Лабораторна робота №7

Тема:Функції

Виконав:

Стоєнко Сергій Максимович

Група виконавця: КН-922Б

Завдання:

1. Переробити програми, що були розроблені під час виконання лабораторних робіт з тем "Масиви" та "Цикли" таким чином, щоб використовувалися функції для обчислення результату.
   1. Параметри одного з викликів функції повинні бути згенеровані за допомогою генератора псевдовипадкових чисел random().
2. Продемонструвати встановлення вхідних даних через аргументи додатка (параметри командної строки). Обробити випадок, коли дані не передались - у цьому випадку вони матимуть значення за умовчуванням, обраними розробником.

Основна частина:

- Опис роботи основної функції:

1. Записуємо окремо від функції main ще 2 функції: find\_nsd яка приймає значення 2 чисел та повертає значення НСД та multiply\_matrix, яка приймає матрицю, яку будемо множити та матрицю, в яку будемо вписувати значення помноженної матриці. Функція нічого не повертає, лише проводить вписує значення помноженної матриці. Більш детальний розбір цих 2 функцій був у попередніх звітах.
   1. Для згенерованих за допомогою генератора псевдовипадкових чисел random() параметрів функції я вибрав функцію find\_nsd та згенерував рандомно 2 числа в яких можна знайти спільний дільник.

- Перелік вхідних даних:

|  |  |
| --- | --- |
| argc[in] | - Записана з командного рядка кількість рядків(стовпців) в матриці |
| argv[in] | - Записана з командного рядка матриця, яку ми хочемо помножити |

|  |  |
| --- | --- |
| first\_number[in] | - Перше випадкове число |
| second\_number[in] | - Друге випадкове число |

|  |  |
| --- | --- |
| argc[in] | - Кількість рядків(стовпців) в матриці |
| argv[in] | - Матриця, яку ми хочемо помножити |
| multiply | - порожня матриця |

- Дослідження результатів роботи програми

-NSD – Найбільший спільний дільник, так як всі оперуючі числа є цілими, так і результуюча змінна є цілою. Тому, її тип – int.

- multiply[] - матриця добутку, так як всі оперуючі числа матриці, що множеться є цілими, так і результативна матриця є цілою. Тому, її тип – int.

- для підтвердження коректності роботи програми find\_nsd з рандомними числами, зупинено відлагодник на строчці „return 0“

(lldb) file dist/main.bin

Current executable set to '/home/serhii/tnp/programing-Stoienko/lab07/dist/main.bin' (x86\_64).

(lldb) b 24

Breakpoint 1: where = main.bin`main + 161 at main.c:24:2, address = 0x0000000000001221

(lldb) run

(lldb) print first\_number

(int) $4 = 44

(lldb) print second\_number

(int) $5 = 63

(lldb) print nsd

(int) $6 = 1

* НСД 44 та 63 дійсно дорівнює 1

(lldb) print first\_number

(int) $19 = 54

(lldb) print second\_number

(int) $20 = 22

(lldb) print nsd

(int) $21 = 2

* НСД 54 та 22 дійсно дорівнює 2

(lldb) print first\_number

(int) $27 = 33

(lldb) print second\_number

(int) $28 = 66

(lldb) print nsd

(int) $29 = 33

* НСД 33 та 66 дійсно дорівнює 33

Бачимо, що функція найбільшого спільного дільника працює коректно.

1. Через командну строку ми можемо встановити вхідні дані за допомогою функції main(int argc, char \*argv[]), де завдяки argc ми дізнаємось чи записано щось у командну строку, а argv буде символьним масивом, де в першому символі буде записано розмір матриці, а всі інші числа будуть самою матрицею, але ці дані будуть у символьному форматі. Тому нам потрібна функція strtol(), яка перетворює строку у число(якщо там записане число).

