**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 № 3**

**з дисципліни “ Основи програмування ”**

**тема “Словники та дерева”**

|  |  |  |
| --- | --- | --- |
| **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**

Реалізувати та використати АТД "Словник" та АТД "Бінарне дерево пошуку".

Навчитись створювати та обходити нелінійні структури даних.

Навчитись створювати і підключати у проект статичні бібліотеки.

**Загальні вимоги до завдання**

Скопіювати у директорію проекту код і допоміжні файли попередньої лабораторної роботи.

1. Словник [2 бал.]

Створити модуль StrStrMap з ключами та значеннями типу char \* (словник рядок-рядок).

Замінити у коді проекту тип даних сутності (за варіантом) на тип StrStrMap.

Додати у рядки CSV тексту перший стовпчик (якщо такого ще немає), що містить унікальне цілочисельне значення для кожного запису. Зберігати значення цього стовпчика у словники з ключем id.

UPD-1 Приклад використання екземпляра StrStrMap замість Student:

char \* String\_allocCopy (const char \* value);

char \* String\_allocFromInt (int value);

char \* String\_allocFromDouble (double value);

StrStrMap \* createStudentMap(int id, const char \* fullname, double score)

{

// create values copies as strings on heap

char \* idValue = String\_allocFromInt(id);

char \* fullnameValue = String\_allocCopy(fullname);

char \* scoreValue = String\_allocFromDouble(score);

// create map on heap and add values

StrStrMap \* map = StrStrMap\_alloc();

StrStrMap\_add(map, "id", idValue);

StrStrMap\_add(map, "fullname", fullnameValue);

StrStrMap\_add(map, "score", scoreValue);

// test checks

assert(StrStrMap\_contains(map, "id"));

assert(StrStrMap\_contains(map, "fullname"));

assert(StrStrMap\_contains(map, "score"));

// return allocated instance

return map;

}

2. Бінарне дерево пошуку [2 бал.]

Реалізувати модуль BSTree (бінарне дерево пошуку, БДП), що може оперувати значеннями вузлів типу StrStrMap і ключами вузлів по ключу id словника.

Додати у програму обробку нової булевої (наявна або ні, без значення) опції командного рядка -b.

За наявності такого ключа після формування списку з елементами StrStrMap додати їх у нове бінарне дерево пошуку.

Вивести вміст побудованого дерева у консоль.

Якщо наявна опція -n - використати її умову з попередньої лабораторної роботи для того, щоби по даному критерію знайти у списку словників всі потрібні записи і видалити їх по ключах з БДП.

Вивести вміст зміненого дерева у консоль.

3. Екранування CSV значень [1 бал.]

Додати до розбору і створення CSV-тексту екранування значень, що містять символи:

коми

подвійних лапок

переходу на новий рядок.

4. Статична бібліотека [1 бал.]

Винести код модуля csv та його залежностей (list) у статичну бібліотеку libcsvlab.

Налаштувати CMake проект для збірки бібліотеки libcsvlab.

Підключати бібліотеку libcsvlab у проект програми (без її встановлення).

**Тексти коду програм**

|  |
| --- |
| **program/main.c** |
| #include <list.h>  #include <csv.h>  #include <addon.h>  #include <strmap.h>  #include <btree.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;  bool totree = false;  char \*inpoot;  char \*outpoot;  parser(argc, argv, &is\_Input, &argument, &N, &is\_Output, &out\_Index, &totree);  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");  }  if(totree)  {  printf("COPY TO TREE: TRUE\n");  }  else  {  printf("COPY TO TREE: FALSE\n");  }    puts("");  List \* dinos;  if (is\_Input == false)  {  dinos = createTestList();  }  else  {  char \* fill = fillStringFromFile(inpoot);  if(fill == NULL)  {  printf("Input file was not found.\n");  dinos = createTestList();  free(fill);  }  else  {  List table;  List\_init(&table);  fill\_the\_table(&table, fill);  dinos = fillDinosListFromTable(&table);  deinitTable(&table);  }  }  printDinosMapList(dinos, NULL);  int len = (int)List\_size(dinos);  if(totree)  {  BSTree bst;  BSTree\_init(&bst);  for (int i = 0; i < len; i++)  {  StrMap \*dino = List\_get(dinos, i);  BSTree\_insert(&bst, dino);  }  printTree(bst.root);  if (argument == true)  {  printf("\nPROCESSING...\n");  traverseAndDelet(bst.root, N, &bst);  printTree(bst.root);  printf("\n");  }  clearBinTree(bst.root);  }  else  {  if(argument == true)  {  processMapList(dinos, N);  printDinosMapList(dinos, NULL);  }  }    if(is\_Output)  {  printDinosMapList(dinos, outpoot);  }  freeListOMaps(dinos);  free(inpoot);  free(outpoot);  return 0;  } |

|  |
| --- |
| **libcsvlab/list.h** |
| #pragma once  #include <stdlib.h>  #include <stdbool.h>  #include <stdio.h>  #include <string.h>  #include <malloc.h>  #include <ctype.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);  char \*createOnHeap(char \*str); |

|  |
| --- |
| **libcsvlab/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;  } |

|  |
| --- |
| **libcsvlab/csv.h** |
| #pragma once  #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[]); |

