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 №2**

**з дисципліни “ Основи програмування”**

**тема “ФОРМАТ ДАНИХ CSV”**

|  |  |  |
| --- | --- | --- |
| **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 роботи**

Опанувати формат даних CSV та навчитись його зчитувати і формувати.  
Реалізувати модуль універсального списку.  
Навчитись збирати проекти з багатьох вихідних файлів за допомогою CMake.

### **Загальні вимоги до завдання**

**UPD-1**: Створити у Google Spreadsheet (або у будь-якому аналозі) таблицю, у яку внести мінімум 5 записів типу **Курс лекцій** зі стовпцями (мінімум 4), що містять дані екземплярів цього типу ([приклад для типу Студент](https://docs.google.com/spreadsheets/d/1RmpAUL6cHRgzPtQAQiGB-qYRsvp5wxcYzn-ptAjec-Y/edit?usp=sharing)).  
Експортувати\завантажити таблицю як CSV-файл і помістити її у директорію проекту як data.csv.

Описати структуру даних типу **Курс лекцій**. Створити у програмі список **testList** із мінімум 5 довільних екземплярів даного типу.

Створити консольну програму, що дозволяє працювати з CSV файлами.

Опції командного рядка (можуть йти у довільному порядку):

1. IN - (перший вільний аргумент) назва вхідного файлу з CSV
2. -n N - аргумент для додаткової обробки
3. -o OUT - назва вихідного файлу

**UPD-2**. [Приклад](https://gist.github.com/PublicHadyniak/2c48933badbd51c025c98130c320b28a) використання функції getopt() для простого отримання значень опцій програми.

Всі опції не обов'язкові. За замовчуванням:

1. Якщо нема вхідного файлу - брати дані з **testList**
2. Якщо нема аргумента обробки - не обробляти список сутностей.
3. Якщо нема назви вихідного файлу - виводити дані у консоль (у довільному вигляді)

Якщо задано шлях до файлу з CSV даними, наприклад:

./a.out data.csv

Зчитати дані з файлу у список списків рядків **inCsvTable** (таблиця рядків, **[модель пам'яті](https://docs.google.com/presentation/d/1L_4QHZRVX38KXFV_WjPoHp5qdSm_c0HoUSkgvZeQViA/edit" \l "slide=id.g5240971098_1_0)**).  
Для цього потрібно реалізувати **універсальний список** (**List**, [інтерфейс](https://docs.google.com/presentation/d/18PQX143prl8iTkqzjQM7h1_GZQ1swnf17oQ0TLarKpw/edit" \l "slide=id.g51cf9cbb50_2_0))  
Вивести отриману таблицю у консоль (форматування при виводі довільне).

**UPD-4**. На основі таблиці **inCsvTable** сформувати список **items** екземплярів типу **Курс лекцій**.  
Приклад для типу **Студент**:

**void** **fillStudentsListFromTable**(List \* csvTable, List \* students);

*// or*

List \* **createStudentsListFromTable**(List \* csvTable);

Якщо задано значення опції -n, наприклад:

./a.out data.csv -n 3

Після формування **items** Знайти всі курси лекцій, у яких загальна тривалість більша N годин. **UPD-4** Видалити зі списку лишні елементи. Для цього можна створити функцію:

**void** **processItems**(List \* items, T n); *// T depends on your variant*

Вивести модифікований **items** у консоль.

Якщо задано значення опції -o, наприклад:

