Лабораторна робота № 5

Тема: Робота з масивами.
Мета роботи: Вивчити засоби мови С для оголошення та ініціалізації статичних масивів, звернення до їх елементів за допомогою індексів та вказівників, введення масивів з клавіатури, генерування випадкових значень елементів та виведення їх на екран.

Теоретичні відомості

Одновимірні масиви

Оголошення

Одновимірний масив (вектор) оголошується так:

```
<тип_даних> <ім'я_масива> [<розмір_масива>];
```

де *тип даних* – це тип елементів масива, при цьому елементами масива не можуть бути функції та елементи типу `void`; *розмір масива* – визначає кількість елементів масива.

На відміну від інших мов, у мові С не перевіряється вихід за межі масива, тому, щоб уникнути помилок у програмі, слід стежити за розмірністю оголошених масивів. Значення розміру масива при його оголошенні не вказують у таких випадках:

- коли при оголошенні масив ініціалізовується;
- коли масив оголошено як формальний параметр функції;
- коли масив оголошено як посилання на масив, явно визначений в іншому модулі.

Звертання до елементів масива виконується за такою схемою:

```
<ім'я_масива> [<значення_індекса>]
```

Оскільки в C/C++ **нумерація індексів починається з нуля**, то значення індексів мають знаходитися в діапазоні від нуля до величини, на одиницю меншу за розмір масива, який визначений при його оголошенні.

Приклад 1. Оголошення масивів:

```
int a[12];     // 12 цілих чисел: a[0], a[1],..., a[11]
float m[30];   // 30 дійсних чисел: m[0], m[1],..., m[29]
double v[15];  /* 15 дійсних чисел з подвійною точністю:
                  v[0], v[1],..., v[14] */
char N[20];    // 20 символів: N[0], N[1],..., N[19]
```

Ініціалізація

При оголошенні масивів їхнім елементам можна присвоювати початкові значення. Якщо реальна кількість ініціалізованих значень є меншою за розмір масива, то решта елементів масива отримують значення 0.

Приклад 2.

```
int a[5]={9,33,-23,8,1};      // a[0]=9, a[1]=33, a[2]=-23, a[3]=8, a[4]=1
```

```
int a[]={9,33,-23,8,1};       /* еквівалентно попередньому оголошенню масива,
                                 компілятор сам визначає довжину масива */
float b[10] = {1.5, -3.8, 10};      // b[0]=1.5, b[1]=-3.8, b[2]=10,
                                    // b[3]=b[4]= … =b[9]=0
```

Доступ до елементів

Для доступу до елементів одновимірного масива за допомогою вказівника використовується така форма:

*(<ім'я_масива> + <індекс>)

Під час програмної реалізації індексний спосіб доступу до елемента масива (<ім'я_масива>[<індекс>]) зводиться до адресного (вказівникового), тобто індексний вираз перетворюється в адресний, а оскільки операції над вказівниками виконуються швидше і, якщо елементи масиву опрацьовуються почергово, то доцільніше використовувати другий спосіб. Якщо ж вибір елементів є випадковим, то, щоб уникнути помилок, краще використовувати перший спосіб. Крім того, перший спосіб є більш наочним і зручнішим для сприйняття, що сприяє кращому читанню програм.

Оскільки ім'я масива є вказівником на перший елемент масива, то вказівнику можна присвоїти адресу першого елемента масива за допомогою такого оператора присвоєння:

```
int *ptr;
ptr = m;
```

Цей вираз аналогічний присвоєнню адреси першого елемента масиву, тобто елемента з нульовим індексом:

```
ptr = &m[0];
```

Також для оголошення масивів можна використовувати *типізовані константи-масиви,* які дають змогу водночас і оголосити масив, і задати його значення як константи:

```
const <тип_даних> <ім'я_масива> [<розмір_масива>] =
{<значення_елементів_масива>};
```

Зауважимо, що **значення елементів констант-масивів змінювати не можна**.

Приклад 3. Оголошення константи-масивів:

```
const int arr[5]={9,3,-7,0,123};      // масив з 5-ти цілих чисел
const C[5]={15,-6,546};    /* масив з 5-ти цілих чисел типу int,
                        де C[0]=15, C[1]=-6, C[2]=546, C[3]=0, C[4]=0 */
const float f[5]={1.5,6,8,7.4,-1.125};     // масив з 5-ти дійсних чисел
```

Крім того, масиви можна оголошувати на основі власного типу даних:

```
typedef <тип_даних> <ім'я_власного_типу_даних> [<розмір_масиву>];
<ім'я_власного_типу_даних> <ім'я_масиву>;
```

Приклад 4. Створити власний тип даних myArray як масив з 10-ти додатних цілих чисел і оголосити масиви a та b, цього типу:

```
typedef unsigned short myArray[10];
myArray a, b;
```

Введення одновимірних масивів

Введення масива **вручну**:

1 спосіб (за допомогою `printf`):

```
const int colCount=5;
for(int i=0; i<colCount; i++)
      {printf("a[%d]=", i); scanf("%d", &a[i]);}
```

2 спосіб (за допомогою `cin`):

```
const int colCount=5;
for(int i=0; i<colCount; i++)
      {cout << "a[" << i << "]="; cin >> a[i];}
```

При цьому числа, що вводяться можна записувати як у стовпчик, так і в рядок через пробіл.

