НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №*6*

із дисципліни «Основи програмування»

на тему

«Динамічні масиви»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-73 | *асистент Громова В.В.* |
| *Садченко М.В.* |  |

Київ — 2018

ЗМІСТ

[ВСТУП 3](#_Toc511687690)

[1 ПОСТАНОВКА ЗАДАЧІ 3](#_Toc511687691)

[1.1 Мета роботи 3](#_Toc511687692)

[1.2 Варіант та завдання лабораторної роботи 3](#_Toc511687693)

[2 ОПИС ПРОГРАМИ 3](#_Toc511687694)

[2.1 Опис коду 3](#_Toc511687695)

[2.3 Опис бібліотек 4](#_Toc511687696)

[ВИСНОВКИ 5](#_Toc511687697)

[ПЕРЕЛІК ПОСИЛАНЬ 5](#_Toc511687698)

[ДОДАТОК А (Відповіді на контрольні запитання) 5](#_Toc511687699)

[ДОДАТОК Б (Код програми) 6](#_Toc511687700)

[ДОДАТОК В (Результат роботи програми) 8](#_Toc511687701)

[ДОДАТОК Г (Блок-схема) 9](#_Toc511687702)

# ВСТУП

Даний звіт створено для детального ознайомлення з лабораторною роботою №6, її завданням, структурою, розробкою програмного коду мовою програмування C, у супроводі детального розбору поставлених вимог.

# 1 ПОСТАНОВКА ЗАДАЧІ

## 1.1 Мета роботи

Набути практичних навичок створення й опрацювання одновимірних і багатовимірних динамічних масивів.

## 1.2 Варіант та завдання лабораторної роботи

Завдання з варіанту №24:

* Знайти різницю між сумами елементів, що лежать на головній і побічної діагоналях матриці М (7х7).

# 2 ОПИС ПРОГРАМИ

## 2.1 Опис коду

Програма починається з того, що користувачеві надається вибір між доступом до самовільного заповнення матриці (кнопка «1»), за допомогою псевдовипадкових чисел («2»), очисткою консолі («8») та виходом з програми («9»). Реалізовано за допомогою циклу do-while, функцій strchr (перевіряє, чи входить введений символ в заданий масив) та getch (<conio.h>) (за її допомогою компьютер отримує символ одразу після натискання; користувачеві непотрібно підтверджувати свій вибір Enter`ом). Далі правильний символ надходить до switch-case, що розмежує кожну з дій окремою підпрограмою. Для того, щоб з одного ‘кейсу’ програма просто не переходила у інший – в кінці кожної з підпрограм ставиться break.

Очистка консолі виконується за допомогою команди system(“cls”) з бібліотеки <stdlib.h>.

Вихід з програми реалізовано командою exit(0), теж з <stdlib.h>.

Програма починається з того, що користувач обирає між способами заповнення матриці. Натиснувши «1», комп’ютер пропонує ввести дані матриці самовільно. Ці дані перевіряються, тому для ‘валідації’ натуральних чисел (функція int valid\_int()) використовуються, окрім циклу while, ще такі функції, як strlen() (рахує кількість елементів у масиві), isdigit() (перевіряє числове значення(чи є елемент цифрою)) та atoi() (конвертує строковий тип до цілих чисел).

Натиснувши «2», користувачеві пропонується обрати межі псевдовипадкових чисел для заповнення. Оскільки ми працюємо з динамічною пам’яттю, для ініціалізації двовимірного масиву застосовано функцію malloc(), що «бронює» пам’ять для даних масиву. Для подальшого «звільнення» пам’яті було використано функцію free() з вказанням необхідного масиву.

Коли двовимірний масив заповнено, програма за простим алгоритмом починає підраховувати спочатку окремі суми головної та побічної діагоналей, а потім їх спільну різницю. Результат виводиться користувачеві на екран.

Після закінчення кожної програми (окрім виходу з програми) відбувається повтор, що дозволяє ще раз обрати між заданими діями, натиснувши «1», «2», «8» чи «9». У всіх функціях повторювані дії здійснювалися за допомогою нескінченного циклу while(1).

