Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Звіт

з лабораторної роботи № 6

з дисципліни “Базові методології та технології програмування”

на тему

“Програмна реалізація алгоритмічних конструкцій повторення”

Виконав студент

академічної групи КІ-20

Микитенко Д. Ю.

Перевірив доцент

Доренський О. П.

Кропивницький-2020

Мета роботи: набути ґрунтовні вміння й практичні навички розроблення циклічних алгоритмів розв’язування обчислювальних задач та їх формального представлення мовою програмування С (ISO/IEC 9899:2018) задля реалізації програмних засобів у вільному кросплатформовому Code::Blocks IDE.

Завдання до лабораторної роботи:

1. Створити програмне забезпечення з лістинга 6.1.

2. Реалізувати програмне забезпечення розв’язування задачі 6.1.

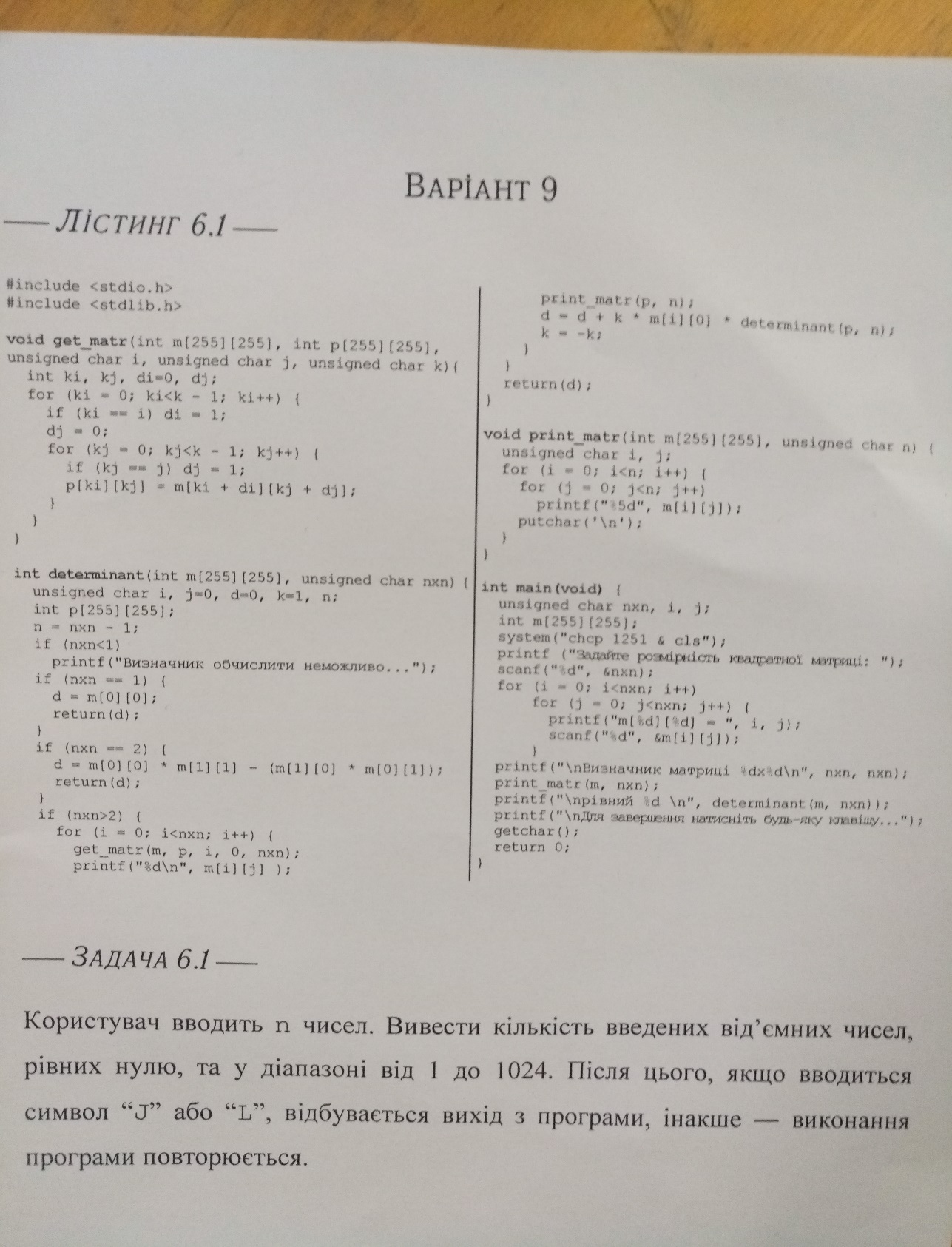


Рисунок 1 – Завдання роботи варіант № 9

1. Згідно з п.3-5 Порядку були створені деки: prj, Software, TestSuite.

Лістинг 6.1 був записаний у IDE Code::Blocks у папці prj.

