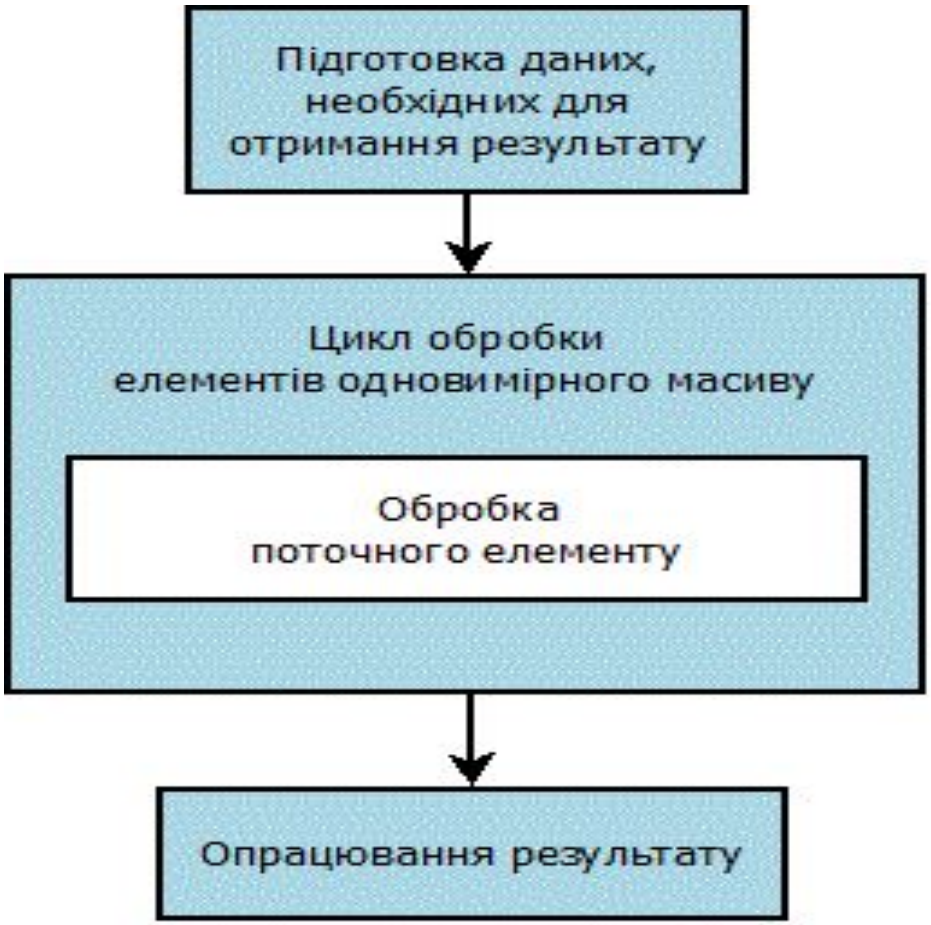
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
Кафедра інженерії програмного забезпечення

Програмування С++

Спеціальність: 121 «Програмна інженерія"

Лектор: доцент Золотухіна О.А.

Нелінійна обробка одновимірних масивів



Лінійна схема обробки одновимірного масиву

ЛІНІЙНА СХЕМА ОБРОБКИ

- Використовується для тих випадків, коли елементи масиву перебираються послідовно із певним кроком
- Для обробки застосовується **один цикл**
- **Схема** є універсальною і застосовується **одна для всіх задач**

НЕЛІНІЙНІ СХЕМИ ОБРОБКИ

- Передбачають обробку елементів не в послідовному порядку (але це не означає відсутність закономірності!)
- Для обробки може застосовуватися **будь-яка кількість циклів**, в тому числі, вкладені цикли або декілька послідовних циклів
- **Для кожної задачі будується своя схема**, яка відображає спосіб вирішення

Як визначити схему – лінійна чи нелінійна?

- Провести аналіз підзадач, з яких складається задача
- Проаналізувати, як змінюються індекси елементів масиву в процесі обробки (промоделювати вирішення задачі вручну)
- Для **лінійної схеми** послідовність індексів може бути представлена деякою прогресією
- Для **нелінійної схеми** представити послідовність індексів у вигляді однієї прогресії неможливо!

Приклад 1

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Обчислити суму елементів одновимірного цілочисельного масиву з 10 елементів.

