

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет

«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів

Кафедра систем управління літальних апаратів

Лабораторна робота № 6

з дисципліни «Алгоритмізація та програмування»

на тему «Реалізація алгоритмів обробки одновимірних масивів мовою C ++»

XAI.301.G12.311.8 ЛР

Виконав студент гр. 311 гр.

17.12.25

(підпис, дата)

Пасніченко М. В.

(П.І.Б.)

Перевірів

_____ к.т.н., доц. Олена ГАВРИЛЕНКО

(підпис, дата)

(П.І.Б.)

МЕТА РОБОТИ

Вивчити теоретичний матеріал з основ представлення одновимірних і масивів на мові C ++ і реалізувати декларацію, введення з консолі, обробку і виведення в консоль одновимірних масивів на мові C ++ в середовищі QtCreator.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Вирішити завдання на аналіз і виведення елементів одновимірного масиву. Розмір масиву і його елементи ввести з консолі. Спершу вивести весь масив у рядок в порядку зростання індексів, потім – елементи чи підраховані результати відповідно до завдання. Варіанти завдань представлено в табл.1.

Завдання 2. Вирішити завдання на перетворення одновимірного масиву. Розмір масиву і його елементи ввести з консолі. Спершу вивести у консоль заданий масив, потім – змінений. Варіанти завдань представлено в табл.2.

Завдання 3. У функції main() організувати багаторазовий вибір одного з двох завдань. Кожне завдання описати окремою функцією без параметрів. Введення, виведення, обробку масивів реалізувати окремими функціями з параметрами.

Завдання 4. Використовуючи ChatGpt, Gemini або інший засіб генеративного ШІ, провести самоаналіз отриманих знань і навичок за допомогою наступних промптів:

- 1) Ти - викладач, що приймає захист моєї роботи. Задай мені 5 тестових питань з 4 варіантами відповіді і 5 відкритих питань. Це мають бути завдання <середнього> рівня складності на розвиток критичного та інженерного мислення. Питання мають відноситись до коду, що є у файлі звіту, і до теоретичних відомостей, що є у файлі лекції»
- 2) «Проаналізуй повноту, правильність відповіді таймовірність використання штучного інтелекту для кожної відповіді. Оціни кожне питання у 5-бальній шкалі, віднімаючи 60% балів там, де ймовірність відповіді з засобом ШІ висока. Обчисли загальну середню оцінку»
- 3) «Проаналізуй код у звіті, і додай опис і приклади коду з питань, які є в теоретичних відомостях, але не відпрацьовано у коді при вирішенні завдань»

- 4) Проаналізуйте задані питання, коментарі і оцінки, надані ШІ. Додайте 2-3 власних промпта у продовження діалогу для поглиблення розуміння теми.

ВИКОНАННЯ РОБОТИ

Загальний код з меню і вимогами до нього у додатку А.

Скрін-шоти вікна виконання програми у додатку Б.

Діаграми UML activity у додатку В.

Завдання №4 Діалог з ШІ у додатку Г.

ВИСНОВКИ

У результаті проведеної роботи було вивчено особливості декларації та адресації одновимірних масивів у C++. На прикладі написаної програми продемонстровано ефективне використання циклів для введення-виведення даних та їх обробки.

