



**ДЕРЖАВНИЙ УНІВЕРСИТЕТ  
ТЕЛЕКОМУНІКАЦІЙ**  
Кафедра інженерії програмного  
забезпечення



# Програмування C++

**Спеціальність:** 121 «Програмна інженерія»

**Лектор:** доцент Золотухіна О.А.

Одновимірний масив

Задачі на лінійну обробку одновимірних масивів

# Класифікація структур даних

- За кількістю елементів, які може вміщувати структура:
  - **Прості** (елементарні) – містять тільки один елемент і не можуть бути поділені на окремі складові (базові типи даних)
  - **Складні** – можуть містити більше одного елемента, їх складовими є інші структури даних
- За однорідністю даних в структурі:
  - **Регулярні** – всі елементи мають однаковий тип та структуру
  - **Нерегулярні** - елементи можуть відрізнятися за типом та структурою
- За способом зв'язку між елементами:
  - **Лінійні** – характер зв'язків лінійний (послідовний)
  - **Нелінійні** – характер зв'язків нелінійний
- За способом доступу до елементів:
  - **Індексовані** – доступ до елементів за одним чи декількома індексами (номерами)
  - **Неіндексовані** – доступ до елементів за іменем
- За порядком розташування елементів:
  - **Впорядковані** – порядок розташування всередині структури даних має значення
  - **Невпорядковані** – порядок елементів не має значення
- За мінливістю:
  - **Статичні** – кількість елементів та зв'язки між ними не змінюються
  - **Напівстатичні** – кількість елементів може змінюватися
  - **Динамічні** – може змінюватися і кількість елементів, і зв'язки між ними
- За типом пам'яті, яка використовується для збереження даних:
  - **Внутрішні** - структури для оперативної пам'яті
  - **Зовнішні** - структури для зовнішньої пам'яті

# Одновимірний масив

**Масив** – це впорядкований іменований набір із фіксованої кількості однотипних даних

**Властивості структури даних «одновимірний масив»:** складна, регулярна, лінійна, індексована, впорядкована, статична, внутрішня

**Синтаксис оголошення:**

*назва\_типу ідентифікатор [цілочисельна\_константа];*

*назва\_типу* – тип елемента масиву

*Ідентифікатор* – ім'я масиву

*цілочисельна\_константа* – кількість елементів масиву

**Зауваження:** вказаним способом оголошуються масиви фіксованої довжини

# Приклади оголошень одновимірного масиву

*Просте оголошення:*

```
int x [10]; // Одновимірний масив з 10 цілих чисел
```

```
int N=10;
```

```
int y [N]; //Помилка – розмір масиву не може бути визначений через змінну
```

*Оголошення з ініціалізацією:*

```
int a[3] = {0, 1, 2}; // кількість ініціалізаторів дорівнює кількості елементів
```

```
double b[5] = {0.1, 0.2, 0.3}; // кількість ініціалізаторів менше кількості елементів
```

```
int c [] = {1, 2, 4, 8, 16}; // кількість елементів масиву визначається за кількістю ініціалізаторів
```

```
int e[3] = {0, 1, 2, 3}; // Помилка - кількість ініціалізаторів більше кількості елементів
```

# Звернення до елементу одновимірного масиву

Оскільки масив є індексованою структурою, то звернення до елементу відбувається із використанням його номеру

Синтаксис звернення до елементу одновимірного масиву:

*ім'я\_масиву [цілочисельний\_вираз]*

```
int x [5];  
for (int i=0; i<5; i++)  
    x[i]=i+1;
```

0	1	2	3	4
1	2	3	4	5

x – ім'я масиву

i – номер елементу в масиві x

x[i] – значення, яке зберігається в масиві x під номером i



0

1

2

3

4

5

6

7

# Зауваження щодо одновимірних масивів

- Оскільки індексація елементів масиву в мовах сімейства С починається з нуля, останній елемент масиву має індекс на одиницю менше, ніж число елементів масиву
- У мові С++ немає можливості вводити і виводити весь масив одним оператором вводу/виводу (крім рядків). Можна вводити і виводити тільки один елемент масиву. Отже, для того щоб ввести весь масив, треба використовувати цикл.
- Не існує операції присвоювання масиву, відповідного описаному раніше способу ініціалізації:

```
int a[3] = {0, 1, 2};    // Оголошення і ініціалізація
int b[3];
b = {0, 1, 2};          // Помилка
```



