НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«Програмування ч.2»

*Звіт з лабораторної роботи №22*

*Тема: «Регулярні вирази»*

Виконав:

ст. гр. КІТ-120А

Старовойтов Н.А.

Перевірив:

Челак В.В.

Харків – 2021

***Мета:*** Отримати навички роботи із регулярними виразами.

***Індивідуальне завдання***

Робота на оцінку “відмінно”.

Поширити попередню лабораторну роботу таким чином:

1. При введенні інформації про “базовий тип” (нема різниці, чи з клавіатури, чи з файлу), організувати перевірку відповідності таким критеріям з використанням регулярних виразів:
   * Можна вводити тільки кириличні символи, латинські символи, цифри, пропуски, розділові знаки;
   * Не повинно бути пропусків та розділових знаків, які повторюються;
   * Перше слово не повинно починатися з маленького символу;
2. У клас-список додати метод, що виводить на екран список усіх об’єктів, які мають одне або більше полів з щонайменше двома словами (перевірку організувати за допомогою регулярних виразів).

**Опис програми**

Функція Regex\_Check проводить перевірку кожного елементу структури на відповідність заданому регулярному виразу. Приймає структуру, повертає результат перевірки. Текст функції:

bool Regex\_Check(struct Bird \*bird){

bool result = true;

regex\_t regex;

regcomp(&regex, "^[0-1]$", 0);

char boool[2];

sprintf(boool, "%d", bird->LOTR);

if (regexec(&regex, boool, 0, NULL, 0) != 0) result = false;

regcomp(&regex, "^ ?[A-ZА-Я]([A-ZА-Яa-zа-я]+ ?)+$", REG\_EXTENDED);

if (regexec(&regex, bird->name, 0, NULL, 0) != 0) result = false;

char num[5];

sprintf(num, "%d", bird->age);

regcomp(&regex, "^[0-9]{1,3}$", REG\_EXTENDED);

if (regexec(&regex, num, 0, NULL, 0) != 0) result = false;

sprintf(num, "%d", bird->home.space);

regcomp(&regex, "^[0-9]{1,4}$", REG\_EXTENDED);

if (regexec(&regex, num, 0, NULL, 0) != 0) result = false;

sprintf(num, "%d", bird->home.height);

regcomp(&regex, "^[0-9]{1,4}$", REG\_EXTENDED);

if (regexec(&regex, num, 0, NULL, 0) != 0) result = false;

sprintf(boool, "%d", bird->home.count\_of\_feeders);

regcomp(&regex, "^[0-9]$", REG\_EXTENDED);

if (regexec(&regex, boool, 0, NULL, 0) != 0) result = false;

regcomp(&regex, "^[0-1]$", 0);

sprintf(boool, "%d", bird->home.if\_nest);

if (regexec(&regex, boool, 0, NULL, 0) != 0) result = false;

regcomp(&regex, "^[0-1]$", 0);

sprintf(boool, "%d", bird->enumSex);

if (regexec(&regex, boool, 0, NULL, 0) != 0) result = false;

return result;

}

Функція Print\_Regex\_Two\_Words проводить перевірку елементу структури «name» (тобто імені/назви птаха) на відповідність заданому регулярному виразу, а саме: назва повинна складатись щонайменше з двох слів. Приймає структуру, якщо результат перевірки позитивний, виводить вміст структури на екран. Текст функції:

void Print\_Regex\_Two\_Words(struct List \* list) {

regex\_t regex;

for (struct Bird \*bird = list->head; bird != NULL; bird = bird->next) {

regcomp(&regex, "^ ?[A-ZА-Яa-zа-я]+ [A-ZА-Яa-zа-я]+", REG\_EXTENDED);

if (regexec(&regex, bird->name, 0, NULL, 0) == 0) {

Print\_bird(bird);