Введення масива за допомогою **генератора випадкових чисел**:

1 спосіб (кожного разу генерується однакова послідовність чисел):

```
#include <stdlib.h>
...
const int colCount=5, Low=6, High=20;
int a[colCount];
for (int i=0; i<colCount; i++)
      a[i] = Low + rand() % (High-Low+1);
```

2 спосіб (кожного разу генерується інша послідовність чисел):

```
#include <stdlib.h>
#include <time.h>
...
const int colCount=5, Low=6, High=20;
srand((unsigned)time(NULL));
for (int i=0; i<colCount; i++)
      a[i] = Low + rand() % (High-Low+1);
```

Виведення одновимірних масивів

1 спосіб (за допомогою `printf`):

```
const int colCount=5;
int a[colCount];
for (int i=0; i<colCount; i++)
      printf("a[%d]=%d\t", i, a[i]);
```

2 спосіб (за допомогою `cout`):

```
// у стовпчик
for (int i=0; i<colCount; i++)
      cout << "a[" << i << "]=" << a[i] << endl;

// у рядок
for (int i=0; i<colCount; i++)
      // розділяти значення за допомогою табуляції
      cout << "a[" << i << "]=" <<  a[i] << "\t" << endl;

// або розділяти значення за допомогою форматування
// cout << setw(5) << "a[" << i << "][" << j << "]=" << setw(2) << a[i][j];
```

Двовимірні масиви

Для зручного зберігання даних часто недостатньо однієї вимірності масива, тоді використовують багатовимірні масиви.

Вимірність масива визначається кількістю його індексів. Елементи одновимірного масива (вектора) мають один індекс, двовимірного (матриці, таблиці) – два індекси: перший – номер рядка, а другий – номер стовпчика. Кількість індексів у масивах необмежена, тому можна створювати масиви різної вимірності.

Оголошення

Оголошення багатовимірного масива має такий вигляд:

<тип_даних> <ім'я_масива> [*<розмір1>*] [*<розмір2>*] … [*<розмірN>*];

При цьому кількість елементів масива дорівнює добутку кількості елементів масива за кожним індексом.

Приклад 5. Оголосити двовимірний масив з 3-х рядків та 4-х стовпчиків (12-ти елементів) цілого типу:

```
int a[3][4];
```

При цьому структура елементів цього масива буде така:

```
a[0][0], a[0][1], a[0][2], a[0][3],
a[1][0], a[1][1], a[1][2], a[1][3],
a[2][0], a[2][1], a[2][2], a[2][3];
```

Під масив виділяється пам'ять, необхідна для розташування всіх його елементів.

Приклад 6. Оголошення двохвимірних масивів:

```
float M1[5][5]; // Матриця 5·5=25 елементів дійсного типу
double M3[4][5][4];   // Тривимірний масив з 4·5·4=80 елементів
                      // дійсного типу
char M2[10][3]; // Двовимірний масив з 10·3=30 елементів
                // символьного типу
```

Ініціалізація

При оголошенні масива його елементи можна ініціалізовувати початковими значення, причому необов'язково всі. У таких випадках неініціалізовані елементи числових масивів отримують значення 0.

Приклад 7. Ініціалізація двохвимірних масивів:

1) `int w[3][3]={{2, 3, 4}, {3, 4, 8}, {1, 0, 9}};`

Тут оголошено та ініціалізовано масив цілих чисел `w[3][3]`. Елементам масиву присвоєно відповідні значення зі списку: `w[0][0]=2, w[0][1]=3, w[0][2]=4, w[1][0]=3` і т.д. Списки, взяті у фігурні дужки, відповідають рядкам масива.

2) `float C[4][3]={1.1, 2, 3, 3.4, 0.5, 6.8, 9.7, 0.9};`

3) `float C[4][3]={{1.1, 2, 3},{3.4, 0.5, 6.8},{9.7, 0.9}};`

4) `float C[][3]={{1.1, 2, 3},{3.4, 0.5, 6.8},{9.7, 0.9}};`

Записи 2)-4) прикладів еквівалентні, при чому в них елементи останнього рядка неініціалізовані, тому вони отримають значення 0.

5) `float C[4][3]={{1.1, 2},{3, 3.4, 0.5},{6.8},{9.7, 0.9}};`

Доступ до елементів

Для доступу до елементів двовимірного масива використовують індексні вирази у такій формі:

```
<ім'я_масива>[<індекс_рядка>][<індекс_стовпчика>]
```

А для доступу до елементів звичайних двовимірних масивів у функціях використовують адресні вирази з вказівником:

```
*(*(<ім'я_масива> + <індекс_рядка>) + <індекс_стовпчика>)
```

Послідовне розміщення елементів двовимірних масивів у пам'яті без жодних проміжків дає змогу звертатись до будь-якого елемента багатовимірного масива, використовуючи адресу його початкового елемента та лише один індексний вираз. Таким чином для масива `a` звертання `*(a)` є посиланням на елемент `a[0][0]`, звертання `*(a+2)` – на елемент `a[0][2]`, звертання `*(a+3)` – на елемент `a[1][0]` і т. д., а звертання `*(a+i*colCount+j)` – на елемент `a[i][j]`.

Розміщення тривимірного масиву відбувається аналогічно.

Для звернення до довільного елемента тривимірного масива з індексами `[i][j][z]` використовують таку форму:

```
<ім'я>[i*colXCount*colZCount + j*colZCount + z]
```

де `colXCount` – кількість стовпчиків по осі X; `colZCount` – кількість стовпчиків по осі Z.

Приклад 8. Оголошення `float W[3][4][5]`, крім самого тривимірного масиву з шістдесяти елементів (3*4*5) типу `float`, створює у програмі масив з чотирьох вказівників на тип `float`, масив з трьох вказівників на масив вказівників на тип `float` і вказівник на масив масивів вказівників на тип `float`. Для доступу до елемента `W[2][3][4]` використовують вказівник, оголошений як `float *p=W[0][0]`, з одним індексним виразом у формі `p[2*4*5+3*5+4]` або `p[59]`.

Багатовимірні масиви також можна оголошувати на основі власного типу даних. Для цього використовується такий запис:

```
typedef <тип_даних> <ім'я_власного_типу_даних>[<розмір1>][<розмір2>]…[<розмір N>];
<ім'я_власного_типу_даних> <ім'я_масива>;
```

Приклад 9. Створити тип даних з ім'ям `myMatrix` як масив цілих чисел з 10-ти рядків та 7-ми стовпчиків і оголосити масиви `D1` і `D2` цього типу:

```
typedef int myMatrix[10][7];
myMatrix D1, D2;
```

Для оголошення багатовимірних масивів також можна використовувати типізовані константмасиви, які обов'язково потрібно ініціалізовувати початковими значеннями. Оголошення таких типізованих константмасивів має такий вигляд:

```
const <тип_даних> <ім'я_масива> [<розмір1>][<розмір2>],…,[<розмірN>]=
{{<значення_елементів_1_рядка>},{<значення_елементів_2_рядка>},…,
{<значення_елементів_N_рядка>}};
```

Приклад 10. Оголошення двовимірного константа-масива:

```
const int arr[2][5]={{9, 3, 0, 12, -5},{-7, 23, 2, 4, 0}};
```

Зауважимо, що **змінювати значення елементів константа-масивів не можна**.