#include <stdio.h>

#include <stdlib.h>

void get\_matr(int m[255][255], int p[255][255], unsigned char i, unsigned char j, unsigned char k){

int ki, kj, di = 0, dj;

for (ki = 0; ki < k - 1; ki++){

if (ki == i) di = 1;

dj = 0;

for (kj = 0; kj < k - 1; k++){

if (kj == j) dj = 1;

p[ki][kj] = m[ki + di][kj + dj];

}

}

}

int determinant(int m[255][255], unsigned char nxn){

unsigned char i, j = 0, d = 0, k = 1, n;

int p[255][255];

n = nxn - 1;

if (nxn < 1)

printf("Визначити область неможливо...");

if (nxn == 1){

d = m[0][0];

return(d);

}

if (nxn == 2){

d = m[0][0] \* m[1][1] - (m[1][0] \* m[0][1]);

return(d);

}

if (nxn > 2){

for (i = 0; i < nxn; i++){

get\_matr(m, p, i, 0, nxn);

printf("%d\n", m[i][j] );

print\_matr(p, n);

d = d + k \* m[i][0] \* determinant(p, n);

k = -k;

}

}

return(d);

}

void print\_matr (int m[255][255], unsigned char n){

unsigned char i, j;

for(i = 0; i < n; i++){

for(j = 0; j < n; j++){

printf("%5d", m[i][j]);

putchar('\n');

}

}

}

int main (void){

unsigned char nxn, i, j;

int m[255][255];

system("chcp 1251 & cls");

printf("Задайте розмірність квадратної матриці: ");

scanf("%d", &nxn);

for (i = 0; i < nxn; i++)

for (j = 0; j < nxn; j++){

printf("m[%d][%d] = ", i, j);

scanf("%d", &m[i][j]);

}

printf("\nВизначник матриці %dx%d\n ", nxn, nxn);

print\_matr(m, nxn);

printf("\nрівний %d \n", determinant(m, nxn));

printf("\nДля завершення натисніть будь-яку клавішу...");

getchar();

return 0;

}

1. За допомогою аналізу був сформований перелік змінних(вхідні та

вихідні значення) для задачі 6.1.

Вихідні дані:

1. С – ціле додатне число, не більше за 100, кількість чисел, що менше за 1024.

Вхідні дані:

1. N – ціле додатне число, не менше за 1 та не більше за 100, кількість чисел.
2. K – ціле число у проміжку від  –32768 до 32767, введене число.
3. Е – символ, відповідає за вихід з програми.

Обмеження:

1. Кількість введених чисел не може бути меншою за 1 та більшою за 100.
2. Кількість чисел, що менші за 1024, також не може бути більшою за 100.
3. Введене число повинне міститься у проміжку від -32768 до 32767.

Математична модель:

1. Якщо k < 1024, c = c + 1;

Алгоритм був сформульований у вербальній формі:

Початок:

1. Вивести анотацію, розробника, контактні дані, версію й рік розробки;
2. Задати значення N;
3. Якщо І більше за N, то перейти на крок 6;
4. Задати значення K;
5. Якщо К < 1024, то С = С + 1;
6. Перейти на крок 3;
7. Вивести С;
8. С = 0;
9. Задати значення Е;
10. Якщо Е не дорівнює J або L, перейти на крок 2.

Кінець.

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int main()

{

char \*locale = setlocale(LC\_ALL, "");

char e;

short n = 0, k = 0, c = 0;

printf("Призначення алгоритму - підрахувати кількість чисел, що менші за 1024\n");

printf("Розробник - Микитенко Даниїл Юрійович, електрона скринька: freezeee682@gmail.com\n");

printf("Версія розробки - 1.0 2020 року\n\n");

for(;;)

{

printf("Введіть кількість чисел, які ви хочете задати: ");

scanf("%hu", &n);

if ( n > 100 || n < 1)

{

printf("\nПомилка: неправильний ввод...\n\n");

return 0;

}

for(int i = 0; i < n; i++)

{

printf("Введіть число № %d: ", i+1);

scanf("%hu", &k);

if (k < 1024)

c++;

}

printf("\nКількість чисел, що менше за 1024, становить: %d", c);

c = 0;

printf("\nЯкщо ви хочете вийти із програми, натисніть J або L\n");

scanf("%c", &e);

scanf("%c", &e);

switch (e)

{

case 'J':

printf("Програма закінчила свою роботу.");

return 0;

case 'L':

printf("Програма закінчила свою роботу.");

return 0;

default:

continue;

}

getch();

}

}

Висновок: у ході виконання лабораторної роботи № 6 треба було записати лістинг й скомпілювати його, а також реалізувати програмне забезпечення задачі 6.1.

Щоб виконати перше завдання достатньо було отримати листок з варіантом роботи у викладача, проаналізувати лістинг, після чого створити проєкт консольного застосування у IDE Code::Blocks й безпосередньо записати сам лістинг 6.1. Під час виконання завдання труднощів не було виявлено й проєкт компілювався. Після цього .exe файл першого завдання був перенесений до теки Software, що була створена до цього на подобі до лабораторної роботи №4.

Задля виконання третього завдання потрібно було уважно проаналізувати дану задачу, задокументувати вхідні та вихідні значення, обмеження, їх типи та побудувати математичну модель. Після цього був створений шаблон програмного забезпечення до задачі 6.1, адже ПЗ не було протестовано. Також потрібно було прослідити за тим, щоб машинний код алгоритму, написаний мовою С, відповідав вимогам, тобто: містив вичерпну інформацію про його призначення (анотація), розробника, контактні дані, версію й рік розробки щодо. Потім було створено 20 тест-сьютів для умови задачі, 2 з яких були тест-сьютами щодо інтерфейсу й вимог, й було перевірено кожен з них для нашого програмного забезпечення, після чого виявилося, що всі тест-сьюти пройшли тестування. Також для задачі 6.1 був розроблений запит, щодо виходу з програми.

Підбиваючи підсумки цієї лабораторної роботи, були закріплені практичні вміння праці з тестуванням ПЗ, закріпленні знання з теми повторень(циклів), умовно «вічних» циклів, а також нових вимог до інтерфейсу програми. Також були здобуті практичні вміння з розробкою ПЗ для окремої задачі та створення інтерфейсу до неї.