Порядок обробки елементів (індекси):

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (арифметична прогресія від 0 до 9 з кроком +1)

або

9, 8, 7, 6, 5, 4, 3, 2, 1, 0 (арифметична прогресія від 9 до 0 з кроком -1)

Схема обробки – лінійна

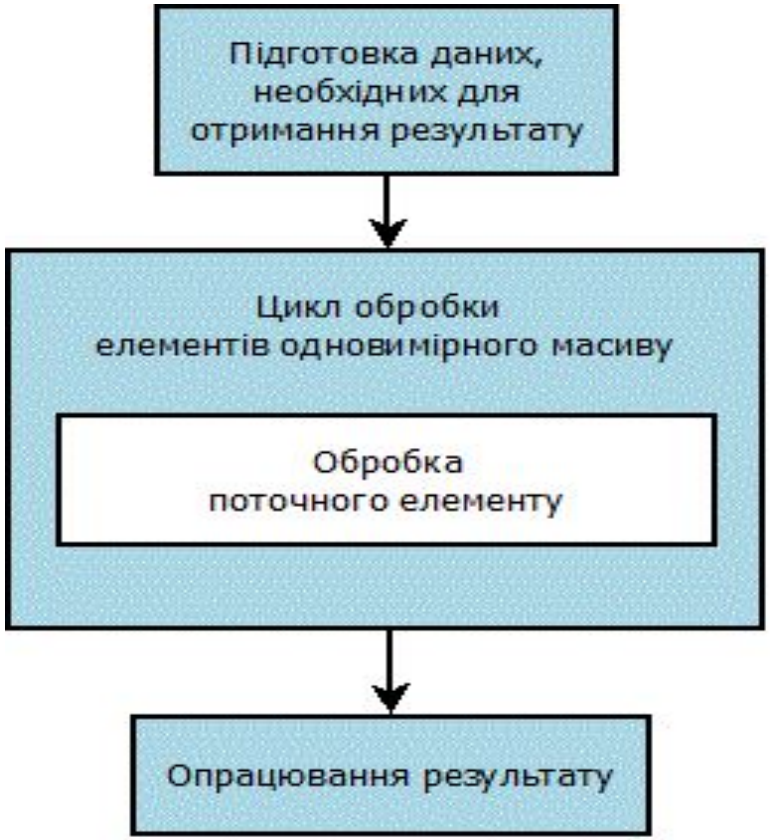
Цикл для обробки:

```
for (int i = 0; i < 10; i++)
```

або

```
for (int i = 9; i >=0; i--)
```

Приклад 1. Реалізація (лінійна схема)



Використовується метод накопичення

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

```
 for (int i = 0; i < 10; 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
```

Приклад 2

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Визначити, чи є в масиві з 10 цілих чисел задане число X.

Порядок обробки елементів (індекси):

*0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (арифметична прогресія від 0 до **9** з кроком +1)*

або

*9, 8, 7, 6, 5, 4, 3, 2, 1, 0 (арифметична прогресія від 9 до **0** з кроком -1)*

Схема обробки – лінійна

Цикл для обробки (?):

