**СТРУКТУРА**

**звіту з лабораторної роботи**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО”**

**ЗВІТ**

**з лабораторної роботи №1**

**з навчальної дисципліни “Програмування складних алгоритмів”**

**Тема: СКЛАДНІСТЬ АЛГОРИТМІВ**

**Варіант №1**

**Виконав студент групи ТР–15**

Руденко Владислав Ігорович

з оцінкою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Перевірив доцент кафедри**

Андрій ОНИСЬКО\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ року

**Київ 2022**

**І. Мета:**

Метою лабораторної роботи є отримання практичних навичок визначати часову складність алгоритму. Навчитися будувати алгоритми з мінімальною часовою складністю для вирішення поставлених задач.

**Теоретична частина.**

**Основні поняття та визначення:**  
Алгоритм — це набір команд, необхідних для вирішення певних задач.   
Кожен алгоритм передбачає існування початкових (вхідних) даних та в результаті роботи призводить до отримання певного результату. Робота кожного алгоритму відбувається шляхом виконання послідовності деяких елементарних дій. Ці дії називають кроками, а процес їхнього виконання називають алгоритмічним процесом. В такий спосіб відзначають властивість дискретності алгоритму.

1. **Параметри алгоритму.  
   -***Детермінованість* - властивість алгоритму, яка передбачає, що в ньому усі вказівки повинні бути чіткими й однозначними  
   -*Масовість*-інваріантність щодо вхідних даних; уточнення поняття розв'язання задачі у загальному вигляді  
   -*Результативність*-отримання необхідного конструктивного об'єкта після закінчення конструктивного процесу  
   -*Спрямованість*-всі конструктивні об'єкти, як множини, упорядковані
2. **Характеристики алгоритму.**

***Мета алгоритму* —** це визначальна властивість алгоритму.

Мета алгоритму підказує нам, де ми можемо застосувати вказаний алгоритм — для сортування чисел в масиві, стискання даних, або походу в магазин. Очевидно, слід підбирати алгоритм, який найкраще підійде для вирішення поставленої задачі.

***Точність алгоритму*** - Важливо розуміти, що не всі задачі потребують отримання вірної відповіді з ймовірністю, рівною 1. Для деяких завдань можуть підійти алгоритми, які дають оцінку ймовірності тієї чи іншої події, або приблизне значення. Набагато важливіше буде отримати приблизну відповідь швидко, ніж витрачати дні, тижні, або навіть роки на отримання точного значення.

***Швидкість алгоритму*** - Вона показує не лише середню швидкість роботи алгоритму, але й можливу зміну швидкості в залежності від кількості вхідних даних.

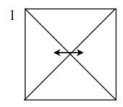
***Методи побудови алгоритму*** - Існують різноманітні методи побудови алгоритмів: метод декомпозиції, метод послідовних наближень, метод випадкового пошуку і т.д. Знання методів побудови алгоритмів може також прислужитися в якості відправної точки при написанні власного алгоритму.