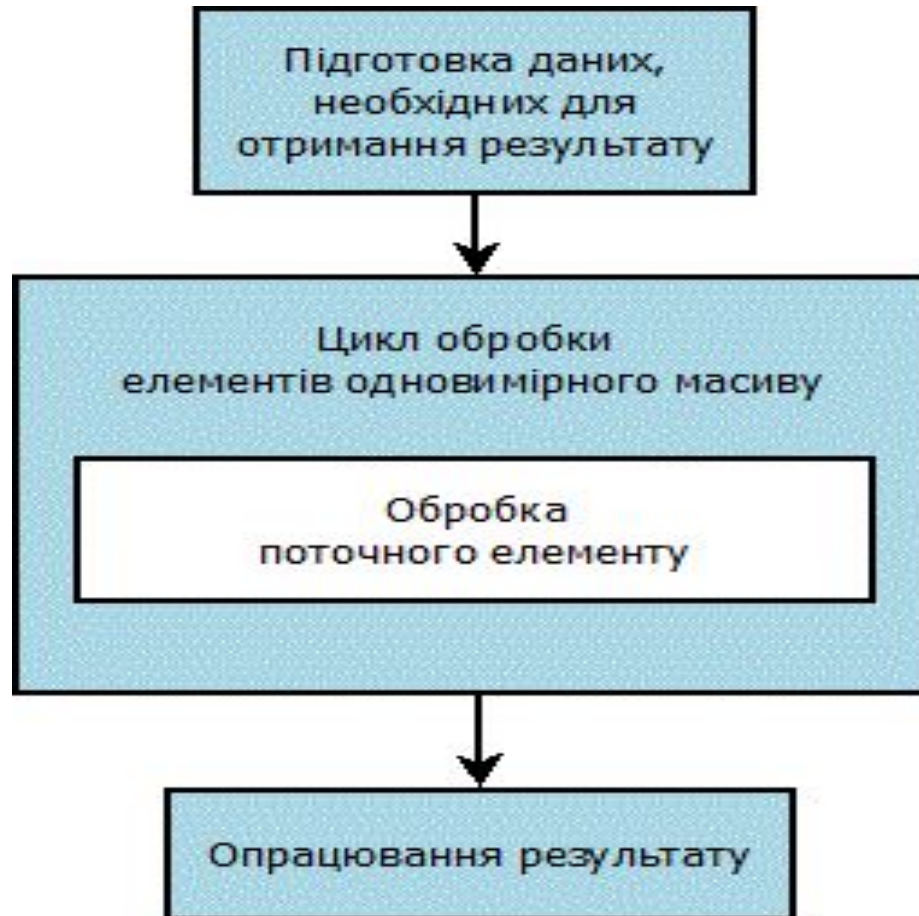
# Особливості обробки одновимірних масивів

- Можна розділити введення даних та їх обробку
- Можна змінювати та обробляти дані, застосовуючи нелінійні схеми обробки
- Лінійна схема обробки масиву практично така сама, як і у випадках обробки послідовності
- Нелінійні схеми обробки залежать від методу вирішення задачі і в кожному окремому випадку різні

## **Важливо!**

- **Якщо схема обробки послідовності елементів є лінійною, то замість масиву з метою економії пам'яті можна використовувати одну буферну змінну для збереження тільки поточного значення послідовності**

# Лінійна схема обробки одновимірного масиву



# Приклади задач обробки одновимірних масивів

# Введення елементів одновимірного масиву з N елементів

```
//блок підготовки даних для отримання результату відсутній  
for (int i = 0; i < N; i++) //цикл обробки елементів одновимірного  
масиву  
{  
    cout << "a" << i + 1 << " = "; //обробка поточного елемента  
    cin >> a[i];      // масиву полягає в його введенні  
}  
//блок опрацювання результату відсутній
```

# Виведення в стовпчик елементів одновимірного масиву довжиною N

//блок підготовки полягає у попередньому формуванні елементів масиву будь яким способом – введення, генерація, розрахунок, тощо

```
for (int i = 0; i < N; i++) //цикл обробки елементів  
одновимірного масиву
```

```
    cout << "a[" << i + 1 << "] = " << a[i] << endl;
```

//обробка поточного елементу масиву – виведення підказки та самого значення елементу

//блок опрацювання результату відсутній

# Заповнити одновимірний масив довжиною N числами починаючи з числа K з кроком h

// блок підготовки полягає у попередньому формуванні  
значень чисел K та h,

// вони можуть буди генеровані, введені або розраховані

```
for (int i = 0; i < N; i++) //цикл обробки елементів  
одновимірного масиву
```

```
    a[i] = K + i*h; //обробка поточного елемента масиву –  
формування значення за формулою
```

//блок опрацювання результату може полягати у виведенні  
масиву чи в подальшому його використанні для інших задач

# Сума елементів одновимірного масиву з N елементів

Використовується метод накопичення

```
S = 0; // підготовка початкового значення для розрахунку  
суми
```

```
for (int i = 0; i < N; i++) //цикл для обробки елементів  
одновимірного масиву
```

```
    S += a[i]; //обробка поточного елементу - додавання  
його до суми
```

```
cout << "Sum = " << S << endl; //опрацювання результату -  
виведення значення суми
```

# Кількість позитивних елементів одновимірного масиву з N елементів

Використовується метод накопичення

```
k = 0; // підготовка початкового значення для розрахунку кількості
```

```
for (int i = 0; i < N; i++) //цикл для обробки елементів  
одновимірного масиву
```

```
    if (a[i] > 0) //обробка поточного елементу – перевірка  
елементу та зміна значення кількості k
```

```
        k++;
```

```
cout << "Positive numbers = " << k << endl; //опрацювання  
результату – виведення кількості
```



# Мінімум одновимірного масиву з N елементів

Використовується метод пошуку мінімуму/максимуму

```
min = a[0]; // підготовка початкового значення для розрахунку
мінімуму

for (int i = 1; i < N; i++) //цикл для обробки елементів
одновимірного масиву

// елемент з номером 0 не обробляється повторно, починаємо з
наступного

    if (a[i] < min) //обробка поточного елементу – перевірка
елементу та зміна значення мінімуму

        min = a[i];

cout << "Min =" << min << endl; //опрацювання результату –
виведення мінімуму
```

# Замінити в одновимірному масиві з N елементів нулі на одиниці

//блок підготовки результату полягає у формуванні елементів масиву: введення, генерація тощо

```
for (int i = 0; i < N; i++) //цикл для обробки елементів одновимірного масиву
```

```
    if (a[i] == 0) //обробка поточного елемента - перевірка елемента
```

```
        a[i] = 1; //та зміна його значення
```

//блок опрацювання результату може полягати у виведенні масиву чи в подальшому його використанні для інших задач