**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-82**  **Мельничук Олексій Геннадійович**  **варіант №15** |  | **Перевірив**  **“\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ 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.

**Постановка завдання**

UPD-1: Створити у Google Spreadsheet (або у будь-якому аналозі) таблицю, у яку внести мінімум 5 записів типу Динозавр зі стовпцями (мінімум 4), що містять дані екземплярів цього типу (приклад для типу Студент).

Експортувати\завантажити таблицю як CSV-файл і помістити її у директорію проекту як data.csv.

Описати структуру даних типу Динозавр. 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):

IN - (перший вільний аргумент) назва вхідного файлу з CSV

-n N - аргумент для додаткової обробки

-o OUT - назва вихідного файлу

UPD-2. Приклад використання функції getopt() для простого отримання значень опцій програми.

Всі опції не обов'язкові. За замовчуванням:

Якщо нема вхідного файлу - брати дані з testList

Якщо нема аргумента обробки - не обробляти список сутностей.

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).

Для цього потрібно реалізувати універсальний список (List, інтерфейс)

Вивести отриману таблицю у консоль (форматування при виводі довільне).

UPD-4. На основі таблиці inCsvTable сформувати список items екземплярів типу Динозавр.

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 всіх динозаврів із середньою масою тіла меншою за N кг. 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).

Модулі:

main - головний модуль

list - універсальний список (void \*, інтерфейс (UPD-3))

csv - функції для роботи із форматом даних CSV (інтерфейс модуля).

Створити у корені проекту директорію build.

Додати у корінь проекту файл CMakeLists.txt із директивами для збірки проекту за допомогою CMake.

Використовувати для збірки проекта такі команди:

cd build

cmake ..

make

**Тексти коду програм**

|  |
| --- |
| **main.c** |
| #include "list.h"  #include "csv.h"  #include "addon.h"  int main(int argc, char \*argv[argc])  {  bool is\_Input = false;  bool argument = false;  int N = -1;  bool is\_Output = false;  int out\_Index = -1;  char \*inpoot;  char \*outpoot;  parser(argc, argv, &is\_Input, &argument, &N, &is\_Output, &out\_Index);  if(is\_Input == true)  {  inpoot = malloc((strlen(argv[1]) + 1) \* sizeof(char));  strcpy(inpoot, argv[1]);  printf("INPUT: %s\n", inpoot);  }  else  {  inpoot = NULL;  printf("NO INPUT\n");  }  if(argument)  {  printf("N: %d\n", N);  }  else if(!argument)  {  printf("NO N\n");  }  if(is\_Output)  {  outpoot = malloc((strlen(argv[out\_Index]) + 1) \* sizeof(char));  strcpy(outpoot, argv[out\_Index]);  printf("OUTPUT: %s\n", outpoot);  }  else  {  outpoot = NULL;  printf("NO OUTPUT\n");  }  List \* dinos;  if (is\_Input == false)  {  dinos = createTestList();  }  else  {  if(inpoot == NULL)  {  dinos = createTestList();  }  else  {  List table;  List\_init(&table);  fill\_the\_table(&table, fillStringFromFile(inpoot));  dinos = fillDinosListFromTable(&table);  deinitTable(&table);  }  }  print\_dinos(dinos);  if(argument == true)  {  processStructList(dinos, N);  }  print\_dinos(dinos);  List table2;  List\_init(&table2);  fillFromDinos(&table2, dinos);  printStringsTable(&table2, outpoot);  deinitTable(&table2);  freeDinoList(dinos);  free(inpoot);  free(outpoot);  return 0;  } |

|  |
| --- |
| **list.h** |
| #pragma once  #include <stdlib.h>  #include <stdbool.h>  #include <stdio.h>  #include <string.h>  #include <malloc.h>  #include <ctype.h>  #include <assert.h>  typedef void \* T;  typedef struct \_\_List List;  struct \_\_List  {  void \*\* items;  size\_t capacity;  size\_t length;  };  void List\_init (List \* self);  void List\_deinit (List \* self);  List \* List\_alloc (void);  void List\_free (List \* self);  void \* List\_get (List \* self, int index);  size\_t List\_size (List \* self);  void List\_add (List \* self, void \* value);  void List\_realloc(List \*self, int newCapacity);  char \*createOnHeap(char \*str); |

