Лабораторна робота №7. Функції

**Автор:** Еткало Сергій Ігорович

**Група:** КН 922Б

**Завдання:**

**1.**Переробити програми, що були розроблені під час виконання лабораторних робіт з тем "Масиви" та "Цикли" таким чином, щоб використовувалися функції для обчислення результату.

**2.**Функції повинні задовольняти основну їх причетність - уникати дублювання коду.

Тому, для демонстрації роботи, ваша програма (функція main()) повинна мати можливість викликати розроблену функцію з різними вхідними даними.

**3.**Слід звернути увагу: параметри одного з викликів функції повинні бути згенеровані за допомогою генератора псевдовипадкових чисел random().

**4.**Слід звернути увагу (#2): продемонструвати встановлення вхідних даних через аргументи додатка (параметри командної строки).

Обробити випадок, коли дані не передались - у цьому випадку вони матимуть значення за умовчуванням, обраними розробником.

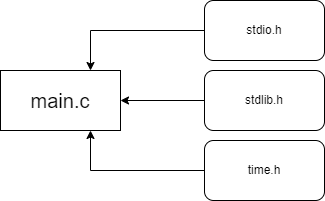
**Опис програми**

***Функціональне призначення***

Ця програма виконує дві операції.

* Множить матрицю саму на себе

***Опис логічної структури***



*(Рис. 1) Графічна структура програми*

***Файл "main.c"***

Головний файл

Це файл, який містить точку входу main, функції squareMatrix та primenumber .

## main(int argc, char \*argv[])

**Аргументи**

argc - зберігаеє кількість введених значень

argv - зберігае самі значення

**Послідовність дій**

* Присвоїти значення аргументам argc і argv, значення цих аргументів ми передаємо у функцію squareMatrix .
* Виклик функції squareMatrix, у параметрах цієї функції при виклику вказуємо аргументи argc і argv.
* Викликаємо функцію primenumber, у параметрах цієї функції при виклику нічого не ваказуємо.

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

*squareMatrix (argc,argv);*

*primenumber ();*

*}*

Ця функція множить матрицю саму на себе:

void squareMatrix (int argc,char \*argv[])

**Аргументи**

argv - зберігає кількість введених значень у командний рядок, та використовується у перевірці.

args - зберігає значення введені у командний рядок які потім використовуються у множенні матриці.

**Послідовність дій**

* Створення змінних a[10][10], MAT[10][10], b, c, i, j, f, z = 2, t, \*N.
  1. *a[10][10] - квадратна матриця, що містить межу 10 рядків і стовпців*
  2. *MAT[10][10] - квадратна матриця, що містить розрахунок матриці а\*а.*
  3. *b і c - змінні що містять у собі кількість рядків і стовпців.*
  4. *t - зберігає значення за умовчанням.*
  5. *i та j - кількість стовпців і рядків матриці, які порівнюються між заданими b та c, та якщо виконується умова вони збільшуються.*
  6. *t - використовується у перетворення значень рядка у число.*
* Перевіряємо чи були введені якісь значення у командний рядок.
* Якщо перевірка була пройдена то перетворюємо перше значення командного рядка у число та присвоюємо змінній t. Після чого присвоюємо змінну t змінним c та b, і запускаємо два цикли у яких записуємо ці значення у матрицю *a*.

if (argc > 1)

{

t = strtol (argv[1], &N, 10);

b = t;

c = t;

for (i = 0; i <= b - 1; i++)

{

for (j = 0; j <= c - 1; j++)

{

a[i][j] = strtol (argv[у], &N, 10);

у++;

}

}

}

* Якщо перевірка не була пройдена то присвоюємо змінну у змінним b та c.

else{

b = у;

c = у;

for (i = 0; i <= b - 1; i++) {

for (j = 0; j <= c - 1; j++){

a[i][j] = strtol (argv[у], &N, 10);

у++;

}

}

}

* Тепер для множення матриці самої на себе запускаємо цикли

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

{

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

{

MAT[i][j] = 0;

for (f = 0; f < c; f++)

{

MAT[i][j] += a[i][f] \* a[f][j];

}

}

}

* Після множення матриці саму на себе виводимо результат. Для цього потрібно запустити 2 цикли які перебирають значення функції *MAT,* та потім використовуючи функцію *printf* виводимо значення на екран.

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

{

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

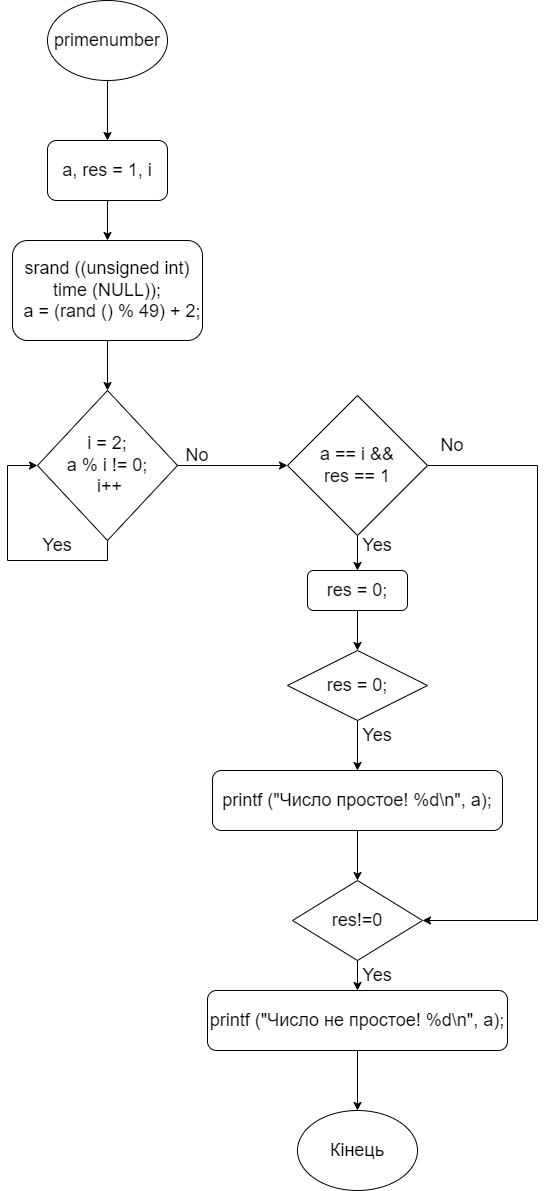
{

printf ("%ld\t", MAT[i][j]);

}

printf ("\n");

}



*(Рис. 2) Схема алгоритму функції lab06*

***Структура проекту лабораторної роботи:***

├── lab07

├── Makefile

├── README.md

└── src

└── main.c

**Висновки:** У цій роботі було набуто навичок роботи з функціями, їх декларація, реалізація та виклик, а також підвищив свій рівень вкористання циклів та умов. Під час тестування програми були отримані результати функції squareMatrix - це множення матриці саму на себе, і робота функції primenumber - це отримання випадково згенерованого числа та перевірка на те просте воно чи ні .