}

}

}

**Схеми алгоритмів функцій**

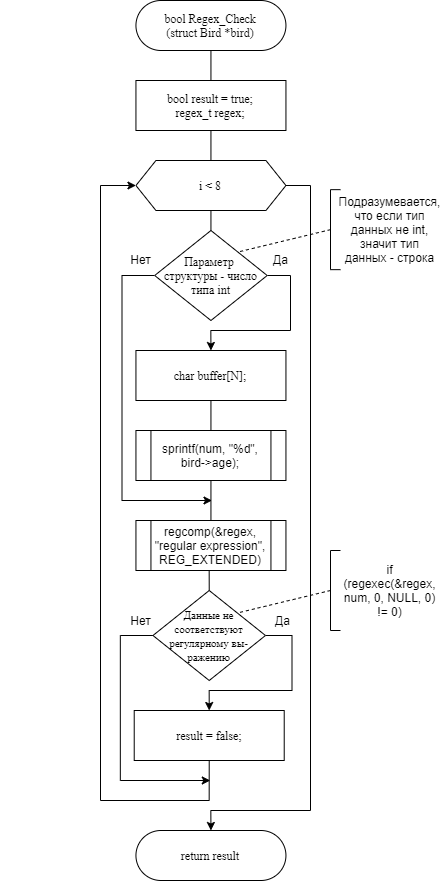


Рисунок 1 — Блок-схема функції Regex\_Check

**Текст програми**

# #include "../Library/library.h"

#define DEBUG

int main() {

system("chcp 65001");

#ifdef DEBUG

Print\_Date();

#endif

printf("\n");

struct List \*list = New\_list();

struct Bird struct\_paste\_bird = {0, "SOME\_BIRD", 27, 250, 40, 2, 0, 1};

struct Bird \*paste\_bird = malloc(sizeof(struct Bird));

\*paste\_bird = struct\_paste\_bird;

Read\_list\_from\_file(list);

int start1, finish1;

while (true) {

int action = 0;

printf("\n|--------------------------------------------------------------------|\n");

printf(" Выберите действие: \n");

printf("\t1. Вывести список на экран\n");

printf("\t2. Вывести список в файл\n");

printf("\t3. Вставить новый элемент в список\n");

printf("\t4. Удалить элемент из списка\n");

printf("\t5. Найти в списке элемент\n");

printf("\t6. Сортировать список по возрасту птицы\n");

printf("\t7. Вывести на экран птиц с именем, содержащим два или больше слова\n");

printf("\t8. Выход\n");

printf("|--------------------------------------------------------------------|\n\n");

printf(" --> ");

scanf("%d", &action);

switch (action) {

case 1:

#ifdef DEBUG

printf("\nВыполняется функция Show\_list");

printf("\nСписок:\n\n");

start1 = clock();

Show\_list(list);

finish1 = clock();

finish1 -= start1;

printf("Время выполнения функции: %dms%s", finish1, "\n");

#else

printf("\nСписок:\n\n");

Show\_list(list);

#endif

break;

case 2:

#ifdef DEBUG

printf("\nВыполняется функция Print\_list\_in\_file");

start1 = clock();

Print\_list\_in\_file(list);

finish1 = clock();

finish1 -= start1;

printf("\n Список занесен в файл 'result.txt'\n");

printf("Время выполнения функции: %dms%s", finish1, "\n\n");

#else

Print\_list\_in\_file(list);

printf("\n Список занесен в файл 'result.txt'\n\n");

#endif

break;

case 3:

printf("\nЭлемент, который будет вставлен: \n\n");

Print\_bird(paste\_bird);

printf("\nВведите позицию, на которую нужно вставить элемент: \n");

printf(" --> ");

int pos;

scanf("%d", &pos);

#ifdef DEBUG

printf("\nВыполняется функция Insert\_element");

start1 = clock();

Insert\_element(list, pos, paste\_bird);

finish1 = clock();

finish1 -= start1;

printf("\nЭлемент вставлен\n");

printf("Время выполнения функции: %dms%s", finish1, "\n\n\n");

#else

Insert\_element(list, pos, paste\_bird);

printf("\nЭлемент вставлен\n\n\n");

#endif

break;

case 4:

printf("\nВведите позицию элемента, который будет удалён: \n");

printf(" --> ");

int num;

scanf("%d", &num);

#ifdef DEBUG

printf("\nВыполняется функция Delete\_element");

start1 = clock();

Delete\_element(list, num);

finish1 = clock();

finish1 -= start1;

printf("\nЭлемент удалён\n");

printf("Время выполнения функции: %dms%s", finish1, "\n\n\n");

#else

Delete\_element(list, num);

printf("\nЭлемент удалён\n\n\n");

#endif

break;

case 5:

printf("\nВведите критерий, по которому искать птицу: \n");

printf("\t1. С кольцом или без\n");

printf("\t2. Имя\n");

printf("\t3. Возраст\n");

printf("\t4. Площадь скворечника\n");

printf("\t5. Высота скворечника\n");

printf("\t6. Кол-во кормушек скворечника\n");

printf("\t7. Является ли скворечник гнездом\n");

printf("\t8. Пол птицы\n");

printf(" --> ");

int number;

scanf("%d", &number);

printf("\n");

