

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 “**Структури даних, функції, вказівники і файлові потоки**”

|  |  |  |
| --- | --- | --- |
| Виконав  студент I курсу  групи КП-62  Лук’янець Михайло Олександрович  (*прізвище, ім’я, по батькові*)  варіант №15 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2016 р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Штрафні бали:   |  |  | | --- | --- | | **Термін здачі** | **Оформлення звіту** | |  |  | | Нараховані бали:   |  |  |  | | --- | --- | --- | | **Корект. програм (2 бала)** | **Відп. на теор. питання (1 бал)** | **Відп. на прогр. питання (2 бала)** | |  |  |  | | Сумарний бал:   |  | | --- | |  | |

Київ 2016

**Мета роботи**

Навчитися працювати зі структурами, файлами, динамічно виділяти пам’ять, а також вдосконалити та закріпити вміння роботи з вказівниками.

**Постановка завдання**

#### **Загальні вимоги до завдання:**

Після входу у програму користувач має можливість зробити вибір:

* Створити новий масив даних
* Зчитати масив даних із файлу

Після цього масив даних можна модифікувати:

* Видалити дані із вказаної позиції у масиві
* Перезаписати дані у вказаній позиції на нововведені
* Перезаписати тільки обране поле даних елемента на вказаній позиції

Також дозволити користувачу виконувати спеціальні команди за варіантом.

Дати можливість користувачу зберегти зміни даних на файлову систему, запропонувавши йому ввести назву дискового файлу, у який збережуться дані.

#### **Завдання №1. Розробка структур і функцій**

1. Описати нову струкутуру даних типу "Динозавр". Визначити у структурі не менше 3-х полів різного типу, серед яких обов'язковими є: строка, ціле число, дробове число, інша структура.
2. Виконати декомпозицію завдання на окремі функції і описати їх прототипи у окремому header-файлі:
   1. Описати функції, одна з яких виконує запис даних із одної структури (через вказівник) у строку, та функцію, що виконує зворотнє перетворення.
   2. Описати функцію, що із тексту створює масив структур та зворотню функцію.
   3. Описати функцію, що видаляє структуру із масиву структур по індексу, функцію, що виконує перезапис структури по індексу даними із іншої структури.
   4. Описати функцію, що отримує вказівник на структуру, строку, що позначає поле у структурі та строку, що позначає нове значення вказаного поля у структурі і виконує перезапис поля заданої структури новим значенням.
   5. Описати функцію, що виконує дію над масивом вказівників на структури: "Знайти всіх динозаврів із середньою масою тіла меншою за Х кг".
3. Реалізувати відповідні функції у окремому файлі з кодом.

#### **Завдання №2. Робота із файловими потоками**

1. Реалізувати функцію для зчитування даних із текстового файлу і перетворення їх у масив структур.
2. Реалізувати функцію для запису масиву структур у текстовий файл.

#### **Завдання №3. Тестування функцій**

1. Протестувати функції, що піддаються тестуванню, за допомогою assert() у окремій функції(-ях). Кожна функція має тестуватись мінімум на 5 тестових випадках.
2. Винести прототипи тестових функцій у окремий header-файл.
3. Винести тестові функції у окремий файл із кодом.

#### **Завдання №4. Розробка CUI-інтерфейсу**

1. Створити CUI - консольний інтерфейс користувача (*просто взаємодія з користувачем через консоль*) для використання описаного функціоналу.
2. Всі прототипи відповідних функцій CUI описати у окремому header-файлі, а їх реалізацію винести у окремий .c файл із кодом.
3. Тестові функції запускати тільки при заданні певного аргумента командного рядка.
4. У файлі із кодом функції main() не повинно бути жодних інших функцій.
5. Весь ввід користувача потрбно перевіряти на валідність щоби не допускати збоїв у роботі програми.