1. **Асимптотична складність.**Для оцінки продуктивності алгоритмів можна використовувати різні підходи. Самий нехитрий - просто запустити кожен алгоритм на кількох завданнях і порівняти час виконання. Інший спосіб - математично оцінити час виконання підрахунком операцій.  
   При оцінці за функцію береться кількість операцій, що зростає найшвидше. Тобто, якщо в програмі одна функція, наприклад, множення, виконується O (n)раз, а складання - O (n 2)раз, то загальна складність програми - O (n 2), Так як в кінці кінців при збільшенні nшвидші (в певний, константнечисло раз) складання стануть виконуватися настільки часто, що будуть впливати на швидкодію куди більше, ніж повільні, але рідкісні множення. символ   
   O ()показує виключно асимптотику!  
    При оцінці O () константи не враховуються. Нехай один алгоритм робить 2500n +1000 операцій, а інший - 2n + 1. Обидва вони мають оцінку O (n), Так як їх час виконання зростає лінійно.
2. **Алгоритми з експоненціальною та поліноміальною складністю.***Експоненціальний алгоритм* **-** для практичних розмірностей задач, як правило, не можуть виконатися повністю. Зазвичай розв'язок задач, що породжують експоненціальні алгоритми, пов'язаний з повним перебором всіх можливих варіантів, і, зважаючи на практичну неможливість реалізації такої стратегії, їх розв'язують іншими шляхами. Якщо навіть існує експоненціальний алгоритм для пошуку оптимального розв'язку деякої задачі, то на практиці застосовуються інші, ефективніші поліноміальні алгоритми для пошуку прийнятного розв'язку (не обов'язково оптимального, а лише наближеного до оптимального).*Поліноміальні алгоритми* -при великих розмірностях задач (які, зазвичай, найцікавіші на практиці), поліноміальні алгоритми можуть бути виконані на сучасних комп'ютерах. Поліноміальні алгоритми також можуть істотно розрізнятися залежно від степеня полінома, що апроксимує залежність

**ІІ. Завдання:**

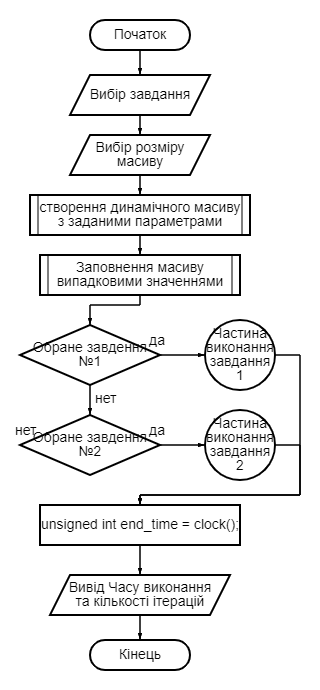
Метою лабораторної роботи є отримання практичних навичок визначати часову

складність алгоритму. Навчитися будувати алгоритми з мінімальною часовою складністю для вирішення поставлених задач.  
  