Введення двовимірних масивів

Здійснювати виведення значень елементів масиву можна лише поелементно, для чого слід організовувати цикли, в яких послідовно змінюватимуться значення індексів елементів.

Введення елементів масива **вручну:**

1 спосіб (за допомогою `printf`):

```
const int rowCount=4, colCount=5;
int a[rowCount][colCount];
for (int i=0; i<rowCount; i++)
      for (int j=0; j<colCount; j++)
            {printf("a[%d][%d]=", i, j); scanf("%d", &a[i][j]);}
```

2 спосіб (за допомогою `cin`):

```
const int rowCount=4, colCount=5;
int a[rowCount][colCount];
for (int i=0; i<rowCount; i++)
      for (int j=0; j<colCount; j++)
            {cout << "a[" << i << "][" << j << "]="; cin >> a[i][j];}
```

Введення масива за допомогою **генератора випадкових чисел**:

1 спосіб (кожного разу генерується однакова послідовність чисел):

```
#include <stdlib.h>
...
const int rowCount=4, colCount=5, Low=6, High=20;
int a[rowCount][colCount];
for (int i=0; i<rowCount; i++)
      for (int j=0; j<colCount; j++)
            a[i][j] = Low + rand() % (High-Low+1);
```

2 спосіб (кожного разу генерується інша послідовність чисел):

```
#include <stdlib.h>
#include <time.h>
...
const int rowCount=4, colCount=5, Low=6, High=20;
srand((unsigned)time(NULL));
for (int i=0; i<rowCount; i++)
      for (int j=0; j<colCount; j++)
            a[i][j] = Low + rand() % (High-Low+1);
```

Виведення двовимірних масивів

1 спосіб (за допомогою `printf`):

```
const int rowCount=4, colCount=5, Low=6, High=20;
int a[rowCount][colCount];
for (int i=0; i<rowCount; i++) {
      for (int j=0; j<colCount; j++) {
            // розділяти значення за допомогою формату
```

```
            printf("a[%d][%d]=%-6d",i,j,a[i][j]);
            // або розділяти значення за допомогою табуляції
            // printf("a[%d][%d]=%d\t",i,j,a[i][j]);
      }
      // перехід на новий рядок
      printf("\n");
}
```

2 спосіб (за допомогою `cout`):

```
const int rowCount=4, colCount=5;
for (int i=0; i<rowCount; i++) {
   for (int j=0; j<colCount; j++) {
      cout << setw(5) << "a[" << i << "][" << j << "]=" << setw(2) << a[i][j];
      // або розділяти значення за допомогою табуляції
      //cout << "a[" << i << "," << j << "]=" << a[i][j] << "\t";
   }
   // перехід на новий рядок
   cout << endl;
}
```

Передавання масивів у функцію

Масив можна передати у функцію за допомогою формального параметра такими способами:

1) **як вказівник**, тоді оголошення функції буде мати такий вигляд:

```
void myFunction(int *param)
{
      ...
}
```

При використанні масива як параметра у функцію передається вказівник на його перший елемент, тобто масив завжди передається за адресою. При цьому інформація про кількість елементів масива втрачається, тому його розмірність потрібно передавати через окремий параметр.

Приклад 11. Обчислити суму елементів одновимірного масива.

```
int sumPointer(const int *arr, int Columns) {
      // calculation sum
      int s = 0;
      for(int i = 0; i < Columns; i++)
            s += arr[i];
      return s;
}
void main(){
      // розмір масива повинен бути константою
      const int colCount = 5;
      //int b[] = {3, 4, 5, 4, 4};  // для статичних елементів
      int b[colCount];              // для випадкових елементів
      const int Low=6, High=20;
      // initialization array
      srand((unsigned)time(NULL));
      for (int i = 0; i < colCount; i++)
            b[i] = Low + rand() % (High-Low+1);
      // output array
      for(int i = 0; i < colCount; i++)
            cout << setw(7) << b[i];
      cout << endl;
      // output sum
      cout << "Array's elements Sum: " << sumPointer(b, colCount) << endl;
}
```

Приклад 12. Обчислити суму елементів двовимірного масива.

```
int sumPointer2(int *arr, int Rows, int Columns) {
	// calculation sum
	int s = 0;
	for (int i = 0; i < Rows; i++)
		for (int j = 0; j < Columns; j++)
			s += *(arr+i*Columns+j);
	return s;
}
void main(){
	// розміри масива мають бути константами
	const int rowCount = 4, colCount = 5;
	int b[rowCount][colCount];
	const int Low = 6, High = 20;
	srand((unsigned)time(NULL));
	for (int i = 0; i < rowCount; i++){
		for (int j = 0; j < colCount; j++) {
			// initialization array
			b[i][j] = Low + rand() % (High-Low+1);
			// output array
			cout << setw(7) << b[i][j];
		}
		cout << endl;
	}
	cout << endl;
	int *p=&b[0][0];
	// output sum
	cout << "Array's elements Sum: " << sumPointer2(p, rowCount, colCount) <<
endl;
}
```

Приклад 12. Написати рекурсивну функцію, яка обчислює мінімальний елемент масиву.