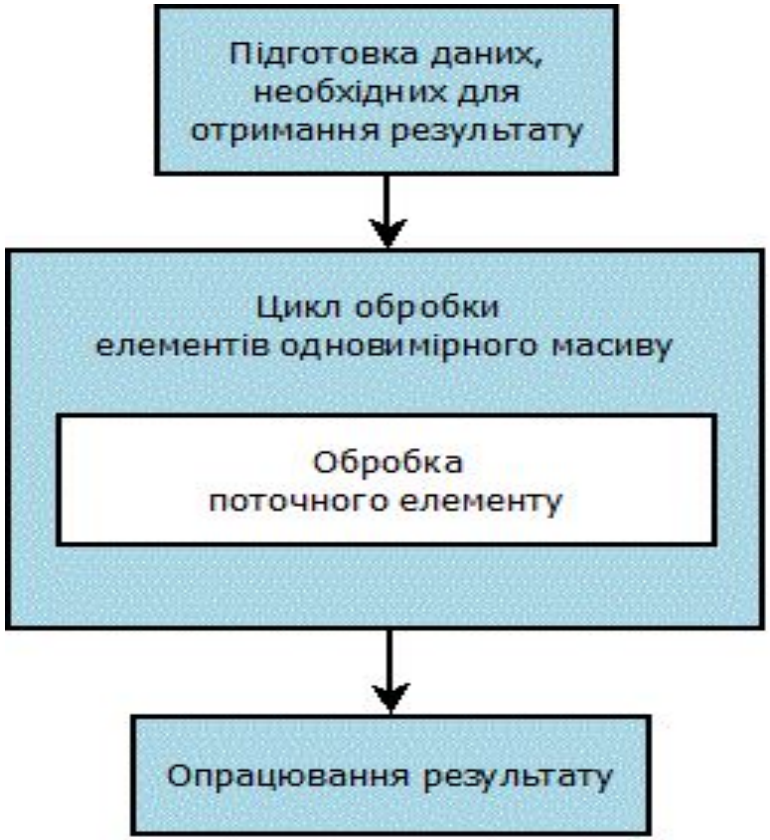
```
for (int i = 0; i < 10; i++)
```

або

```
for (int i = 9; i >=0; i--)
```

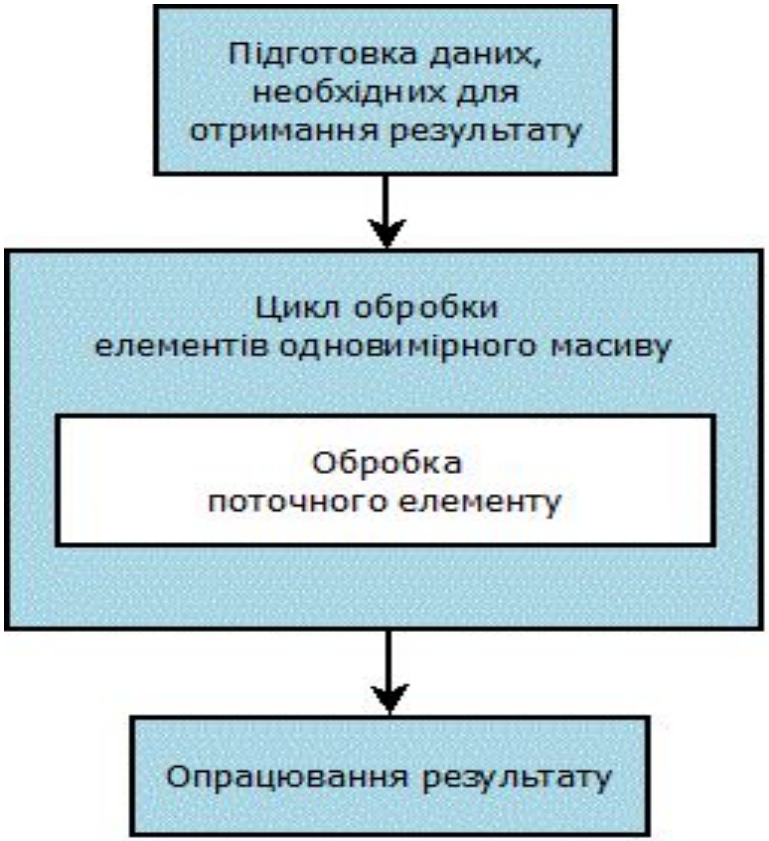
або … ???

Приклад 2. Реалізація (лінійна схема) Варіант 1 – використання методу лінійного пошуку




```
int i = 0; // починаємо пошук з початку масиву


//цикл для обробки елементів одновимірного масиву

while(i < 10 && a[i]!=X) //поки не дійшли до кінця масиву та не знайдено елемент, що дорівнює X, перебираємо елементи масиву

    i++;  //обробка поточного елементу полягає в переході до наступного елементу


//опрацювання результату – перевірка місця зупинки циклу: якщо зупинились в межах масиву, то в цій позиції знайдено X

if (i < 10) cout << "X found in position " << i <<endl;

else cout << "X not found in array"<<endl;
```

