**Лабораторна робота 6 (С2)**
**Застосування масивів та функцій**

**Ціль:** Практика в організації ітераційних та арифметичних циклів. Здобуття навичок обробки масивів з використанням функцій. Організація динамічних масивів.

***Робота складається з двох частин***

**1 частина роботи**

**2 частина роботи**

**Приклад виконання перша частина(С) тут**

**Приклад виконання друга частина(С) тут**

# 1 частина роботи. Постановка задачі

## "Обчислення функцій з використанням їх розкладання в статечний ряд"

**Ціль:**Практика в організації ітераційних та арифметичних циклів, використання функцій.

**Постановка задачі**

Для змінюється від a до b з кроком (ba)/k, де (k=10), обчислити функцію f(x), використовуючи її розкладання в статечний ряд у трьох випадках:

а) для заданого n;

б) для заданої точності ε(ε=0.0001);

с) для "точного" значення (за аналітичною формулою).
Для порівняння знайти відносну похибку обчислення функції значення функції про_погр = ABS( (точ_знач – наближ_знач)/ точ_знач)

**Програма повинна містити три функції користувача, що виконують розрахунок значення Y трьома способами.**

**Варіанти**

| № | функція | діапазон зміни аргументу | n | сума |
|---|---|---|---|---|
| 1 | $y = 3^X$ | $0{,}1 \le x \le 1$ | 10 | $S = 1 + \frac{\ln 3}{1!}x + \frac{\ln^2 3}{2!}x^2 + \ldots + \frac{\ln^n 3}{n!}x^n$ |
| 2 | $y = -\ln\left\|2\sin\frac{x}{2}\right\|$ | $\frac{\pi}{5} \le x \le \frac{9\pi}{5}$ | 40 | $S = \cos x + \frac{\cos 2x}{2} + \ldots + \frac{\cos nx}{n}$ |
| 3 | $y = \sin X$ | $0{,}1 \le x \le 1$ | 10 | $S = x - \frac{x^3}{3!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$ |
| 4 | $y = XarctgX - \ln\sqrt{1+x^2}$ | $0{,}1 \le x \le 0{,}8$ | 10 | $S = \frac{x^2}{2} - \frac{x^4}{12} + \ldots + (-1)^{n+1}\frac{x^{2n}}{2n(2n-1)}$ |
| 5 | $y = e^x$ | $1 \le x \le 2$ | 15 | $S = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!}$ |
| 6 | $y = e^{x\cos\pi/4} \cdot \cos(x\sin\pi/4)$ | $0{,}1 \le x \le 1$ | 25 | $S = 1 + \frac{\cos\frac{\pi}{4}}{1!}x + \ldots + \frac{\cos n\frac{\pi}{4}}{n!}x^n$ |
| 7 | $y = \cos x$ | $0{,}1 \le x \le 1$ | 10 | $S = 1 - \frac{x^2}{2!} + \ldots + (-1)^n \frac{x^{2n}}{(2n)!}$ |
| 8 | $y = \frac{x\sin\pi/4}{1 - 2x\cos\pi/4 + x^2}$ | $0{,}1 \le x \le 0{,}8$ | 40 | $S = x\sin\frac{\pi}{4} + x^2\sin 2\frac{\pi}{4} + \ldots + x^n \sin n\frac{\pi}{4}$ |
| 9 | $y = \frac{1}{4}\ln\frac{1+x}{1-x} + \frac{1}{2}arctgX$ | $0{,}1 \le x \le 0{,}8$ | 3 | $S = x + \frac{x^5}{5} + \ldots + \frac{x^{4n+1}}{4n+1}$ |

