Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

Звіт

про виконання лабораторних та практичних робіт блоку № 3

На тему: «Цикли. Вкладені Цикли. Завершення виконання циклів. Функції. Простір імен. Перевантаження функцій. Функції з змінною кількістю параметрів (еліпсис). Рекурсія. Вбудовані функції.»

з дисципліни: **«Основи програмування»**

до:

ВНС Лабораторної Роботи № 2

ВНС Лабораторної Роботи № 3

ВНС Лабораторної Роботи № 7

Практичних Робіт до блоку № 3

Виконала:

Студентка групи ШІ-13

Ходацька Аліна Віталіївна

Львів 2024

Тема роботи:

Одновимірні масиви. Двовимірні Масиви. Вказівники та Посилання. Динамічні масиви. Структури даних. Вкладені структури. Алгоритми обробки та робота з масивами та структурами.

Мета роботи:

Навчитися працювати з одновимірними, двовимірними та динамічними масивами, вказівниками та посиланнями, структурами та вкладеними структурами.

Теоретичні відомості:

1. Цикли: for, while, do-while
2. Застосування операторів break та continue
3. Вкладені цикли
4. Оголошення та виклик функцій
5. Передача параметрів
6. Повернення значення функцій
7. Перевантаження функцій
8. Функції зі змінною кількістю параметрів (еліпсис)
9. Рекурсія

<u>Опрацювання теоретичного матеріалу :</u>

1.https://cherto4ka.xyz/2018/10/21/%D1%86%D0%B8%D0%BA%D0%BB%D0%B8-%D0%B2-%D1%81/

2. https://acode.com.ua/urok-73-operatory-break-i-continue/

3. https://w3schoolsua.github.io/cpp/cpp_functions.html#gsc.tab=0

4. https://acode.com.ua/urok-15-funktsiyi-i-operator-return/

5. https://acode.com.ua/urok-108-perevantazhennya-funktsij/#toc-0

6. https://acode.com.ua/urok-113-rekursiya-i-chysla-fibonachchi/

Виконання роботи :

Завдання №1: Practice task “Менеджмент бібліотеки”

Створити просту програму керування бібліотекою. Книги в бібліотеці є, користувачі можуть їх взяти або повернути.

Програма повинна

- Перерахувати всі книги.
- Дозволити взяти книгу (за наявності).
- Дозволити повернення книги.

Структури даних

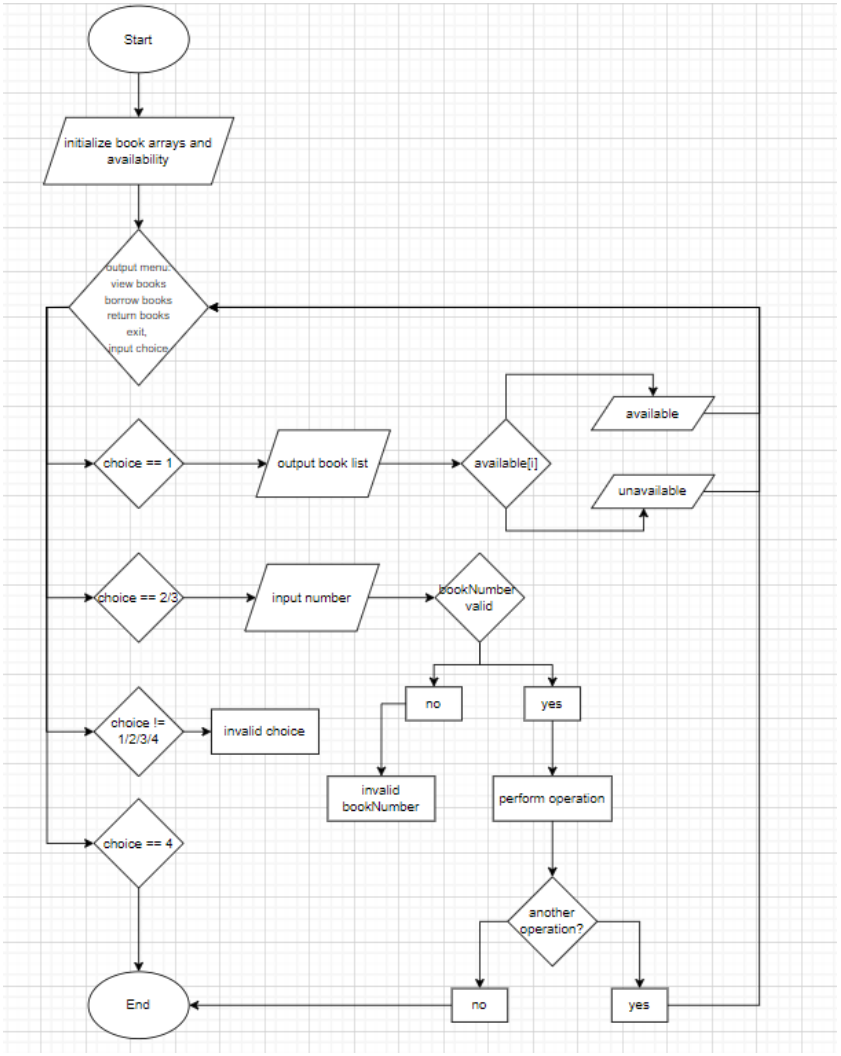
- Використовувати масив або вектор для зберігання назв книг.
- Використовувати інший масив або вектор для збереження стану доступності кожної книги.