Створимо рекурсивну функцію `minimum()`, яка повертатиме індекс мінімального елемента. Вона матиме три параметри: 1) вказівник на початок масива `mas`; 2) розмір масива `n`; 3) індекс елемента, з якого починається підмасив. Цей підмасив рекурсивно зменшується поки його розмір не становитиме 1. Коли у ньому залишиться один елемент (`k==n-1`), то повертається значення `k`. На наступному кроці рекурсивного повернення порівнюються `mas[k]` (останній елемент) та `mas[a]` (передостанній елемент) і повертається індекс меншого з них. Продовжуючи, на кожному кроці порівнюються мінімум, отриманий на попередніх кроках, та `mas[a]` (поточний елемент).

```
int minimum(int *mas, int n, int k) {
	if(k == n-1) return k;
	int a = minimum(mas, n, k+1); // Виклик для підмасиву розміром на 1 менше
	if(mas[a]<mas[k]) return a;
		else return k;
}
void main() {
	int i, n, min;
	cout << "Уведіть розмір масиву: "; cin >> n;
	int *mas = new int[n];
	for (i=0;i<n;i++)
		{ cout << "Уведіть mas[" << i << "] = "; cin >> mas[i]; }
	cout << endl;
	min = minimum (mas, n, 0) ; // Виклик для всього масиву (k=0)
	cout << "Мінімум: mas[" << min << "] = " << mas[min] << endl;
	return 0;
}
```

Приклад 13. Увести масив з 10-ти дійсних чисел і створити рекурсивну функцію, яка записує масив у зворотній послідовності.

```
void invertItem(double *a, int i, int j) {
      double t=a[i];
      a[i]=a[j]; a[j]=t;
      i++; j--;
      if (i<j) invertItem(a, i, j);
}
void main() {
      double a[10]; int i;
      cout << "Увести 10 дійсних чисел:" << endl;
      for (i=0; i<10; i++) cin >> a[i];
      invertItem(a, 0, 9);
      for (i=0; i<10; i++) cout << a[i] << " ";
      return 0;
}
```

Результати виконання програми:

```
Увести 10 дійсних чисел:
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
```

2) як масив визначеного розміру: якщо розмірність масива є константою, то її можна вказати, по-перше, при оголошенні формального параметра і, по-друге, як верхню межу циклів опрацювання масива всередині функції.

Оголошення функції з масивом визначеного розміра як параметра має такий вигляд:

```
<тип> <ім'я_функції> (<тип> <параметр>[<розмірність>])
{
      ...
}
```

Приклад 14. Обчислити суму елементів одновимірного масива.

```
int sumDef(const int arr[5]) {
      // calculation sum
      int s = 0;
      for(int i = 0; i < 5; i++)
            s += arr[i];
      return s;
}
 void main(){
      // розмір масива повинен бути константою
      const int colCount = 5;
      //int b[colCount] = {3, 4, 5, 4, 4};      // для статичних елементів
      int b[colCount];                          // для випадкових елементів
      const int Low = 6, High = 20;
      // initialization array
      srand((unsigned)time(NULL));
      for (int i = 0; i < 5; i++)
            b[i] = Low + rand() % (High-Low+1);
      // output array
      for(int i = 0; i < 5; i++)
            cout << setw(7) << b[i];
      cout << endl;
      // output sum
      cout << "Array's elements Sum: " << sumDef(b) << endl;
}
```

Приклад 15. Обчислити суму елементів двовимірного масива

```
int sumDef2(const int arr[4][5]) {
```

```
	// calculation sum
	int s = 0;
	for (int i = 0; i < 4; i++)
		for (int j = 0; j < 5; j++)
			s += arr[i][j];
	return s;
}
void main(){
	// розміри масива мають бути константами
	const int rowCount = 4, colCount = 5;
	int b[rowCount][colCount];
	const int Low = 6, High = 20;
	srand((unsigned)time(NULL));
	for (int i = 0; i < rowCount; i++){
		for (int j = 0; j < colCount; j++){
			// initialization array
			b[i][j] = Low + rand() % (High-Low+1);
			// output array
			cout << setw(7) << b[i][j];
		}
		cout << endl;
	}
	// output sum
	cout << "Array's elements Sum: " << sumDef2(b) << endl;
}
```

Приклад 16. Увести матрицю дійсних чисел розмірності 4x3 й обчислити за допомогою функції мінімальний елемент матриці та його індекси.
До функції слід передати два додаткових параметри -indi та indj, в які буде записано обидва індекси мінімального елемента. Ці параметри мають передаватися за посиланням (чи за вказівником), щоб у тілі функції можна було їх змінити і повернути ці зміни у точку виклику.

```
double matrix_min(double a[4][3], int& indi, int& indj) {
	double min=a[0][0];
	indi=0; indj=0;
	for(int i=0; i<4; i++)
		for(int j=0; j<3; j++)
			if (a[i][j]<min)
				{ min=a[i][j]; indi=i; indj=j; }
	return min;
}
void main() {
	double a[4][3], m;
	int i, j, ind_i=0, ind_j=0;
	for(i=0;i<4;i++)
		for(j=0;j<3;j++)
			cin >> a[i][j];
m = matrix_min(a, ind_i, ind_j);
	cout << m << "(" << ind_i << ", " << ind_j << ")";
}
```

3) як масив невизначеного розміру : тоді оголошення функції має такий вигляд:

```
<тип> <ім'я_функції> (<тип> <параметр>[])
{
	...
}
```

Приклад 17. Обчислити суму елементів одновимірного масива.

```
int sumNoDef(const int arr[], int Columns) {
	// calculation sum
	int s = 0;
```

```
	for(int i = 0; i < Columns; i++)
		s += arr[i];
	return s;
}
 void main(){
	// розмір масива повинен бути константою
	const int colCount = 5;
	//int b[colCount] = {3, 4, 5, 4, 4};	// для статичних елементів
	int b[colCount];				// для випадкових елементів
	const int Low = 6, High = 20;
	// initialization array
	srand((unsigned)time(NULL));
	for (int i = 0; i < colCount; i++)
		b[i] = Low + rand() % (High-Low+1);
	// output array
	for(int i = 0; i < colCount; i++)
		cout << setw(7) << b[i];
	cout << endl;
	// output sum
	cout << "Array's elements Sum: " << sumNoDef(b, colCount) << endl;
}
```

