

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

#### Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №3**

з дисципліни

**«Паралельні та розподілені обчислення»**

#### Тема «Робота з компіляторами мови С в режимі командного рядка»

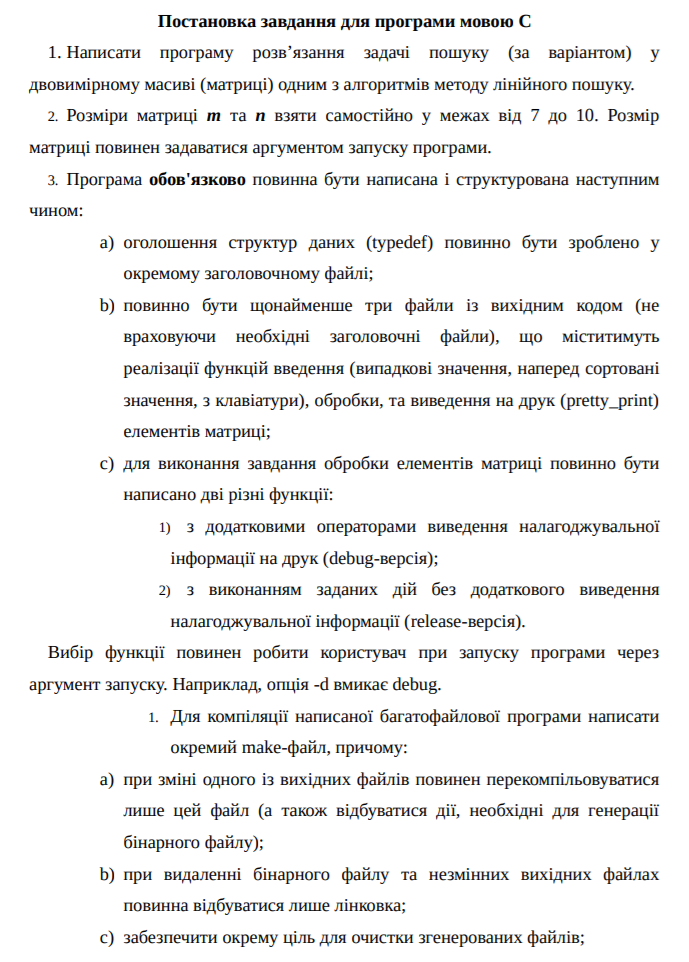
Виконав: студент ІІ курсу

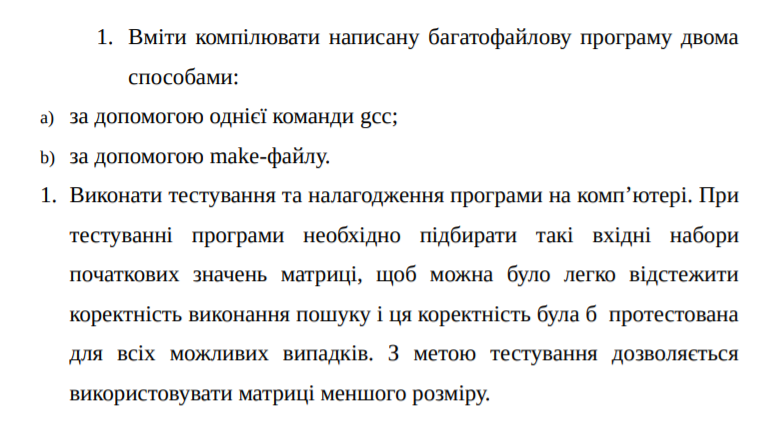
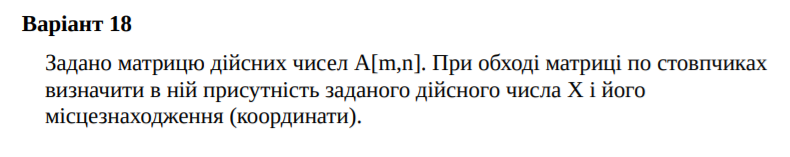
ФПМ групи КВ-91

Селетков В. Р.

Перевірив: Марченко О. О.

Київ – 2021





**Текст програми мовою С**

**main.c**

#include "input.h"

#include "output.h"

#include "data.h"

#include "func.h"

#include <ctype.h>

int main (int argc, char\*\* argv) {

int j = 0, key = 0, flag = 1;

n = -1, m = -1;

for (int i = 1; i < argc; i++) {

flag = 1;

if ((argv[i][0] == '-' && argv[i][1] == 'n') || (argv[i][0] == '-' && argv[i][1] == 'm')) {

j = 0;

while (argv[i + 1][j] != '\0') {

if (!isdigit(argv[i + 1][j])) {

flag = 0;

break;

}

j++;

}

if (atoi(argv[i + 1]) >= 0 && flag == 1) {

if (argv[i][1] == 'n')

n = atoi(argv[i + 1]);

else if (argv[i][1] == 'm')

m = atoi(argv[i + 1]);

}

else {

printf("\nerror size\n");

exit(1);

}

i++;

}

else if (argv[i][0] == '-' && argv[i][1] == 'd')

key = 1;