| | | | | |
|---|---|---|---|---|
| 10 | $y = e^{\cos x}\cos(\sin x)$ | $0{,}1 \le x \le 1$ | 20 | $S = 1 + \frac{\cos x}{1!} + \ldots + \frac{\cos nx}{n!}$ |
| 11 | $y = (1 + 2x^2)e^{x^2}$ | $0{,}1 \le x \le 1$ | 10 | $S = 1 + 3x^2 + \ldots + \frac{2n+1}{n!}x^{2n}$ |
| 12 | $y = -\frac{1}{2}\ln(1 - 2x\cos\frac{\pi}{3} + x^2)$ | $0{,}1 \le x \le 0{,}8$ | 35 | $S = \frac{x\cos\frac{\pi}{3}}{1} + \frac{x^2\cos 2\frac{\pi}{3}}{2} + \ldots + \frac{x^n\cos n\frac{\pi}{3}}{n}$ |
| 13 | $y = \frac{1}{2}\ln x$ | $0{,}2 \le x \le 1$ | 10 | $S = \frac{x-1}{x+1} + \frac{1}{3}(\frac{x-1}{x+1})^3 + \ldots + \frac{1}{2n+1}(\frac{x-1}{x+1})^{2n+1}$ |
| 14 | $y = \frac{1}{4}(x^2 - \frac{\pi^2}{3})$ | $\frac{\pi}{5} \le x \le \pi$ | 20 | $S = -\cos x + \frac{\cos 2x}{2^2} + \ldots + (-1)^n\frac{\cos nx}{n^2}$ |
| 15 | $y = \frac{1+x^2}{2}arctgX - \frac{x}{2}$ | $0{,}1 \le x \le 1$ | 30 | $S = \frac{x^3}{3} - \frac{x^5}{15} + \ldots + (-1)^{n+1}\frac{x^{2n+1}}{4n^2-1}$ |
| 16 | $y = \frac{\pi^2}{8} - \frac{\pi}{4}\lvert x\rvert$ | $\frac{\pi}{5} \le x \le \pi$ | 40 | $S = \cos x + \frac{\cos 3x}{3^2} + \ldots + \frac{\cos(2n-1)x}{(2n-1)^2}$ |
| 17 | $y = \frac{e^x + e^{-x}}{2}$ | $0{,}1 \le x \le 1$ | 10 | $S = 1 + \frac{x^2}{2!} + \ldots + \frac{x^{2n}}{(2n)!}$ |
| 18 | $y = \frac{1}{2} - \frac{\pi}{4}\lvert\sin x\rvert$ | $0{,}1 \le x \le 0{,}8$ | 50 | $S = \frac{\cos 2x}{3} + \frac{\cos 4x}{15} + \ldots + \frac{\cos 2nx}{4n^2-1}$ |
| 19 | $y = e^{2x}$ | $0{,}1 \le x \le 1$ | 20 | $S = 1 + \frac{2x}{1!} + \ldots + \frac{(2x)^n}{n!}$ |
| 20 | $y = (\frac{x^2}{4} + \frac{x}{2} + 1)e^{x/2}$ | $0{,}1 \le x \le 1$ | 30 | $S = 1 + 2\frac{x}{2} + \ldots + \frac{n^2+1}{n!}(\frac{x}{2})^n$ |
| 21 | $y = arctgX$ | $0{,}1 \le x \le 1$ | 40 | $S = x - \frac{x^3}{3} + \ldots (-1)^n\frac{x^{2n+1}}{2n+1}$ |
| 22 | $y = (1 - \frac{x^2}{2})\cos x - \frac{x}{2}\sin x$ | $0{,}1 \le x \le 1$ | 35 | $S = 1 - \frac{3}{2}x^2 + \ldots + (-1)^n\frac{2n^2+1}{(2n)!}x^{2n}$ |
| 23 | $y = 2(\cos^2 x - 1)$ | $0{,}1 \le x \le 1$ | 15 | $S = -\frac{(2x)^2}{2} + \frac{(2x)^4}{24} + \ldots + (-1)^n\frac{(2x)^{2n}}{(2n)!}$ |

| | | | | |
|---|---|---|---|---|
| 24 | $y = \ln(\frac{1}{2+2x+x^2})$ | $-2 \le x \le -0{,}1$ | 40 | $S = -(1+x)^2 + \frac{(1+x)^4}{2} + \ldots + (-1)^n \frac{(1+x)^{2n}}{n}$ |
| 25 | $y = \frac{e^x - e^{-x}}{2}$ | $0{,}1 \le x \le 1$ | 20 | $S = x + \frac{x^3}{3!} + \ldots + \frac{x^{2n+1}}{(2n+1)!}$ |

**Методичні вказівки**

1. Алгоритм розв'язання задачі зводиться до циклу змінної x. У тілі циклу викликаються три функції – дві і комбінація бібліотечних (результат їх роботи вважається умовно точним значенням). Необхідно спроектувати та реалізувати користувацькі функції розрахунку F(x) = S за двома вказаними вище алгоритмами.

2. Результати розрахунків надрукувати у вигляді таблички:

```
Обчислення функці
   X      Y      Y1              Y2          погр1 погр2
:::::: ::::     .....           .....       ..... .....
                .....           .....       ..... .....
...... ....     .....           .....       ..... .....
...... ....     .....           .....       ..... .....
```

Тут X значення параметра; Y1 значення суми для заданого n; Y2 значення суми для заданої точності; Y-точне значення функції; погр1, погр2 - відносні похибки наближених обчислень.