|  |
| --- |
| **libcsvlab/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;  int commacount = 0;  List \*curr;  while (1)  {  //IF IT'S A FIRST LETTER OF  if ((p == (char \*)csvString || \*(p - 1) == '\n') && state == true)  {  //BASICALLY CREATING A NEW ROW OF STRINGS  List \*new = List\_alloc();  curr = new;  }  if (state == true)  {  if (\*p == '"')  {  buf[bufX] = \*p;  bufX++;  buf[bufX] = \*p;  bufX++;  }  else  {  if(\*p == ' ' || \*p == ',' || \*p == '\n')  {  if (\*p == ' ')  {  for (int i = strlen(buf); i > 0; i--)  {  buf[i] = buf[i - 1];  }  buf[0] = '"';  bufX++;  buf[bufX] = \*p;  bufX++;  state = false;  }  else if (\*p == ',')  {  if(buf[0] == '\0')  {  buf[0] = '"';  buf[1] = '"';  buf[2] = '\0';  List\_add(curr, createOnHeap(&buf[0]));  printf("{ADDED %s}\n", buf);  for (int i = 0; i < bufX + 1; i++)  {  buf[i] = '\0';  }  commacount++;  bufX = 0;  p++;  continue;  }  if (\*(p + 1) == ' ' || \*(p + 1) == '\n')  {  for (int i = strlen(buf); i > 0; i--)  {  buf[i] = buf[i - 1];  }  buf[0] = '"';  bufX++;  buf[bufX] = \*p;  bufX++;  state = false;  }  else if (\*(p+1) != ' ' || \*(p+1) != '\n')  {  buf[bufX] = '\0';  List\_add(curr, createOnHeap(&buf[0]));  printf("{ADDED %s}\n", buf);  for (int i = 0; i < bufX + 1; i++)  {  buf[i] = '\0';  }  commacount++;  //  bufX = 0;  }  }  else if(\*p == '\n')  {  if (/\*isdigit(\*(p + 1)) &&\*/ commacount == 5)  {  buf[bufX] = '\0';  List\_add(curr, createOnHeap(&buf[0]));  printf("{ADDED %s}\n", buf);  for (int i = 0; i < bufX + 1; i++)  {  buf[i] = '\0';  }  List\_add(table, curr);  printf("!ADDED NEW ROW!\n\n");  commacount = 0;  bufX = 0;  p++;  continue;  }  else  {  state = false;  for (int i = strlen(buf); i > 0; i--)  {  buf[i] = buf[i - 1];  }  buf[0] = '"';  bufX++;  buf[bufX] = \*p;  bufX++;  }  }  }  else  {  if (\*p == '\0')  {  buf[bufX] = '\0';  List\_add(curr, createOnHeap(&buf[0]));  printf("{ADDED %s}\n", buf);  for (int i = 0; i < bufX + 1; i++)  {  buf[i] = '\0';  }  List\_add(table, curr);    printf("!ADDED NEW ROW!\n\n");  bufX = 0;  }  else  {  buf[bufX] = \*p;  bufX += 1;  }  }  }  }  else if(state == false) // open brackets reading  {  if (\*p == '"')  {  buf[bufX] = \*p;  bufX++;  buf[bufX] = \*p;  bufX++;  }  else  {  if(\*p == ' ' || \*p == ',' || \*p == '\n')  {  if (\*p == ' ')  {  buf[bufX] = \*p;  bufX++;  }  else if (\*p == ',')  {  if (\*(p + 1) == ' ' || \*(p + 1) == '\n')  {  buf[bufX] = \*p;  bufX++;  }  else  {  buf[bufX] = '"';  bufX++;  buf[bufX] = '\0';  state = true;  List\_add(curr, createOnHeap(&buf[0]));  printf("{ADDED %s}\n", buf);  for (int i = 0; i < bufX + 1; i++)  {  buf[i] = '\0';  }  commacount++;  //  bufX = 0;  }  }  else if (\*p == '\n')  {  if (/\*isdigit(\*(p + 1)) && \*/commacount == 5)  {  state = true;  buf[bufX] = '"';  bufX++;  buf[bufX] = '\0';  List\_add(curr, createOnHeap(&buf[0]));  printf("{ADDED %s}\n", buf);  for (int i = 0; i < bufX + 1; i++)  {  buf[i] = '\0';  }  List\_add(table, curr);  printf("!ADDED NEW ROW!\n\n");  commacount = 0;  bufX = 0;  }  else  {  buf[bufX] = \*p;  bufX++;  }  }  }  else  {  if(\*p == '\0')  {  buf[bufX] = '"';  bufX++;  buf[bufX] = '\0';  List\_add(curr, createOnHeap(&buf[0]));  printf("{ADDED %s}\n", buf);  for (int i = 0; i < bufX + 1; i++)  {  buf[i] = '\0';  }    List\_add(table, curr);  printf("!ADDED NEW ROW!\n\n");  //  bufX = 0;  }  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);  } |

|  |
| --- |
| **libcsvlab/addon.h** |
| #pragma once  #include "list.h"  #include "csv.h"  #include "strmap.h"  #include <assert.h>  typedef struct dino  {  int id;  char \* name;  char \* per;  char \* fam;  int disc;  double mass;  } Dino;  void parser(int argc, char \*argv[], bool \* is\_Input, bool \* argument, int \* N, bool \* is\_Output, int \* out\_Index, bool \* totree);  void processMapList(List \* list, int N);  char \* fillStringFromFile(char from[]);  List \* createTestList();  void freeDinoList(List \* list);  void fillFromDinos(List \* table, List \* dinos);  List \* fillDinosListFromTable(List \* table);  char \* String\_allocCopy(const char \* value);  char \* String\_allocFromInt(int value);  char \* String\_allocFromDouble(double value);  StrMap \* createDinoMap(Dino \* dino);  void printDinosMapList(List \* DinoMaps, char \* where);  void fillDinoMapList(List \* list, int DinoArrLen, Dino DinosArr[]);  void freeListOMaps(List \* maps); |

