НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп'ютерних систем

Лабораторна робота №1

з дисципліни

«Паралельні та розподілені обчислення»

ТЕМА: «Робота з компіляторами мов C та Java в режимі командного рядка»

Виконав:

студент групи КВ-04

Забродський Віталій Миколайович

Київ – 2022

Постановка задачі

1.Написати програму розв’язання задачі пошуку (за варіантом) у двовимірному масиві (матриці) одним з алгоритмів методу лінійного пошуку.
Розміри матриці m та n взяти самостійно у межах від 7 до 10. Розмір матриці повинен задаватися аргументом запуску програми.
Програма обов'язково повинна бути написана і структурована таким чином:

a) оголошення структур даних (typedef) повинно бути зроблено у окремому заголовочному файлі;

b) повинно бути щонайменше три файли із вихідним кодом (не враховуючи необхідні заголовочні файли), що міститимуть реалізації функцій введення (випадкові значення, наперед сортовані значення, з клавіатури), обробки, та виведення на друк (pretty_print) елементів матриці;

c) для виконання завдання обробки елементів матриці повинно бути написано дві різні функції:

1) з додатковими операторами виведення налагоджувальної інформації на друк (debug-версія);

2) з виконанням заданих дій без додаткового виведення налагоджувальної інформації (release-версія).

4. Вибір функції повинен робити користувач при запуску програми через аргумент запуску. Наприклад, опція -d вмикає debug режим.
5. Для компіляції написаної багатофайлової програми написати окремий make-файл, причому:

a) при зміні одного із вихідних файлів повинен перекомпільовуватися лише цей файл (а також відбуватися дії, необхідні для генерації бінарного файлу);

b) при видаленні бінарного файлу та незмінних вихідних файлах повинне відбуватися лише лінкування (компоновка бінарного файлу з об’єктних);

c) забезпечити окрему ціль для очистки згенерованих файлів;

6. Забезпечити можливість компіляції написаної багатофайлової програми двома способами:

a) за допомогою однієї команди gcc;

b) за допомогою make-файлу. Марченко О.І., Марченко О.О.

7.Виконати тестування та налагодження програми на комп’ютері. При тестуванні програми необхідно підбирати такі вхідні набори початкових значень матриці, щоб можна було легко відстежити коректність виконання пошуку і ця коректність була б протестована для всіх можливих випадків. З метою тестування дозволяється використовувати матриці меншого розміру.

Варіант 5

Задано матрицю дійсних чисел A[n][n]. У побічній діагоналі матриці знайти перший мінімальний і останній максимальний елементи, а також поміняти їх місцями.

Код програми

MAIN.C

```c
#include "process.h"
#include "realeseAndDebug.h"
#include "memory.h"


int main(int argc, char** argv)
{
    char flag;
    switch(flag=verbose_flag(argc,argv))
    {
    case'r':
        make_arr();
        realese();
        break;

    case'd':
        make_arr();
        debug();
        break;
    }
    del_arr();

}
```

INPUT.C

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "Typedef.h"
void inputRand(){
    int i,j;
    srand(time(0));
    for(i = 0; i < n; i++){
        for(j = 0; j < n; j++){
            arr[i][j]=-50 + rand() % 100;
        }
    }
}
void inputSorted() {
    int i,j;
    inputRand();
    double number = 0;
    for(i = 0; i < n ; i++) {
        for(j = 0; j < n; j++) {
            arr[i][j] = number;
            number++;
        }
```

```
    }
}
void inputKeyboard(){
    int i,j;
    for(i = 0; i < n; i++){
        for(j = 0; j < n; j++){
            scanf("%d",&arr[i][j]);
        }
    }
}
```

INPUT.H

```
#ifndef INPUT_H_INCLUDED
#define INPUT_H_INCLUDED
void inputRand(void);
void inputSorted(void);
void inputKeyboard(void);



#endif // INPUT_H_INCLUDED
```

MEMORY.C

```
#include<stdlib.h>
#include"Typedef.h"



void make_arr()
{
    arr = (int**)malloc(n * sizeof(int*));
    for(int i = 0; i < n; ++i)
    {
        *(arr+i) = (int*)malloc(n * sizeof(int));
    }
}

void del_arr()
{
    for(int i = 0;i < n; ++i)
    {
        free(arr[i]);
    }
    free(arr);
}
```

MEMORY.H

```
#ifndef MEMORY_H_INCLUDED
#define MEMORY_H_INCLUDED
void make_arr();
void del_arr();