**Зміст звіту**

1. Постановка задачі.
2. Варіант завдання.
3. Математична модель (формули, якими виконуються обчислення доданків ряду).
4. програма.
5. Отримані результати.

# 2 частина роботи."Функції та масиви"

**Ціль:**Організувати обробку масивів із використанням функцій, навчитися передавати масиви як параметри функцій.

### Постановка задачі

Використовуючи функції, вирішити вказане у варіанті завдання. Масив повинен передаватися у функцію як параметр.

### Варіанти

1. Визначити скільки елементів двовимірного масиву менше за будь-який елемент на головній діагоналі.
2. Написати функцію для обміну рядків двовимірного масиву за допомогою її відсортувати масив за елементами третього стовпця.
3. Написати процедуру для підсумовування матриць. З її допомогою скласти вихідну матрицю та транспоновану (тобто отриману поворотом вихідної на 90°).
4. Написати функцію для видалення рядка із двовимірного масиву. Рядки, що залишилися, повинні бути розташовані щільно, відсутні елементи замінюються 0. За допомогою розроблених функцій виключити з масиву рядки з номерами від A до B.
5. Визначити чи є матриця ортонормованою, тобто такою, що скалярний добуток кожної пари різних рядків дорівнює 0, а скалярний добуток рядка самої на себе дорівнює 1.
6. Елемент матриці є сідловою точкою, якщо він є найменшим у своєму рядку і найбільшим у своєму стовпці (або навпаки: найбільшим у своєму рядку і найменшим у своєму стовпці). Для заданої матриці визначити всі сідлові точки.
7. Написати процедуру обміну стовпця та рядки двомірного масиву. З її допомогою поміняти місцями ті рядки та стовпці, перші елементи яких збігаються.
8. Написати функцію транспонування квадратної матриці (тобто повороту вихідної матриці на 90°). З її допомогою визначити, чи є задана матриця симетричною. (Матриця називається симетричною, якщо транспонована матриця дорівнює вихідній).
9. Написати функцію для обчислення суми елементів квадратної матриці, які розташовані нижче за головну діагональ. З її допомогою знайти максимальне значення такої суми у n матрицях.

10. Написати функцію, яка перевіряє чи є негативні елементи у зазначеному рядку двовимірного масиву. Видалити з масиву всі рядки з негативними елементами, віддалений рядок заповнюється 0 і переноситься на кінець масиву.

11. Написати функцію, яка перевіряє за зростанням або зменшенням упорядкований зазначений рядок двовимірного масиву. Упорядкувати за зростанням усі рядки двовимірного масиву, які невпорядковані за спаданням.

12. Написати функцію для пошуку максимального елемента у зазначеному рядку двовимірного масиву. Зрушити в двовимірному масиві всі рядки циклічно вправо на кількість елементів, що дорівнює максимальному елементу в цьому рядку.

13. Визначити чи можна у двовимірному масиві знайти такий стовпець, який розбиває масив на два так, що сума елементів у першому більша, ніж сума елементів у другому. Сам стовпець у частини, що розбиваються, не входить.

14. Обчислити добуток всіх стовпців масиву, у яких перший елемент більше елементів розташованих на головній та побічній діагоналі.

15. Заданий двовимірний масив. Знайти суму елементів першого шпальти без одного останнього елемента, суму елементів другого шпальти без двох останніх, суму елементів третього шпальти без трьох останніх і т. д. Останній шпальт не обробляється. Серед знайдених сум знайти максимальну.

16. Заданий двовимірний масив N x N. Дозволяється довільно переставляти елементи всередині будь-якого стовпця. Перевірити, чи можна виконавши кінцеву кількість перестановок у стовпцях, розташувати на побічній діагоналі елементи так, щоб він зростав.

17. Заданий двовимірний масив N x M. Знайти в ньому підмасив 3 x 3 сума елементів якого максимальна. N та M можуть бути не кратні трьом.

18. Заданий двовимірний масив N x N. Послідовно розглядаються квадратні під масиви, верхній верхній елемент яких лежить на побічній діагоналі. У кожному під масиві знаходиться максимальний елемент. Шляхом перестановок рядків і стовпців (цілком) елемент треба перемістити в правий верхній кут підмасиву. Перевірити чи вийшла на побічній діагоналі спадна послідовність елементів.

19. Задано рядок із $N^2$ цифр. Чи можна встановити, розбивши рядок на підрядки довжиною N, записати їх у рядки двовимірного масиву N x N по одній цифрі в одному елементі так, щоб вони в першому стовпці розташувалися в порядку зростання.

20. Знайти мінімальний із повторюваних елементів двовимірного масиву.

21. Знайти максимальний з елементів двомірного масиву, що повторюються.