|  |
| --- |
| **libcsvlab/addon.c** |
| #include "addon.h"  void parser(int argc, char \*argv[argc], bool \* is\_Input, bool \* argument, int \* N, bool \* is\_Output, int \* out\_Index, bool \* totree)  {  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 || strcmp(argv[i], "-b") != 0)  {  \*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 && strcmp(argv[i], "-b") != 0)  {  \*is\_Output = true;  \*out\_Index = i;  }  }  if(strcmp(argv[i], "-b") == 0)  {  \*totree = true;  }  }  }  }  char \* fillStringFromFile(char from[])  {  char ch = 0;  int count = 0;  FILE \*file;  if((file = fopen(from, "r")) == NULL)  {  return NULL;  }  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(sizeof(char) \* count);  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 processMapList(List \* list, int N)  {  int i = 0;  while(i < List\_size(list))  {  StrMap \*dino = List\_get(list, i);  char copy[100];  strcpy(copy, StrMap\_getVal(dino, "mass"));  int mass = atoi(copy);  printf("\n(%d)\n", mass);  if(mass < N)  {  for(int k = 0; k < 6; k++)  {  free((char\*)dino->items[k].value);  }  free(dino->items);  free(dino);  for(int j = i; j < List\_size(list) - 1; j++)  {  list->items[j] = list->items[j+1];  }  list->length--;  }  else  {  i++;  }    }  }  char \* String\_allocCopy(const char \* value)  {  char \*meme = malloc(sizeof(char) \* (strlen(value) + 1));  strcpy(meme, value);  return meme;  }  char \* String\_allocFromInt(int value)  {  char meme[10];  sprintf(meme, "%d", value);  return String\_allocCopy(meme);  }  char \* String\_allocFromDouble(double value)  {  char buf[100];  sprintf(buf, "%f", value);  char \* meme = String\_allocCopy(buf);  return meme;  }  StrMap \* createDinoMap(Dino \* dino)  {  char \* idValue = String\_allocFromInt(dino->id);  char \* fullnameValue = String\_allocCopy(dino->name);  char \* existed = String\_allocCopy(dino->per);  char \* family = String\_allocCopy(dino->fam);  char \* discovery = String\_allocFromInt(dino->disc);  char \* massval = String\_allocFromDouble(dino->mass);  // printf("%s\n", idValue);  // printf("%s\n", fullnameValue);  // printf("%s\n", existed);  // printf("%s\n", family);  // printf("%s\n", discovery);  // printf("%s\n", massval);  StrMap \* map = StrMap\_alloc();  StrMap\_add(map, "id", idValue);  StrMap\_add(map, "name", fullnameValue);  StrMap\_add(map, "period", existed);  StrMap\_add(map, "family", family);  StrMap\_add(map, "discovered", discovery);  StrMap\_add(map, "mass", massval);  assert(StrMap\_contains(map, "id"));  assert(StrMap\_contains(map, "name"));  assert(StrMap\_contains(map, "period"));  assert(StrMap\_contains(map, "family"));  assert(StrMap\_contains(map, "discovered"));  assert(StrMap\_contains(map, "mass"));  // for(int i = 0; i < 6; i++)  // {  // printf("|%s - ", map->items[i].value);  // }  // printf("\n\n");  return map;  }  void printDinosMapList(List \* DinoMaps, char \* where)  {  printf("PRINTING List\n");  FILE \*output;  if(where == NULL)  {  output = stdout;  puts("Id,Name,Period,Family,\"Discovered, year\",\"Weight, kg\"");  }  else  {  output = fopen(where, "w");  }  for(int i = 0; i < List\_size(DinoMaps); i++)  {  StrMap \* doono = List\_get(DinoMaps, i);  fprintf(output, "%s,%s,%s,%s,%s,%s\n", StrMap\_getVal(doono, "id"),  StrMap\_getVal(doono, "name"), StrMap\_getVal(doono, "period"),  StrMap\_getVal(doono, "family"), StrMap\_getVal(doono, "discovered"),  StrMap\_getVal(doono, "mass"));  }  puts("");  if(where != NULL)  {  fclose(output);  }  }  void fillDinoMapList(List \* list, int DinoArrLen, Dino DinosArr[])  {  for (int i = 0; i < DinoArrLen; i++)  {  Dino \* dino = &DinosArr[i];  StrMap \* studentMap = createDinoMap(dino);  List\_add(list, studentMap);  }  }  List \* createTestList()  {  List \* dinos = List\_alloc();  Dino DinosArr[] = {  {223, "Dinosaur", "A long time ago", "Rawr XD", 1337, 228},  {56, "Diego", "1800s", "Brando", 1870, 78},  {47, "Dr. Ferdinand", "1800s", "Unknown", 1800, 70},  };  const int DinosArrLen = sizeof(DinosArr) / sizeof(DinosArr[0]);  fillDinoMapList(dinos, DinosArrLen, DinosArr);  return dinos;  }  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();  char id[100];  char disc[100];  char mass[100];  sprintf(id, "%i", dino->id);  sprintf(disc, "%i", dino->disc);  sprintf(mass, "%f", dino->mass);  List\_add(newrow, createOnHeap(id));  List\_add(newrow, createOnHeap(dino->name));  List\_add(newrow, createOnHeap(dino->per));  List\_add(newrow, createOnHeap(dino->fam));  List\_add(newrow, createOnHeap(disc));  List\_add(newrow, createOnHeap(mass));  List\_add(table, newrow);  }  }  List \* fillDinosListFromTable(List \* table)  {  List \* dinos = List\_alloc();  for(int i = 0; i < (int)table->length; i++)  {  List \*row = List\_get(table, i);  char \* idValue = String\_allocCopy(List\_get(row, 0));  // printf("%s\n", idValue);  char \* fullnameValue = String\_allocCopy(List\_get(row, 1));  // printf("%s\n", fullnameValue);  char \* existed = String\_allocCopy(List\_get(row, 2));  // printf("%s\n", existed);  char \* family = String\_allocCopy(List\_get(row, 3));  // printf("%s\n", family);  char \* discovery = String\_allocCopy(List\_get(row, 4));  // printf("%s\n", discovery);  char \* massval = String\_allocCopy(List\_get(row, 5));  // printf("%s\n\n", massval);  StrMap \* map = StrMap\_alloc();  StrMap\_add(map, "id", idValue);  StrMap\_add(map, "name", fullnameValue);  StrMap\_add(map, "period", existed);  StrMap\_add(map, "family", family);  StrMap\_add(map, "discovered", discovery);  StrMap\_add(map, "mass", massval);  List\_add(dinos, map);  }  return dinos;  }  void freeListOMaps(List \* maps)  {  printf("{FREEING %d MAPS}\n", (int)List\_size(maps));  for(int i = 0; i < (int)List\_size(maps); i++)  {  StrMap \*dino = List\_get(maps, i);  for (int k = 0; k < 6; k++)  {  free((char\*)dino->items[k].value);  }  free(dino->items);  free(dino);  }  free(maps->items);  free(maps);  } |