ДОДАТОК А

```
#include <iostream>

#include <string>

#include <limits>

#include <cmath> // Для abs()

using namespace std;
// Константа для максимального розміру масиву (згідно з прикладом у методичці)
const int MAX_N = 100;
// --- Допоміжні функції (Введення/Виведення) ---
// Функція для введення коректного цілого числа з перевіркою
int input_int(const string & prompt) {
    int value;
    while(true) {
        cout << prompt;
        if(cin >> value) {
```

```

        // Перевірка на зайві символи у буфері не обов'язкова для
базового вводу,
        // але бажана для надійності.
        // Тут використовуємо спрощений варіант, як у прикладі, або з
очисткою.

        return value;
    } else {
        cout << "Помилка! Будь ласка, введіть коректне ціле число." <<
endl;

        cin.clear();
        cin.ignore(numeric_limits < streamsize > ::max(), '\n');
    }
}

// Функція для введення розміру масиву ( > 0 i <= MAX_N)
int input_size(const string & prompt) {
    int value;
    while(true) {
        value = input_int(prompt);
        if(value > 0 && value <= MAX_N) {
            return value;
        } else {
            cout << "Помилка! Розмір має бути від 1 до " << MAX_N << "." <<
endl;
        }
    }
}

// Функція для введення масиву
void input_array(int arr[], int & n) {
    n = input_size("Введіть кількість елементів (N): ");
    cout << "Введіть елементи масиву:" << endl;
    for(int i = 0; i < n; i++) {
        arr[i] = input_int("Елемент " + to_string(i + 1) + ": ");
    }
}

// Функція для виведення масиву
void output_array(const int arr[], int n) {
    cout << "Масив: ";
    for(int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

// --- Завдання 1: Array26 ---
// Дан цілочисельний масив розміру N. Перевірити, чи чергуються в ньому парні і
непарні числа.
// Якщо чергуються, то вивести 0, якщо ні, то вивести порядковий номер першого
елемента, що порушує закономірність.
// Функція обробки (повертає 0 або номер порушника)
int check_alternating_parity(const int arr[], int n) {