#endif // MEMORY_H_INCLUD
```

PRETTY_PRINT.C

```c
#include <stdio.h>
#include "Typedef.h"
void outputArr(){
    int i,j;
    for(i = 0; i < n; i++ ){
        for(j = 0; j < n ; j++ )
            printf("%d\t",arr[i][j]);
        printf("\n");
    }
    printf("\n");
}
```

PRETTY_PRINT.H

```c
#ifndef PRETTY_PRINT_H_INCLUDED
#define PRETTY_PRINT_H_INCLUDED

void outputArr();

#endif // PRETTY_PRINT_H_INCLUDED
```

PROCESS.C

```c
#include <stdio.h>
#include "Typedef.h"
#include <getopt.h>
#include <stdlib.h>

void findAndSwapRelease()
{
    int max_element=0, min_element=0, i, tmp;

    for(i=0; i<n; i++)
    {
        if (arr[n-i-1][i]<arr[n-min_element-1][min_element])
            min_element=i;
    }

    for(i=0; i<n; i++)
    {
        if (arr[n-i-1][i]>=arr[n-max_element-1][max_element])
            max_element=i;
    }

    tmp=arr[n-min_element-1][min_element];
    arr[n-min_element-1][min_element]=arr[n-max_element-1][max_element];
    arr[n-max_element-1][max_element]=tmp;

}
```

```
void findAndSwapDebug()
{
    int max_element=0, min_element=0, i, tmp;

    for(i=0; i<n; i++)
    {
        if (arr[n-i-1][i]<arr[n-min_element-1][min_element])
            min_element=i;
        printf("min element is %d on position [%d][%d]\n",arr[n-
min_element-1][min_element], n-min_element-1, min_element);
    }
    printf("\n\n");
    for(i=0; i<n; i++)
    {
        if (arr[n-i-1][i]>=arr[n-max_element-1][max_element])
            max_element=i;
        printf("max element is %d on position [%d][%d]\n",arr[n-
max_element-1][max_element], n-max_element-1, max_element);
    }
    printf("\n\n");
    printf("Success!!!!! \nFinal results:\nmin element is %d on position
[%d][%d]\nmax element is %d on position [%d][%d]",arr[n-min_element-
1][min_element], n-min_element-1, min_element,arr[n-max_element-
1][max_element], n-max_element-1, max_element);
    tmp=arr[n-min_element-1][min_element];
    arr[n-min_element-1][min_element]=arr[n-max_element-1][max_element];
    arr[n-max_element-1][max_element]=tmp;
    printf("\nResult:\n");
}

char verbose_flag(int argc,char*argv[])
{
    char choise='r';
    char options;
    while((options = getopt(argc,argv, "drn:"))!=-1)
    {
        switch(options)
        {
        case'r':
            choise = 'r';
            break;

        case'd':
            choise = 'd';
            break;

        case'n':
            n = atoi(optarg);
            break;
        }
    }
    return choise;
}
```

PROCESS.H

```
#ifndef PROCESS_H_INCLUDED
#define PROCESS_H_INCLUDED

void findAndSwapRelease(void);
void findAndSwapDebug(void);
char verbose_flag(int argc,char*argv[]);


#endif // PROCESS_H_INCLUDED
```

realeseAndDebug.c

```
#include"input.h"
#include"pretty_print.h"
#include"process.h"
#include <stdio.h>



void realese()
{
    printf("_____________________REALESE_____________________\n");
    printf("-----With random array and relese function--------\n");
    inputRand();
    outputArr();
    findAndSwapRelease();
    outputArr();

    printf("------With sorted array and release function------------\n");
    inputSorted();
    outputArr();
    findAndSwapRelease();
    outputArr();

    printf("--------With array from keyboard and release function---------
-\n");
    inputKeyboard();
    outputArr();
    findAndSwapRelease();
    outputArr();
}


void debug()
{
    printf("_____________________DEBUG_____________________\n");
    printf("-----With random array and debug function--------\n");
    inputRand();
    outputArr();
    findAndSwapDebug();
    outputArr();

    printf("------With sorted array and debug function------------\n");
    inputSorted();
    outputArr();
```

```
    findAndSwapDebug();
    outputArr();

    printf("--------With array from keyboard and debug function----------
\n");
    inputKeyboard();
    outputArr();
    findAndSwapDebug();
    outputArr();
}
```

REALESEANDDEBUG.H

```
#ifndef REALESEANDDEBUG_H_INCLUDED
#define REALESEANDDEBUG_H_INCLUDED

void realese();
void debug();