**Тексти коду програм**

|  |
| --- |
| main.c |
| #include <progbase.h>  #include <string.h>  #include <ctype.h>  #include "interface.h"  #include "dino.h"  #include "totest.h"  #include "saves.h"  **int** main(**int** argc, **char** \* argv[]) {  **char** key = '\0';  **if**(argc > 1 && (!strcmp(argv[1], "-test"))){  test();  }  **const struct** conShift MAIN\_SHIFT = { 0, -10 + MAX\_SIZE};  **const char** \*START\_TEXT = "Hello User! Welcome to jurasic's park pre-alpha version 0.0.0.0.0.0.1\n" \  "You can choose what to do via pressing needed button. Press h for info\nPress q to quit program\n\0";  **const char** \*START\_INFO = "Press 1 to rewrite certain field\n"  "Press 2 to rewrite one dino\n"  "Press 3 to delete needed position\n"  "Press 4 to save data\n"  "Press 5 to copy data from one dino to another\n"  "Press 6 to find all dinos with mass less than X\n"  "Press 7 to add dino if possible\n"  "Press 8 to fill with some random data ALL fields(for tests only!)\n"  "\n"  "Press h to return\n\0";  redraw(MAIN\_SHIFT.cols , MAIN\_SHIFT.rows );  **int** flagMode = start("Press 1 for new array. Press 2 for existing array");  **char** readFileName[100] = "";  **if**(flagMode == 2){  **char** \* readFileTemp = getStringInter("read file name");  strcpy( readFileName , readFileTemp);  free(readFileTemp);  }  **struct** dino \* arr = readFile(readFileName);  cleanCanvas();  printText(START\_TEXT);  printStructArr(arr);  **while**(key != 'q'){  key = conGetChar();  **switch** (key) {  **case** '1':  ;  **char** \* number = getStringInter("number of a dino");  **char** \* toChange = getStringInter("new data");  **char** \* field = getStringInter("field to rewrite");  **if**(atoi(number) >= MAX\_SIZE || !isdigit(number[0])){  free(number);  free(toChange);  free(field);  **continue**;  }  changeField(&arr[atoi(number)], field, toChange, arr);  cleanCanvas();  printText(START\_TEXT);  printStructArr(arr);  printResult("field was changed");  free(number);  free(toChange);  free(field);  **break**;  **case** '2':  ;  **char** \* numberOfDino = getStringInter("number of a dino");  **char** \* name = getStringInter("name");  **char** \* age = getStringInter("age");  **char** \* friend = getStringInter("number of friend");  **char** \* mass = getStringInter("mass");  **if**(atoi(numberOfDino) >= MAX\_SIZE || !isdigit(numberOfDino[0])){  printResult("Not correct number");  free(numberOfDino);  free(name);  free(age);  free(friend);  free(mass);  **continue**;  }  changeField(&arr[atoi(numberOfDino)], "name", name, arr);  changeField(&arr[atoi(numberOfDino)], "age", age, arr);  changeField(&arr[atoi(numberOfDino)], "friend", friend, arr);  changeField(&arr[atoi(numberOfDino)], "mass", mass, arr);  cleanCanvas();  printText(START\_TEXT);  printStructArr(arr);  printResult("Data was changed");  free(numberOfDino);  free(name);  free(age);  free(friend);  free(mass);  **break**;  **case** '3':  ;  **char** \* numberToDelete = getStringInter("number of a dino to delete");  **if**(!isdigit(numberToDelete[0])){  printResult("Not correct number");  free(numberToDelete);  **continue**;  }  delete(arr, atoi(numberToDelete));  free(numberToDelete);  cleanCanvas();  printText(START\_TEXT);  printStructArr(arr);  printResult("Dino was deleted. Or not");  **break**;  **case** '4':  ;  **char** \* saveFileName = getStringInter("save file name");  **if**(!saveFile(arr, saveFileName)){  printResult("Data saved");  } **else** {  printResult("Save was absolutely sabotaged");  }  free(saveFileName);  **break**;  **case** '5':  ;  **char** \* first = getStringInter("dino to rewrite");  **char** \* second = getStringInter("dino to copy");  **if**(!isdigit(first[0]) || !isdigit(second[0]) ||  atoi(first) >= MAX\_SIZE || atoi(second) >= MAX\_SIZE){  printResult("Not correct numbers");  free(first);  free(second);  **continue**;  }  rewrite(arr, atoi(first), atoi(second));  cleanCanvas();  printText(START\_TEXT);  printStructArr(arr);  printResult("Data copied");  free(first);  free(second);  **break**;  **case** '6':  ;  **char** \* massLower = getStringInter("mass");  **char** \* result = findByMass(arr, atof(massLower));  cleanCanvas();  printText(START\_TEXT);  printStructArr(arr);  **if**(strlen(result) == 0){  printResult("Not found");  } **else** {  printResult(result);  }  free(result);  free(massLower);  **break**;  **case** '7':  ;  **char** \* data = getStringInter("data in format name age mass number of friend");  cleanCanvas();  **if**(!writeFromStr(arr, data)){  printResult("Added successfully");  } **else** {  printResult("Added not successfully");  }  printText(START\_TEXT);  printStructArr(arr);  free(data);  **break**;  **case** '8':  fillRandArr(arr);  cleanCanvas();  printText(START\_TEXT);  printStructArr(arr);  printResult("Dinos were randomized");  **break**;  **case** 'h':  cleanCanvas();  printInfoText(START\_INFO);  cleanCanvas();  printText(START\_TEXT);  printStructArr(arr);  **break**;  **default**: ;  }  }  free(arr);  exitInter();  **return** 0;  } |

