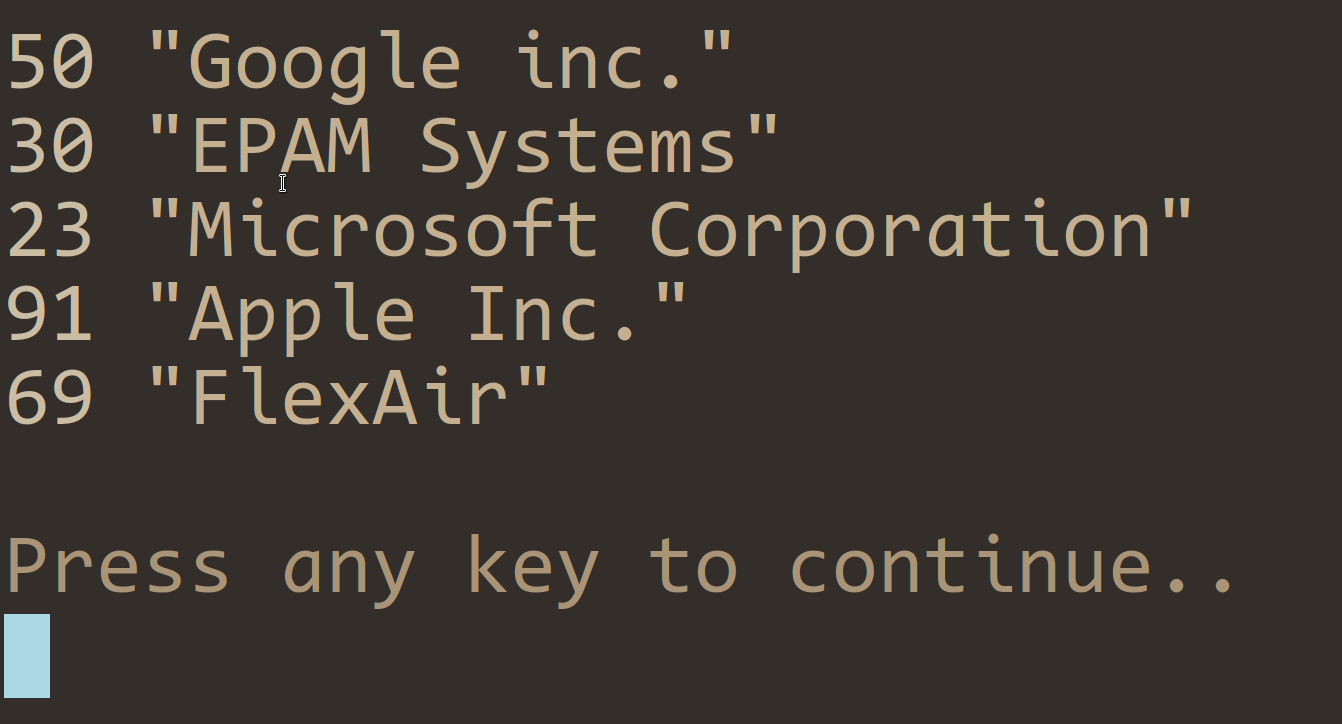
**Вибір директорії з даними**

****

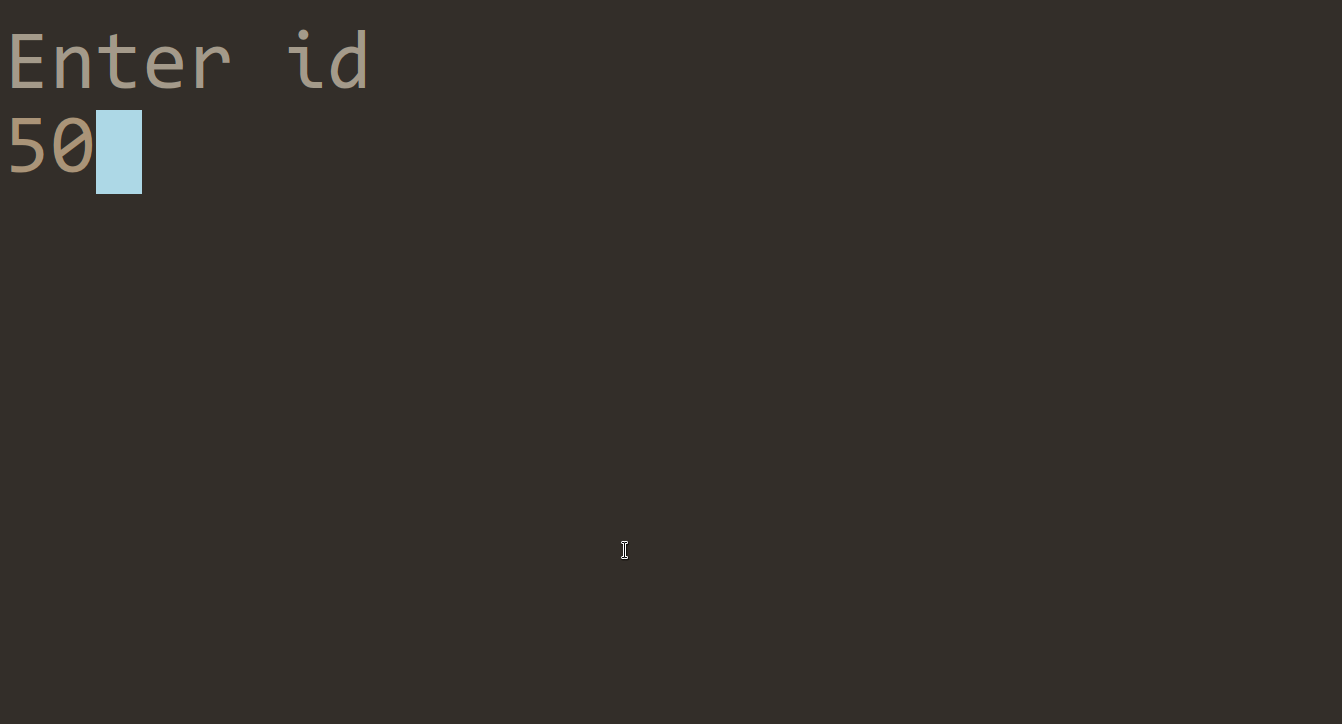
**Меню**

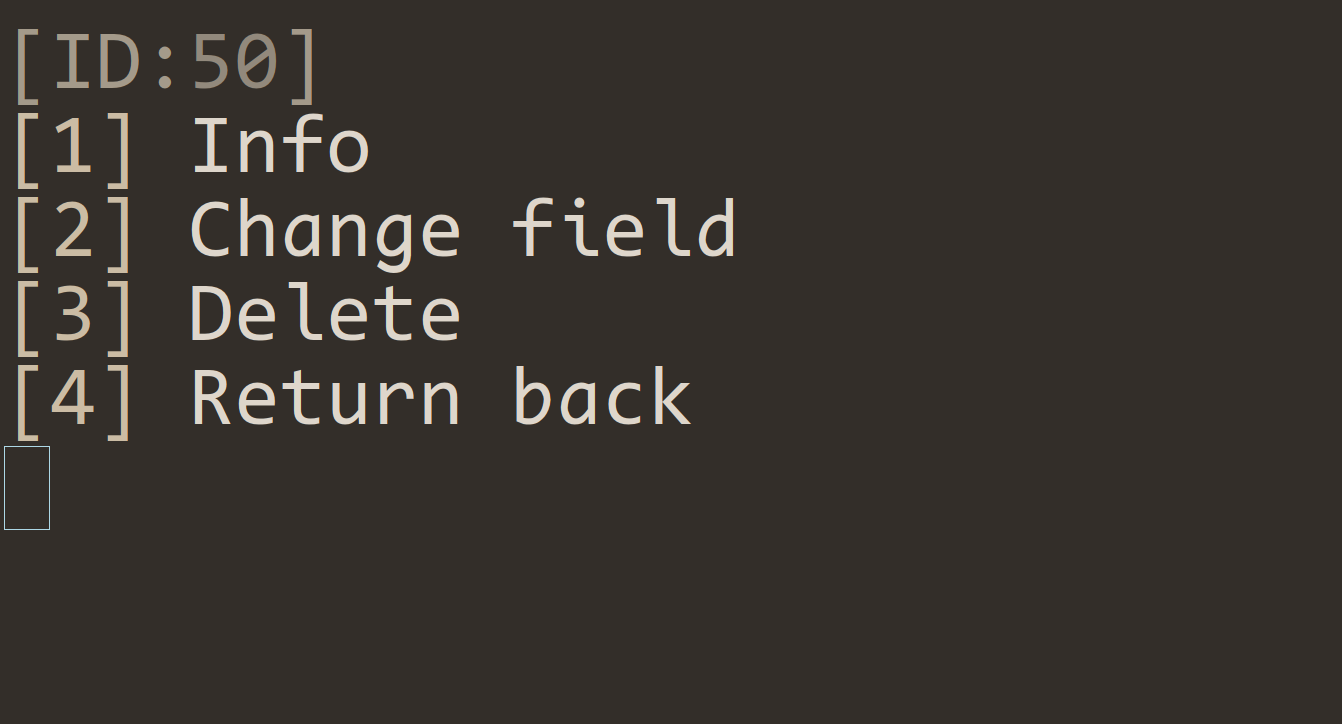
****

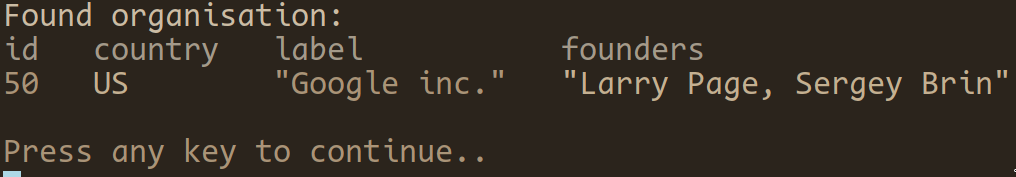
**Список сутностей**

****

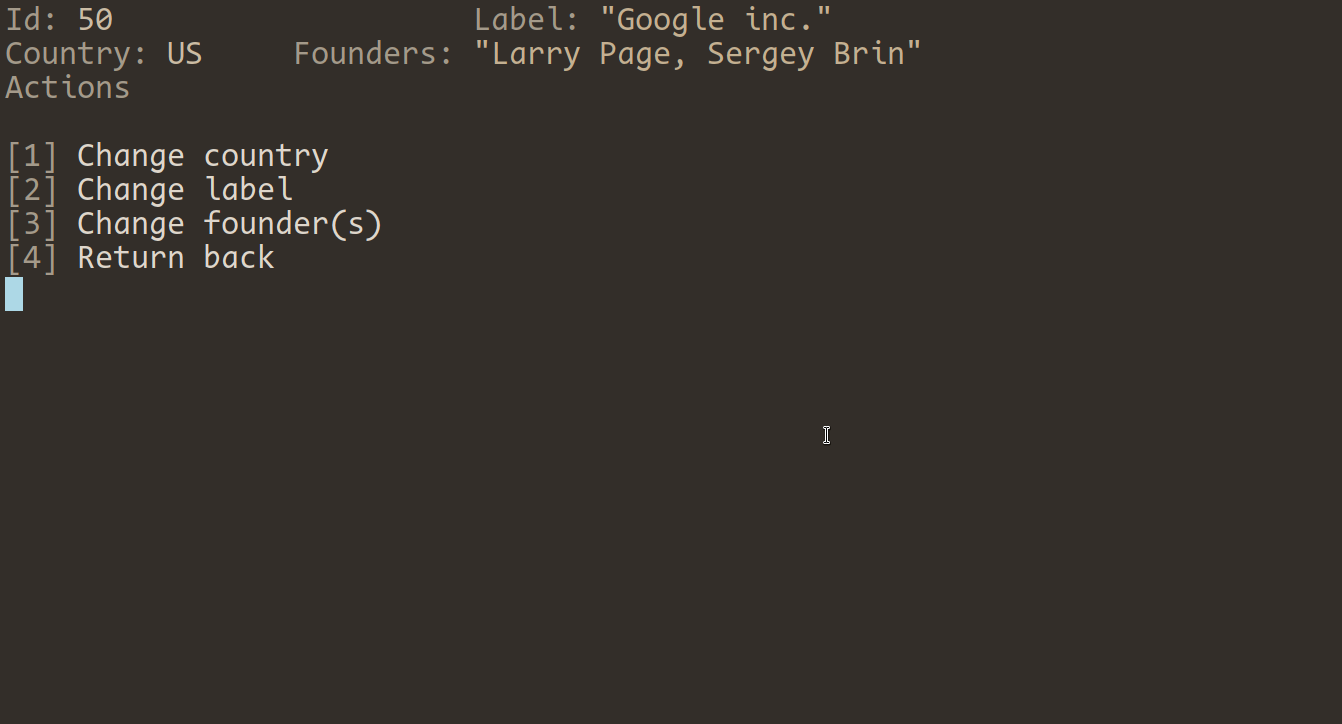