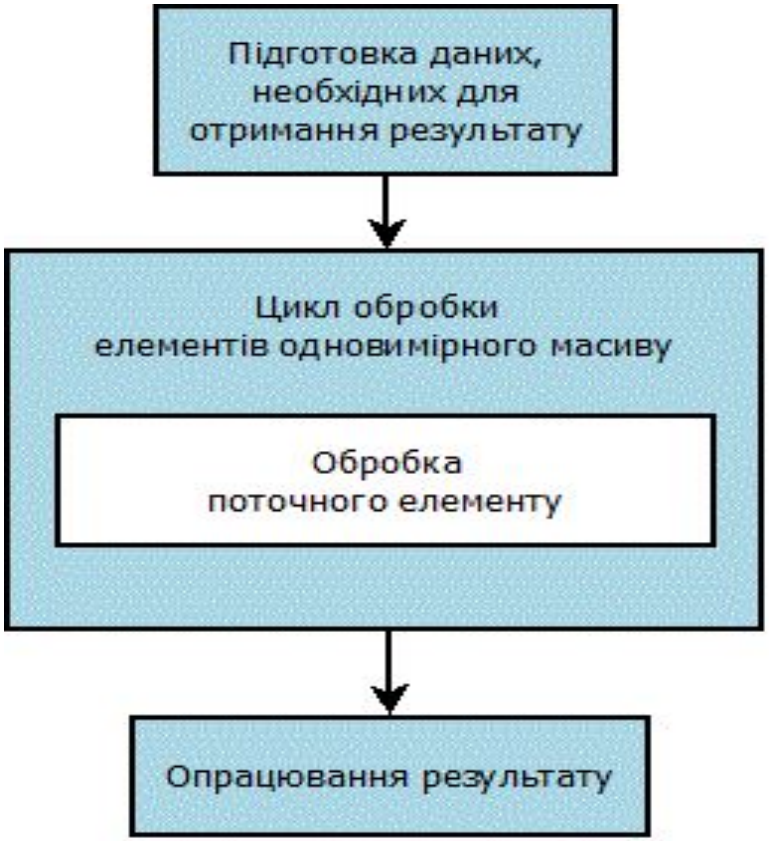
Приклад 2. Реалізація (лінійна схема) Варіант 2 – використання методу накопичення (рахуємо, скільки разів в масиві зустрічається X)




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


 for (int i = 0; i < 10; i++) //цикл для обробки елементів одновимірного масиву

      if (a[i] == X) //обробка поточного елементу – перевірка елементу та накопичення кількості

            k++;


//опрацювання результату – перевірка кількості знайдених значень X

if (k > 0) cout << "X found in array" <<endl;

else cout << "X not found in array"<<endl;
```

***Зауваження**: алгоритм може бути доповнений перевіркою k > 0 всередині циклу, що дозволить припинити перебір елементів масиву, як тільки знайдено значення X. Але для випадку, коли в масиві все ж таки немає значення X, така модифікація призводить до збільшення обчислювальної складності*

Приклад 2. Реалізація (лінійна схема) Варіант 3 – використання прапорця (змінна-прапорець змінює своє значення, коли знайдено X)




```
// прапорець flag має 2 значення: 0 відповідає ситуації, коли значення X відсутнє в масиві, 1 відповідає ситуації, коли X є в масиві

flag = 0;// припускаємо, що в масиві немає значення X


 for (int i = 0; i < 10 && fjag==0; i++) //цикл для обробки елементів одновимірного масиву: перебираємо елементи масиву, поки вони не закінчаться і прапорець не змінить свого значення

      if (a[i] == X) //обробка поточного елементу – перевірка елементу та зміна прапорця

            flag=1;

//опрацювання результату – перевірка прапорця

if (flag == 1) cout << "X found in array" <<endl;

else cout << "X not found in array"<<endl;
```

***Зауваження**: в загальному випадку прапорець може приймати будь-які значення – кількість значень залежить від кількості ситуацій, що обробляються. Для вказаної задачі прапорець може бути змінною логічного типу.*

Приклад 3

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Визначити, чи впорядкований масив з 10 цілих чисел за зростанням елементів.

Порядок обробки елементів (індекси):

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (арифметична прогресія від 0 до 9 з кроком +1 ?)

або

9, 8, 7, 6, 5, 4, 3, 2, 1, 0 (арифметична прогресія від 9 до 0 з кроком -1 ?)

Схема обробки – ?

Цикл для обробки (?):

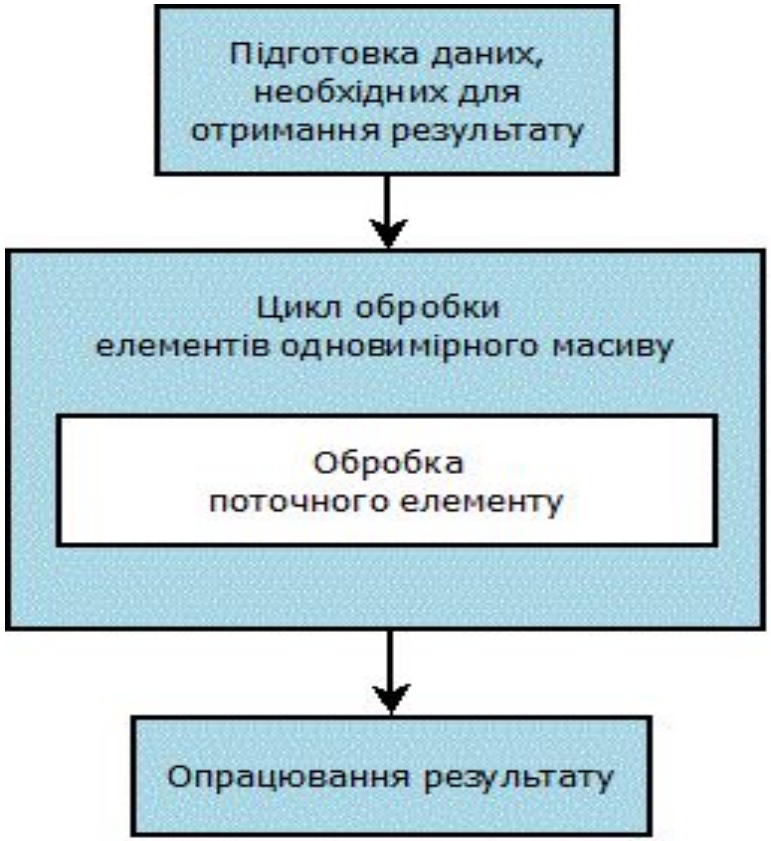
```
for (int i = 0; i < 10; i++)
```

або