**Завдання №1**В кожному рядку знайти максимальний елемент. Поміняти місцями рядки з найбільшим і

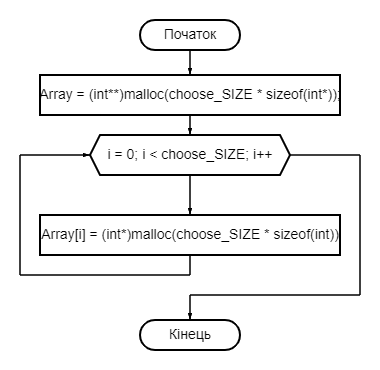
найменшим максимальними елементам  
  
**Завдання №2**  


**ІІІ. Результати виконання лабораторної роботи.**

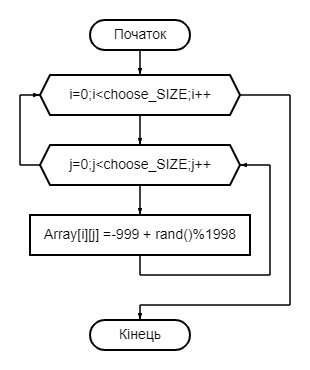
**Блок Схеми:**

****

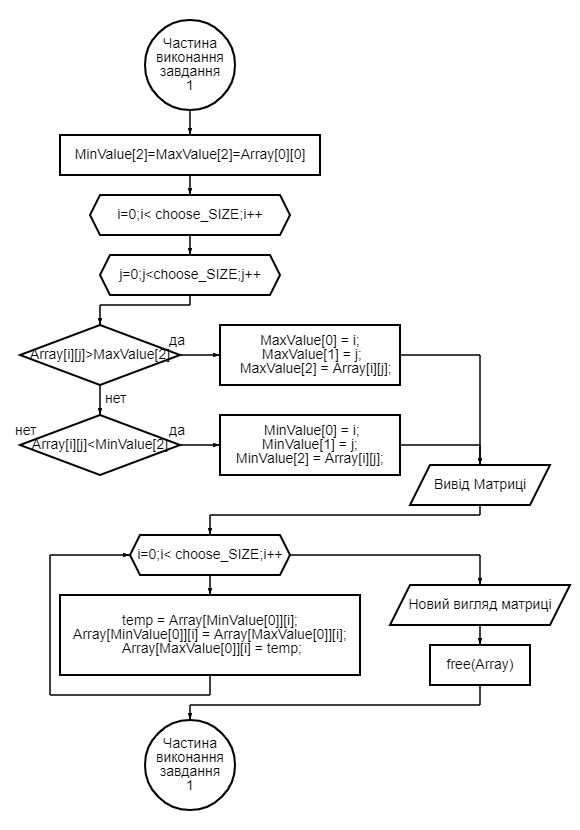
Блок-Схема 1 (Основний Алгоритм)

****

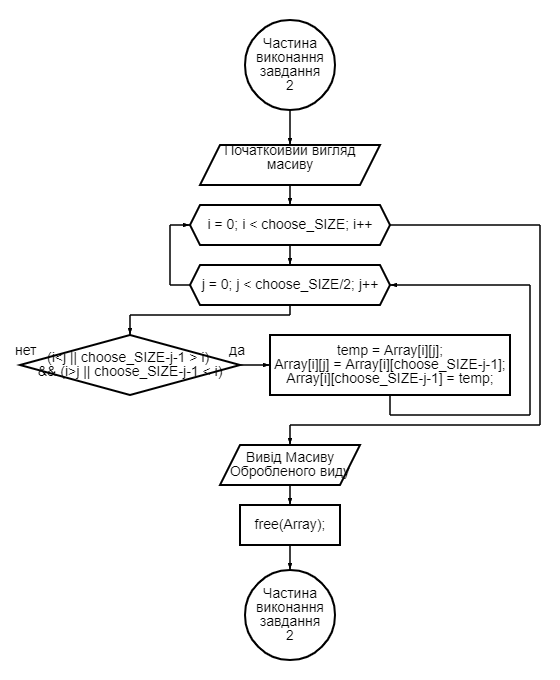
Блок-Схема 2 (Створення динамічного масиву)



Блок-Схема 3 (Заповнення масиву випадковими числами)



Блок-Схема 4 (Блок виконання Завдання 1)

****

Блок-Схема 5 (Блок виконання Завдання 2)

**Опис програми (написана на мові Сі):  
-** Спочатку було створенно соновну структуру, яка створює масив необхідної величини, та заповнення його випадковими числами. Після чого за допомогою оператора switch\* слідує виконання необхідного завдання. Після виконання блоку завдання програма повідомляє користувача про окінчення опрацювання програми ,а також оголошення часу, витраченого на опрацювання, та ітерацій, які було використано під час роботи.