**Робота з окремою сутністю**

****

****

****

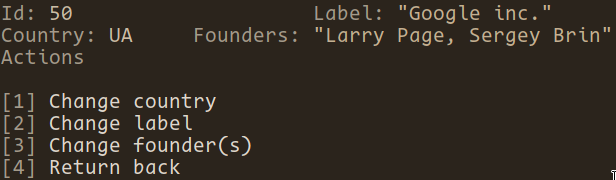
**Меню зміни поля**

****

**Зміна поля сутності**

****

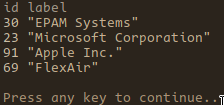
**Результат**

****

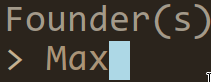
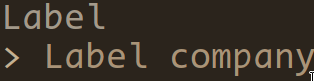
**Видалення сутності**

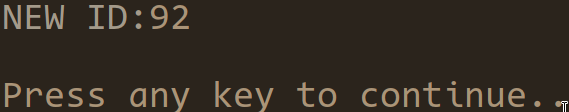
****

**Результат**

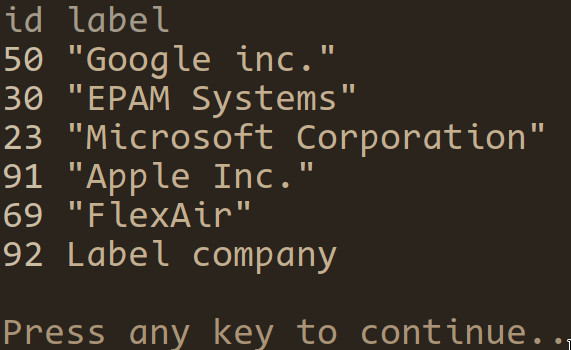
****

**Додавання нової сутності**

****

****

**Результат**

****

**Висновки**

Було розділено код консольної програми на модулі інтерфейсу користувача та файлового сховища даних. Використати стандартні класи потоків C++. Ми додали меню для роботи з сутністю організацій. Ми використовували формат даних CSV , зчитували його і форматували. Було використано Cmake для збірки проекту з багатьох вихідних файлів. Компіляція всього коду відбувалась за допомогою компілятора g++.