```
for (int i = 9; i >=0; i--)
```

Приклад 3. Реалізація (лінійна схема)

Задача визначення впорядкованості масиву може бути зведена до задачі з прикладу 2, але в якості X виступає не якесь конкретне значення в масиві, а ситуація «чи присутня в масиві невпорядкована за зростанням пара сусідніх елементів». Тобто, маємо справу зі звичайним лінійним пошуком.




```
int i = 0; // починаємо пошук з початку масиву


//цикл для обробки елементів одновимірного масиву

while(i < 9 && a[i]<a[i+1])    //поки не дійшли до кінця масиву та не знайдено невпорядковану пару, перебираємо елементи масиву. Оскільки розглядаються не окремі елементи, а пари, то рухаємось до передостаннього елементу (для останнього елементу пари не існує)

    i++;  //обробка поточного елементу полягає в переході до наступного елементу


//опрацювання результату – перевірка місця зупинки циклу: якщо зупинились в межах масиву, то масив впорядковано

if (i < 9) cout << "The array is sorted in ascending order" << endl;

else cout << " The array is not sorted in ascending order" << endl;
```

Приклад 4

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Визначити, чи є масив з 10 цілих чисел паліндромом (читається зліва направо та справа наліво однаково).

Порядок обробки елементів (індекси):

?

Схема обробки ?

Цикл для обробки ?

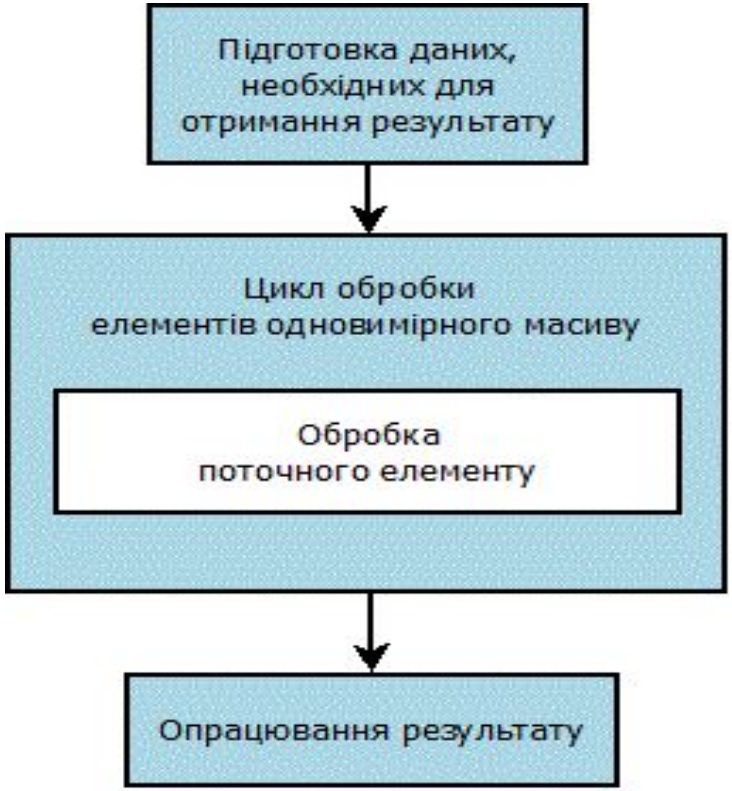
Приклад 4. Реалізація (лінійна схема)

Для побудови алгоритму спочатку проаналізуємо індекси елементів, що порівнюються між собою:

0 ↔ 9 1 ↔ 8 2 ↔ 7 … i ↔ 9-i …

Не важко помітити, що індекси обох елементів, які порівнюються, залежать лише від значення і, таким чином, маємо справу з лінійною схемою обробки.

В цілому задача може бути зведена до лінійного пошуку ситуації «знайдено неспівпадаючу пару» (приклади 2, 3). Ключова відмінність – діапазон даних, який обробляється, - нам треба порівняти лише половину масиву.