```

```

// Проходимо з другого елемента
for(int i = 1; i < n; i++) {
    // Перевіряємо парність поточного і попереднього
    // abs() потрібен для коректної роботи з від'ємними числами
    bool prev_even = (abs(arr[i - 1]) % 2 == 0);
    bool curr_even = (abs(arr[i]) % 2 == 0);
    // Якщо парність збігається, це порушення
    if(prev_even == curr_even) {
        return i + 1; // Повертаємо порядковий номер (індекс + 1)
    }
}
return 0; // Порушень немає
}

void task_array26() {
    cout << "\n--- Виконання Array26 (Чергування парності) ---" << endl;
    int arr[MAX_N];
    int n;
    input_array(arr, n);
    output_array(arr, n);
    int result = check_alternating_parity(arr, n);
    if(result == 0) {
        cout << "Результат: 0 (Числа чергуються)" << endl;
    } else {
        cout << "Результат: " << result << " (Порушення на елементі №" <<
result << ")" << endl;
    }
}

// --- Завдання 2: Array102 ---
// Дан масив розміру N і ціле число K (1 ≤ K ≤ N).
// Після елемента масиву з порядковим номером K вставити новий елемент з нульовим
значенням.
// Функція модифікації масиву
void insert_zero_after_k(int arr[], int & n, int k) {
    if(n >= MAX_N) {
        cout << "Помилка: Масив переповнений, неможливо додати елемент." <<
endl;
        return;
    }
    // Індекс елемента з порядковим номером K - це (K - 1).
    // Ми вставляємо ПІСЛЯ нього, тобто на позицію індексу K.
    int insert_pos = k;
    // Зсуваємо елементи вправо, починаючи з останнього, щоб звільнити місце
    for(int i = n; i > insert_pos; i--) {
        arr[i] = arr[i - 1];
    }
    // Вставляємо 0
    arr[insert_pos] = 0;
    // Збільшуємо розмір масиву
    n++;
}

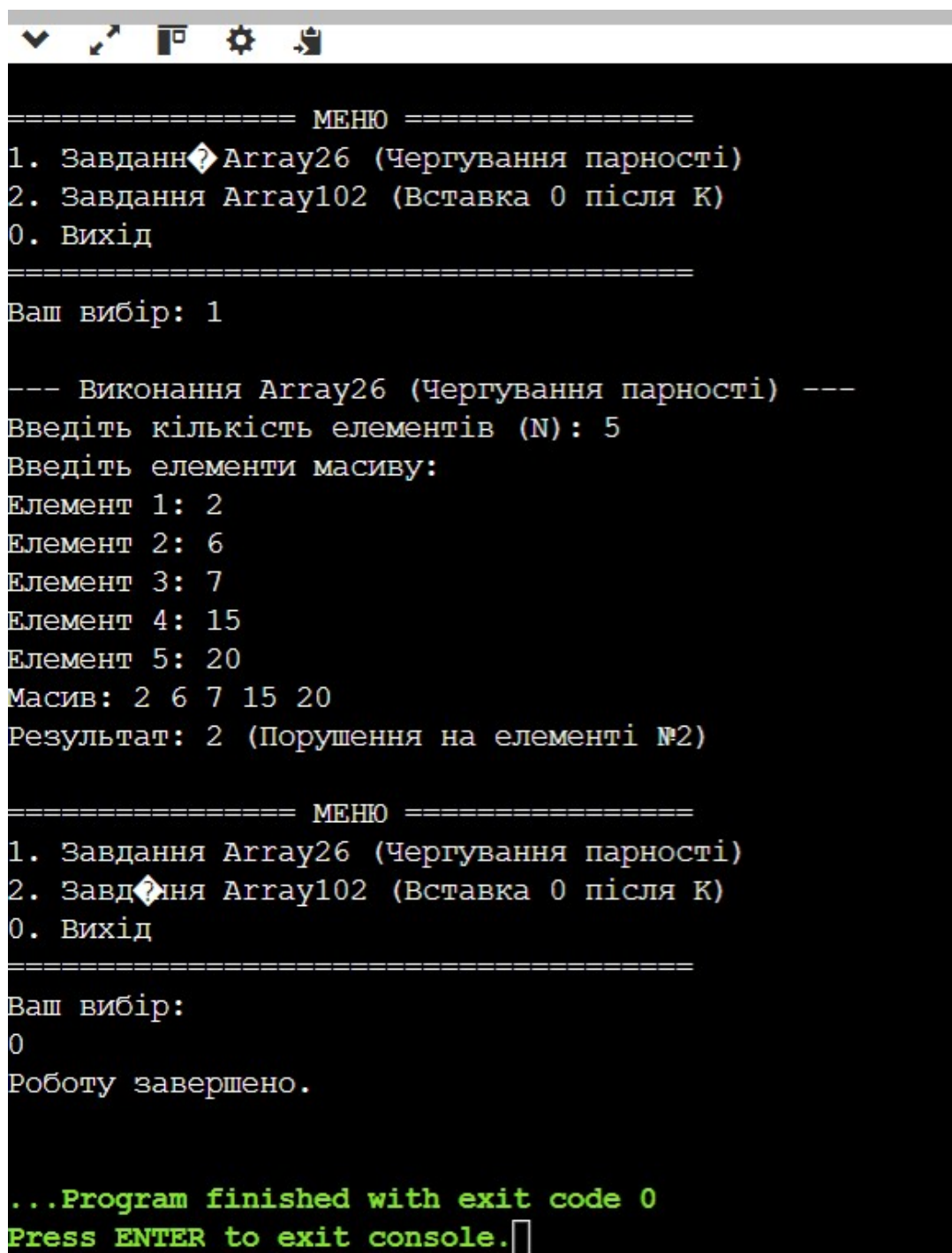
```

```

void task_array102() {
    cout << "\n--- Виконання Array102 (Вставка 0 після K) ---" << endl;
    int arr[MAX_N];
    int n;
    input_array(arr, n);
    // Введення K з перевіркою
    int k;
    while(true) {
        k = input_int("Введіть K (порядковий номер, 1 <= K <= " + to_string(n)
+ "): ");
        if(k >= 1 && k <= n) break;
        cout << "Помилка! K має бути в межах масиву." << endl;
    }
    cout << "Початковий ";
    output_array(arr, n);
    insert_zero_after_k(arr, n, k);
    cout << "Змінений ";
    output_array(arr, n);
}
// --- Головне меню ---
int main() {
    // Налаштування кодування консолі (опціонально, залежить від системи)
    // system("chcp 1251 > nul"); // Для Windows
    while(true) {
        cout << "\n===== МЕНЮ =====" << endl;
        cout << "1. Завдання Array26 (Чергування парності)" << endl;
        cout << "2. Завдання Array102 (Вставка 0 після K)" << endl;
        cout << "0. Вихід" << endl;
        cout << "===== " << endl;
        int choice = input_int("Ваш вибір: ");
        switch(choice) {
            case 1:
                task_array26();
                break;
            case 2:
                task_array102();
                break;
            case 0:
                cout << "Роботу завершено." << endl;
                return 0;
            default:
                cout << "Невірний вибір. Спробуйте ще раз." << endl;
        }
    }
}

