**ПРИКЛАДИ ВИРІШЕННЯ ЗАДАЧ ДЛЯ САМОСТІЙНОГО ОПРАЦЮВАННЯ**

**Обробка послідовностей та деякі задачі лінійної обробки одновимірних масивів**

Зверніть увагу, що в прикладах надано ***фрагменти коду!***

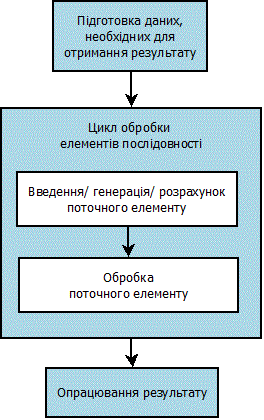
Для того, щоб отримати реальні результати, треба додати необхідні фрагменти, пов’язані із підключенням бібліотек, ініціалізацією відповідних змінних, введенням даних тощо.

**ОБРОБКА ПОСЛІДОВНОСТЕЙ.**

Головний нюанс вирішення задач на обробку послідовності, в якій поточний елемент представлено однією змінною, полягає в тому, що ***обробка даних завжди суміщена з введенням/генерацією/розрахунком даних***.

***Схема обробки даних послідовності завжди лінійна!***

Загальна схема обробки виглядає наступним чином:



1. Перед циклом обробки виконується підготовка даних, необхідних для формування результату. В залежності від задачі блок підготовки може бути відсутнім.
2. На кожному кроці циклу обробки відбувається формування поточного елементу та його обробка.
3. Після завершення циклу обробки відбувається опрацювання результату (виведення, перевірка+виведення, використання для вирішення інших задач тощо).
4. **Сума елементів послідовності довжиною N**

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

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

{

cout << "a"<<i<<" = ";

cin >> a; //введення поточного елементу

S += a; //обробка поточного елементу – додавання його до суми

}

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

1. **Кількість нулів в послідовності довжиною N**

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

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

{

cout << "a"<<i<<" = ";

cin >> a; //введення поточного елементу

if (a == 0) //обробка поточного елементу – перевірка елементу та

k++; //зміна значення кількості k

}

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

1. **Мінімум в послідовності довжиною N**

cout << "a1 = "; // підготовка початкового значення для розрахунку мінімуму:

cin >> a; // перший елемент послідовності обробляється окремо

min = a;

for (int i = 2; i <= N; i++) //цикл для обробки елементів послідовності,

//перший елемент не обробляється повторно, починаємо з другого

{

cout << "a"<<i<<" = ";

cin >> a; //введення поточного елементу

if (a < min) //обробка поточного елементу – перевірка елементу та

min = a; //зміна значення мінімуму

}

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

1. **Максимум в послідовності довжиною N**

cout << "a1 = "; // підготовка початкового значення для розрахунку максимуму:

cin >> a; // перший елемент послідовності обробляється окремо

max = a;

for (int i = 2; i <= N; i++) //цикл для обробки елементів послідовності:

//перший елемент не обробляється повторно, починаємо з другого

{

cout << "a"<<i<<" = ";

cin >> a; //введення поточного елементу

if (a > max) //обробка поточного елементу – перевірка елементу та

max = a; //зміна значення максимуму

}

cout << "Max =" << max <<endl; //опрацювання результату – виведення максимуму

1. **Середнє арифметичне послідовності довжиною N**

Avg = 0; // підготовка початкового значення для розрахунку середнього арифметичного

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

{

cout << "a"<<i<<" = ";

cin >> a; //введення поточного елементу

Avg += a; //обробка поточного елементу – формування суми елементів

}

cout << "Avg =" << Avg/N <<endl; //опрацювання результату – розрахунок та виведення

//середнього арифметичного

1. **Середнє арифметичне ненульових елементів послідовності довжиною N**

Avg = 0; // підготовка початкових значень для розрахунку середнього арифметичного

k = 0; // ненульових елементів: для розрахунку необхідно знайти суму та

// кількість ненульових елементів

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

{

cout << "a" << i << " = ";

cin >> a; //введення поточного елементу

if (a != 0) //обробка поточного елементу

{

Avg += a; // формування суми ненульових елементів

k++; // та їх кількості

}

}

if (k == 0) //опрацювання результату – перевірка, чи існує середнє арифметичне ненульових

cout << "All elements are 0" << endl;

else

cout << "Avg =" << Avg / k << endl;

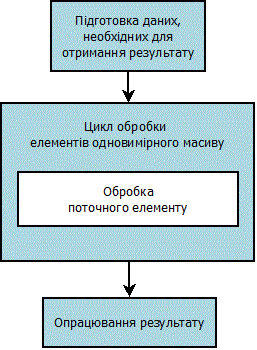
**ОБРОБКА ОДНОВИМІРНИХ МАСИВІВ.**

Однією з особливостей вирішення задач на обробку одновимірних масивів є те, що ми ***можемо розділити введення даних та їх обробку***. Крім того, в масиві, на відміну від послідовності, представленої однією коміркою пам’яті, ми ***можемо змінювати та обробляти дані, застосовуючи нелінійні схеми обробки***.

Лінійна схема обробки масиву практично така сама, як і у випадку обробки послідовності.

***Нелінійні схеми залежать від методу вирішення задачі і в кожному окремому випадку різні.***

***Лінійна схема*** обробки одновимірного масиву виглядає наступним чином:

****

1. **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 елементу масиву

cin >> a[i]; // полягає в його введенні

}

//блок опрацювання результату відсутній

1. **Заповнити одновимірний масив довжиною N числами починаючи з числа К 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 елементу масиву – формування значення за формулою

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

1. **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

1. **Сума елементів одновимірного масиву з N елементів**

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

1. **Кількість позитивних елементів одновимірного масиву з N елементів**

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

k++; //зміна значення кількості k

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

1. **Мінімум одновимірного масиву з N елементів**

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 (min < a[i]) //обробка поточного елементу – перевірка елементу та

min = a[i]; //зміна значення мінімуму

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

1. **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 елементу – перевірка елементу та

a[i] = 1; //зміна його значення

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