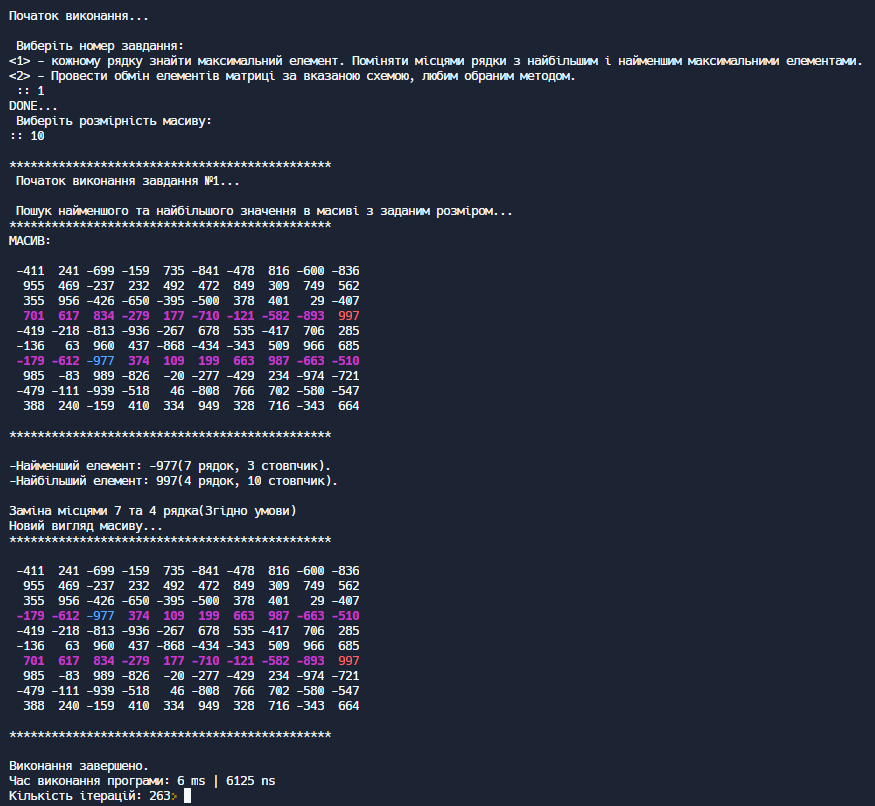
- Блок завдання 1 виконує первинну обробку заданого масиву, вишукуючи та запам’ятовуючи місцезнаходження найбільшого та найменшого значення в цьому масиві. При знаходженні однакових найбільших\найменших значення програма запам’ятовую тільке те значення, яке було знайдено раніше, але повідомляє користувачу про наявність таких значень в графічній формі (якщо такі значення присутні), після чого відбувається заміна необіхних рядків та виведення результатів на екран.

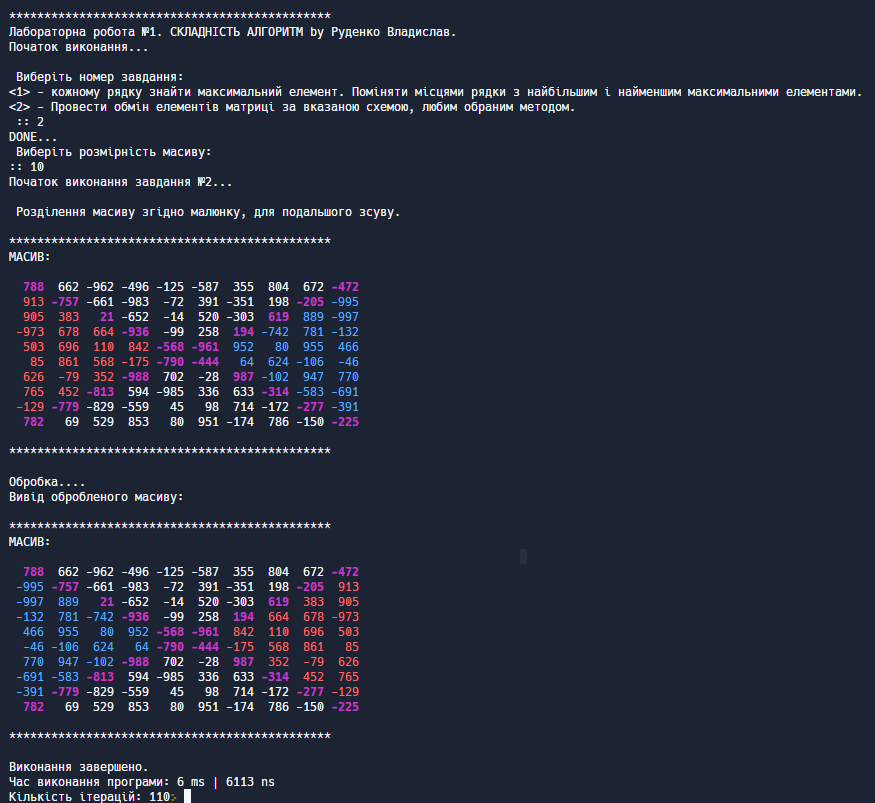
-Блок завдання 2 не виконує пошук яких-небудь значення, що зменшує кількість ітерацій. Натомість програма одразу виконує обробку масиву, переставляючи необхідні ячейки масиву, при цьому виводить на екран початковий та оброблений стани масиву.  
  