|  |
| --- |
| **list.c** |
| #include "list.h"  void List\_init (List \* self)  {  self->capacity = 2;  self->length = 0;  self->items = malloc(sizeof(void \*) \* self->capacity);  if (self->items == NULL)  {  printf("Allocation error\n");  abort();  }  }  void List\_deinit (List \* self)  {  free(self->items);  }  List \* List\_alloc (void)  {  List \* self = malloc(sizeof(List));  List\_init(self);  return self;  }  void List\_free (List \* self)  {  List\_deinit(self);  free(self);  }  void List\_realloc(List \*self, int newCapacity)  {  void \*\*new = realloc((self->items), sizeof(void\*) \* newCapacity);  if (new == NULL)  {  free(self->items);  fprintf(stderr, "Reallocation error");  abort();  }  self->items = new;  self->capacity = newCapacity;  // printf("Reallocated\n");  }  void List\_add(List \* self, T value)  {  self->items[self->length] = value;  self->length += 1;  if (self->length == self->capacity)  {  // realloc  int newCapacity = self->capacity + 1;  List\_realloc(self, newCapacity);  }  }  T List\_get (List \* self, int index)  {  return self->items[index];  }  size\_t List\_size (List \* self)  {  return self->length;  }  char \*createOnHeap(char \*str)  {  char \*meme = malloc(sizeof(char) \* (strlen(str) + 1));  strcpy(meme, str);  return meme;  } |

|  |
| --- |
| **csv.h** |
| #include "list.h" // generic list (array implementation)  void fill\_the\_table(List \*table, char file[]);  void deinitTable(List \* table);  void printStrings(List \*pl, FILE \* where);  void printStringsTable(List \*table, char where[]); |

|  |
| --- |
| **csv.c** |
| #include "csv.h"  void printStrings(List \*pl, FILE \* where)  {  for (int i = 0; i < List\_size(pl); i++)  {  char \*item = List\_get(pl, i);  if (i == 0) fprintf(where, "%s", item);  else fprintf(where, ",%s", item);  }  fprintf(where, "\n");  }  void printStringsTable(List \*table, char where[])  {  printf("PRINTING TABLE\n");  FILE \*output;  if(where == NULL)  {  output = stdout;  puts("Name,Period,Family,\"Discovered, year\",\"Weight, kg\"");  }  else  {  output = fopen(where, "w");  }  for (int i = 0; i < List\_size(table); i++)  {  List \*row = List\_get(table, i);  printStrings(row, output);  }  fclose(output);  }  void fill\_the\_table(List \*table, char file[])  {  char \*csvString = file;  char buf[100];  for (int i = 0; i < 100; i++)  {  buf[i] = '\0';  }  //BUFFER TRACKER  int bufX = 0;  bool state = true;  // CsvReadState state = CsvReadState\_Normal;  //CHAR POINTER TO READ THE STRING  char \*p = csvString;  List \*curr;  while (1)  {  //IF IT'S A FIRST LETTER OF  if (p == (char \*)csvString || \*(p - 1) == '\n')  {  List \*new = List\_alloc();  curr = new;  }  if (state == true)  {  if (\*p == '"')  {  //fook it it's escape gotta move to escape rn  state = false;  }  else  {  if (\*p == '\n' || \*p == ',' || \*p == '\0')  {  buf[bufX] = '\0';  List\_add(curr, createOnHeap(&buf[0]));  for (int i = 0; i < bufX + 1; i++)  {  buf[i] = '\0';  }  if (\*p == '\n' || \*p == '\0')  {  List\_add(table, curr);  }  //  bufX = 0;  }  else  {  buf[bufX] = \*p;  bufX += 1;  }  }  }  else // open brackets reading  {  if (\*p == '"')  {  if (\*(p + 1) == '"')  {  buf[bufX] = '"';  bufX += 1;  p += 1;  }  else  {  buf[bufX] = '\0';  List\_add(curr, createOnHeap(&buf[0]));  for (int i = 0; i < bufX + 1; i++)  {  buf[i] = '\0';  }  bufX = 0;  //  if (\*(p + 1) == ',' || \*(p + 1) == '\n')  {  if (\*(p + 1) == '\n')  {  List\_add(table, curr);  }  p += 1;  state = true;  }  else  {  fprintf(stderr, "CSV format error: invalid char '%c' after escape\n", \*(p + 1));  abort();  }  }  }  else  {  buf[bufX] = \*p;  bufX += 1;  }  }  if (\*p == '\0')  {  break;  }  p += 1;  }  free(csvString);  return;  }  void deinitTable(List \* table)  {  for (int i = 0; i < List\_size(table); i++)  {  List \*row = List\_get(table, i);  for (int j = 0; j < List\_size(row); j++)  {  char \*str = List\_get(row, j);  free(str);  }  List\_free(row);  }  List\_deinit(table);  } |