22. У двовимірному масиві знайти середнє арифметичне першого стовпця та кількість елементів у кожному з наступних стовпців, що перевищують середнє арифметичне попереднього стовпця.

23. Заданий одновимірний масив, що складається з N цілих чисел. Сформувати на його основі двовимірний масив N x N так, щоб сума елементів у першому стовпці дорівнювала першому елементу одномірного масиву, сума елементів у другому стовпці дорівнювала другому елементу одномірного масиву і т. д. Нулі не використовувати.

24. Визначити скільки елементів двовимірного масиву більше за мінімальний елемент на головній діагоналі.

25. Визначити, скільки елементів двовимірного масиву менше максимального елемента на головній діагоналі.

## Методичні вказівки

Алгоритм розв'язання багатьох задач зводиться до розробки необхідної за умовою задачі функції та її багаторазового застосування для ряду стовпців або рядків.
***Рекомендується користуватися динамічними масивами.***

Рекомендується також розробити функцію виведення на консоль значень елементів масиву і використовувати її для візуалізації значень до і після перетворення масивів.

Значення елементів масиву можна встановити з використанням функції отримання випадкових чисел – random чи rand.

Докладніше про статичні масиви можна подивитися **тут**.

Детальніше про динамічні масиви можна подивитися **тут.**

## Зміст звіту

1. Постановка задачі.
2. Варіант завдання
3. Текст програми.
4. Результат розв'язання конкретного варіанта.

# Приклад рішення лабораторної роботи № 2(С)

```
   /**
     1 частина роботи. Постановка задачі
    "Обчислення функцій з використанням їх розкладання в статечний ряд"
    Постановка задачі
      Для x, що змінюється від a до b з кроком (ba)/k,
      де (k=10), обчислити функцію f(x),
      використовуючи її розкладання в статечний ряд у трьох випадках:
     а) для заданого n;
     б) для заданої точності e(e=0.0001);
     с) для "точного" значення (за бібліотечними функціями).
     Для порівняння знайти відносну похибку обчислення функції значення функції
      o_погр = ABS((точ_знач – наближ_знач)/ точ_знач)
     Програма повинна містити три функції користувача,
     виконують розрахунок значення У трьома способами.

      y = (exp (x) - exp (-x)) / 2
      0.1 <= x <= 1.0
      k=20

     an = x ** (2 * n + 1) / (2 * n + 1)!
     a_нове=a_старе * x**2/( 2*(2n+3)*(n+1) )
     n починається з НУЛЯ)
    **/

// додаємо до "початкового коду" програми ПРОТОТИПИ бібліотечних функцій
#include <iostream>
#include <stdio.h>
#include <math.h>
using namespace std; // використовуємо "стандартний простір імен" за умовчанням
/* Прототипи функцій користувача */
double f1(double x); // розраховує значення f(x) через набір функцій БІБЛІОЧКИ
double f2(double x, int n); // Розраховує значення f(x)
```

```
                                  // через ряд Маклорена з фіксованою кількістю доданків n
double f3(double x, double eps); // Розраховує значення f(x)
                                   // через ряд Маклорена із заданою точністю eps
// Головна програма
int main(int argc, char *argv[], char *env[]) { // "загальний вигляд" заголовка ГОЛОВНОЇ
процедури


    cout << "Привіт із кодеблоку" << endl;
    printf("Робота № 2 Варіант № 25\n");
    double a=0.1L, b=1.0L; // інтервал аргументу x
    int ktoch=10; // кількість точок значень функцій на відрізку [a, b]
    int n=20; // кільк. доданків при розрахунку через ряд Маклорена з фіксованою кількістю
доданків
    double x // значення аргумен
           y1, // значення f(x) через набір функцій БІБЛІОЧКИ - умовно ТОЧНЕ значення
           y2, // значення f(x) через ряд Маклорена з фіксованою кількістю доданків n
           y3 // значення f(x)через ряд Маклорена із заданою точністю eps
           er1, // відносна похибка y % між y1 та y2
           er2; // відносна похибка y % між y1 та y3
     double eps=1e-4L; //задана точністю eps
     double h=(ba)/(double) ktoch; //Крок зміна аргументу
    // Висновок заголовка таблиці
    printf("+-------------------------------------------------------------------------
+\n");
    printf("| x | y1 | y2 | y3 | er1 \%% | er2 \%% |\n");
    printf("+-------------------------------------------------------------------------
+\n");
    x=a;
    while ( x <= b) {
       y1 = f1 (x);
       y2=f2(x,n);
       y3=f3(x, eps);
       er1 = fabs ((y1-y2) / y1) * 100.0L;
       er2 = fabs ((y1-y3) / y1) * 100.0L;
```

```
      printf("| %6.2g | %12.5g | %12.5g | %12.5g | %8.2g | %8.2g |\n", x, y1, y2, y3, er1,
er2);
      x+=h;
    };
    printf("+-------------------------------------------------------------------------------
+\n");
    return 0;
};
double f1(double x) {
// розраховує значення f(x) через набір функцій БІБЛІОЧКИ
  return (double) (exp (x) - exp (-x)) / 2.0;
};
double f2(double x, int kmax) {
  double a, sum = 0.0L;
  a = x; // значення 1-го доданку при k=0
  sum+=a;
  for(int i=1;i<= kmax; i++){
    a=a*( (x*x)/(double) ( 2*( 2*i+3)*(i+1) ) );
    sum+=a;
  };
  return sum;
};
double f3(double x, double e){
  double a, sum = 0.0L;
  int k=0;
  a = x; // значення 1-го доданку при k=0
  sum+=a;
  while( fabs(a) > e) {
    a=a*( (x*x)/(double) ( 2*( 2*k+3)*(k+1) ) );
    sum+=a;
    k++;
  };
  return sum;
}
```