|  |
| --- |
| dino.h |
| //  // Created by tgifr on 03.12.16.  //  #include <pbconsole.h>  #include <progbase.h>  #include <string.h>  #include <time.h>  #ifndef LAB3\_DYNO\_H  #define LAB3\_DYNO\_H  #define MAX\_SIZE 10  #define NAMES\_SIZE 10  **const char** \* DIN0\_NAMES[10] ;  **struct** dino {  **char** name[100];  **int** age;  **double** mass;  **struct** dino \* friend;  **int** number;  };  **int** writeFromStr(**struct** dino \*arr, **char** \*str);  **char** \* writeInStr(**struct** dino \* dyno);  **struct** dino \* writeFromText(**char** \* text);  **char** \*writeInText(**struct** dino \*arr);  **void** delete(**struct** dino \* arr, **int** toDelete);  **void** rewrite(**struct** dino \* arr, **int** toRewrite, **int** toCopy);  **void** changeField(**struct** dino \*dyno, **char** \*field, **char** \*toChange, **struct** dino \*arr);  **char**\* findByMass(**struct** dino \*arr, **double** mass);  **void** initializeArr(**struct** dino \* arr);  **void** fillRandArr(**struct** dino \* arr);  #endif //LAB3\_DYNO\_H |

|  |
| --- |
| dino.c |
| //  // Created by tgifr on 03.12.16.  //  #include <ctype.h>  #include "dino.h"  **const char** \* DIN0\_NAMES[10] = { "Kvadrat", "Jurasic", "Star", "Katzchen", "Jet",  "Half", "life", "3", "Night\_Fury", "Moonshine"};  **int** writeFromStr(**struct** dino \*arr, **char** \*str){  **int** i = 0;  **char** test[100] = "";  strcpy(test, str);  **char** \* temp = strtok(test, " ");  **while**(strcmp(arr[i].name, "") || arr[i].age != 0 || arr[i].mass != 0.0){  i++;  **if**(i == MAX\_SIZE){  **return** EXIT\_FAILURE;  }  }  **if**(temp != NULL && isalpha(temp[0])){  strcpy(arr[i].name, temp);  temp = strtok(NULL, " ");  }  **if**(temp != NULL && isdigit(temp[0])){  arr[i].age = atoi(temp);  temp = strtok(NULL, " ");  }  **if**(temp != NULL && isdigit(temp[0])){  arr[i].mass = atof(temp);  temp = strtok(NULL, " ");  }  **if**(temp != NULL && atoi(temp) < MAX\_SIZE && atoi(temp) > -1 && isdigit(temp[0])){  arr[i].friend = &arr[atoi(temp)];  }  **return** EXIT\_SUCCESS;  }  **char** \* writeInStr(**struct** dino \* dyno){  **char** \*result = malloc(200 \* **sizeof**(**char**));  **if**(result == NULL){  **return** 0;  }  sprintf(result, "%s %i %.3f %i\n", dyno->name, dyno->age, dyno->mass, dyno->friend->number);  **return** result;  }  **struct** dino \* writeFromText(**char** \* text){  **char** temp[200] = "";  **struct** dino \* arr = malloc(MAX\_SIZE \* **sizeof**(**struct** dino));  **if**(arr == NULL){  **return** NULL;  }  initializeArr(arr);  **char** toCopy[2] = "";  toCopy[1] = '\0';  **for** (**int** i = 0; text[i] != '\0'; ++i) {  toCopy[0] = text[i];  strcat(temp, toCopy);  **if**(text[i] == '\n'){  temp[i] = '\0';  **if** (!writeFromStr(arr, temp)) {  **break**;  }  strcpy(temp, "");  }  }  **return** arr;  }  **char** \*writeInText(**struct** dino \*arr){  **char** \* temp;  **char** result[1000] = "";  **for** (**int** i = 0; i < MAX\_SIZE; ++i) {  temp = writeInStr(&arr[i]);  strcat(result, temp);  free(temp);  }  temp = malloc((strlen(result) + 5 ) \* **sizeof**(**char**));  **if**(temp == NULL){  **return** 0;  }  strcpy(temp, result);  **return** temp;  }  **void** delete(**struct** dino \* arr, **int** toDelete){  **if**(toDelete >= MAX\_SIZE || toDelete < 0){  **return**;  }  strcpy(arr[toDelete].name, "");  arr[toDelete].age = 0;  arr[toDelete].mass = 0.0;  arr[toDelete].friend = &arr[0];  }  **void** rewrite(**struct** dino \* arr, **int** toRewrite, **int** toCopy){  **if**(toRewrite >= MAX\_SIZE || toCopy >= MAX\_SIZE || toCopy < 0 || toRewrite < 0){  **return**;  }  strcpy(arr[toRewrite].name, arr[toCopy].name);  arr[toRewrite].age = arr[toCopy].age;  arr[toRewrite].mass = arr[toCopy].mass;  arr[toRewrite].friend = arr[toCopy].friend;  }  **void** changeField(**struct** dino \*dyno, **char** \*field, **char** \*toChange, **struct** dino \*arr) {  **if**(!strcmp("name", field)){  strcpy(dyno->name, toChange);  } **else if**(!strcmp("age", field)){  dyno->age = atoi(toChange);  } **else if**(!strcmp("mass", field)){  dyno->mass = atof(toChange);  } **else if**(!strcmp("friend", field) && atoi(toChange) < MAX\_SIZE){  dyno->friend = &arr[atoi(toChange)];  }  }  **char**\* findByMass(**struct** dino \*arr, **double** mass){  **char** temp[300] = "";  **for** (**int** i = 0; i < MAX\_SIZE; ++i) {  **if**(arr[i].mass < mass){  strcat(temp, arr[i].name);  strcat(temp, " ");  }  }  **char** \* result = malloc((strlen(temp) + 5)\* **sizeof**(**char**));  **if**(result == NULL){  **return** "";  }  strcpy(result, temp);  **return** result;  }  **void** initializeArr(**struct** dino \* arr){  **for**(**int** i = 0; i < MAX\_SIZE; i ++){  strcpy(arr[i].name ,"");  arr[i].number = i;  arr[i].friend = &arr[0];  arr[i].age = 0;  arr[i].mass = 0.0;  }  }  **void** fillRandArr(**struct** dino \* arr){  srand(time(0));  **for**(**int** i = 0; i < MAX\_SIZE; i ++){  strcpy(arr[i].name, DIN0\_NAMES[rand() % 10]);  arr[i].friend = &arr[ rand() % NAMES\_SIZE];  arr[i].age = (rand() % 1000) + 1;  arr[i].mass = 50.0 / ((rand() % 50) + 1) ;  }  } |