```
int i = 0; // починаємо пошук з початку масиву


//цикл для обробки елементів одновимірного масиву

while(i < 5 && a[i]==a[9-i])      //поки не дійшли до середини масиву та не знайдено неспівпадаючу пару, перебираємо елементи масиву. Рухаємось до середини масиву.

    i++;   //обробка поточного елементу полягає в переході до наступного елементу


//опрацювання результату – перевірка місця зупинки циклу: якщо зупинились в межах першої половини масиву, то масив містить не співпадаючу пару і не є паліндромом

if (i < 5) cout << "The array is not palindrom" << endl;

else cout << " The array is palindrom" << endl;
```

Приклад 5

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Замінити в масиві з 10 цілих чисел від’ємні елементи на середнє арифметичне всіх елементів масиву.

Порядок обробки елементів (індекси):

?

Схема обробки ?

Цикл для обробки ?

Приклад 5. Реалізація (декілька лінійних схем поспіль)

Задача розбивається на дві, кожна з яких має лінійну схему обробки.
Підзадача 1: розрахунок середнього арифметичного.
Підзадача 2: заміна елементів масиву.

```
// Підзадача 1
Avg = 0;// підготовка початкового значення для розрахунку середнього арифметичного
 for (int i = 0; i < 10; i++) //цикл для обробки елементів одновимірного масиву
      Avg += a[i];  //обробка поточного елементу – додавання його до суми
Avg/=10; //опрацювання результату для Підзадачі 1– розрахунок середнього арифметичного


// Підзадача 2
// блоком підготовки для неї виступає Підзадача 1
for (int i = 0; i < 10; i++) //цикл для обробки елементів одновимірного масиву
     if (a[i] < 0) //обробка поточного елементу – перевірка елементу та заміна на значення Avg
     a[i]=Avg;
//блок опрацювання результату для Підзадачі 2 може полягати у виведенні масиву чи в подальшому його використанні для інших задач
```

Приклад 6

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Для масиву з 10 цілих елементів побудувати всі можливі комбінації цих елементів.

Порядок обробки елементів (індекси):

?

Схема обробки ?

Цикл для обробки ?

Приклад 6. Реалізація (схема обробки нелінійна)

Для побудови алгоритму спочатку проаналізуємо індекси елементів, що комбінуються між собою:

$0 \leftrightarrow 0$ $0 \leftrightarrow 1$ $0 \leftrightarrow 2$ … $0 \leftrightarrow 9$

$1 \leftrightarrow 0$ $1 \leftrightarrow 1$ $1 \leftrightarrow 2$ … $1 \leftrightarrow 9$

…

$9 \leftrightarrow 0$ $9 \leftrightarrow 1$ $9 \leftrightarrow 2$ … $9 \leftrightarrow 9$

Пари індексів не утворюють лінійну послідовність, тому для обробки масиву неможливо застосувати лінійну схему (треба більше одного циклу обробки). Якщо позначити перший індекс через *i*, а другий через *j*, то маємо наступну закономірність: для кожного значення індексу *i* індекс *j* змінюється від 0 до 9. Таким чином, необхідно 2 вкладених цикли для обробки масиву. Оскільки індекс *i* змінюється повільніше, то він повинен бути в зовнішньому циклі.