-

**Результати роботи:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поріврнянн часу виконання двох завданнь | | | | |
| Розмір матриці | К-ітерацій  *(завдання 1)* | К-ітерацій  *(завдання 2)* | Час виконання *(завдання 1) µs* | Час виконання *(завдання 2) µs* |
| **10x10** | **263** | **110** | **6125** | **6113** |
| **50x50** | **5180** | **3050** | **9317** | **10252** |
| **100x100** | **20345** | **12350** | **15453** | **17417** |
| **500x500** | **501524** | **311750** | **191362** | **252901** |

**Результати роботи у вигляді скріншотів:**

****

Скріншот 1 (Завдання №1)  
 ****

Скріншот 2 (Завдання №2)

**Посилання на repl.it:**

<https://replit.com/join/wsyoteyrhc-hetik>

**IV. Висновки.**

В ході виконання Лабораторної роботи №1 було набуто практичні навички визначення складності алгоритму. Було побудовано алгоритм з мінімальною часовою складністю для вирішення поставленої задачі. Було створено алгоритм опрацювання матриці, який включав в себе 2 задачі: пошук і заміна рядків по параметрам, та перестановка елементів матриці згідно з графічним завданням. Під час виконання було виявлено та виправлено ряд помилок.

**Програмний код:**

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <time.h>

#define RED "\x1b[31m"

#define BLUE "\x1b[34m"

#define YELLOW "\033[1;35m"

#define RESET "\x1b[0m"

int main(void)

{

srand(time(NULL));

int \*\*Array;

int choose\_variant, choose\_SIZE, again = 0, i, j, temp, INTER = 0;

int MinValue[3];

int MaxValue[3];

// 1ше значення - рядок, 2ге- стовпчик, 3е- значення елементу

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nЛабораторна робота №1. СКЛАДНІСТЬ АЛГОРИТМ by Руденко Владислав.\nПочаток виконання...\n\n Виберіть номер завдання: \n<1> - кожному рядку знайти максимальний елемент. Поміняти місцями рядки з найбільшим і найменшим максимальними елементами. \n<2> - Провести обмін елементів матриці за вказаною схемою, любим обраним методом. \n :: ");

scanf("%i", &choose\_variant);

printf("DONE...\n Виберіть розмірність масиву:\n:: ");

scanf("%i", &choose\_SIZE);

Array = (int\*\*)malloc(choose\_SIZE \* sizeof(int\*));

for (int i = 0; i < choose\_SIZE; i++)

Array[i] = (int\*)malloc(choose\_SIZE \* sizeof(int));

for(i=0;i<choose\_SIZE;i++)

for(j=0;j<choose\_SIZE;j++)

Array[i][j] =-999 + rand()%1998;

switch(choose\_variant)

{

case 1:

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Початок виконання завдання №1...\n\n Пошук найменшого та найбільшого значення в масиві з заданим розміром... \n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nМАСИВ:\n\n");

MinValue[2]=MaxValue[2]=Array[0][0];

for(i=0;i< choose\_SIZE;i++)

for(j=0;j<choose\_SIZE;j++)

{

if (Array[i][j]>MaxValue[2])

{

MaxValue[0] = i;

MaxValue[1] = j;

MaxValue[2] = Array[i][j];

INTER = INTER+3;

}

if (Array[i][j]<MinValue[2])

{

MinValue[0] = i;

MinValue[1] = j;

MinValue[2] = Array[i][j];

INTER = INTER+3;

}

INTER = INTER+2;

}

for (i = 0; i < choose\_SIZE; i++)