|  |
| --- |
| saves.h |
| //  // Created by tgifr on 03.12.16.  //  #include "dino.h"  #ifndef LAB3\_SAVES\_H  #define LAB3\_SAVES\_H  **struct** dino \* readFile(**char** \* readFileName);  **int** saveFile(**struct** dino \* arr, **char** \* saveFileName);  #endif //LAB3\_SAVES\_H |

|  |
| --- |
| saves.c |
| **//**  **// Created by tgifr on 03.12.16.**  **//**  **#include <stdio.h>**  **#include "dino.h"**  **struct dino \* readFile(char \* readFileName){**  **FILE \* read;**  **char temp[150] = "";**  **struct dino \* arr = malloc(MAX\_SIZE \* sizeof(struct dino));**  **initializeArr(arr);**  **if((read = fopen(readFileName, "r")) == NULL){**  **return arr;**  **}**  **while(!feof(read)){**  **fgets(temp, 150, read);**  **if(writeFromStr(arr, temp)){**  **break;**  **}**  **}**  **fclose(read);**  **return arr;**  **}**  **int saveFile(struct dino \* arr, char \* saveFileName){**  **FILE \* save;**  **if((save = fopen(saveFileName, "w")) == NULL){**  **return EXIT\_FAILURE;**  **}**  **char \* toWrite = writeInText(arr);**  **fprintf(save, "%s", toWrite);**  **free(toWrite);**  **fclose(save);**  **return EXIT\_SUCCESS;**  **}** |