**fgd**

|  |
| --- |
| **addon.h** |
| #pragma once  #include "list.h"  #include "csv.h"  typedef struct dino  {  char \* name;  char \* per;  char \* fam;  char \* disc;  char \* mass;  } Dino;  void parser(int argc, char \*argv[], bool \* is\_Input, bool \* argument, int \* N, bool \* is\_Output, int \* out\_Index);  void processStructList(List \* list, int N);  char \* fillStringFromFile(char from[]);  List \* createTestList();  void print\_dinos(List \* dinos);  void freeDinoList(List \* list);  void fillFromDinos(List \* table, List \* dinos);  List \* fillDinosListFromTable(List \* table); |

|  |
| --- |
| **addon.c** |
| #include "addon.h"  void print\_dinos(List \* dinos)  {  printf("PRINTING LIST OF STRUCTURES\n");  for(int i = 0; i < (int)List\_size(dinos); i++)  {  Dino \* din = List\_get(dinos, i);  if(!printf("%s, %s, %s, %s, %s\n", din->name, din->per, din->fam, din->disc, din->mass))  {  printf("UNSUCCESSFUL\n");  }  }  puts("");  return;  }  void parser(int argc, char \*argv[argc], bool \* is\_Input, bool \* argument, int \* N, bool \* is\_Output, int \* out\_Index)  {  if(argc > 1)  {  for(int i = 1; i < argc; i++)  {  if(i == 1)  {  if(strstr(argv[1], "."))  {  \*is\_Input = true;  }  }  if(strcmp(argv[i], "-n") == 0 || strcmp(argv[i], "-N") == 0)  {  i++;  if(strcmp(argv[i], "-o") != 0 || i != argc - 1)  {  \*argument = true;  \*N = atoi(argv[i]);  }  }  if(strcmp(argv[i], "-o") == 0)  {  i++;  if(i == argc)  {  \*is\_Output = false;  puts("No file was found.");  }  else if(strcmp(argv[i], "-n") != 0 && i < argc)  {  \*is\_Output = true;  \*out\_Index = i;  }  }  }  }  }  char \* fillStringFromFile(char from[])  {  FILE \*file;  char ch = 0;  int count = 0;  file = fopen(from, "r");  while((ch = fgetc(file)) != EOF)  {  if(ch == '\r')  {  //THANK YOU, CSV, VERY COOL  }  else  {  count++;  }  }  count++;  fclose(file);  file = fopen(from, "r");  char \*csvString = malloc(count \* sizeof(char));  for(int i = 0; i < count; i++)  {  csvString[i] = '\0';  }  char \*roll = csvString;  while((ch = fgetc(file)) != EOF)  {  if(ch == '\r')  {  //THANK YOU, CSV, VERY COOL  }  else  {  \*roll = ch;  roll++;  }  }  \*(roll-1) = '\0';  fclose(file);  return csvString;  }  void processStructList(List \* list, int N)  {  int i = 0;  while(i < List\_size(list))  {  Dino \*dino = List\_get(list, i);  if(atoi(dino->mass) < N)  {  free(dino->name);  free(dino->per);  free(dino->fam);  free(dino->disc);  free(dino->mass);  free(dino);  for(int j = i; j < List\_size(list) - 1; j++)  {  list->items[j] = list->items[j+1];  }  list->length--;  }  else  {  i++;  }    }  }  List \* createTestList()  {  List \* dinos = List\_alloc();  Dino \* dino1 = malloc(sizeof(Dino));  dino1->name = createOnHeap("Dinasaur");  dino1->per = createOnHeap("A long time ago");  dino1->fam = createOnHeap("Rawr XD");  dino1->disc = createOnHeap("1337");  dino1->mass = createOnHeap("228");  List\_add(dinos, dino1);  Dino \* dino2 = malloc(sizeof(Dino));  dino2->name = createOnHeap("Diego");  dino2->per = createOnHeap("1800s");  dino2->fam = createOnHeap("Brando");  dino2->disc = createOnHeap("1870");  dino2->mass = createOnHeap("78");  List\_add(dinos, dino2);  Dino \* dino3 = malloc(sizeof(Dino));  dino3->name = createOnHeap("Dr. Ferdinand");  dino3->per = createOnHeap("1800s");  dino3->fam = createOnHeap("Unknown");  dino3->disc = createOnHeap("1800s");  dino3->mass = createOnHeap("80");  List\_add(dinos, dino3);  return dinos;  }  void freeDinoList(List \* list)  {  for (int i = 0; i < List\_size(list); i++)  {  Dino \* din = List\_get(list, i);  free(din->name);  free(din->per);  free(din->fam);  free(din->disc);  free(din->mass);  free(din);  }  List\_free(list);  }  void fillFromDinos(List \* table, List \* dinos)  {  for(int i = 0; i < List\_size(dinos); i++)  {  Dino \* dino = List\_get(dinos, i);  List \*newrow = List\_alloc();  List\_add(newrow, createOnHeap(dino->name));  List\_add(newrow, createOnHeap(dino->per));  List\_add(newrow, createOnHeap(dino->fam));  List\_add(newrow, createOnHeap(dino->disc));  List\_add(newrow, createOnHeap(dino->mass));  List\_add(table, newrow);  }  }  List \* fillDinosListFromTable(List \* table)  {  List \* dinos = List\_alloc();  for(int i = 0; i < (int)table->length; i++)  {  Dino \* dino = malloc(sizeof(Dino));  List \*row = List\_get(table, i);  dino->name = createOnHeap(List\_get(row, 0));  dino->per = createOnHeap(List\_get(row, 1));  dino->fam = createOnHeap(List\_get(row, 2));  dino->disc = createOnHeap(List\_get(row, 3));  dino->mass = createOnHeap(List\_get(row, 4));  List\_add(dinos, dino);  }  return dinos;  } |

|  |
| --- |
| **data.csv** |
| Plotosaurus,Cretaceous,Mosasauridae,1941,2400  Coelophysis,Triassic,Coelophysidae,1887,20  Echinodon,Cretaceous,Heterodontosauridae,1861,7  Saltasaurus,Cretaceous,Saltasauridae,1977,7000  Diplodocus,Jurassic,Diplodocidae,1877,13000 |

|  |
| --- |
| **CMakeLists.txt** |
| cmake\_minimum\_required(VERSION 3.6.1)  file(GLOB SOURCES "\*.c")  add\_executable(exec\_project ${SOURCES}) |

**Робота коду**

Програма будується з допомогою Сmake.

Програма сканує ввід і з’ясовує наявність потрібних параметрів.