|  |
| --- |
| **libcsvlab/strmap.h** |
| #pragma once  #include <stdlib.h>  #include <stdbool.h>  #include <stdio.h>  #include <string.h>  typedef struct KeyValue KeyValue;  struct KeyValue  {  const char \* key;  const char \* value;  };  typedef struct \_\_StrMap StrMap;  struct \_\_StrMap {  KeyValue \* items;  size\_t capacity;  size\_t size;  };  int KeyValue\_compare(const KeyValue \* a, const KeyValue \* b);  void StrMap\_init(StrMap \* self);  void StrMap\_deinit(StrMap \* self);  size\_t StrMap\_size (StrMap \* self);  KeyValue StrMap\_getKV (StrMap \* self, int index);  const char \* StrMap\_getVal (StrMap \* self, const char \* key);  void StrMap\_set (StrMap \* self, int index, KeyValue value);  const char \* StrStrMap\_setNgetVal (StrMap \* self, const char \* key, const char \* value);  void StrMap\_removeAt (StrMap \* self, int index);  const char \* StrStrMap\_removeNgetVal (StrMap \* self, const char \* key);  void StrMap\_add (StrMap \* self, const char \* key, const char \* value); // add value and sort, tip: use insertion sort  void StrMap\_remove (StrMap \* self, KeyValue value); // < O(n), tip: find index and \_removeAt()  int StrMap\_indexOf (StrMap \* self, KeyValue value); // < O(n), tip: use non-linear search method  bool StrMap\_contains (StrMap \* self, const char \* key); // < O(n), tip: find index and check it for -1  void StrMap\_clear (StrMap \* self);  StrMap \* StrMap\_alloc();  int getStrMap\_key(StrMap \* self); |

|  |
| --- |
| **libcsvlab/strmap.c** |
| #include "strmap.h"  StrMap \* StrMap\_alloc()  {  StrMap \* new = malloc(sizeof(StrMap));  StrMap\_init(new);  return new;  }  void StrMap\_init(StrMap \* self)  {  self->capacity = 16;  self->items = malloc(sizeof(KeyValue) \* self->capacity);  self->size = 0;  }  void StrMap\_deinit(StrMap \* self)  {  free(self->items);  }  size\_t StrMap\_size (StrMap \* self)  {  return self->size;  }  KeyValue StrMap\_getKV (StrMap \* self, int index)  {  return self->items[index];  }  const char \* StrMap\_getVal (StrMap \* self, const char \* key)  {  KeyValue kv;  kv.key = key;  int index = StrMap\_indexOf(self, kv);  if (index == -1)  {  abort();  }  KeyValue kvf = StrMap\_getKV(self, index);  return kvf.value;  }  void StrMap\_set (StrMap \* self, int index, KeyValue value)  {  self->items[index] = value;  }  const char \* StrStrMap\_setNgetVal (StrMap \* self, const char \* key, const char \* value)  {  KeyValue kv;  kv.key = key;  kv.value = value;  int index = StrMap\_indexOf(self, kv);  if (index == -1)  {  abort();  }  KeyValue kvOld = StrMap\_getKV(self, index);  StrMap\_set(self, index, kv);  return kvOld.value;  }  void StrMap\_removeAt (StrMap \* self, int index)  {  // @todo check index  for (int i = index; i < self->size - 1; i++)  {  self->items[i] = self->items[i + 1];  }  self->size -= 1;  }  int KeyValue\_compare(const KeyValue \* a, const KeyValue \* b)  {  return strcmp(a->key, b->key);  }  void StrMap\_realloc(StrMap \*self, int newCapacity)  {  KeyValue \*new = realloc((self->items), sizeof(void\*) \* newCapacity);  if (new == NULL)  {  free(self->items);  // printf("Reallocation error\n");  abort();  }  self->items = new;  self->capacity = newCapacity;  // printf("Reallocated\n");  }  void StrMap\_add (StrMap \* self, const char \* key, const char \* value)  {  KeyValue kv;  kv.key = key;  kv.value = value;    // self->items[self->size] = kv;  // self->size += 1;  for (int i = self->size; i >= 0; i--)  {  if (i == 0)  {  // stop  self->items[i] = kv;  }  else  {  int cmp = KeyValue\_compare(&kv, &self->items[i - 1]);  if (cmp >= 0)  {  // stop  self->items[i] = kv;  break;  }  else  {  self->items[i] = self->items[i - 1];  }  }  }  self->size += 1;  if (self->size == self->capacity)  {  // realloc  int newCapacity = self->capacity + 1;  StrMap\_realloc(self, newCapacity);  }  }  void StrMap\_remove (StrMap \* self, KeyValue kv)  {  int index = StrMap\_indexOf(self, kv);  if (index == -1) return;  for (int i = index; i < self->size - 1; i++)  {  self->items[i] = self->items[i + 1];  }  self->size -= 1;  }  const char \* StrStrMap\_removeNgetVal (StrMap \* self, const char \* key)  {  KeyValue kv;  kv.key = key;  int index = StrMap\_indexOf(self, kv);  if (index == -1)  {  abort();  }  KeyValue kvf = StrMap\_getKV(self, index);  StrMap\_removeAt(self, index);  return kvf.value;  }  int StrMap\_indexOf (StrMap \* self, KeyValue kv)  {  //self->items  KeyValue \* item = bsearch(&kv, self->items, self->size, sizeof(KeyValue), (\_\_compar\_fn\_t)KeyValue\_compare);  if (item == NULL) return -1;  return item - self->items;  }  bool StrMap\_contains (StrMap \* self, const char \* key)  {  KeyValue kv;  kv.key = key;  return StrMap\_indexOf(self, kv) >= 0;  }  void StrMap\_clear (StrMap \* self)  {  self->size = 0;  }  int getStrMap\_key(StrMap \* self)  {    KeyValue \* kv = &(self->items[2]);  int keey = atoi(kv->value);  printf("keyis %d\n", keey);  return keey;  } |