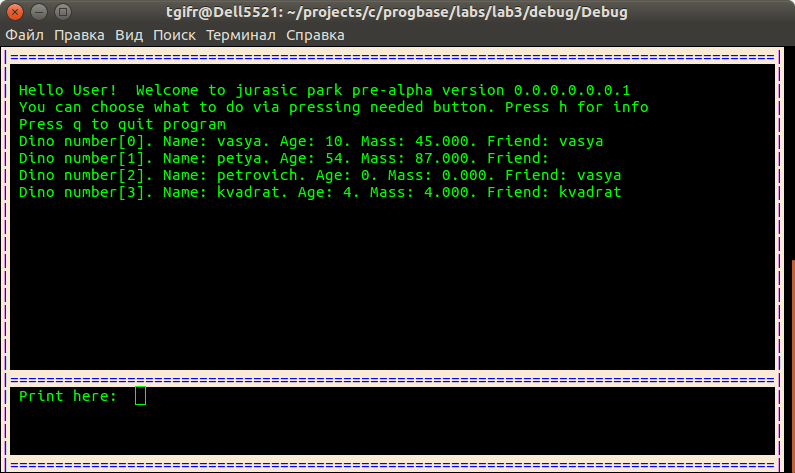
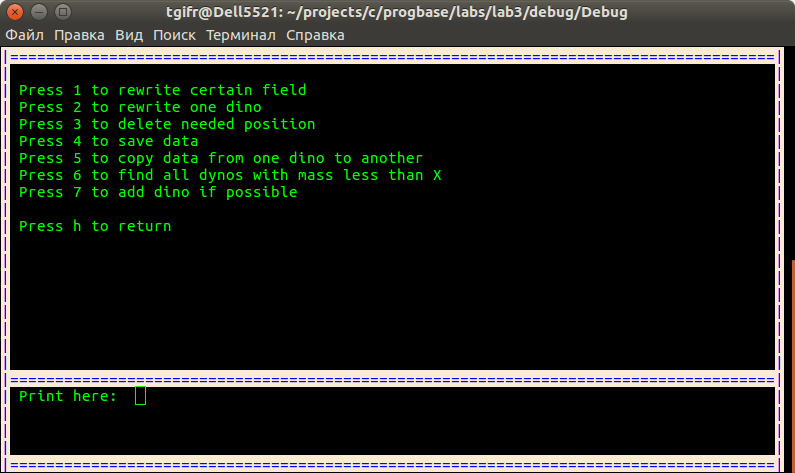
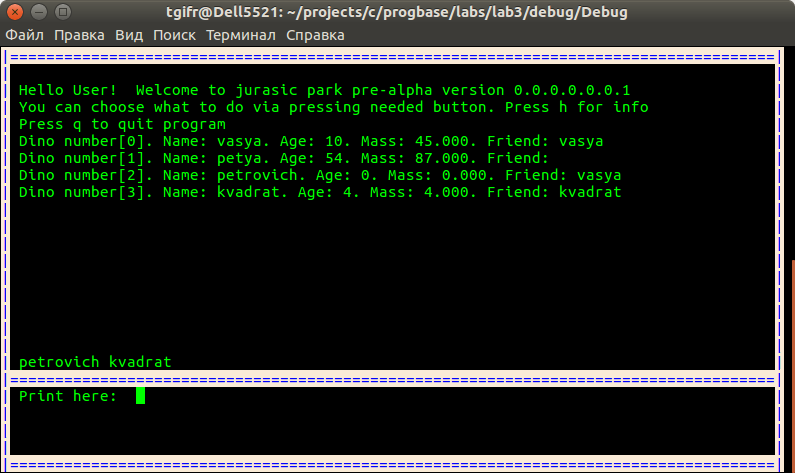
|  |
| --- |
| **interface.h** |
| **//**  **// Created by tgifr on 03.12.16.**  **//**  **#include <pbconsole.h>**  **#include <progbase.h>**  **#include <string.h>**  **#include "dino.h"**  **#ifndef INTERFACE\_H**  **#define INTERFACE\_H**  **void redraw(int cols, int rows);**  **void clearAnim();**  **void printText(const char \*TEXT);**  **void printInfoText(const char \*HELP);**  **struct conShift{**  **short cols;**  **short rows;**  **};**  **void conDefPos();**  **void cleanInp();**  **void cleanCanvas();**  **char \* getStringInter(char \* whatGet);**  **void exitInter();**  **void printStructArr(struct dino (\* arr));**  **int start(const char \*START\_TEXT);**  **void printResult(char \* result);**  **#endif //LAB3\_INTERFACE\_H** |