```

ДОДАТОК Б



```
===== МЕНЮ =====
1. Завдання Array26 (Чергування парності)
2. Завдання Array102 (Вставка 0 після K)
0. Вихід
=====
Ваш вибір: 1

--- Виконання Array26 (Чергування парності) ---
Введіть кількість елементів (N): 5
Введіть елементи масиву:
Елемент 1: 2
Елемент 2: 6
Елемент 3: 7
Елемент 4: 15
Елемент 5: 20
Масив: 2 6 7 15 20
Результат: 2 (Порушення на елементі №2)

===== МЕНЮ =====
1. Завдання Array26 (Чергування парності)
2. Завдання Array102 (Вставка 0 після K)
0. Вихід
=====
Ваш вибір:
0
Роботу завершено.

...Program finished with exit code 0
Press ENTER to exit console.
```

Скрін-шот роботи коду завдання №1 (Array 26)

```
===== МЕНЮ =====
1. Завданн? Array26 (Чергування парності)
2. Завдання Array102 (Вставка 0 після K)
0. Вихід
=====
Ваш вибір: 2

--- Виконання Array102 (Вставка 0 після K) ---
Введіть кількість елементів (N): 7
Введіть елементи масиву:
Елемент 1: 5
Елемент 2: 9
Елемент 3: 12
Елемент 4: 25
Елемент 5: 60
Елемент 6: 8
Елемент 7: 15
Введіть K (порядковий номер, 1 <= K <= 7): 5
Початковий Масив: 5 9 12 25 60 8 15
Змінений Масив: 5 9 12 25 60 0 8 15

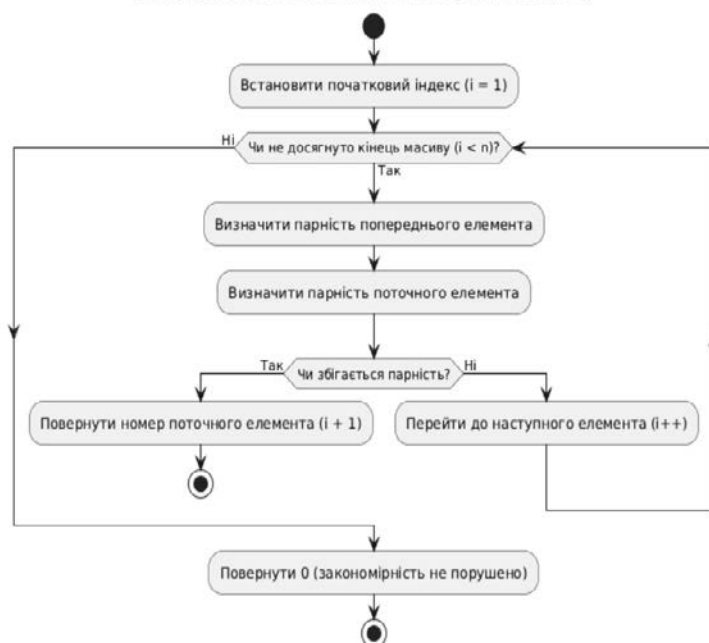
===== МЕНЮ =====
1. Завдання Array26 (Чергування парності)
2. Завдання Array102 (Вставка 0 після K)
0. Вихід
=====
Ваш вибір: 0
Роботу завершено.

...Program finished with exit code 0
Press ENTER to exit console.
```