## 2.3 Опис бібліотек

Для початку, опишемо використані бібліотеки у програмі:

<stdio.h> - відповідає за ввод\вивід даних;

<stdlib.h> - відповідає за функції, що займаються виділенням пам'яті, контроль процесу виконання програми, перетворення типів та інші;

<string.h> - містить функції для роботи з нуль-термінованими рядками і різними функціями роботи з пам'яттю;

<ctype.h> - містить оголошення функцій для класифікації символів;

<conio.h> - застосовується для створення текстового інтерфейсу.

# ВИСНОВКИ

Під час виконання лабораторної роботи були вивчені нюанси роботи із динамічними масивами у мові програмування С. Було застосовано функції, що функцію malloc() для динамічного виділення пам’яті та free() для звільнення пам’яті.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Програмування мовою С. Методичні вказівки до виконання лабораторних робіт із кредитного модуля «Програмування-2. С.» для студентів спеціальності 113 «Прикладна математика» [Текст] / В. В. Громова. — К. : НТУУ «КПІ», 2016. —5- 115 с.

# ДОДАТОК А (Відповіді на контрольні запитання)

1. *Поняття статичної та динамічної пам’ті.*

*Статична пам’ять* – SRAM (Static Random Access Memory) здатна зберігати інформацію в статичному режимі, тобто скільки завгодно довго за відсутності обігу (але за наявності живлячої напруги). Але це не єдина її перевага. SRAM має більш високу швидкодію, ніж DRAM, і може працювати на тій же частоті, що й сучасні процесори.

Динамі́чна операти́вна па́м'ять або DRAM (Dynamic Random Access Memory) — один із видів комп'ютерної пам'яті із довільним доступом (RAM), найчастіше використовується як ОЗП сучасних комп'ютерів.

1. *Функції управління оперативною пам’яттю.*

void\* malloc(size\_t size);

void\* calloc(size\_t num, size\_t size);

void\* realloc(void\* memblock, size\_t size);

void free(void\* memblock);

1. *Коли потрібно використовувати динамічні масиви?*

На відміну від цього місце для зберігання динамічних масивів виділяється та вивільняється в процесі виконання програми. В одних випадках виділення та вивільнення пам’яті система здійснює автоматично.

Наприклад, коли відводиться пам’ять для зберігання локальних масивів у процедурах і функціях. В інших випадках користувачеві надається можливість запросити ділянку пам’яті потрібного розміру та вивільнити її згодом. Тільки в такий спосіб у програмах можна завести масив змінного розміру.

1. *Як розподілити пам’ять під динаміні масиви?*

На відміну від цього місце для зберігання *динамічних* *масивів* виділяється та вивільняється в процесі виконання програми. В одних випадках виділення та вивільнення пам’яті система здійснює автоматично. Наприклад, коли відводиться пам’ять для зберігання локальних масивів у процедурах і функціях. В інших випадках користувачеві надається можливість запросити ділянку пам’яті потрібного розміру та вивільнити її згодом. Тільки в такий спосіб у програмах можна завести масив змінного розміру.

За динамічного розподілу пам’яті, як один із варіантів, для масивів потрібно описати відповідний указівник і просвоїти йому значення за допомогою функції calloc().

1. *Як вивільнити пам’ять, зайняту динамічним масивом?*

void free(void\* memblock);

Ця функція слугує для вивільнення раніше виділеної області

динамічної пам’яті.

Вивільнити пам’ять, зайняту під n масивів — рядків:

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

free(uk[i]);

1. *Як звернутися до і-го елементу динамічного масиву?*

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

uk[i] =(*<тип>*\*) malloc(m \* sizeof(*<тип>*));

# ДОДАТОК Б (Код програми)

#include <stdio.h>

#include <string.h> // strlen(), strchr()

#include <stdlib.h> // atoi(), exit(0), system("cls")

#include <ctype.h> // isdigit()

#include <conio.h> // getch()

#include <time.h> // rand()

#define SIZE 7

int valid\_int() {

char str[65];

int num;

while (1) {

int smb = 0;

scanf("%s", str); *//input of value*

int len = strlen(str);

for (int i=0; i<len; ++i) {

if (isdigit(str[i]) == 0) {

++smb;