|  |
| --- |
| **libcsvlab/btree.h** |
| #pragma once  #include "strmap.h"  #include "addon.h"  #include "strmap.h"  #include <stdio.h>  typedef struct \_\_BinTree BinTree;  struct \_\_BinTree  {  int key;  StrMap \* map; // set on init/alloc  BinTree \* left; // set to NULL on init  BinTree \* right; // set to NULL on init  };  typedef struct \_\_BSTree BSTree;  struct \_\_BSTree  {  BinTree \* root; // a pointer to the root tree node  };  void BinTree\_init (BinTree \* self, StrMap \* map);  BinTree \* BinTree\_alloc (StrMap \* map);  void BinTree\_free (BinTree \* self);  void clearBinTree(BinTree \* node);  static void printValueOnLevel(BinTree \* node, char pos, int depth);  void print(BinTree \* node, char pos, int depth);  void printTree(BinTree \* root);  void BSTree\_init(BSTree \* self);  void BSTree\_deinit(BSTree \* self);  StrMap SMdelete(BinTree \* node, int key, BinTree \* parent);  void modifyTreeOnDelete(BinTree \* node, BinTree \* parent);  void BSTree\_insert (BSTree \* self, StrMap \* map); // add unique  bool BSTree\_lookup (BSTree \* self, int key); // check for value with a key  StrMap \* BSTree\_search (BSTree \* self, int key); // get the value for a key  StrMap BSTree\_delete (BSTree \* self, int key); // delete the value for a key  void BSTree\_clear (BSTree \* self); // delete all values  void traverseAndDelet(BinTree \* node, int N, BSTree \* root); |