}

if (n == -1 || m == -1) {

printf("\nerror arguments\n");

exit(1);

}

if (key == 1) {

printf("\n\t\t\tdebug version\n");

create();

random\_input();

printf("\nrandom matrix debug\n");

output();

debug();

sorted\_input();

printf("\nsorted matrix debug\n");

output();

debug();

user\_input();

printf("\nuser matrix debug\n");

output();

debug();

delete();

printf("\n\t\t debug version finished\n");

}

else {

printf("\n\t\t\trelease version\n");

create();

random\_input();

printf("\nrandom matrix release\n");

output();

release();

sorted\_input();

printf("\nsorted matrix release\n");

output();

release();

user\_input();

printf("\nuser matrix release\n");

output();

release();

delete();

printf("\n\t\t release version finished\n");

}

return 0;

}

**data.h**

#ifndef DATA\_H

#define DATA\_H

#include <stdio.h>

#include <stdlib.h>

int n;

int m;

int m\_x;

int m\_y;

float \*\*A;

float x;

void create();

void delete();

#endif //DATA\_H

**data.c**

#include "data.h"

void create() {

A = (float\*\*)malloc(sizeof(float\*) \* m);

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

A[i] = (float\*)malloc(sizeof(float) \* n);

}

void delete() {

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

free (A[i]);

free(A);

}

**func.h**

#ifndef FUNC\_H

#define FUNC\_H

void release();

void debug();

#endif //FUNC\_H

**func.c**

#include "func.h"

#include "data.h"

void release() {

printf("\n\n\t\t\trelease function\n");

m\_x = -1, m\_y = -1, x = 3.0;

int flag = 0;

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

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

if (A[j][i] == x) {

m\_y = j;

m\_x = i;

flag = 1;

break;

}

if (flag == 1)

break;

}

if (m\_x == -1 && m\_y == -1)

printf("\nx = %.1f, wasn't found\n\n\t\trelease function finished\n\n", x);

else

printf("\nx = %.1f, was found\ncoordinates: m\_x = %d, m\_y = %d\n\n\t\trelease function finished\n\n", x, m\_x, m\_y);

}

void debug() {

printf("\n\n\t\t\tdebug function\n");

m\_x = -1, m\_y = -1, x = 3.0;

int flag = 0;

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

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

if (A[j][i] == x) {

m\_y = j;

m\_x = i;

flag = 1;

break;

}

if (flag == 1)

break;

}

if (m\_x == -1 && m\_y == -1)

printf("\nx = %.1f, wasn't found\n\n\t\tdebug function finished\n\n", x);

else

printf("\nx = %.1f, was found\ncoordinates: m\_x = %d, m\_y = %d\n\n\t\tdebug function finished\n\n", x, m\_x, m\_y);

}

**input.h**

#ifndef INPUT\_H

#define INPUT\_H

void random\_input();

void sorted\_input();

void user\_input();

#endif //INPUT\_H

**input.c**

#include "input.h"

#include "data.h"

void random\_input() {

srand(time(NULL));

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

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

A[j][i] = (-500 + rand() % 1000) / 10.0;

}

void sorted\_input() {

srand(time(NULL));

float counter = 0.0;

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

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

counter = (((i + 1) \* (j + 1) \* n) / (i + j + 1 + (rand() % 10))) + counter;

A[j][i] = counter;

}

}

void user\_input() {

printf("Input matrix:\n\n");

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

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

scanf("%f", &A[j][i]);

printf("\n");

}

}

**output.h**

#ifndef OUTPUT\_H

#define OUTPUT\_H

void output();

#endif //OUTPUT\_H

**output.c**

#include "output.h"

#include "data.h"

void output() {

printf("\n");

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

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

printf("%8.1f", A[i][j]);

printf("\n");

}

}

**Командні рядки для компілювання та запуску програми мовою C**

Компілювання:

gcc -o lab3 data.c func.c input.c main.c output.c

Запуск:

./lab3 -m 7 -n 8

**Make-файл для компілювання та запуску програми мовою C**

**makefile**

.PHONY: greet build lab3 rebuild clean\_all clean\_obj clean\_lab3

greet:

@echo "chose your option:"

@echo "<==build==> : build project"

@echo "<==rebuild==> : clean / build / rebuild project"

@echo "<==clean\_all==> : clean all files"

@echo "<==clean\_obj==> : clean obj"

@echo "<==clean\_lab3==> : clean lab3"

@echo "<==lab3==> : link project"

build: lab3

./lab3 -m 7 -n 8

rebuild: clean\_all lab3

./lab3 -d -n 7 -m 8

clean\_all: clean\_obj clean\_lab3

clean\_obj:

rm -rvf \*.o

clean\_lab3:

rm -rvf lab3

lab3: output.o main.o func.o data.o input.o

gcc -o lab3 output.o main.o func.o data.o input.o

output.o: output.c output.h

gcc -c -o output.o output.c

main.o: main.c

gcc -c -o main.o main.c

input.o: input.c input.h

gcc -c -o input.o input.c

func.o: func.c func.h

gcc -c -o func.o func.c

data.o: data.c data.h

gcc -c -o data.o data.c

**Тестування програми**