* Перевіримо роботу функції за допомогою функції printf():
* serhii@serhii-VirtualBox:~/tnp/programing-Stoienko/lab07/dist$ ./main.bin 4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
* 90 100 110 120
* 202 228 254 280
* 314 356 398 440
* 426 484 542 600
* serhii@serhii-VirtualBox:~/tnp/programing-Stoienko/lab07/dist$ ./main.bin 3 1 2 3 4 5 6 7 8 9
* 30 36 42
* 66 81 96
* 102 126 150
* serhii@serhii-VirtualBox:~/tnp/programing-Stoienko/lab07/dist$ ./main.bin 3 1 2 3 4 5 6 7 8 9 10 11 12
* 30 36 42
* 66 81 96
* 102 126 150
* serhii@serhii-VirtualBox:~/tnp/programing-Stoienko/lab07/dist$ ./main.bin 4 1 2 3 4 5 6 7 8 9 10 11 12
* Segmentation fault (core dumped)
* serhii@serhii-VirtualBox:~/tnp/programing-Stoienko/lab07/dist$ ./main.bin
* 0 0 0
* 0 0 0
* 0 0 0

Як бачимо, результати співпадають з реальними матрицями, тому функція множення матриці працює коректно.

Структура проекту лабораторної роботи:

lab07

├── doc

│ └── lab07.docx

├── README.md

└── src

└── main.c

Висновки: при виконанні лабораторної роботи були набуті навички створення функцій на мові С, зокрема: навчились генерувати рандомні числа, перетворювати строку у числа, оптимізувати код, перетворивши його у функцію, встановлювати вхідні дані через аргумент додатка.

Код проекту:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int find\_nsd(int first\_number, int second\_number);

void multiply\_matrix(int argc, char \*argv[], int \*multiply);

int main(int argc, char \*argv[])

{

srand((unsigned int)time(0));

int multiply[15][15];

multiply\_matrix(argc, argv, (int \*)multiply);

int first\_number = random() % 100;

int second\_number = random() % 100;

int nsd = find\_nsd(first\_number, second\_number);

// show on screen

printf(" first number = %d\n second number = %d\n nsd = %d\n",

first\_number, second\_number, nsd);

return 0;

}

int find\_nsd(int first\_number, int second\_number)

{

// create first and second number

int larger\_number;

int nsd = 1;

//we need to check which of the numbers are larger

//and assign new valuble to variable larger\_number

//for knowing, what number is larger for cycle for

if (second\_number >= first\_number) {

larger\_number = second\_number;

} else {

larger\_number = first\_number;

}

for (int i = 1; i <= larger\_number; i++) {

// if numbers are both evenly divisible by i, then i

// is common factor, the last i, that gonna be common

// facor of the numbers, is largest

if (first\_number % i == 0 && second\_number % i == 0) {

nsd = i;

}

}

return nsd;

}

void multiply\_matrix(int argc, char \*argv[], int \*multiply)

{

long int mas[15][15];

long int array\_size;

//

long int symbol\_number = 2;

char \*nul;

if (argc > 1)

// Перевіряємо чи були введені данні у командний рядок.

{

// Знаходимо розмір матриці, він записан першим в масиві символів

array\_size = strtol(argv[1], &nul, argc);

} else {

// Якщо ні, то тоді буде нульова матриця розмірами 3 на 3

array\_size = 3;

}

for (int row = 0; row < array\_size; row++)

{

for (int coll = 0; coll < array\_size; coll++) {

mas[row][coll] =

strtol(argv[symbol\_number], &nul, argc);

symbol\_number++;

}

}

// Так як треба множити матрицю на саму себе, то це повинна бути

// квадратична матриця, бо інакше число рядків та стовпців не будуть

// співпадати, а тоді не можна множити(за правилами множення).

// рядок

for (int row = 0; row < array\_size; row++) {

// стовпчик

for (int coll = 0; coll < array\_size; coll++) {

// даємо значення елементу матриці

multiply[array\_size \* row + coll] = 0;

// скільки разів множемо рядок матриці на стовпчик

for (int times = 0; times < array\_size; times++) {

// Помножимо рядок матриці на стовпчик матриці та отримуємо

// елемент на рядку та стовпчику матриці добутка

multiply[array\_size \* row + coll] +=

mas[row][times] \* mas[times][coll];

}

}

}

for (int row = 0; row < array\_size; row++) {

int new\_line = 0;

// стовпчик

for (int coll = 0; coll < array\_size; coll++) {

printf("%d\t", multiply[array\_size \* row + coll]);

new\_line++;

if (new\_line == array\_size) {

new\_line = 0;

printf("\n");

}

}

}

}