```
for (int i = 0; i < 10; i++) //цикл для зміни індексу першого елементу в парі
    for (int j = 0; j < 10; j++) //цикл для зміни індексу другого елементу в парі
    cout << a[i] << " - " << a[j] << endl; //обробка елементів – виведення їх на екран
```

Приклад 7

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Для масиву з 10 цілих елементів побудувати всі можливі комбінації цих елементів (пари, що відрізняються тільки порядком слідування індексів, виводити один раз).

Порядок обробки елементів (індекси):

?

Схема обробки ?

Цикл для обробки ?

Приклад 7. Реалізація (схема обробки нелінійна)

Задача схожа на приклад 6, але має відмінність в діапазонах індексів. Проаналізуємо їх аналогічно попередньому прикладу (червоним позначено ті пари, які необхідно ***виключити з розгляду***):

0 ↔ 0 0 ↔ 1 0 ↔ 2 … 0 ↔ 9

1 ↔ 0 1 ↔ 1 1 ↔ 2 … 1 ↔ 9

2 ↔ 0 **2 ↔ 1** 2 ↔ 2 … 2 ↔ 9

…

9 ↔ 0 **9 ↔ 1** **9 ↔ 2** **…** 9 ↔ 9

Як і в прикладі 6, пари індексів не утворюють лінійну послідовність, тому для обробки масиву неможливо застосувати лінійну схему (треба більше одного циклу обробки). Якщо позначити перший індекс через *i*, а другий через *j*, то маємо наступну закономірність: для кожного значення індексу *i* індекс *j* змінюється від

***i* до 9**.

Таким чином, маємо 2 вкладених цикли для обробки масиву. Відмінність від прикладу 6 полягає в інших діапазонах зміни індексу *j*.

```
for (int i = 0; i < 10; i++) //цикл для зміни індексу першого елементу в парі
    for (int j = i; j < 10; j++) //цикл для зміни індексу другого елементу в парі
    cout << a[i] << " - " << a[j] << endl; //обробка елементів – виведення їх на екран
```

Приклад 8

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Визначити кількість унікальних елементів в одновимірному масиві з 10 цілих чисел.

Порядок обробки елементів (індекси):

?

Схема обробки ?

Цикл для обробки ?

Приклад 8. Реалізація (схема обробки нелінійна)

Загальний результат (кількість унікальних елементів масиву) передбачає використання методу накопичення:

1. Початкове значення кількості =0.
2. Перебираємо всі елементи масиву та обробляємо кожен наступним чином: якщо елемент унікальний, то збільшуємо кількість на 1.

Таким чином, маємо «зовнішню задачу» визначення кількості та «внутрішню задачу» визначення унікальності елементу, що передбачає використання вкладених циклів: зовнішній відповідає за перебір елементів, внутрішній (1 чи декілька циклів – в залежності від використаного методу) відповідає за процес визначення унікальності поточного елементу.

Для визначення унікальності можемо застосовувати метод лінійного пошуку, прапорець чи підрахунок кількості збігів (***див. приклад 2***).

Наведено варіант реалізації, який використовує лінійний пошук:
масив розбивається на 2 частини, в кожній з яких шукаємо збіг – до поточного елементу та після нього.

```
int Count = 0; //кількість унікальних елементів на початку дорівнює 0 (метод накопичення)


for (int i = 0; i < 10; i++) //цикл для перебирання всіх елементів масиву

{

	int j = 0; //починаємо звіряти поточний елемент з тими, які лівіше за нього

	while (j < i && a[i]!=a[j]) //поки не дійшли до поточного і-го елементу та не зустріли збіг

		j++; //переходимо до наступного елементу

	if (j < i) //якщо ми зупинились раніше, ніж досягнули поточного і-го елементу, то маємо збіг – значить, і-й елемент не унікальний, треба перейти до розгляду наступного

	   continue; //продовжуємо зовнішній цикл

	j=i+1; //продовжуємо шукати збіг у фрагменті справа від і-го елементу

	while (j < 9 && a[i]!=a[j])

		j++;

	if (j < 9) //якщо зупинились у фрагменті справа від і-го елементу, значить маємо збіг і елемент не унікальний

	   continue; //продовжуємо зовнішній цикл

	else  Count++; //в усіх інших випадках не маємо жодного збігу, значить збільшуємо кількість унікальних елементів