Приклад 18. Обчислити суму елементів двовимірного масива

```
int sumNoDef2(int arr[], int Rows, int Columns) {
	// calculation sum
	int s = 0;
	for (int i = 0; i < Rows; i++)
		for (int j = 0; j < Columns; j++)
			s += arr[i*Columns+j];
	return s;
}
void main(){
	// розміри масива мають бути константами
	const int rowCount = 4, colCount = 5;
	int b[rowCount][colCount];
	const int Low = 6, High = 20;
	srand((unsigned)time(NULL));
	for (int i = 0; i < rowCount; i++){
		for (int j = 0; j < colCount; j++){
			// initialization array
			b[i][j] = Low + rand() % (High-Low+1);
			// output array
			cout << setw(7) << b[i][j];
		}
		cout << endl;
	}
	// output sum
	cout << "Array's elements Sum: " << sumNoDef2(&b[0][0], rowCount,
colCount) << endl;
}
```

Повернення масивів з функцій

Щоб функція повертала одновимірний масив, потрібно її оголосити як функцію, що повертає вказівник:

```
int * myFunction()
{
	...
}
```

Приклад 19. Ініціалізувати одновимірний масив випадковими числами.

```
int* init(int *arr, int Columns){
	const int Low = 6, High = 20;
```

```
	// initialization array
	srand((unsigned)time(NULL));
	for (int i = 0; i < Columns; i++)
		arr[i] = Low + rand() % (High-Low+1);
	return arr;
}
void main(){
	// розмір масива повинен бути константою
	const int colCount = 5;
	int b[colCount];
	// initialization array
	int *p = init(b, colCount);
	// output array
	for(int i = 0; i < colCount; i++)
		cout << setw(7) << p[i];
}
```

Приклад 20. Ініціалізувати одновимірний масив випадковими числами.

```
int * createInit(){
	static int arr[5];
	/* set the seed */
	srand((unsigned)time(NULL));
	for (int i = 0; i < 5; ++i){
		arr[i] = rand();
		cout << " arr[" << i << "] = " << arr[i] << endl;
	}
	return arr;
}
void main(){
	int *p;
	p = createInit();
	for (int i = 0; i < 5; i++)
		cout << "*(p + " << i << ") = " << *(p + i) << endl;
}
```

Зауважимо, що не можна повертати з функції вказівник на локальну змінну, оскільки пам'ять, виділена локальним змінним при вході у функцію, звільняється одразу ж після повернення з неї.

Приклад 21.

```
int* f() {
	int a = 5;
	return &a;  // ПОМИЛКА!
}
```

Хід роботи

Завдання 1.

1. Написати програму для виконання певних дій над **статичним** одновимірним масивом, реалізувавши доступ до елементів масива двома способами: за допомогою індексів та вказівників. Кожний спосіб реалізувати окремою функцією.
2. Алгоритм формування початкового масива реалізувати двома способами: за допомогою введення даних з клавіатури та випадкової генерації чисел. Кожний спосіб реалізувати окремою функцією.
3. Вивести на екран елементи початкового масива. Якщо є кілька початкових масивів, то вивести усі. Якщо потрібно визначити певні індекси, значення

тощо, то вивести їх. Якщо масив був змінений, то вивести модифікований масив.

4. **Використання глобальних змінних у підпрограмах не допускається**. Інформація у підпрограми повинна передаватися лише за допомогою параметрів.
5. Введення даних, виведення даних і виконання певних дій над масивами потрібно реалізувати в окремих функціях. У головній програмі потрібно виконувати лише їхній виклик. Введення-виведення даних супроводжувати відповідними повідомленнями.
6. Побудувати блок-схему алгоритму програми та блок-схеми алгоритмів усіх функцій з індексним доступом до елементів масива.

№ з/п	Завдання
1.	Написати підпрограму, яка обчислює середнє арифметичне непарних елементів одновимірного масиву (вектора) із n елементів цілого типу.
2.	Написати підпрограму, яка міняє місцями перший елемент із найменшим парним елементом одновимірного масиву (вектора) із n елементів цілого типу.
3.	Написати підпрограму, яка шукає максимальний та мінімальний елементи одновимірного масиву (вектора) із n елементів цілого типу.
4.	Написати підпрограму, яка міняє місцями елементи одновимірного масиву (вектора) із n елементів цілого типу так, щоб вони розмістилися в зворотному порядку: $a_n, a_{n-1}, \ldots, a_2, a_1$.
5.	Написати підпрограму, яка обчислює суму індексів непарних елементів одновимірного масиву (вектора) із n елементів цілого типу.
6.	Написати підпрограму, яка обчислює середнє арифметичне елементів одновимірного масиву (вектора) із n елементів цілого типу з парними індексами.
7.	Написати підпрограму, яка міняє місцями останній елемент із найбільшим непарним елементом одновимірного масиву (вектора) із n елементів цілого типу.
8.	Написати підпрограму, яка шукає індекси найбільшого та найменшого елементів одновимірного масиву (вектора) із n елементів цілого типу.
9.	Написати підпрограму, яка обчислює середнє арифметичне максимального та мінімального елементів одновимірного масиву (вектора) із n елементів цілого типу.
10.	Написати підпрограму, яка обчислює суму елементів одновимірного масиву (вектора) із n елементів цілого типу з непарними індексами.
11.	Написати підпрограму, яка обчислює суму максимального та мінімального елементів одновимірного масиву (вектора) із n елементів цілого типу.
12.	Написати підпрограму, яка шукає найменший парний елемент одновимірного масиву (вектора) із n елементів цілого типу.
13.	Написати підпрограму, яка шукає індекс найбільшого парного елемента одновимірного масиву (вектора) із n елементів цілого типу.