|  |
| --- |
| **interface.c** |
| **//**  **// Created by tgifr on 03.12.16.**  **//**  **#include "interface.h"**  **unsigned long int MILLIS\_WAIT = 30;**  **const struct consize DEF\_SIZE = {87, 25};**  **const char \* EXIT\_TEXT = " \_\_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \n"**  **" / \_\_\_\_|/ \_\_ \\ / \_\_ \\| \_\_ \\| \_ \\ \\ / / \_\_\_\_|\n"**  **" | | \_\_| | | | | | | | | | |\_) \\ \\\_/ /| |\_\_ \n"**  **" | | |\_ | | | | | | | | | | \_ < \\ / | \_\_| \n"**  **" | |\_\_| | |\_\_| | |\_\_| | |\_\_| | |\_) | | | | |\_\_\_\_ \n"**  **" \\\_\_\_\_\_|\\\_\_\_\_/ \\\_\_\_\_/|\_\_\_\_\_/|\_\_\_\_/ |\_| |\_\_\_\_\_\_|\n";**  **void redraw(int cols, int rows){**  **int i = 0;**  **conResize(DEF\_SIZE.rows + (unsigned short)rows, DEF\_SIZE.cols + (unsigned short)cols);**  **conClear();**  **const struct consize actSize = { DEF\_SIZE.cols + (unsigned short)cols, DEF\_SIZE.rows + (unsigned short)rows};**  **conSetAttr(*BG\_WHITE*);**  **conSetAttr(*FG\_INTENSITY\_BLUE*);**  **for(i = 1; i <= actSize.cols; i ++){**  **conMove(1, i );**  **printf("=");**  **conMove(actSize.rows - 5, i);**  **printf("=");**  **conMove(actSize.rows, i );**  **printf("=");**  **fflush(stdout);**  **sleepMillis(MILLIS\_WAIT/3);**  **}**  **for(i = 1; i <= actSize.rows; i ++){**  **conMove(i , 1);**  **printf("|");**  **conMove(i , actSize.cols);**  **printf("|");**  **fflush(stdout);**  **sleepMillis(MILLIS\_WAIT);**  **}**  **conReset();**  **conMove( actSize.rows - 4, 3);**  **printf("Print here: ");**  **}**  **void clearAnim(){**  **int i = 0;**  **int j = 0;**  **const struct consize size = conGetSize();**  **for(i = 1; i <= size.cols; i ++){**  **conSetAttr(*BG\_INTENSITY\_BLACK*);**  **for(j = 1; j <= size.rows; j ++){**  **conMove(j, i);**  **printf(" ");**  **conMove(j, i + 1);**  **printf(" ");**  **}**  **conSetAttr(*BG\_DEFAULT*);**  **if (i != 1) {**  **for(j = 1; j <= size.rows; j ++){**  **conMove(j, i - 1);**  **printf(" ");**  **}**  **}**  **if (i == size.cols) {**  **for(j = 1; j <= size.rows; j ++){**  **conMove(j, i);**  **printf(" ");**  **}**  **}**  **fflush(stdout);**  **sleepMillis(MILLIS\_WAIT);**  **}**  **conReset();**  **conDefPos();**  **}**  **void printText(const char \*TEXT){**  **int i = 0, yShift = 0;**  **conMove(3, 3);**  **for(i = 0; i < strlen(TEXT); i ++){**  **if(TEXT[i] == '\n'){**  **yShift++;**  **conMove(3 + yShift, 3);**  **} else printf("%c", TEXT[i]);**  **}**  **conDefPos();**  **}**  **void printInfoText(const char \*HELP){**  **char key = '\0';**  **cleanCanvas();**  **printText(HELP);**  **while (key != 'h'){**  **key = conGetChar();**  **}**  **cleanInp();**  **}**  **void conDefPos(){**  **struct consize actSize = conGetSize();**  **conMove(actSize.rows - 4, 3 + sizeof("Print here: "));**  **}**  **void cleanInp(){**  **int i = 0;**  **struct consize actSize = conGetSize();**  **for(i = sizeof("Print here: ") + 3; i < actSize.cols - 2; i++){**  **conMove(actSize.rows - 4, i);**  **printf(" ");**  **conMove(actSize.rows - 3, i - sizeof("Print here: "));**  **printf(" ");**  **}**  **conMove(actSize.rows - 6, 3 );**  **printf(" ");**  **conDefPos();**  **}**  **void cleanCanvas(){**  **int i = 0;**  **int j = 0;**  **conReset();**  **struct consize actSize = conGetSize();**  **for(j = 2; j < actSize.rows - 5; j ++){**  **for(i = 2; i < actSize.cols; i++){**  **conMove(j, i);**  **printf(" ");**  **}**  **}**  **conDefPos();**  **}**  **char \* getStringInter(char \* whatGet){**  **struct consize actSize = conGetSize();**  **conMove(actSize.rows - 6, 3 );**  **printf("Please, enter %s:", whatGet);**  **conDefPos();**  **char \* result = getString();**  **cleanInp();**  **return result;**  **}**  **void exitInter(){**  **clearAnim();**  **printText(EXIT\_TEXT);**  **sleepMillis(MILLIS\_WAIT \* 100);**  **conClear();**  **}**  **void printStructArr(struct dino \*arr){**  **for(int i = 0; i < MAX\_SIZE ; i ++){**  **conMove(6 + i, 3);**  **if(strcmp(arr[i].name, "") || arr[i].age != 0 || arr[i].mass != 0.0) {**  **printf("Dino number[%i]. Name: %s. Age: %i. Mass: %.3f. Friend: %s", i, arr[i].name, arr[i].age,**  **arr[i].mass, arr[i].friend->name);**  **}**  **}**  **conDefPos();**  **}**  **int start(const char \*START\_TEXT){**  **conMove(3,3);**  **char key = '\0';**  **printf("%s", START\_TEXT);**  **while(key != '1' && key != '2'){**  **key = conGetChar();**  **}**  **return key - '0';**  **}**  **void printResult(char \* result){**  **struct consize actSize = conGetSize();**  **conMove(actSize.rows - 6 , 3);**  **printf("%s", result);**  **conDefPos();**  **}** |

