Національний технічний університет України

«Київський політехнічний інститут»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих

комп’ютерних систем

**ЛАБОРАТОРНА РОБОТА №2**

з дисципліни

**«Паралельні та розподілені обчислення”**

ТЕМА: «Робота з компіляторами мов С та Java в режимі командного

рядка»

Підготував: студент групи КВ-42

Сахнік Іван

Перевірив(ла):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Київ 2016**

**Варіант 18**

Задано матрицю дійсних чисел A[m,n]. При обході матриці по стовпчиках

визначити в ній присутність заданого дійсного числа X і його місцезнаходження.

**Постановка завдання для програми мовою C**

1. Написати програму розв’язання задачі пошуку (за варіантом) у

двовимірному масиві (матриці) одним з алгоритмів методу лінійного пошуку.

2. Розміри матриці m та n взяти самостійно у межах від 7 до 10.

3. Програма обов'язково повинна бути написана і структурована наступним

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. 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

повинна відбуватися лише лінковка;

c) забезпечити окрему ціль для очистки згенерованих файлів;

5. Вміти компілювати написану багатофайлову програму двома способами:

a) за допомогою однієї команди gcc;

b) за допомогою make-файлу.

6. 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.

**Постановка завдання для програми мовою Java**

1. Написати консольну програму розв’язання задачі пошуку (за варіантом) у

двовимірному масиві (матриці) одним з алгоритмів методу лінійного

пошуку.

2. Розміри матриці m та n взяти самостійно у межах від 7 до 10.

3. При написанні програми повинно бути щонайменше три класи, один із

яких буде відповідати за пошук елементу в матриці, другий відповідати за

ввід-вивід матриці, а третій — головний клас, що міститиме метод main.

4. Для компіляції та запуску написаної програми написати окремий

make-файл, причому забезпечити окремі цілі для очистки згенерованих

файлів, а також генерації JAR-архіву.

5. Вміти компілювати написану програму двома способами:

a) за допомогою однієї команди javac;

b) за допомогою make-файлу.

6. 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.

**Текст програми мовою С:**

#ifndef ARRAY\_H

#define ARRAY\_H

#include<stdio.h>

#include<stdlib.h>

#define N 5

#define M 5

int Random\_array(int \*\*array);

int Sort\_array(int \*\*array);

int Enter\_array(int \*\*array);

int Output\_array(int \*\*array);

void Search(int \*\*array, int value);

#endif //ARRAY\_H

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#include"array.h"

#include<time.h>

int Random\_array(int \*\*array)

{

int i,j;

srand(time(0));

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

for(j=0; j<N; j++)

array[i][j] = rand()%10;

return 0;

}

int Sort\_array(int \*\*array)

{

int i,j;

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

for(j=0; j<N; j++)

array[i][j] = j + i\*N;

return 0;

}

int Enter\_array(int \*\*array)

{

int i,j;

printf("Input array:\n");

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

for(j=0; j<N; j++)

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

return 0;

}

#include"array.h"

int Output\_array(int \*\*array)

{

int i,j;

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

{

for(j=0; j<N; j++)

printf("%4d ",array[i][j]);

printf("\n");

}

return 0;

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#include"array.h"

void Search(int \*\*array, int value)

{

int index\_i = -1, index\_j = -1;

int i, j;

for (j=0; j<N; j++)

{

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

if (array[i][j] == value)

{

index\_i = i;

index\_j = j;

break;

}

if (index\_i != -1 && index\_j != -1)

break;

}

if (index\_i != -1 && index\_j != -1)

{

index\_i += 1;

index\_j += 1;

printf("Element %d found at position %d-%d \n",value,index\_i,index\_j);

}

else

printf("Element %d not found \n",value);

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#include"array.h"

int main()

{

int \*\* array, i;

array = (int \*\*)malloc(sizeof(int \*)\*M);

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

array[i] = (int \*)malloc(sizeof(int)\*N);

Random\_array(array);

//Sort\_array(array);

//Enter\_array(array);

Output\_array(array);

Search(array,5);

return 0;