Створюється список структур типу “динозавр”.

Якщо не було вказано файл вводу або він не був знайдений, то список заповнюється захаркоджено.

Якщо файл таблиці був знайдений, то створюється список списків, який заповнюється значеннями з таблиці, а потім список структур заповнюється з цього списку списків, який потім очищується.

Якщо було вказане значення після -n, то зі списку структур видаляються структури, у яких числове значення менше за введене.

Потім створюється новий список списків, який заповнюється із списку структур і виводиться або у консоль, або у файл виводу, який був вказаний у вводі.

Списки очищуються і програма завершує роботу.

**Приклади результатів**

Результат у консолі:

|  |
| --- |
| aleksey@auv:~/Desktop/local code/VARIOUS/lab2t/nproject/build$ cmake ..  -- The C compiler identification is GNU 7.3.0  -- The CXX compiler identification is GNU 7.3.0  -- Check for working C compiler: /usr/bin/cc  ^[[A-- Check for working C compiler: /usr/bin/cc -- works  -- Detecting C compiler ABI info  -- Detecting C compiler ABI info - done  -- Detecting C compile features  ^[[A-- Detecting C compile features - done  -- Check for working CXX compiler: /usr/bin/c++  -- Check for working CXX compiler: /usr/bin/c++ -- works  -- Detecting CXX compiler ABI info  -- Detecting CXX compiler ABI info - done  -- Detecting CXX compile features  -- Detecting CXX compile features - done  -- Configuring done  -- Generating done  -- Build files have been written to: /home/aleksey/Desktop/local code/VARIOUS/lab2t/nproject/build  aleksey@auv:~/Desktop/local code/VARIOUS/lab2t/nproject/build$ make  Scanning dependencies of target exec\_project  [ 20%] Building C object CMakeFiles/exec\_project.dir/addon.c.o  [ 40%] Linking C executable exec\_project  [100%] Built target exec\_project  aleksey@auv:~/Desktop/local code/VARIOUS/lab2t/nproject/build$ ./exec\_project  NO INPUT  NO N  NO OUTPUT  PRINTING LIST OF STRUCTURES  Dinasaur, A long time ago, Rawr XD, 1337, 228  Diego, 1800s, Brando, 1870, 78  Dr. Ferdinand, 1800s, Unknown, 1800s, 80  PRINTING LIST OF STRUCTURES  Dinasaur, A long time ago, Rawr XD, 1337, 228  Diego, 1800s, Brando, 1870, 78  Dr. Ferdinand, 1800s, Unknown, 1800s, 80  PRINTING TABLE  Name,Period,Family,"Discovered, year","Weight, kg"  Dinasaur,A long time ago,Rawr XD,1337,228  Diego,1800s,Brando,1870,78  Dr. Ferdinand,1800s,Unknown,1800s,80  aleksey@auv:~/Desktop/local code/VARIOUS/lab2t/nproject/build$ ./exec\_project data.csv  INPUT: data.csv  NO N  NO OUTPUT  PRINTING LIST OF STRUCTURES  Plotosaurus, Cretaceous, Mosasauridae, 1941, 2400  Coelophysis, Triassic, Coelophysidae, 1887, 20  Echinodon, Cretaceous, Heterodontosauridae, 1861, 7  Saltasaurus, Cretaceous, Saltasauridae, 1977, 7000  Diplodocus, Jurassic, Diplodocidae, 1877, 13000  PRINTING LIST OF STRUCTURES  Plotosaurus, Cretaceous, Mosasauridae, 1941, 2400  Coelophysis, Triassic, Coelophysidae, 1887, 20  