|  |
| --- |
| **libcsvlab/btree.c** |
| #include "btree.h"  void BinTree\_init (BinTree \* self, StrMap \* map)  {  self->key = getStrMap\_key(map);  self->map = map;  self->left = NULL;  self->right = NULL;  }  BinTree \* BinTree\_alloc (StrMap \* map)  {  BinTree \* self = malloc(sizeof(BinTree));  BinTree\_init(self, map);  return self;  }  void BinTree\_free (BinTree \* self)  {  free(self);  }  void clearBinTree(BinTree \* node)  {  if (node == NULL) return;  clearBinTree(node->left);  clearBinTree(node->right);  BinTree\_free(node);  }  static void printValueOnLevel(BinTree \* node, char pos, int depth)  {  for (int i = 0; i < depth; i++) {  printf("....");  }  printf("%c: ", pos);  if (node == NULL) {  printf("(null)\n");  } else {  printf("[%d]: ",node->key);  for(int i = 0; i < 6; i++)  {  printf("%s - ", node->map->items[i].value);  }  printf("\n");  }  }  void print(BinTree \* node, char pos, int depth)  {  bool isNotNull = (node != NULL) && (node->left != NULL || node->right != NULL);  if (isNotNull) print(node->right, 'R', depth + 1);  printValueOnLevel(node, pos, depth);  if (isNotNull) print(node->left, 'L', depth + 1);  }  void printTree(BinTree \* root)  {  printf("\nPRINTING TREE\n");  print(root, '+', 0);  }  void BSTree\_init(BSTree \* self)  {  self->root = NULL;  }  void BSTree\_deinit(BSTree \* self)  {  BinTree\_free(self->root);  }  static void insert(BinTree \* node, BinTree \* newNode)  {  if (newNode->key == node->key)  {  fprintf(stderr, "`%i` Already exists\n", newNode->key);  abort();  }  if (newNode->key < node->key)  {  if (node->left == NULL)  {  node->left = newNode;  }  else  {  insert(node->left, newNode);  }  }  else if (newNode->key > node->key)  {  if (node->right == NULL)  {  node->right = newNode;  }  else  {  insert(node->right, newNode);  }  }  }  void BSTree\_insert (BSTree \* self, StrMap \* map)  {  BinTree \* newNode = BinTree\_alloc(map);  if (self->root == NULL)  {  self->root = newNode;  }  else  {  insert(self->root, newNode);  }  }  static bool lookup(BinTree \* node, int key)  {  if (node == NULL) return false;  if (node->key == key) return true;  if (key < node->key) return lookup(node->left, key);  if (key > node->key) return lookup(node->right, key);  else return false;  }    bool BSTree\_lookup (BSTree \* self, int key)  {  return lookup(self->root, key);  }  static StrMap \* search(BinTree \* node, int key)  {  if (node == NULL) return NULL;  if (node->key == key) return node->map;  if (key < node->key) return search(node->left, key);  if (key > node->key) return search(node->right, key);  else return NULL;  }    StrMap \* BSTree\_search (BSTree \* self, int key)  {  return search(self->root, key);  }  static BinTree \* searchMin(BinTree \* node)  {  if (node == NULL) return NULL;  if (node->left == NULL) return node;  return searchMin(node->left);  }  void modifyTreeOnDelete(BinTree \* node, BinTree \* parent)  {  if (node->left == NULL && node->right == NULL)  {  // case A  if (parent->left == node) parent->left = NULL;  else parent->right = NULL;  }  else if (node->left == NULL || node->right == NULL)  {  // case B  BinTree \* child = (node->left != NULL) ? node->left : node->right;  if (parent->left == node) parent->left = child;  else parent->right = child;  }  else  {  // case C  BinTree \* minNode = searchMin(node->right);  int minKey = minNode->key;  StrMap deletedValue = SMdelete(node->right, minKey, parent);  BinTree \* newMin = BinTree\_alloc(&deletedValue);  newMin->left = node->left;  newMin->right = node->right;  if (parent->left == node) parent->left = minNode;  else parent->right = minNode;    }  }  StrMap SMdelete(BinTree \* node, int key, BinTree \* parent)  {  if (node == NULL)  {  fprintf(stderr, "`%i` not found\n", key);  abort();  }  if (key < node->key) return SMdelete(node->left, key, node);  else if (key > node->key) return SMdelete(node->right, key, node);  else {  modifyTreeOnDelete(node, parent);  StrMap old = \*node->map;  BinTree\_free(node);  return old;  }  }  StrMap BSTree\_delete (BSTree \* self, int key)  {  BinTree fakeRoot;  fakeRoot.left = self->root;  StrMap old = SMdelete(self->root, key, &fakeRoot);  self->root = fakeRoot.left;  return old;  }  void traverseAndDelet(BinTree \* node, int N, BSTree \* root)  {  if (node == NULL) return;  traverseAndDelet(node->left, N, root);  if(atoi(node->map->items[3].value) < N)  {  BinTree \* next = node->left;  BinTree \* rext = node->right;  BSTree\_delete(root, node->key);  traverseAndDelet(next, N, root);  traverseAndDelet(rext, N, root);  return;  }  traverseAndDelet(node->right, N, root);  } |

|  |
| --- |
| **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 |

|  |
| --- |
| **program/CMakeLists.txt** |
| cmake\_minimum\_required(VERSION 2.8.9)  project(a.out)  include\_directories(${CMAKE\_SOURCE\_DIR}/../libcsvlab)  link\_directories(${CMAKE\_SOURCE\_DIR}/../libcsvlab/build)  file(GLOB SOURCES "\*.c")  add\_executable(${PROJECT\_NAME} ${SOURCES})  target\_link\_libraries(${PROJECT\_NAME} csvlab) |

|  |
| --- |
| **libcsvlab/CMakeLists.txt** |
| cmake\_minimum\_required(VERSION 2.8.9)  project(csvlab)  file(GLOB SOURCES "\*.c")  #Generate the shared (STATIC, SHARED or MODULE) library from the sources  add\_library(${PROJECT\_NAME} STATIC ${SOURCES}) |

**Робота коду**

Програма і бібліотека будуюься з допомогою Сmake.

Спочатку через термінал у папці libcsvlab/build виконуються команди cmake .. i make, потім ті ж самі команди виконуються у папці program/build.

Програма сканує ввід і з’ясовує наявність потрібних параметрів.

Створюється список мапів.

Якщо не було вказано файл вводу або він не був знайдений, то список заповнюється з допомогою хардкоду.

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

Якщо був вказаний параметр -b, то будується бінарне дерево пошуку і туди заносяться мапи по індексу (перше значення в таблиці) і вміст дерева виноситься у консоль.

Якщо було вказане значення після -n, то зі списку мапів чи дерева видаляються мапи по ключу ‘mass’, у яких числове значення менше за введене.

Якщо вказаний параметр -o, то програма видруковує зі списку мапів значення у файл, що вказаний після параметра.

Список (і дерево) очищується і програма завершує роботу.