**Результат**

```
Привіт із кодеблоку
Робота №2 Варіант №25
+-------------------------------------------------------------------------+
| x | y1 | y2 | y3 | er1% | er2% |
+-------------------------------------------------------------------------+
| 0,1  | 0,10017 | 0,10005 | 0,10017 | 0,12 | 2e-008  |
| 0,19 | 0,19115 | 0,19034 | 0,19115 | 0,42 | 9,3e-007 |
| 0,28 | 0,28367 |  0,2811 | 0,28367 | 0,91 | 9,4e-006 |
| 0,37 | 0,3785  | 0,37254 | 0,3785  | 1,6  | 5e-005   |
| 0,46 | 0,4764  | 0,46489 | 0,4764  | 2,4  | 5,3e-007 |
| 0,55 | 0,57815 | 0,55838 | 0,57815 | 3,4  | 2,2e-006 |
| 0,64 | 0,68459 | 0,65324 | 0,68459 | 4,6  | 7,3e-006 |
| 0,73 | 0,79659 | 0,7497  | 0,79659 | 5,9  | 2e-005   |
| 0,82 | 0,91503 | 0,84801 | 0,91503 | 7,3  | 5,1e-005 |
| 0,91 | 1,0409  | 0,94843 | 1,0409  | 8,9  | 8,6e-007 |
| 1    | 1,1752  | 1,0512  | 1,1752  | 11   | 2,1e-006 |
+-------------------------------------------------------------------------+

Process returned 0 (0x0) execution time : 0.050 s
Press any key to continue.
```