14.	Написати підпрограму, яка обчислює суму індексів максимального та мінімального елементів одновимірного масиву (вектора) із n елементів цілого типу.
15.	Написати підпрограму, яка шукає найбільший непарний елемент одновимірного масиву (вектора) із n елементів цілого типу.
16.	Написати підпрограму, яка обчислює середнє арифметичне індексів парних елементів одновимірного масиву (вектора) із n елементів цілого типу.
17.	Написати підпрограму, яка міняє місцями елементи одновимірного масиву (вектора) із $2n$ елементів цілого типу так, щоб вони розмістилися в такому порядку:a_2, a_1, a_4, a_3, ..., a_{2n}, a_{2n-1}. (кожний елемент з парним індексом міняється місцями з попереднім елементом).
18.	Написати підпрограму, яка обчислює суму парних елементів одновимірного масиву (вектора) із n елементів цілого типу.
19.	Поміняти місцями елементи одновимірного масиву (вектора) із $2n$ елементів цілого типу так, щоб вони розмістилися в такому порядку: a_{n+1}, ..., a_{2n}, a_1, …, a_n (перша половина елементів вектора міняється місцями з другою).
20.	Впорядкувати за спаданням елементи одновимірного масиву (вектора) із n елементів цілого типу.
21.	Обчислити середнє арифметичне максимального та мінімального елементів одновимірного масиву (вектора) із n елементів цілого типу.
22.	Замінити від’ємні елементи масиву, розташовані у непарних стовпчиках, їхніми модулями та знайти суму цих елементів.
23.	Обчислити добуток елементів масиву, розташованих після максимального за модулем елементу.
24.	Обчислити суму елементів масиву, розташованих після першого додатного елементу.
25.	Обчислити середнє арифметичне елементів масиву, розташованих між першим і другим додатними елементами.
26.	Обчислити добуток елементів масиву, розташованих між першим і другим нульовими елементами у парних стовпчиках.
27.	Написати підпрограму, яка міняє місцями максимальний та мінімальний елементи одновимірного масиву (вектора) із n елементів цілого типу.
28.	Написати підпрограму, яка шукає індекс найменшого непарного елемента одновимірного масиву (вектора) із n елементів цілого типу.
29.	Написати підпрограму, яка обчислює кількість непарних елементів одновимірного масиву (вектора) із n елементів цілого типу.
30.	Написати підпрограму, яка обчислює середнє арифметичне індексів максимального та мінімального елементів одновимірного масиву (вектора) із n елементів цілого типу.

Завдання 2.

1. Написати програму для виконання певних дій над **статичним** одновимірним масивом, реалізувавши доступ до елементів масива двома способами: за допомогою індексів та вказівників. Кожний спосіб реалізувати окремою функцією.
2. Алгоритм формування початкового масива реалізувати двома способами: за допомогою введення даних з клавіатури та випадкової генерації чисел. Кожний спосіб реалізувати окремою функцією.
3. Вивести на екран елементи початкового масива. Якщо є кілька початкових масивів, то вивести усі. Якщо потрібно визначити певні індекси, значення тощо, то вивести їх. Якщо масив був змінений, то вивести модифікований масив.
4. **Використання глобальних змінних у підпрограмах не допускається**. Інформація у підпрограми повинна передаватися лише за допомогою параметрів.
5. Введення даних, виведення даних і виконання певних дій над масивами потрібно реалізувати в окремих функціях. У головній програмі потрібно виконувати лише їхній виклик. Введення-виведення даних супроводжувати відповідними повідомленнями.
6. Побудувати блок-схему алгоритму програми та функції, що виконує вказані дії над масивом.

№ з/п	Завдання
1.	Визначити значення та порядкові номери двох найбільших елементів вектора дійсних чисел.
2.	Визначити значення та порядкові номери максимального від'ємного елемента вектора дійсних чисел.
3.	Визначити значення та порядкові номери мінімального додатного елемента вектора цілих чисел.
4.	Включити у відсортований за спаданням значень масив цілих чисел введене з клавіатури ціле число так, щоб не порушити загальну впорядкованість елементів масива.
5.	Серед елементів першого вектора дійсних чисел, які не входять у другий вектор дійсних чисел, знайти значення та порядкові номери максимального елемента.
6.	Відсортувати елементи сформованого із двох векторів цілих чисел вектора за зростанням.
7.	Записати у два масиви усі дільники двох цілих чисел. Крім того, спільні дільники записати у третій масив. Утворити число, дільниками якого є числа, записані у третій масив, і вивести його на екран.
8.	Цифри цілої частини дійсного числа записати у масив. Утворити та вивести на екран ціле число зі зворотним порядком цифр.
9.	Вивести на екран елементи одновимірного масива цілих чисел, значення яких зустрічається більше одного разу.
10.	Визначити кількість повторень кожного елемента одновимірного масива

	цілих чисел. Елементи масива можуть повторюватися.
11.	Ввести масив цілих додатних десяткових чисел. Виконати переведення кожного числа з десяткової системи числення у двійкову. Для переведення використати алгоритм виділення залишків від цілочисельного ділення десяткового числа на 2. Двійкові зображення чисел вивести на екран.
12.	Циклічно зсунути всі елементи масива цілих чисел на K позицій вліво. Значення K ввести з клавіатури. При циклічному зсуві вліво початковий елемент масива записується на місце зсунутого кінцевого елемента.
13.	Створити новий масив, вибравши з кожної пари двох сусідніх елементів одновимірного масива цілих чисел максимальний. Визначити значення мінімального елемента створеного масива.
14.	Перевірити, чи вектор цілих чисел є спадаючим. Якщо так, то записати елементи масива у порядку зростання (без застосування алгоритму сортування), інакше – відсортувати його за спаданням.
15.	Перші 10 простих чисел натурального ряду записати в одновимірний масив. Визначити середнє арифметичне значення елементів цього масива.
16.	Сформувати вектор з 20 непарних чотирицифрових випадкових чисел без повторень елементів. Визначити мінімальний і максимальний елементи вектора та поміняти їх місцями.
17.	Сформувати вектор з усіх простих чисел, що потрапляють у заданий інтервал (a,b). Межі інтервала ввести з клавіатури. Сформований масив вивести на екран.
18.	Ввести масив цілих чисел та циклічно зсунути усі його елементи на K позицій вправо. Значення K утворюється випадковим чином і є числом, що не перевищує 10. При циклічному зсуві вправо кінцевий елемент масива записується на місце зсунутого початкового елемента.
19.	Сформувати вектор із N цілих випадкових чисел, які приймають значення на відрізку [a,b]. Межі відрізка ввести з клавіатури. Визначити середнє арифметичне значення елементів масива. Знайти квадратичне відхилення кожного числа від середнього значення та записати його у новий вектор. Знайти значення та номери елементів з найбільшим відхиленням.
20.	Сформувати масив з 10 випадкових цілих чисел, що належать заданому інтервалу (a,b]. Межі інтервала ввести з клавіатури. Елементи масива не повинні повторюватися. Відсортувати сформований масив за спаданням значень елементів.
21.	Сформувати масив з перших 20 чисел Фібоначі. Врахувати, що перші два числа Фібоначі дорівнюють 1, а кожне наступне є сумою двох попередніх.
22.	Сформувати масив з усіх дільників введеного з клавіатури натурального числа.
23.	Виконати перетворення цілого десяткового числа у шістнадцяткову систему числення. Використати алгоритм виділення залишків при цілочисельному діленні введеного числа на 16. Залишки записувати в