**Приклади результатів**

Результат у консолі:

|  |
| --- |
| **libcsvlab/build** |
| aleksey@auv:~/Desktop/lab3/libcsvlab/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  -- Check for working C compiler: /usr/bin/cc -- works  -- Detecting C compiler ABI info  -- Detecting C compiler ABI info - done  -- Detecting C compile features  -- 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/lab3/libcsvlab/build  aleksey@auv:~/Desktop/lab3/libcsvlab/build$ make  Scanning dependencies of target csvlab  [ 16%] Building C object CMakeFiles/csvlab.dir/addon.c.o  [ 33%] Building C object CMakeFiles/csvlab.dir/btree.c.o  [ 50%] Building C object CMakeFiles/csvlab.dir/csv.c.o  [ 66%] Building C object CMakeFiles/csvlab.dir/list.c.o  [ 83%] Building C object CMakeFiles/csvlab.dir/strmap.c.o  [100%] Linking C static library libcsvlab.a  [100%] Built target csvlab  aleksey@auv:~/Desktop/lab3/libcsvlab/build$ |

|  |
| --- |
| **program/build** |
| aleksey@auv:~/Desktop/lab3/program/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  -- Check for working C compiler: /usr/bin/cc -- works  -- Detecting C compiler ABI info  -- Detecting C compiler ABI info - done  -- Detecting C compile features  -- 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/lab3/program/build  aleksey@auv:~/Desktop/lab3/program/build$ make  Scanning dependencies of target a.out  [ 50%] Building C object CMakeFiles/a.out.dir/main.c.o  [100%] Linking C executable a.out  [100%] Built target a.out  aleksey@auv:~/Desktop/lab3/program/build$ ./a.out  NO INPUT  NO N  NO OUTPUT  COPY TO TREE: FALSE  PRINTING List  Id,Name,Period,Family,"Discovered, year","Weight, kg"  223,Dinosaur,A long time ago,Rawr XD,1337,228.000000  56,Diego,1800s,Brando,1870,78.000000  47,Dr. Ferdinand,1800s,Unknown,1800,70.000000  {FREEING 3 MAPS}  aleksey@auv:~/Desktop/lab3/program/build$ ./a.out data.csv  INPUT: data.csv  NO N  NO OUTPUT  COPY TO TREE: FALSE  PRINTING List  Id,Name,Period,Family,"Discovered, year","Weight, kg"  123,Plotosaurus,Cretaceous,Mosasauridae,1941,2400  435,Coelophysis,Triassic,Coelophysidae,1887,20  2654,Echinodon,Cretaceous,Heterodontosauridae,1861,7  876,Ssaltasaurus,Cretaceous,Saltasauridae,1977,7000  908,Diplodocus,Jurassic,"random dinosaur ""name""  with different options to check",1877,13000  {FREEING 5 MAPS}  aleksey@auv:~/Desktop/lab3/program/build$ ./a.out data.csv -n 600  INPUT: data.csv  N: 600  NO OUTPUT  COPY TO TREE: FALSE  PRINTING List  Id,Name,Period,Family,"Discovered, year","Weight, kg"  123,Plotosaurus,Cretaceous,Mosasauridae,1941,2400  435,Coelophysis,Triassic,Coelophysidae,1887,20  2654,Echinodon,Cretaceous,Heterodontosauridae,1861,7  876,Ssaltasaurus,Cretaceous,Saltasauridae,1977,7000  908,Diplodocus,Jurassic,"random dinosaur ""name""  with different options to check",1877,13000  (2400)  (20)  (7)  (7000)  (13000)  PRINTING List  Id,Name,Period,Family,"Discovered, year","Weight, kg"  123,Plotosaurus,Cretaceous,Mosasauridae,1941,2400  876,Ssaltasaurus,Cretaceous,Saltasauridae,1977,7000  908,Diplodocus,Jurassic,"random dinosaur ""name""  with different options to check",1877,13000  {FREEING 3 MAPS}  aleksey@auv:~/Desktop/lab3/program/build$ ./a.out data.csv -n 600 -o output2.csv  INPUT: data.csv  N: 600  OUTPUT: output2.csv  COPY TO TREE: FALSE  PRINTING List  Id,Name,Period,Family,"Discovered, year","Weight, kg"  123,Plotosaurus,Cretaceous,Mosasauridae,1941,2400  435,Coelophysis,Triassic,Coelophysidae,1887,20  2654,Echinodon,Cretaceous,Heterodontosauridae,1861,7  876,Ssaltasaurus,Cretaceous,Saltasauridae,1977,7000  908,Diplodocus,Jurassic,"random dinosaur ""name""  with different options to check",1877,13000  (2400)  (20)  (7)  (7000)  (13000)  PRINTING List  Id,Name,Period,Family,"Discovered, year","Weight, kg"  123,Plotosaurus,Cretaceous,Mosasauridae,1941,2400  876,Ssaltasaurus,Cretaceous,Saltasauridae,1977,7000  908,Diplodocus,Jurassic,"random dinosaur ""name""  with different options to check",1877,13000  PRINTING List  {FREEING 3 MAPS}  aleksey@auv:~/Desktop/lab3/program/build$  aleksey@auv:~/Desktop/lab3/program/build$ ./a.out data.csv -b  INPUT: data.csv  NO N  NO OUTPUT  COPY TO TREE: TRUE  PRINTING List  Id,Name,Period,Family,"Discovered, year","Weight, kg"  123,Plotosaurus,Cretaceous,Mosasauridae,1941,2400  435,Coelophysis,Triassic,Coelophysidae,1887,20  2654,Echinodon,Cretaceous,Heterodontosauridae,1861,7  876,Ssaltasaurus,Cretaceous,Saltasauridae,1977,7000  908,Diplodocus,Jurassic,"random dinosaur ""name""  