./a.out -o out.csv

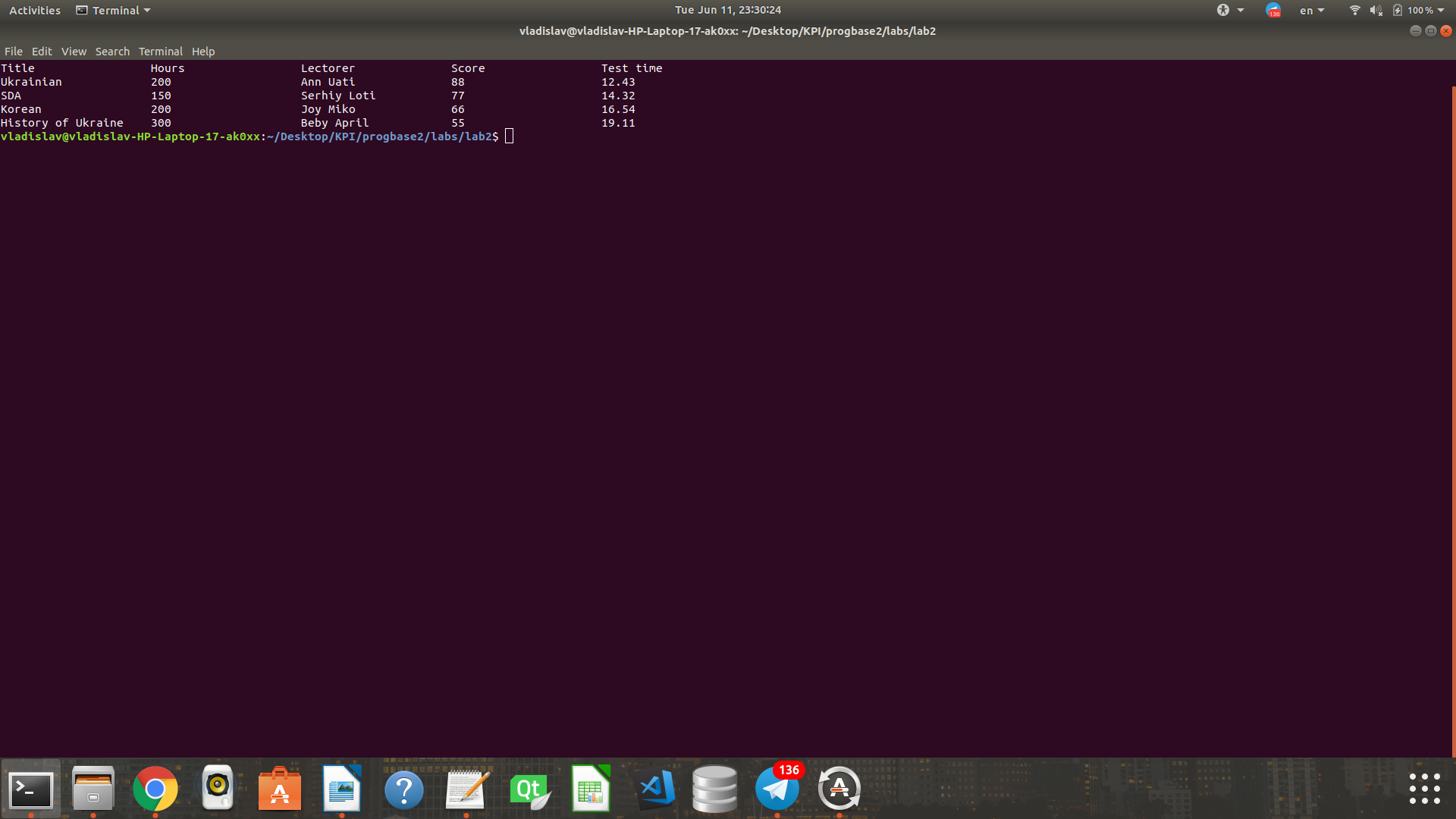
**UPD-4** Перетворити модифікований **items** у нову таблицю **outCsvTable** (список списків рядків) і зберегти таблицю у вихідний файл у форматі CSV.