---

## Приклад рішення лабораторної роботи № 2(С) другої частини

```
  /**
     2 частина роботи. Постановка задачі
25. Визначити скільки елементів двовимірного масиву менше максимального елемента на головній
діагоналі.
    **/

// додаємо до "початкового коду" програми ПРОТОТИПИ бібліотечних функцій
```

```
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <ctime>
#include <cstdlib>
using namespace std; // використовуємо "стандартний простір імен" за умовчанням
// Головна програма
int main(int argc, char *argv[], char *env[]) {// "загальний вигляд" заголовка ГОЛОВНОЇ
процедури


    cout << "Привіт із кодеблоку" << endl;
    printf("Робота № 2-2 Варіант № 25\n");
    printf("Визначити скільки елементів двовимірного масиву \nменше максимального елемента на
головній діагоналі\n");
    /* визначимося з вихідними даними
      використовуватимемо динамічний масив
      розмірності nxm
      За умовою завдання передбачається, що квадратна матриця n=m
      Значення елементів "для налагодження" задаватимемо
      набором випадкових цілих чисел
    */
    int n, m; // Розмірність матриці
    // int i, j; // поточні індекси
    int maxd; // максимальний елемент головної діагоналі
    int krez=0; // кількість елементів двовимірного масиву менша за максимальний елемент на
головній діагоналі
    int * * mat = NULL; // покажчик на матрицю
    printf("Введіть розмірність матриці (ціле число):");
    scanf("%d", &n);
    if (n < 0) {
        printf("Помилка значення розмірності матриці... n=%d\nДосвідання....\n", n);
        return -1;
    }
    m=n;
```

```
    // "Створимо матрицю" виділимо під неї ВП
    mat = (int * *) calloc (sizeof (int *), n); // покажчик на вектор покажчиків int *
    if ( mat == NULL ) {
           printf("*** Помічено до об'єднаної пам'яті для мат => %f Kb\n",
                     (float) (sizeof (int *) * n) / 1024.0);
           return -1;
     };
     for(int ii=0; ii < n ; ii++) mat[ii]=NULL; // Вектор ініціалізували NULL - це не
обов'язково
     for(int ii=0; ii < n ; ii++) { // у кожен елемент вектора запишемо адресу "рядки" матриці
          mat[ii]=( int *) calloc( sizeof(int), m );
          if ( mat[ii] == NULL ) {
               printf("*** Використовується для збереження пам'яті для mat[%d] => %f Kb\n",
                         ii, (float) (sizeof(int) * m)/1024.0);
               // Необхідно раніше захоплену ВП повернути системі.
               for(int iii=0;iii <= ii; iii++) if ( mat[iii] !=NULL) free(mat[iii]);
               if (mat! = NULL) free (mat);
               return -1;
          };
     };
     // пам'ять під матрицю виділено задамо значення елементів
     // Ініціалізація датчика випадкових чисел.
     //Включення автоматичної рандомізації
     srand(time(0));


    for(int i=0; i < n; i++)
        for(int j=0; j < m; j++)
           mat[i][j]=rand() % 100; // rand() повертає випадкове число з [0,RAND_MAX],
                                   // Зазвичай RAND_MAX=32767, операція % - залишок від розподілу
     // Виведемо вихідну матрицю
     printf("початкова матриця\n");
      for(int i=0; i < n; i++) {
        for(int j=0; j < m; j++)
```

```
            printf("%4d", mat[i][j]);
         printf("\n");
       };
    // знайдемо maxd максимальний елемент головної діагоналі
    maxd=mat[0][0];
    for(int i=1; i < n; i++)
        if (mat[i][i] > maxd) maxd=mat[i][i];
    printf("максимальний елемент головної діагоналі %d\n", maxd);

    // krez знайдемо кількість елементів двовимірного масиву менше максимального елемента на
головній діагоналі
    // Елементи, що лежать на головній діагоналі враховувати не будемо
    for(int i=0; i < n; i++)
        for(int j=0; j < m; j++){
           if ( i == j ) continue;
           if (mat [i] [j] <maxd) krez ++;
        };
    printf("необхідна кількість елементів %d\n", krez);
   // Повернімо ВП системі
   if (mat != NULL ) {
        for(int ii=0; ii < n; ii++)
            if (mat[ii]! = NULL) free(mat[ii]);
        free(mat);
   };
   return 0;
}
```

## Результат роботи

```
Привіт із кодеблоку
Робота №2-2 Варіант №25
Визначити скільки елементів двовимірного масиву
менше максимального елемента на головній діагоналі
Введіть розмірність матриці (ціле число):5
вихідна матриця
```

```
  80 17 89 41 2
  65 89 56 65 3
  67 77 0 21 42
   8 67 72 74 9
  95 65 27 47 13
максимальний елемент головної діагоналі 89
кількість елементів 18
Process returned 0 (0x0) execution time : 2.670 s
Press any key to continue.
```

---

# 1 «статичні» масиви

## 1.1. Визначення масиву

Визначення масиву містить тип елементів, ім'я масиву та кількість елементів у масиві.

```
int mas [10];
```


Т. е. Індекси елементів в масиві mas можуть змінюватися від 0 до 9, всього в масиві 10 елементів.

## 1.2. Ініціалізація масиву

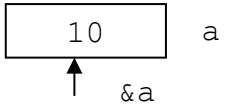
Ініціалізація масивів можлива при визначенні:

```
double d[] = {1, 2, 3, 4, 5};
```

Довжина масиву обчислюється компілятором за кількістю значень, перерахованих у фігурних дужках.

## 1.3. Вказівники

Кожна змінна у програмі це об'єкт, що має ім'я та значення на ім'я можна звернутися до змінної та отримати її значення. Оператор присвоєння( = ) виконує зворотне дію: імені змінної ставиться у відповідність значення.




```
a=10;
```

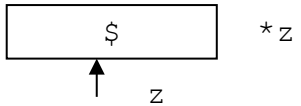
Вираз &a дозволяє отримати адресу ділянки пам'яті, виділеної змінної a. Операція & застосовується лише до об'єктів, які мають ім'я та розміщені в пам'яті.