with different options to check",1877,13000  keyis 123  keyis 435  keyis 2654  keyis 876  keyis 908  PRINTING TREE  ............R: (null)  ........R: [2654]: 1861 - Heterodontosauridae - 2654 - 7 - Echinodon - Cretaceous -  ................R: [908]: 1877 - "random dinosaur ""name""  with different options to check" - 908 - 13000 - Diplodocus - Jurassic -  ............L: [876]: 1977 - Saltasauridae - 876 - 7000 - Ssaltasaurus - Cretaceous -  ................L: (null)  ....R: [435]: 1887 - Coelophysidae - 435 - 20 - Coelophysis - Triassic -  ........L: (null)  +: [123]: 1941 - Mosasauridae - 123 - 2400 - Plotosaurus - Cretaceous -  ....L: (null)  {FREEING 5 MAPS}  aleksey@auv:~/Desktop/lab3/program/build$ ./a.out data.csv -b -n 600  INPUT: data.csv  N: 600  NO OUTPUT  COPY TO TREE: TRUE  PRINTING List  Id,Name,Period,Family,"Discovered, year","Weight, kg"  123,Plotosaurus,Cretaceous,Mosasauridae,1941,2400  435,Coelophysis,Triassic,Coelophysidae,1887,20  2654,Echinodon,Cretaceous,Heterodontosauridae,1861,7  876,Ssaltasaurus,Cretaceous,Saltasauridae,1977,7000  908,Diplodocus,Jurassic,"random dinosaur ""name""  with different options to check",1877,13000  keyis 123  keyis 435  keyis 2654  keyis 876  keyis 908  PRINTING TREE  ............R: (null)  ........R: [2654]: 1861 - Heterodontosauridae - 2654 - 7 - Echinodon - Cretaceous -  ................R: [908]: 1877 - "random dinosaur ""name""  with different options to check" - 908 - 13000 - Diplodocus - Jurassic -  ............L: [876]: 1977 - Saltasauridae - 876 - 7000 - Ssaltasaurus - Cretaceous -  ................L: (null)  ....R: [435]: 1887 - Coelophysidae - 435 - 20 - Coelophysis - Triassic -  ........L: (null)  +: [123]: 1941 - Mosasauridae - 123 - 2400 - Plotosaurus - Cretaceous -  ....L: (null)  PROCESSING...  PRINTING TREE  ........R: [908]: 1877 - "random dinosaur ""name""  with different options to check" - 908 - 13000 - Diplodocus - Jurassic -  ....R: [876]: 1977 - Saltasauridae - 876 - 7000 - Ssaltasaurus - Cretaceous -  ........L: (null)  +: [123]: 1941 - Mosasauridae - 123 - 2400 - Plotosaurus - Cretaceous -  ....L: (null)  {FREEING 5 MAPS}  aleksey@auv:~/Desktop/lab3/program/build$ ./a.out data.csv -b -n 600 -o output.csv  INPUT: data.csv  N: 600  OUTPUT: output.csv  COPY TO TREE: TRUE  PRINTING List  Id,Name,Period,Family,"Discovered, year","Weight, kg"  123,Plotosaurus,Cretaceous,Mosasauridae,1941,2400  435,Coelophysis,Triassic,Coelophysidae,1887,20  2654,Echinodon,Cretaceous,Heterodontosauridae,1861,7  876,Ssaltasaurus,Cretaceous,Saltasauridae,1977,7000  908,Diplodocus,Jurassic,"random dinosaur ""name""  with different options to check",1877,13000  keyis 123  keyis 435  keyis 2654  keyis 876  keyis 908  PRINTING TREE  ............R: (null)  ........R: [2654]: 1861 - Heterodontosauridae - 2654 - 7 - Echinodon - Cretaceous -  ................R: [908]: 1877 - "random dinosaur ""name""  with different options to check" - 908 - 13000 - Diplodocus - Jurassic -  ............L: [876]: 1977 - Saltasauridae - 876 - 7000 - Ssaltasaurus - Cretaceous -  ................L: (null)  ....R: [435]: 1887 - Coelophysidae - 435 - 20 - Coelophysis - Triassic -  ........L: (null)  +: [123]: 1941 - Mosasauridae - 123 - 2400 - Plotosaurus - Cretaceous -  ....L: (null)  PROCESSING...  PRINTING TREE  ........R: [908]: 1877 - "random dinosaur ""name""  with different options to check" - 908 - 13000 - Diplodocus - Jurassic -  ....R: [876]: 1977 - Saltasauridae - 876 - 7000 - Ssaltasaurus - Cretaceous -  ........L: (null)  +: [123]: 1941 - Mosasauridae - 123 - 2400 - Plotosaurus - Cretaceous -  ....L: (null)  PRINTING List  {FREEING 5 MAPS}  aleksey@auv:~/Desktop/lab3/program/build$ |

|  |
| --- |
| **output.csv** |
| 123,Plotosaurus,Cretaceous,Mosasauridae,1941,2400  435,Coelophysis,Triassic,Coelophysidae,1887,20  2654,Echinodon,Cretaceous,Heterodontosauridae,1861,7  876,Ssaltasaurus,Cretaceous,Saltasauridae,1977,7000  908,Diplodocus,Jurassic,"random dinosaur ""name""  with different options to check",1877,13000 |

|  |
| --- |
| **output2.csv** |
| 123,Plotosaurus,Cretaceous,Mosasauridae,1941,2400  876,Ssaltasaurus,Cretaceous,Saltasauridae,1977,7000  908,Diplodocus,Jurassic,"random dinosaur ""name""  with different options to check",1877,13000 |

**Висновки**

Завдання допомогло розібратися з роботою бінарних дерев та відображень, обходити та проводити операції з іх вмістом.

Також була проведена робота зі збіркою власних програм і статичних бібліотек з допомогою Cmake.