Echinodon, Cretaceous, Heterodontosauridae, 1861, 7  Saltasaurus, Cretaceous, Saltasauridae, 1977, 7000  Diplodocus, Jurassic, Diplodocidae, 1877, 13000  PRINTING TABLE  Name,Period,Family,"Discovered, year","Weight, kg"  Plotosaurus,Cretaceous,Mosasauridae,1941,2400  Coelophysis,Triassic,Coelophysidae,1887,20  Echinodon,Cretaceous,Heterodontosauridae,1861,7  Saltasaurus,Cretaceous,Saltasauridae,1977,7000  Diplodocus,Jurassic,Diplodocidae,1877,13000  aleksey@auv:~/Desktop/local code/VARIOUS/lab2t/nproject/build$ ./exec\_project data.csv -n 800  INPUT: data.csv  N: 800  NO OUTPUT  PRINTING LIST OF STRUCTURES  Plotosaurus, Cretaceous, Mosasauridae, 1941, 2400  Coelophysis, Triassic, Coelophysidae, 1887, 20  Echinodon, Cretaceous, Heterodontosauridae, 1861, 7  Saltasaurus, Cretaceous, Saltasauridae, 1977, 7000  Diplodocus, Jurassic, Diplodocidae, 1877, 13000  PRINTING LIST OF STRUCTURES  Plotosaurus, Cretaceous, Mosasauridae, 1941, 2400  Saltasaurus, Cretaceous, Saltasauridae, 1977, 7000  Diplodocus, Jurassic, Diplodocidae, 1877, 13000  PRINTING TABLE  Name,Period,Family,"Discovered, year","Weight, kg"  Plotosaurus,Cretaceous,Mosasauridae,1941,2400  Saltasaurus,Cretaceous,Saltasauridae,1977,7000  Diplodocus,Jurassic,Diplodocidae,1877,13000  aleksey@auv:~/Desktop/local code/VARIOUS/lab2t/nproject/build$ ./exec\_project data.csv -n 800 output.csv  INPUT: data.csv  N: 800  NO OUTPUT  PRINTING LIST OF STRUCTURES  Plotosaurus, Cretaceous, Mosasauridae, 1941, 2400  Coelophysis, Triassic, Coelophysidae, 1887, 20  Echinodon, Cretaceous, Heterodontosauridae, 1861, 7  Saltasaurus, Cretaceous, Saltasauridae, 1977, 7000  Diplodocus, Jurassic, Diplodocidae, 1877, 13000  PRINTING LIST OF STRUCTURES  Plotosaurus, Cretaceous, Mosasauridae, 1941, 2400  Saltasaurus, Cretaceous, Saltasauridae, 1977, 7000  Diplodocus, Jurassic, Diplodocidae, 1877, 13000  PRINTING TABLE  Name,Period,Family,"Discovered, year","Weight, kg"  Plotosaurus,Cretaceous,Mosasauridae,1941,2400  Saltasaurus,Cretaceous,Saltasauridae,1977,7000  Diplodocus,Jurassic,Diplodocidae,1877,13000  aleksey@auv:~/Desktop/local code/VARIOUS/lab2t/nproject/build$ ./exec\_project data.csv -n 800 -o output.csv  INPUT: data.csv  N: 800  OUTPUT: output.csv  PRINTING LIST OF STRUCTURES  Plotosaurus, Cretaceous, Mosasauridae, 1941, 2400  Coelophysis, Triassic, Coelophysidae, 1887, 20  Echinodon, Cretaceous, Heterodontosauridae, 1861, 7  Saltasaurus, Cretaceous, Saltasauridae, 1977, 7000  Diplodocus, Jurassic, Diplodocidae, 1877, 13000  PRINTING LIST OF STRUCTURES  Plotosaurus, Cretaceous, Mosasauridae, 1941, 2400  Saltasaurus, Cretaceous, Saltasauridae, 1977, 7000  Diplodocus, Jurassic, Diplodocidae, 1877, 13000  PRINTING TABLE  aleksey@auv:~/Desktop/local code/VARIOUS/lab2t/nproject/build$ |

**Висновки**

Робота допомогла розібратися в універсальних списках, нагадала роботу з файлами та командним рядком, навчила роботі з Cmake та файлами формату .csv.