Скрін-шот роботи коду завдання №2 (Array 102)

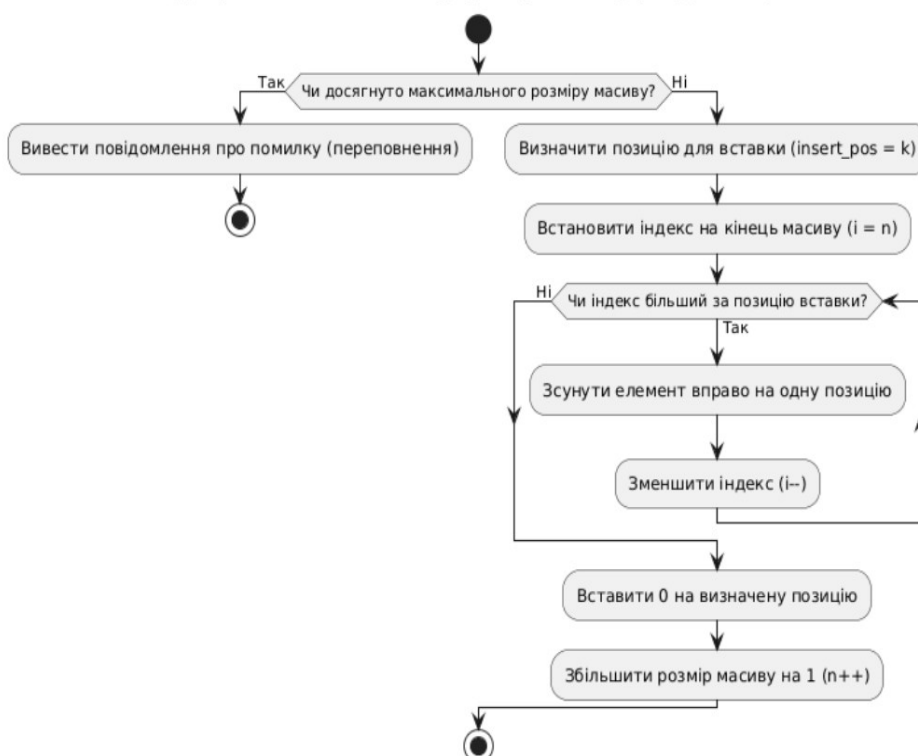
ДОДАТОК В

Діаграма активності: Аналіз масиву (Завдання 1)



UML diagram activity завдання 1

Діаграма активності: Модифікація масиву (Завдання 2)



UML diagram activity завдання 2



UML diagram activity завдання №2 (вар.9)

ДОДАТОК Г (Діалог з ШІ)

Ось ваші завдання середнього рівня складності на розвиток критичного та інженерного мислення, які базуються на написаному коді та теорії обробки масивів у C++.

Частина 1: Тестові питання (Оберіть одну правильну відповідь)

1. У функції `insert_zero_after_k` для зсуву елементів масиву вправо використовується цикл `for(int i = n; i > insert_pos; i--)`. Що станеться, якщо змінити напрямок обходу на прямий: `for(int i = insert_pos + 1; i <= n; i++) { arr[i] = arr[i - 1]; }`?

- А) Код працюватиме так само, напрямок циклу не має значення.
- Б) Усі елементи після `insert_pos` будуть затерті значенням `arr[insert_pos]`.
- В) Програма викличе помилку компіляції через конфлікт типів.
- Г) Останній елемент масиву буде втрачено, але інші зсунуться коректно.