	одновимірний масив. Перетворене число вивести на екран.
24.	Коефіцієнти многочлена $p_n(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x^1 + a_0$ є елементами масива a[n]. Обчислити значення многочлена в точці x за схемою Горнера: $p_n(x) = (...((a_n x + a_{n-1})x + a_{n-2})x + ... + a_1)x + a_0$.
25.	Відсортувати вектор дійсних чисел за зростанням значень елементів, використавши метод сортування вставками: якщо перші k елементів вже впорядковані, то береться (k+1)-й елемент і розміщується серед перших k елементів так, щоб ряд із (k+1)-го елемента був впорядкований.
26.	Виконати нормування елементів вектора цілих чисел, розділивши кожен з них на значення максимального елемента. Якщо максимальний елемент дорівнює 0, то вивести відповідне повідомлення.
27.	Із неспільних елементів (без повторень значень вибраних елементів) двох векторів цілих чисел утворити третій вектор.
28.	Із спільних елементів (без повторень значень вибраних елементів) двох векторів цілих чисел утворити третій вектор. За відсутності спільних елементів вивести відповідне повідомлення.
29.	Виконати злиття впорядкованих за зростанням значень векторів дійсних чисел в один впорядкований масив (без застосування алгоритму сортування).
30.	Створити одновимірний масив, записавши у нього менші елементи з кожної пари елементів двох масивів дійсних чисел однакової розмірності. Кожна пара елементів визначається однаковим значенням індексів. Знайти максимальний елемент утвореного масиву.

Завдання 3.

1. Написати програму для виконання певних дій над **статичним** двовимірним масивом, реалізувавши доступ до елементів масива двома способами: за допомогою індексів та вказівників. Кожний спосіб реалізувати окремою функцією.
2. Алгоритм формування початкового масива реалізувати двома способами: за допомогою введення даних з клавіатури та випадкової генерації чисел. Кожний спосіб реалізувати окремою функцією.
3. Вивести на екран елементи початкового масива. Якщо є кілька початкових масивів, то вивести усі. Якщо потрібно визначити певні індекси, значення тощо, то вивести їх. Якщо масив був змінений, то вивести модифікований масив.
4. **Використання глобальних змінних у підпрограмах не допускається**. Інформація у підпрограми повинна передаватися лише за допомогою параметрів.
5. Введення даних, виведення даних і виконання певних дій над масивами потрібно реалізувати в окремих функціях. У головній програмі потрібно виконувати лише їхній виклик. Введення-виведення даних супроводжувати відповідними повідомленнями.

6. Побудувати блок-схему алгоритму програми та функції, що виконує вказані дії над масивом.

№ з/п	Завдання
1.	Поміняти місцями найменший елемент масива з його останнім елементом.
2.	Замінити всі додатні елементи заданого рядка масива одиницями.
3.	Замінити найбільший елемент заданого стовпчика масива нулем.
4.	Обчислити суму елементів парних рядків масива.
5.	Зменшити всі елементи масива, більші за число K, на 10.
6.	Обчислити суму кожних двох сусідніх елементів заданого рядка масива.
7.	Замінити найменший елемент головної діагоналі масива нулем.
8.	Обчислити кількість невід'ємних елементів масива.
9.	Обчислити найбільший елемент масиву і відняти його від усіх елементів масива.
10.	Обчислити суму елементів заданого рядка масива, розміщених через один елемент, починаючи з першого.
11.	Записати у масив B елементи масива A у зворотному порядку.
12.	Записати на місце першого елемента масива його найбільший елемент.
13.	Обчислити середнє арифметичне додатних елементів масива.
14.	Обчислити найбільший серед від'ємних елементів заданого рядка масива.
15.	Обчислити найменший непарний елемент масива.
16.	Замінити всі від'ємні елементи масива нулями.
17.	Обчислити кількість нульових елементів масива.
18.	Обчислити кількість непарних елементів масива, кратних 3.
19.	Обчислити добуток додатних елементів заданого стовпчика масива.
20.	Обчислити суму елементів заданого рядка масива, які більші за 12.
21.	Обчислити суму додатних елементів масива.
22.	Обчислити середнє арифметичне значення парних елементів масива.
23.	Записати у масив B всі елементи масива A, менші за 10.
24.	Обчислити найбільший елемент бічної діагоналі масива.
25.	Записати у масив B всі додатні числа масива A, зберігаючи порядок їх слідування у заданому масиві.
26.	Записати у масив B всі від'ємні числа та нулі масива A, зберігаючи порядок їх слідування у заданому масиві.
27.	Записати у масив B всі ненульові елементи масива A.
28.	Збільшити у два рази всі елементи масива, менші за 20.
29.	Обчислити найбільший парний елемент масива A(5,5).
30.	Обчислити суму елементів масива, які кратні 5.