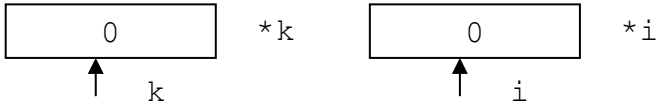
Маючи можливість визначити адресу змінної за допомогою &, треба мати можливість працювати з цією адресою: зберігати її, передавати, перетворювати. І тому вводиться поняття покажчика. Покажчик - це змінна, значенням якої є адреса об'єкта конкретного типу. Нульова адреса позначається константою NULL, яка визначена в заголовному файлistdio.h. Щоб визначити покажчик треба повідомити об'єкт якого типу посилається цей покажчик.

```
char *z;
int *k, *i;
float *f;
```

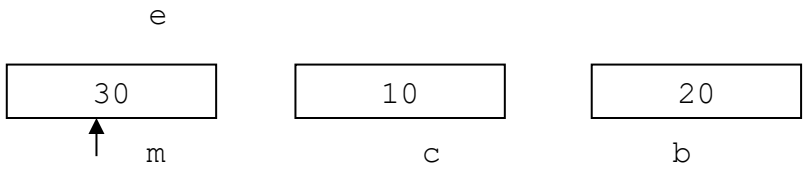
* - це операція розіменування. Операндом цієї операції завжди є покажчик. Результат операції - це об'єкт, який адресує покажчик операнд.

```
*z='$';
```




```
*k=*i=0;
```




Приклад:

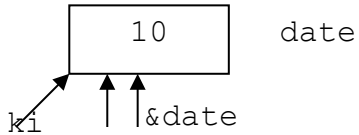
```
int e, c, b, * m;
. . . . . .
m = &e;
* m = c + b;
```

Операції над покажчиками.

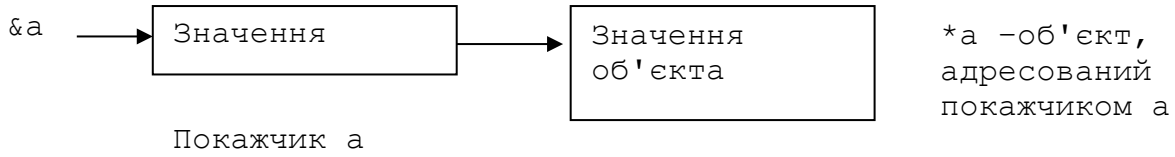
- привласнення (=);
- отримання значення об'єкта, який посилається покажчик (*);
- отримання адреси самого покажчика (&).

Приклад:

```
int date = 10;
int *i, *k;;
i = &date;
k = i;
z = NULL;
```




Подібно до будь-яких змінних змінна типу покажчик має ім'я, арес у пам'яті та значення.




За допомогою унарних операцій ++ і - числові значення змінних типу покажчик змінюються по-різному, залежно від типу даних, з яким пов'язані ці змінні.

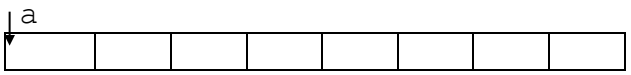
Приклад:

```
char *z;
int *k, *i;
float *f;
. . . . . .
z++; // значення змінюється на 1
i++; // значення змінюється на 2
f++; // значення змінюється на 4
```

Т. е. при зміні покажчика на 1, покажчик переходить до початку наступного (попереднього) поля тієї довжини, яка визначається типом об'єкта, що адресується покажчиком.

### 1.4. Вказівники та масиви

Ім'я масиву без індексу є покажчиком-константою, тобто адресою першого елемента масиву (a[0]).




*** a = = a [0];**
***(a+1) = = a[1];**
**. . . . . .**
***(a+i) = =a[i];**

Відповідно до синтаксису в Сі існують тільки одномірні масиви, але їх елементами, у свою чергу, також можуть бути масиви.

int a[5][5];
Для двовимірного масиву:
**a[m][n] = = *(a[m]+n) = = *(*(a+m)+n);**

---

# 1 динамічні масиви

При традиційному визначенні масиву: тип ім'я_масиву [кількість_елементів]; загальна кількість пам'яті, що виділяється під масив, визначається визначенням і дорівнює кількість_елементів * sizeof(тип).

Але іноді буває потрібно, щоб пам'ять під масив виділялася для вирішення конкретного завдання, причому її розміри заздалегідь не відомі і не можуть бути фіксовані.

Формування масивів зі змінними розмірами можна організувати за допомогою покажчиків та засобів динамічного розподілу пам'яті двома способами:

1) з використанням бібліотечних функцій, описаних у заголовних файлах alloc.h та stdlib.h (стандартний Сі);

2) з використанням операцій new та delete (Сі++).