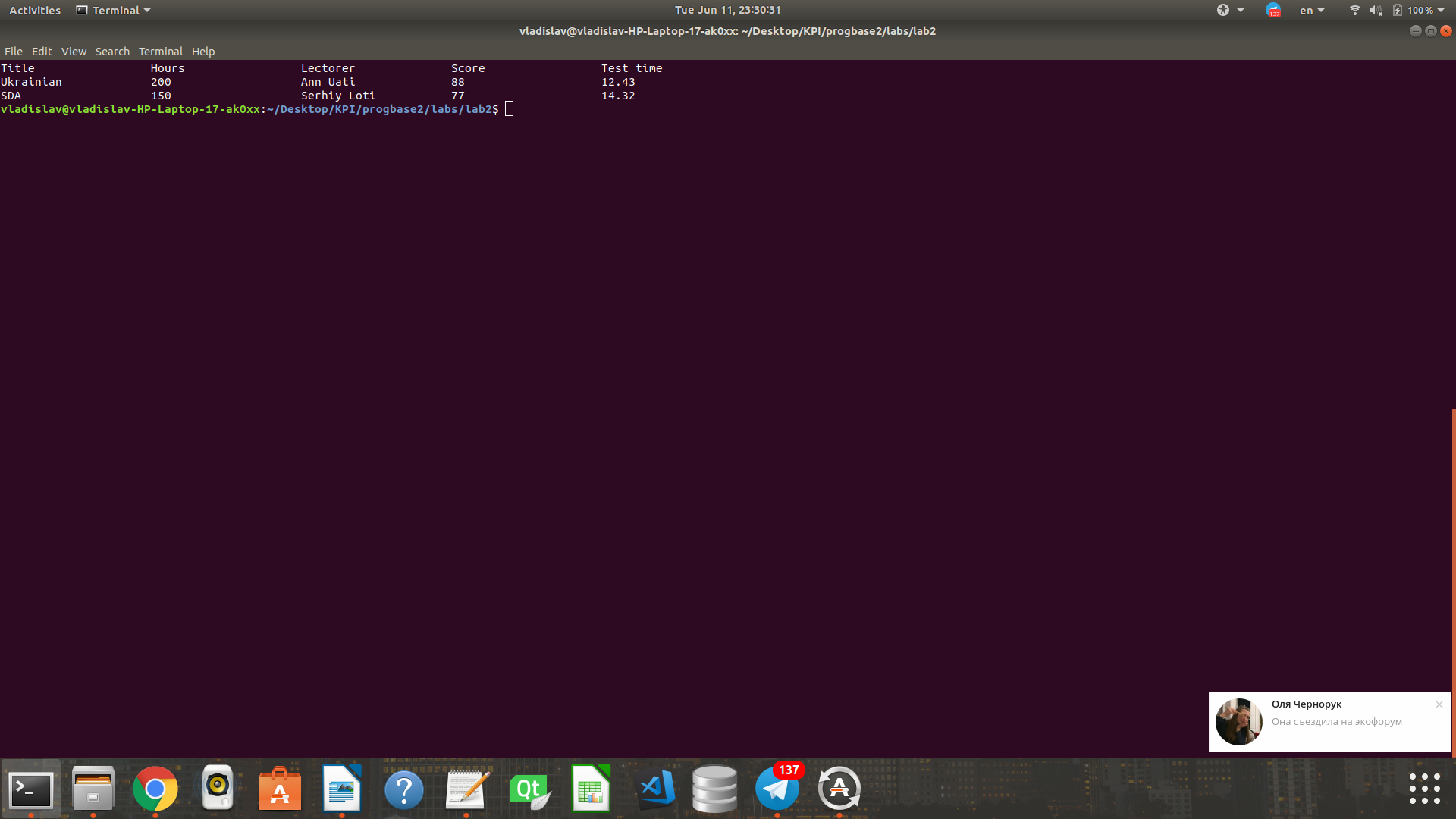
**UPD-1**. Перевірити коректність формату згенерованого файлу імпортувавши його у Google Spreadsheet (або аналог).