|  |
| --- |
| **totest.h** |
| **//**  **// Created by tgifr on 03.12.16.**  **//**  **#include <assert.h>**  **#ifndef LAB3\_TOTEST\_H**  **#define LAB3\_TOTEST\_H**  **void test();**  **#endif //LAB3\_TOTEST\_H** |

|  |
| --- |
| **totest.c** |
| **//**  **// Created by tgifr on 03.12.16.**  **//**  **#include "totest.h"**  **#include "dino.h"**  **#include "saves.h"**  **void test(){**  **struct dino \* arr = readFile("blah");**  **assert(!writeFromStr(arr, "blah"));**  **assert(!writeFromStr(arr, "blah 2 5 8 "));**  **assert(!writeFromStr(arr, "blah 5 45 45 "));**  **assert(!writeFromStr(arr, ""));**  **for (int i = 0; i < MAX\_SIZE; ++i) {**  **arr[i].age = 1;**  **}**  **assert(writeFromStr(arr, "blah"));**  **struct dino din = {"vasya", 10, 10, &din, 0};**  **struct dino din2 = {"vasya2", 150, 150, &din, 4};**  **assert(!strcmp(writeInStr(&din2), "vasya2 150 150.000 0\n"));**  **assert(strcmp(writeInStr(&din2), "vasygdshgkdsfghlkdhgsaa2 150 150.000 0\n"));**  **assert(!strcmp(writeInStr(&din), "vasya 10 10.000 0\n"));**  **assert(strcmp(writeInStr(&din), "vasya2 150 150.000 0\n"));**  **din.friend = &din2;**  **assert(!strcmp(writeInStr(&din), "vasya 10 10.000 4\n"));**  **assert(writeFromText("vasya 10 10.000 4\n")[0].age == 10);**  **assert(writeFromText("vasya 10 10.000 4\n")[2].age == 0);**  **assert(writeFromText("vasya 10 10.000 4\n")[0].mass == 10.000);**  **assert(writeFromText("vasya 10 10.000 4\n")[9].number == 9);**  **assert(writeFromText("vasya 1550 10.000 4\n")[0].age == 1550);**  **delete(arr, 1);**  **delete(arr, 1);**  **delete(arr, 2);**  **delete(arr, 3);**  **delete(arr, 6);**  **assert(arr[1].age == 0);**  **assert(arr[1].age == 0);**  **assert(arr[2].age != 1);**  **assert(arr[3].mass != 1);**  **assert(arr[6].age == 0);**  **assert(din.age == 10);**  **changeField(&din, "age", "20", arr);**  **assert(din.age == 20);**  **changeField(&din, "name", "vas", arr);**  **assert(!strcmp(din.name, "vas"));**  **changeField(&din, "mass", "30.45", arr);**  **assert(din.mass == 30.45);**  **changeField(&din, "friend", "5", arr);**  **assert(din.friend == &arr[5]);**  **changeField(&din, "age", "20", arr);**  **assert(din.age == 20);**  **free(arr);**  **}** |

**Приклади результатів**



**Висновки**

Виконавши дану лабораторну роботу було використано можливості мови С з викорситання динамічної пам’яті, структур данних, вказівників та файлових потоків.

Була додана можливість зберігання результатів роботи і їх завантаження при запуску.

Було проведено тестування деяких основних функцій програми.

Дотадково був створений користувацький консольний інтерфейс.