Завдання 4.

1. Написати програму для виконання певних дій над **статичним** двовимірним масивом, реалізувавши доступ до елементів масива двома способами: за допомогою індексів та вказівників. Кожний спосіб реалізувати окремою функцією.
2. Алгоритм формування початкового масива реалізувати двома способами: за допомогою введення даних з клавіатури та випадкової генерації чисел. Кожний спосіб реалізувати окремою функцією.
3. Вивести на екран елементи початкового масива. Якщо є кілька початкових масивів, то вивести усі. Якщо потрібно визначити певні індекси, значення тощо, то вивести їх. Якщо масив був змінений, то вивести модифікований масив.
4. **Використання глобальних змінних у підпрограмах не допускається**. Інформація у підпрограми повинна передаватися лише за допомогою параметрів.
5. Введення даних, виведення даних і виконання певних дій над масивами потрібно реалізувати в окремих функціях. У головній програмі потрібно виконувати лише їхній виклик. Введення-виведення даних супроводжувати відповідними повідомленнями.
6. Побудувати блок-схему алгоритму програми та функції, що виконує вказані дії над масивом.

№ з/п	Завдання
1.	Визначити, чи квадратна матриця цілих чисел симетрична відносно бічної діагоналі.
2.	Визначити, чи квадратна матриця цілих чисел є магічним квадратом, для якого суми елементів кожного рядка і стовпчика є однаковими.
3.	Елементи квадратної матриці цілих чисел, які не належать відрізку [a,b] записати в окремий вектор. Межі інтервала ввести з клавіатури.
4.	Знайти скалярний добуток рядка, в якому знаходиться найбільший елемент квадратної матриці дійсних чисел, на стовпчик з найменшим елементом. Значення елементів матриці не повинні повторюватися. Скалярний добуток – це сума добутків елементів рядка на відповідні елементи стовпчика.
5.	Поміняти місцями мінімальний та максимальний елементи (вважати, що такі елементи єдині) прямокутної матриці дійсних чисел.
6.	Утворити одновимірний масив, присвоївши його k-му елементу значення 1, якщо елементи k-го рядка прямокутної матриці дійсних чисел впорядковані за спаданням, і значення 0 – в іншому випадку.
7.	Відсортувати кожний рядок прямокутної матриці дійсних чисел за зростанням значень елементів.
8.	Відсортувати кожний стовпчик прямокутної матриці дійсних чисел за спаданням значень елементів.
9.	Визначити номери двох рядків прямокутної матриці дійсних чисел з найбільшим скалярним добутком. Скалярний добуток – це сума добутків

	двох елементів рядків, розміщених в одному стовпчику.
10.	Вилучити з прямокутної матриці дійсних чисел рядок з мінімальним значенням добутку усіх його елементів.
11.	Вилучити з прямокутної матриці дійсних чисел стовпчик з найбільшою сумою елементів.
12.	Поміняти місцями два рядки прямокутної матриці дійсних чисел з найбільшою і найменшою сумою елементів.
13.	Визначити кількість різних елементів прямокутної матриці цілих чисел (елементи, що повторюються рахуються один раз).
14.	Визначити кількості повторень значень елементів прямокутної матриці цілих чисел.
15.	Виконати циклічний зсув рядків прямокутної матриці цілих чисел на K позицій вниз. Значення K ввести з клавіатури. При циклічному зсуві вниз елементи останнього рядка матриці заносяться на відповідні місця зсунутих вниз елементів першого рядка.
16.	Виконати циклічний зсув стовпчиків прямокутної матриці цілих чисел на K позицій вправо. Значення K є цілим випадковим числом, що не перевищує 5. При циклічному зсуві вправо елементи останнього стовпчика матриці заносяться на відповідні місця зсунутих вправо елементів першого стовпчика.
17.	Вилучити з прямокутної матриці цілих чисел рядок і стовпчик, на перетині яких знаходиться максимальний елемент.
18.	Вивести на екран індекси усіх сідлових точок прямокутної матриці дійсних чисел. Елемент матриці називається сідловою точкою, якщо виконується умова: $\max_i \min_j a[i][j] = \min_j \max_i a[i][j]$.
19.	Знайти мінімальні елементи в кожному рядку прямокутної матриці дійсних чисел і записати їх в окремий одновимірний масив.
20.	Знайти максимальні елементи в кожному стовпчику прямокутної матриці дійсних чисел і записати їх в окремий одновимірний масив.
21.	Написати програму для множення прямокутної матриці дійсних чисел A[n][m] на вектор дійсних чисел B[m].
22.	Написати програму для множення прямокутної матриці дійсних чисел A[n][m] на матрицю дійсних чисел C[m][p].
23.	Знайти мінімальний елемент прямокутної матриці цілих чисел, який не входить до складу елементів вектора цілих чисел.
24.	Знайти мінімальний елемент квадратної матриці дійсних чисел серед елементів, які знаходяться нижче головної діагоналі.
25.	Знайти максимальний елемент квадратної матриці дійсних чисел серед елементів, які знаходяться нижче бічної діагоналі.
26.	Знайти максимальний елемент квадратної матриці дійсних чисел серед елементів, які знаходяться вище головної діагоналі.
27.	Знайти мінімальний елемент квадратної матриці дійсних чисел серед елементів, які знаходяться вище бічної діагоналі.
28.	Якщо сума елементів головної діагоналі квадратної матриці дійсних чисел більша від суми елементів бічної діагоналі, то виконати

	траспонування матриці – заміну рядків на відповідні стовпчики.
29.	Визначити чи квадратна матриця цілих чисел ортонормована, тобто така, в якій скалярний добуток кожної пари різних рядків дорівнює 0, а скалярний добуток кожного рядка на себе самого дорівнює 1. Скалярний добуток двох векторів однакової розмірності – це сума добутків їхніх елементів з однаковими індексами.
30.	Визначити, чи квадратна матриця цілих чисел симетрична відносно головної діагоналі.