#ifdef DEBUG

printf("\nВыполняется функция Find\_bird");

start1 = clock();

Find\_bird(list, number);

finish1 = clock();

finish1 -= start1;

printf("Время выполнения функции: %dms%s", finish1, "\n\n\n");

#else

Find\_bird(list, number);

#endif

break;

case 6:

#ifdef DEBUG

printf("\nВыполняется функция Sort\_birds\_by\_age");

start1 = clock();

Sort\_birds\_by\_age(list);

finish1 = clock();

finish1 -= start1;

printf("\nСписок отсортирован \n");

printf("Время выполнения функции: %dms%s", finish1, "\n\n\n");

#else

Sort\_birds\_by\_age(list);

printf("\nСписок отсортирован \n");

#endif

break;

case 7:

#ifdef DEBUG

printf("\nВыполняется функция Print\_Regex\_Two\_Words\n\n");

start1 = clock();

Print\_Regex\_Two\_Words(list);

finish1 = clock();

finish1 -= start1;

printf("Время выполнения функции: %dms%s", finish1, "\n\n\n");

#else

Print\_Regex\_Two\_Words(list);

#endif

break;

case 8:

#ifdef DEBUG

printf("\nВыполняется функция Free\_list");

start1 = clock();

Free\_list(list);

finish1 = clock();

finish1 -= start1;

free(list);

printf("\nВремя выполнения функции: %dms%s", finish1, "\n");

printf("\nЗавершение работы программы...\n");

return 0;

#else

Free\_list(list);

free(list);

printf("\nЗавершение работы программы...\n");

return 0;

#endif

default:

printf("\nERROR: Вы ввели некорректный вариант\n");

break;

}

}

}

**Результати роботи програми**

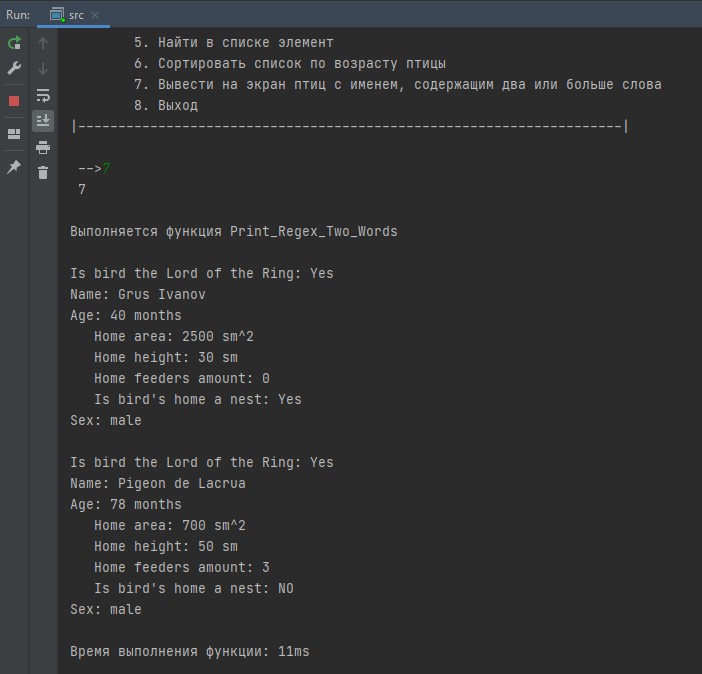


Рисунок 2 — Результат успішного виконання програми

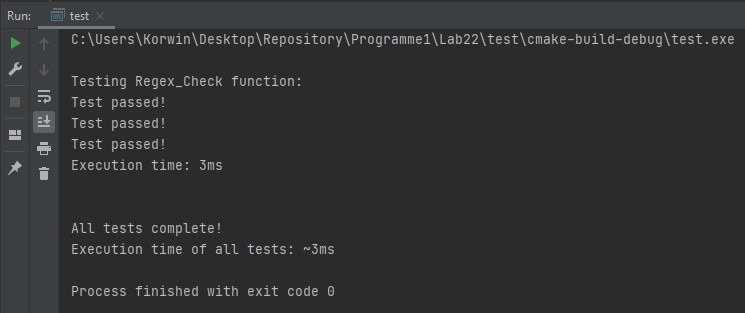


Рисунок 3 — Результат успішного виконання тестів

**Висновки**

Під час виконання даної лабораторної роботи було отримано навички роботи із регулярними виразами.