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

.PHONY: clean

Laba2: main.o input.o search.o output.o

gcc main.o input.o search.o output.o -o Laba2

search.o: search.c

gcc -c search.c -o search.o

input.o: input.c

gcc -c input.c -o input.o

output.o: output.c

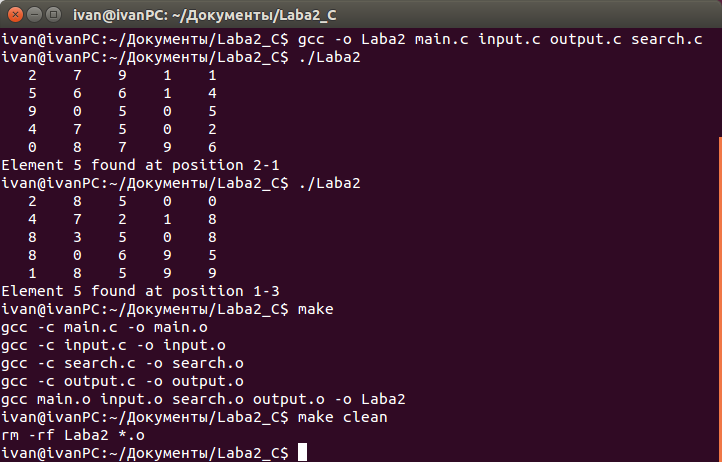
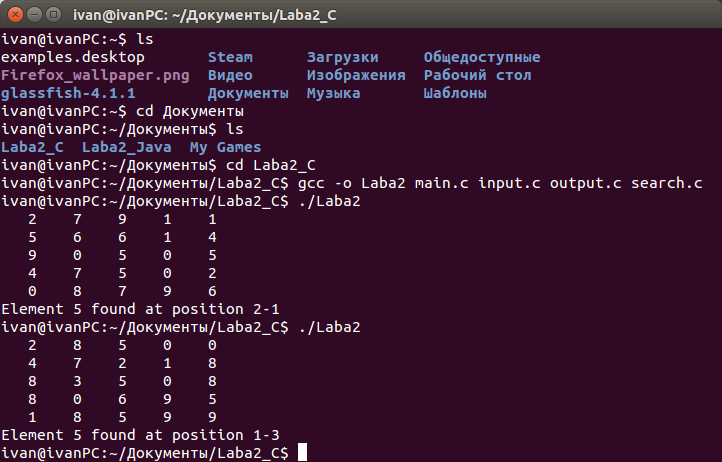
gcc -c output.c -o output.o

main.o: main.c

gcc -c main.c -o main.o

clean:

rm -rf Laba2 \*.o



**Текст програми мовою Java:**

import Inp.RandomArray;

import Out.OutputArray;

import Sh.Search;

public class Laba2{

public static void main(String[] args){

int m=5;

int n=5;

int array[][] = new int [m][n];

RandomArray.Input\_array(array,m,n);

OutputArray.Output\_array(array,m,n);

Search.Search\_value(array,m,n,5);

}

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

package Inp;

import java.util.Random;

public class RandomArray{

public static void Input\_array(int array[][], int m, int n){

Random generator = new Random();

int gn;

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

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

gn=generator.nextInt(10);

array[i][j] = gn;

}

}

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

package Out;

public class OutputArray{

public static void Output\_array(int array[][], int m, int n){

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

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

System.out.print(array[i][j]+" ");

System.out.println();

}

}

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

package Sh;

public class Search{

public static void Search\_value(int array[][], int m, int n, int value){

int index\_i = -1, index\_j = -1;

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

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

if (array[i][j] == value){

index\_i = i;

index\_j = j;

break;

}

if (index\_i != -1 || index\_j != -1)

break;

}

if (index\_i != -1 || index\_j != -1){

index\_i += 1;

index\_j += 1;

System.out.println("Element "+value+" found at position "+index\_i+"-"+index\_j);

}

else

System.out.println("Element "+value+" not found");

}

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

.PHONY: clean

Laba2.class: Laba2.java Inp/RandomArray.class Sh/Search.class Out/OutputArray.class

javac Laba2.java

Inp/RandomArray.class: RandomArray.java

javac -d . RandomArray.java

Out/OutputArray.class: OutputArray.java

javac -d . OutputArray.java

Sh/Search.class: Search.java

javac -d . Search.java

generateJar:

jar -cfm Laba2.jar main.manifest Laba2.class Inp/RandomArray.class Out/OutputArray.class Sh/Search.class

clean:

rm -rf \*.class

rm -r Out Sh Inp Laba2.jar