{

for (j = 0; j < choose\_SIZE; j++){

if (Array[i][j]==MinValue[2])

printf(BLUE"%5.d"RESET, Array[i][j]);

else if (Array[i][j]==MaxValue[2])

printf(RED"%5.d"RESET, Array[i][j]);

else if (i==MinValue[0] || i==MaxValue[0])

printf(YELLOW"%5.d"RESET, Array[i][j]);

else

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

}

printf("\n");

}

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n-Найменший елемент: %i(%i рядок, %i стовпчик).\n-Найбільший елемент: %i(%i рядок, %i стовпчик).\n\nЗаміна місцями %i та %i рядка(Згідно умови)", MinValue[2],MinValue[0]+1,MinValue[1]+1,MaxValue[2],MaxValue[0]+1,MaxValue[1]+1, MinValue[0]+1,MaxValue[0]+1);

if(MinValue[0]!=MaxValue[0])

for(i=0;i< choose\_SIZE;i++)

{

temp = Array[MinValue[0]][i];

Array[MinValue[0]][i] = Array[MaxValue[0]][i];

Array[MaxValue[0]][i] = temp;

INTER = INTER+3;

}

printf("\nНовий вигляд маcиву...\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

for (i = 0; i < choose\_SIZE; i++)

{

for (j = 0; j < choose\_SIZE; j++){

if (Array[i][j]==MinValue[2])

printf(BLUE"%5.d"RESET, Array[i][j]);

else if (Array[i][j]==MaxValue[2])

printf(RED"%5.d"RESET, Array[i][j]);

else if (i==MinValue[0] || i==MaxValue[0])

printf(YELLOW"%5.d"RESET, Array[i][j]);

else

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

}

printf("\n");

}

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

free(Array);

break;

case 2:

printf("Початок виконання завдання №2...\n\n Розділення масиву згідно малюнку, для подальшого зсуву.\n \n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nМАСИВ:\n\n");

for (i = 0; i < choose\_SIZE; i++)

{

for (j = 0; j < choose\_SIZE; j++)

{

if (i==j || choose\_SIZE-i-1 ==j)

printf(YELLOW"%5.d"RESET, Array[i][j]);

else if((i<j || choose\_SIZE-j-1 > i) && (i>j || choose\_SIZE-j-1 < i))

{

if(j<choose\_SIZE/2)

printf(RED"%5.d"RESET, Array[i][j]);

else

printf(BLUE"%5.d"RESET, Array[i][j]);

}

else

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

}

printf("\n");

}

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\nОбробка....\n");

for (i = 0; i < choose\_SIZE; i++)

for (j = 0; j < choose\_SIZE/2; j++){

if((i<j || choose\_SIZE-j-1 > i) && (i>j || choose\_SIZE-j-1 < i)){

temp = Array[i][j];

Array[i][j] = Array[i][choose\_SIZE-j-1];

Array[i][choose\_SIZE-j-1] = temp;

INTER = INTER+3;

}

INTER++;

}

printf("Вивід обробленого масиву:\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nМАСИВ:\n\n");

for (i = 0; i < choose\_SIZE; i++)

{

for (j = 0; j < choose\_SIZE; j++)

{

if (i==j || choose\_SIZE-i-1 ==j)

printf(YELLOW"%5.d"RESET, Array[i][j]);

else if((i<j || choose\_SIZE-j-1 > i) && (i>j || choose\_SIZE-j-1 < i))

{

if(j<choose\_SIZE/2)

printf(BLUE"%5.d"RESET, Array[i][j]);

else

printf(RED"%5.d"RESET, Array[i][j]);

}

else

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

}

printf("\n");

}

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

free(Array);

break;

default:

printf("Хибний вибір...\nНажаль, ви маєте вибрати варіант серед запропонованих.");

return 0;

break;

}

unsigned int end\_time = clock();

printf("\nВиконання завершено.\nЧас виконання програми: %i ms | %i ns\nКількість ітерацій: %i", end\_time/1000,end\_time, INTER);

}