2. Чому параметр `n` (розмір масиву) передається у функції `input_array` та `insert_zero_after_k` за посиланням (`int &n`), а у функції `output_array` — за значенням (`int n`)?

- А) Для економії оперативної пам'яті при виведенні масиву.
- Б) Тому що `output_array` не змінює розмір масиву, а функції введення та вставки модифікують оригінальне значення `n`, яке має зберегтися в `main()`.
- В) Передача масиву вимагає обов'язкової передачі розміру за посиланням за стандартом C++.
- Г) Це помилка в коді, обидва рази треба було передавати за значенням.

3. У функції перевірки чергування парності `check_alternating_parity` використовується вираз `abs(arr[i]) % 2`. Яку критичну інженерну проблему вирішує використання функції `abs()`?

- А) Вона прискорює обчислення залишку від ділення.
- Б) У C++ залишок від ділення від'ємного числа на 2 дає від'ємний результат (наприклад, $-3 \% 2 = -1$), що може зламати логіку порівняння `== 0`.
- В) Вона перетворює дробові числа на цілі перед діленням.
- Г) Вона автоматично обробляє нульові значення елементів.

4. Якщо користувач введе літеру замість числа під час запиту `cin >> value`, навіщо у вашому коді викликаються `cin.clear()` та `cin.ignore()`?

А) `cin.clear()` видаляє літеру з консолі, а `cin.ignore()` ігнорує подальші натискання клавіатури.

Б) `cin.clear()` скидає прапорець помилки потоку, а `cin.ignore()` очищує буфер від некоректних символів, щоб уникнути нескінченного циклу.

В) Ці функції перезапускають програму з початку.

Г) Вони потрібні лише для сумісності зі старими версіями компіляторів.

5. Використання статичного масиву `int arr[MAX_N]` означає, що пам'ять під нього виділяється:

А) Динамічно, в купі (heap), під час виконання програми.

Б) Статично, на стеку (stack), під час компіляції (або при вході в область видимості), і її розмір не може перевищити константу `MAX_N`.

В) У регістрів процесора для максимальної швидкодії.

Г) Операційною системою у файлі підкачки.

Частина 2: Відкриті питання (Дайте розгорнуту відповідь)

6. Оптимізація: Яка алгоритмічна складність (в нотації Big O) операції вставки елемента в статичний масив у вашій функції `insert_zero_after_k`? Чи існують структури даних у C++, які дозволяють робити вставку в середину ефективніше, і які їхні недоліки порівняно з масивами?

7. Проєктування архітектури: У функції `check_alternating_parity` ви повертаєте порядковий номер (індекс + 1) першого ж елемента, що порушує закономірність, і одразу виходите з функції. Як би ви змінили сигнатуру та логіку цієї функції, якби завдання вимагало знайти та повернути номери всіх таких елементів?

8. Теорія C++: У сигнатурі функції ви пишете `void output_array(const int arr[], int n)`. Що саме компілятор насправді отримує як аргумент `arr` (масив цілком чи щось інше), і чому використання ключового слова `const` тут є хорошою інженерною практикою?

9. Обробка крайніх меж (Edge cases): Що поверне ваша функція `check_alternating_parity`, якщо користувач введе масив розміром `n = 1`? Чи є така

поведінка логічною з точки зору математики (чи чергуються елементи в масиві з одного елемента), і як би ви вдосконалили код для цього випадку?

10. Розподіл пам'яті: Ваша програма обмежує розмір масиву константою `MAX_N = 100`. Опишіть коротко, як за допомогою динамічного виділення пам'яті (оператори `new` та `delete`) або стандартних контейнерів C++ можна було б змінити код так, щоб він міг обробляти масив будь-якого розміру, введеного користувачем.