### 1.1. Формування динамічних масивів із використанням бібліотечних функцій

Для виділення та звільнення динамічної пам'яті використовуються функції

| Функція | Прототип та короткий опис |
|---|---|
| malloc | void * malloc(unsigned s)<br>Повертає покажчик на початок області динамічної пам'яті довжиною s байт, при невдалому завершенні повертає NULL |
| calloc | void * calloc(unsigned n, unsigned m)<br>Повертає покажчик початку області динамічної пам'яті для розміщення n елементів довжиною по m байт кожен, при невдалому завершенні повертає NULL |
| realloc | void * realloc (void * p, unsigned s)<br>Змінює розмір блоку раніше виділеної динамічної пам'яті до розміру s байт, адресу початку змінного блоку, при невдалому завершенні повертає NULL |
| free | void * free (void p)<br>Звільняє раніше виділену ділянку динамічної пам'яті, p - адреса першого байта |

Приклад:
Функція для формування одновимірного динамічного масиву

```
int * make_mas(int n)
(
int *mas;
mas = (int *) malloc (n * sizeof (int));
for(int i=0;i<n;i++)
mas[i]=random(10);
return mas;
}
```

Для виділення пам'яті використовується функція malloc, параметром якої є розмір ділянки пам'яті, що виділяється, рівний n * sizeof (int). Оскільки функція malloc повертає нетипізований покажчик void*, необхідно виконати перетворення отриманого нетипізованого покажчика в покажчик int*.

Звільнити виділену пам'ять можна функцією free(mas).

### 1.2. Формування динамічних масивів з використанням операцій new та delete

Для динамічного розподілу пам'яті використовуються операції new та delete.
Операція
**new имя_типа**
або
**new имя_типа ініціалізатор**

дозволяє виділити і зробити доступними вільна ділянка пам'яті, розміри якої відповідають типу даних, що визначається ім'ям типу. У виділену ділянку заноситься значення, яке визначається ініціалізатором, який не є обов'язковим параметром. У разі успішного виділення пам'яті операція повертає адресу початку виділеної ділянки пам'яті, якщо ділянка не може бути виділена, то повертається NULL.

Приклади:

```
1) int *i;
   i=new int(10);

2) float *f;
   f=new float;

3)  int * mas = new [5];
```

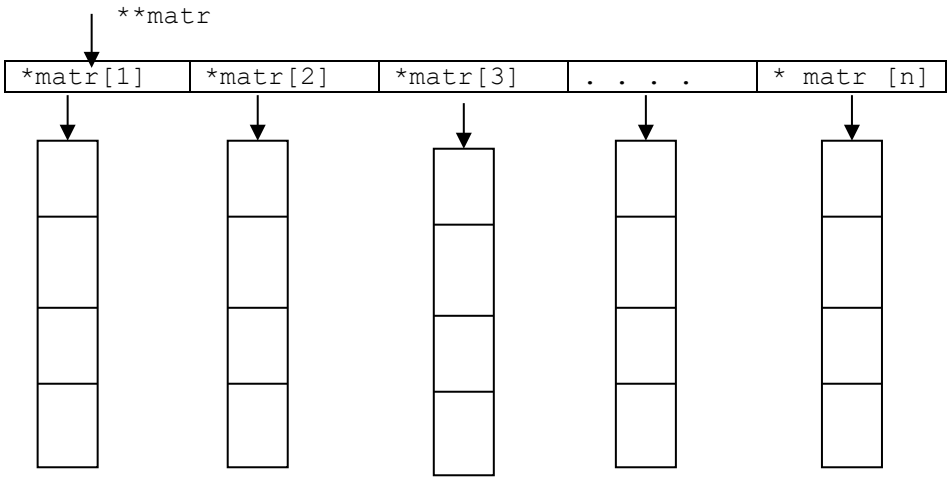
У прикладах 1, 2 показано, як виділити пам'ять під скалярні змінні, приклад 3 показує виділення пам'яті під масив змінних.

Операціяdelete покажчик звільняє ділянку пам'яті раніше виділений операцією new.
Приклад:

Функція для формування двовимірного динамічного масиву

```
int ** make_matr(int n)
{
int **matr;
int i,j;
matr=new int*[n];
for (i=0;i<n;i++)
  {
     matr[i]=new int[n];
     for (j=0;j<n;j++)
     matr[i][j]=random(10);
   };
return matr;
}
```

При формуванні матриці спочатку виділяється пам'яті для масиву покажчиків на одновимірні масиви, а потім у циклі з параметром виділяється пам'ять під n одновимірних масивів.




Щоб звільнити пам'ять, необхідно виконати цикл для звільнення одновимірних масивів.

```
for(int i=0;i<n;i++)
delete matr [i];
```

Після цього звільняємо пам'ять на яку вказує покажчик matr

```
delete [] matr;
```