**break**;

}

}

if (smb == 0) **break**;

else printf("need natural number!: ");

}

num = atoi(str); *//convert str to int*

return num;

}

int \*\*init\_fill(char choice) {

*// INITIALIZATION of MATRIX*

int \*\*matrix = (int \*\*) malloc(SIZE \* sizeof(int \*));

for (int i = 0; i < SIZE; ++i) {

matrix[i] = (int \*) malloc(SIZE \* sizeof(int));

}

*// FILLING of MATRIX*

if (choice == 'u') {

for (int i = 0; i < SIZE; ++i) {

for (int j = 0; j < SIZE; ++j) {

printf("matrix[%d][%d] = ", i+1, j+1);

matrix[i][j] = valid\_int();

}

}

}

else if (choice == 'r') {

printf("Enter max range: ");

int max\_ran = valid\_int();

for (int i = 0; i < SIZE; ++i) {

for (int j = 0; j < SIZE; ++j) {

matrix[i][j] = rand()%max\_ran;

printf("matrix[%d][%d] = %d**\n**", i+1, j+1, matrix[i][j]);

}

}

}

return matrix;

}

void calculating(int \*\*matrix) {

int main\_diag = 0, sec\_diag = 0;

*// ELEMENTS SUM of MAIN and SECONDARY DIAGONALS*

for (int i = 0; i < SIZE; ++i) {

for (int j = 0; j < SIZE; ++j) {

if (i == j) {

main\_diag += matrix[i][j];

}

if (i+j == SIZE-1) {

sec\_diag += matrix[i][j];

}

}

}

if (main\_diag >= sec\_diag)

printf("**\n**main\_diag - secondary\_diag = %d**\n**", main\_diag - sec\_diag);

else printf("**\n**secondary\_diag - main\_diag = %d**\n**", sec\_diag - main\_diag);

}

void main() {

while (1) {

int choice;

fflush(stdin);

printf("**\n**Press '1' to create user matrix.**\n**"\

"Press '2' to get random matrix.**\n**"\

"Press '8' to clear console.**\n**"\

"Press '9' to get out program.**\n**");

do {

choice = getch();

} while(!strchr("1289", choice));

switch (choice) {

case '1': {

*// USER matrix*

int \*\*matrix = init\_fill('u');

calculating(matrix);

*// MATRIX FREEING*

for (int i = 0; i < SIZE; ++i) {

int \*currentIntPtr = matrix[i];

free(currentIntPtr);

}

**break**;

}

case '2': {

time\_t t;

*// Initializes random number generator*

srand((unsigned) time(&t));

*// Initialization and filling by RANDOM*

int \*\*matrix = init\_fill('r');

calculating(matrix);

*// MATRIX FREEING*

for (int i = 0; i < SIZE; ++i) {

int \*currentIntPtr = matrix[i];

free(currentIntPtr);

}

**break**;

}

case '8': {

printf(" 7 ballov :)**\n**");

system("cls");

**break**;

}

case '9': exit(0); **break**;

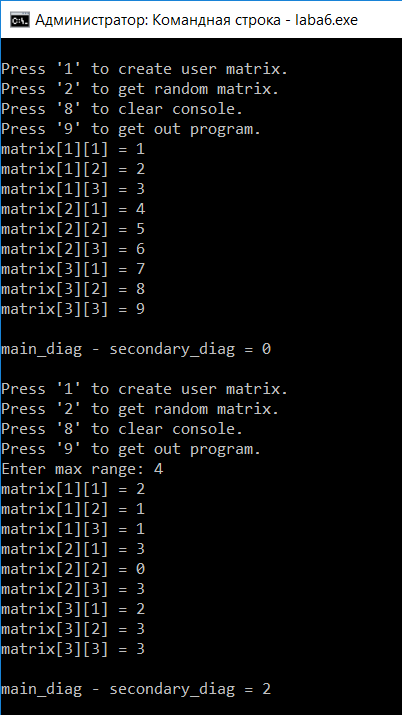
default: **break**;

}

}

}

# ДОДАТОК В (Результат роботи програми)



# ДОДАТОК Г (Блок-схема)