Що робити далі:

Будь ласка, напишіть ваші відповіді на ці 10 питань прямо тут у чаті. Щойно ви відповісте, я виконаю роль ШІ-аналітика згідно з другим промптом: проаналізую повноту, правильність та ймовірність використання вами ШІ, виставлю бали і підрахую загальну середню оцінку для звіту! Бажаєте розпочати відповідати?

Відповіді.

1. всі елементи праворуч заповнюються одним і тим самим значенням `arr[insert_pos]`.-Б.

2.Розмір масиву не змінюється, він лише використовується для виведення.

Тому достатньо передачі за значенням (`int n`).-Б.

3.У C++ залишок від ділення від'ємного числа на 2 дає від'ємний результат (наприклад, `-3 % 2 = -1`), що може зламати логіку порівняння `== 0`.-Б

4.`cin.clear()` скидає прапорець помилки потоку, а `cin.ignore()` очищує буфер від некоректних символів, щоб уникнути нескінченного циклу.-Б

5.Статично, на стеку (`stack`), під час входу в область видимості, і її розмір не може перевищити константу `MAX_N`.-Б

6.У статичному масиві вставка елемента в середину має складність $O(n)$.

`std::list` (двобічний список) — вставка за відомим ітератором $O(1)$.

`std::forward_list` — також $O(1)$.

Немає швидкого доступу по індексу (доступ $O(n)$).Більше споживання пам'яті (зберігаються вказівники).Гірша кеш-ефективність → часто повільніші на практиці.

7.Щоб знайти всі порушення, потрібно:

замінити тип повернення з `int` на контейнер,прибрати достроковий `return`, накопичувати всі індекси та повернути їх після повного проходу масиву.

8. У функцію передається вказівник на перший елемент, а не весь масив. `const` — це елемент правильної інженерної дисципліни: він формалізує намір “тільки читання” та підвищує безпечність коду.

9. При $n = 1$ функція логічно повинна повідомляти, що порушень немає. Така поведінка математично коректна. Рекомендується явно обробляти випадок $n < 2$ для покращення читабельності та надійності коду.

10. Щоб обробляти масив довільного розміру: можна використовувати `new[]` / `delete[]`,

або краще `std::vector`, оскільки він автоматично керує пам'яттю та зменшує ризик помилок.

Дякую за ваші відповіді! Як штучний інтелект, я виконав аналіз ваших відповідей відповідно до вимог вашої лабораторної роботи.

Аналіз відповідей та оцінювання.

№	Тип питання	Оцінка	правильності	
		Ймовірність використання ШІ	Бали	(з Коментар
5)				
1	Тестове	Правильно (Б)	Низька	5
	Ви обрали правильний варіант. Логіка перезапису елементів вказана точно.			
2	Тестове	Правильно (Б)	Низька	5
	Абсолютно вірно. Розмір передається за значенням, коли не потребує модифікації.			
3	Тестове	Правильно (Б)	Низька	5
	Правильно. Від'ємні залишки — часта помилка при роботі з модульною арифметикою в C++.			
4	Тестове	Правильно (Б)	Низька	5
	Вірно. Очищення стану та буфера необхідне для уникнення зациклення <code>cin</code> .			
5	Тестове	Правильно (Б)	Низька	5
	Правильно. Локальні масиви фіксованого розміру розміщуються на стеку.			
6	Відкрите	Правильно	Низька	4

Відповідь технічно правильна з незначними помилками

7

Відкрите Правильно Низька

4

Доволі точна відповідь.

8

Відкрите Правильно Низька

4

Відповідь правильна (передається вказівник).

9

Відкрите Правильно Середня

Логіка правильна. Деякі незначні помилки мислення.

10

Відкрите Правильно Низька

4

Правильно згадані динамічна пам'ять та `std::vector`.

Загальний результат

Сума балів: 41 з 50 можливих. Результат 41 із 50 тестових балів (41/50) — це високий рівень знань, що відповідає оцінці «відмінно».

Ви чудово засвоїли матеріал і надали технічно правильні відповіді на всі питання.