}

cout << "Number of unique elements " << Count << endl; //виведення результату
```

Приклад 9

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Зсунути елементи масиву з 10 цілих чисел циклічно на 1 вправо.

Порядок обробки елементів (індекси):

?

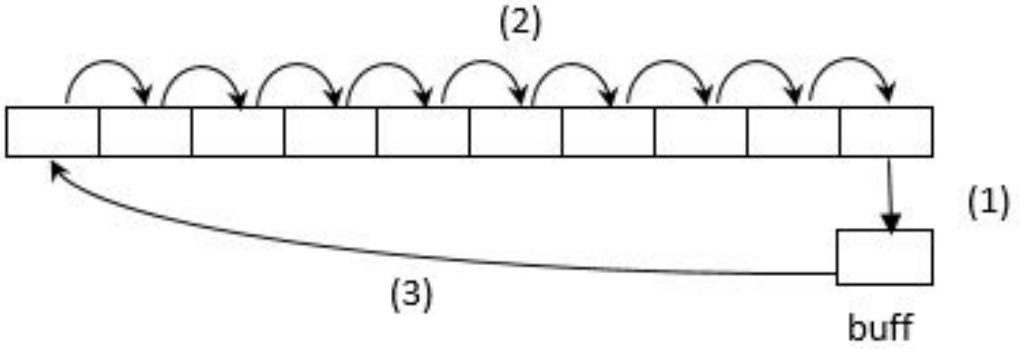
Схема обробки ?

Цикл для обробки ?

Приклад 9. Реалізація (схема обробки лінійна)

Циклічний зсув вправо передбачає декілька етапів (див. рис.):

1. Запам’ятати значення крайнього справа елементу в буфері.
2. Зсунути елементи вправо: кожен поточний елемент замінюється на елемент, що стоїть зліва від нього. Заміни треба робити починаючи з останнього елементу.
3. Записати в крайній зліва елемент масиву значення з буферу.




```
buff=a[9]; // запам’ятати значення крайнього справа елементу в буфері
for (int i = 9; i > 0; i--) //цикл для зсуву елементів вправо
    a[i]=a[i-1];
a[0]=buff; //записати в крайній зліва елемент масиву значення з буферу
```

Приклад 10

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Зсунути елементи масиву з 10 цілих чисел циклічно на 1 вліво.

Порядок обробки елементів (індекси):

?

Схема обробки ?

Цикл для обробки ?

Приклад 10. Реалізація (схема обробки лінійна)

Циклічний зсув вліво відбувається за схемою, подібною до циклічного зсуву вправо (відмінності позначено червоним):

1. Запам’ятати значення крайнього **зліва** елементу в буфері.
2. Зсунути елементи **вліво**: кожен поточний елемент замінюється на елемент, що стоїть **справа** від нього. Заміни треба робити починаючи з **першого** елементу.
3. Записати в крайній **справа** елемент масиву значення з буферу.

```
buff=a[0]; // запам’ятати значення крайнього зліва елементу в буфері
for (int i = 0; i < 9; i++) //цикл для зсуву елементів вправо
    a[i]=a[i+1];
a[9]=buff; //записати в крайній справа елемент масиву значення з буферу
```

Приклад 11

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Вставити в масив з 10 цілих чисел число X на позицію k (розмір масиву збільшиться на 1 елемент).

Порядок обробки елементів (індекси):

?

Схема обробки ?

Цикл для обробки ?

Приклад 11. Реалізація (схема обробки лінійна)

Вставка значення реалізується із використанням підзадачі зсуву на один елемент вправо: кожен поточний елемент починаючи з k-го замінюється на елемент, що стоїть зліва від нього.
Заміни треба робити починаючи з останнього елементу.
Після зсуву на позицію k записується значення X.

Важливо! *В подальших циклах, які обробляють даний масив, слід використовувати збільшений розмір масиву.*

```
for (int i = 9; i > k; i--) //цикл для зсуву елементів вправо
    a[i]=a[i-1];
a[k]=X; //записати на позицію k значення X
```

Приклад 12

12	7	32	11	1	0	43	2	-3	-15
0	1	2	3	4	5	6	7	8	9

Задача: Видалити число з позиції k в масиві з 10 цілих чисел (розмір масиву зменшиться на 1 елемент).

Порядок обробки елементів (індекси):

?

Схема обробки ?

Цикл для обробки ?

Приклад 12. Реалізація (схема обробки лінійна)

Видалення значення реалізується із використанням підзадачі зсуву на один елемент вліво: кожен поточний елемент починаючи з k-го замінюється на елемент, що стоїть справа від нього.
Заміни треба робити починаючи з крайнього зліва елементу.

Важливо! *В подальших циклах, які обробляють даний масив, слід використовувати зменшений розмір масиву.*

```
for (int i = k; i < 9; i++) //цикл для зсуву елементів вліво
    a[i]=a[i+1];
```