**Тексти коду програм**

|  |
| --- |
| **csv.c** |
| #include <stdlib.h>  #include <string.h>  #include <stdio.h>  #include "list.h"  void Csv\_addInt(List \*row, int value)  {  char buffer[100];  sprintf(buffer, "%i", value);  char \*bufOnHeap = malloc(strlen(buffer) + 1);  strcpy(bufOnHeap, buffer);  List\_add(row, bufOnHeap);  }  void Csv\_addDouble(List \*row, double value)  {  char buffer[100];  sprintf(buffer, "%lf", value);  char \*bufOnHeap = malloc(strlen(buffer) + 1);  strcpy(bufOnHeap, buffer);  List\_add(row, bufOnHeap);  }  void Csv\_addString(List \*row, const char \*value)  {  char \*bufOnHeap = malloc(strlen(value) + 1);  strcpy(bufOnHeap, value);  List\_add(row, bufOnHeap);  }  int Csv\_int(List \*row, int index)  {  if (index >= 0 && index < row->size)  {  int buff = atoi(row->items[index]);  return buff;  }  else  {  printf("ERROR: invalid index\n");  return -1;  }  }  double Csv\_double(List \*row, int index)  {  if (index >= 0 && index < row->size)  {  float buffer = atof(row->items[index]);  return buffer;  }  else  {  printf("ERROR: invalid index\n");  return -1;  }  }  int Csv\_string(List \*row, int index, char \*buf, int nBuf)  {  if (strlen(List\_get(row, index)) + 1 > nBuf)  {  return -1;  }  else  {  strcpy(buf, List\_get(row, index));  return 0;  }  }  void Csv\_addRow(List \*table, List \*row)  {  List\_add(table, row);  }  List \*Csv\_row(List \*table, int index)  {  return table->items[index];  }  void Csv\_fillTableFromString(List \*csvTable, const char \*csvStr)  {  List \*row = NULL;  char buf[10000];  int bufX = 0;  const char \*p = csvStr;  while (1)  {  if (\*p == ',' || \*p == '\n' || \*p == '\0')  {  buf[bufX] = '\0';  if (bufX == 0 && \*p == '\0' && row == NULL)  {  break;  }  if (row == NULL)  {  row = List\_alloc();  }  char \*bufOnHeap = malloc(strlen(buf) + 1);  strcpy(bufOnHeap, buf);  List\_add(row, bufOnHeap);  if (\*p == '\n' || \*p == '\0')  {  List\_add(csvTable, row);  row = NULL;  }  bufX = 0;  }  else  {  buf[bufX] = \*p;  bufX += 1;  }  if (\*p == '\0')  {  break;  }  p += 1;  }  }  int Csv\_fillStringFromTable(List \*csvTable, char \*buf, int nBuf)  {  buf[0] = '\0';  int buffLength = 0;  for (size\_t i = 0; i < List\_size(csvTable); i++)  {  List \*rowItem = List\_get(csvTable, i);  for (size\_t j = 0; j < List\_size(rowItem); j++)  {  char \*value = List\_get(rowItem, j);  strcat(buf, value);  if (j != List\_size(rowItem) - 1)  {  strcat(buf, ",");  }  buffLength += strlen((char\*)List\_get(rowItem,j));  }  if (i != List\_size(csvTable) - 1)  {  strcat(buf, "\n");  }  }  if (buffLength > nBuf)  return -1;  return 0;  }  char \*Csv\_createStringFromTable(List \*csvTable)  {  char \*newString = malloc(sizeof(char) \* 10000);  newString[0] = '\0';  Csv\_fillStringFromTable(csvTable, newString, 10000);  return newString;  }  void Csv\_clearTable(List \*csvTable)  {  for (size\_t i = 0; i < List\_size(csvTable); i++)  {  List \*temp = List\_get(csvTable, i);  for (size\_t j = 0; j < List\_size(temp); j++)  {  free(List\_get(temp, j));  }  List\_free(List\_get(csvTable, i));  }  } |

**￼Приклади результатів**

****



**Висновки**

Виконавши дану лабораторну роботу ми навчилися працювати із форматом даних CSV(«сі-ес-ві») .

При введені певних аргументів командного рядка, відбувається взаємодія із стовпцем «TEST TIME».

Програма компілювалася за допомогою компілятора gcc.