#endif // REALESEANDDEBUG_H_INCLUDED
```

TYPEDEF.C

```
#include "Typedef.h"
int** arr;
int n = 3;
```

TYPEDEF.H

```
#ifndef TYPEDEF_H_INCLUDED
#define TYPEDEF_H_INCLUDED

extern int n;
extern int** arr;


#endif // TYPEDEF_H_INCLUDED
```

MAKEFILE

```
.PHONY:build clean rebuild run greet debug realese

greet:
	@echo "Terminating make - please specify target explicitly"
	@echo "    build   : fast rebuild / build"
	@echo "    rebuild : full rebuild"
	@echo "    run     : run after fast rebuild / build"
	@echo "    clean   : perform full clean"
build: create

rebuild:clean create

run:build
	./lab1


debug: build
	./lab1 -d


realese: build
	./lab1 -r


clean:
	rm -rvf *.o lab1


main.o:main.c process.h realeseAndDebug.h memory.h
	gcc -c -o main.o main.c
process.o:process.c Typedef.h
	gcc -c -o process.o process.c
realeseAndDebug.o: realeseAndDebug.c input.h process.h pretty_print.h
	gcc -c -o realeseAndDebug.o realeseAndDebug.c
memory.o: memory.c Typedef.h
	gcc -c -o memory.o memory.c
```

```
input.o:input.c Typedef.h

	gcc -c -o input.o input.c

pretty_print.o:pretty_print.c Typedef.h

	gcc -c -o pretty_print.o pretty_print.c

Typedef.o:Typedef.c Typedef.h

	gcc -c -o Typedef.o Typedef.c

create:main.o process.o realeseAndDebug.o memory.o input.o pretty_print.o
Typedef.o

	gcc -o lab1 main.o process.o realeseAndDebug.o memory.o input.o
pretty_print.o Typedef.o
```

Команди для роботи з терміналом

gcc -o lab1 main.c input.c pretty_print.c process.c memory.c Typedef.c realeseAndDebug.c

./lab1

/lab1 -d -запуск дебаг режиму

/lab1 -r -n3 -запуск реліз режиму з кількістю n=3

Команди для роботи з Makefile

make build

make rebuild

make run

make debug

make realese

make cleane

Тест програми

```
student@virt-linux:~/codeblocks-workspace/lab1$ gcc -o lab1 main.c input.c pretty_print.c process.c memory.c Typedef.c realeseAndDebug.c
student@virt-linux:~/codeblocks-workspace/lab1$ ./lab1 -r -n7
                    REALESE
-----With random array and relese function--------
5       42      12      41      -7      4       43
1       34      17      45      -21     0       -46
7       -11     -19     -50     -48     10      2
39      6       16      41      -17     -35     -14
-8      7       -9      -1      -49     -46     -10
-5      -40     35      -2      44      3       -7
26      5       -50     35      44      33      36

5       42      12      41      -7      4       -48
1       34      17      45      -21     0       -46
7       -11     -19     -50     43      10      2
39      6       16      41      -17     -35     -14
-8      7       -9      -1      -49     -46     -10
-5      -40     35      -2      44      3       -7
26      5       -50     35      44      33      36

------With sorted array and release function------------
0       1       2       3       4       5       6
7       8       9       10      11      12      13
14      15      16      17      18      19      20
21      22      23      24      25      26      27
28      29      30      31      32      33      34
35      36      37      38      39      40      41
42      43      44      45      46      47      48

0       1       2       3       4       5       42
7       8       9       10      11      12      13
14      15      16      17      18      19      20
21      22      23      24      25      26      27
28      29      30      31      32      33      34
35      36      37      38      39      40      41
6       43      44      45      46      47      48
```

```
student@virt-linux:~/codeblocks-workspace/lab1$ make build
gcc -c -o main.o main.c
gcc -c -o process.o process.c
gcc -c -o realeseAndDebug.o realeseAndDebug.c
gcc -c -o memory.o memory.c
gcc -c -o input.o input.c
gcc -c -o pretty_print.o pretty_print.c
gcc -c -o Typedef.o Typedef.c
gcc -o lab1 main.o process.o realeseAndDebug.o memory.o input.o pretty_print.o T
ypedef.o
student@virt-linux:~/codeblocks-workspace/lab1$ make run
gcc -o lab1 main.o process.o realeseAndDebug.o memory.o input.o pretty_print.o Typedef.o
./lab1
                    REALESE
-----With random array and relese function--------
-2      -31     38
49      34      -3
-28     -24     22

-2      -31     -28
49      34      -3
38      -24     22
```