Умови задачі :

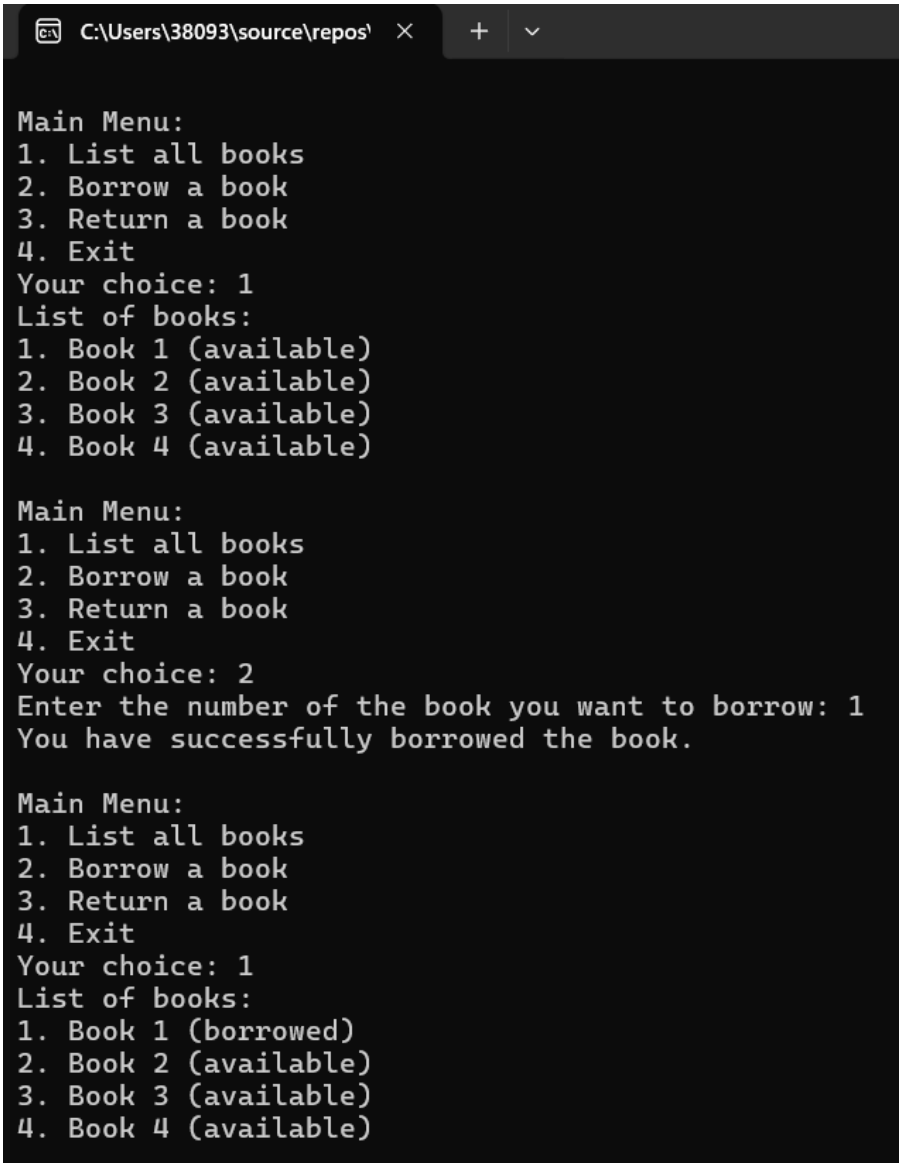
- while: продовжувати працювати, доки користувач не вирішить вийти.
- do while: Після кожної операції (позичити, повернути, перерахувати) запитувати користувача, чи хоче він виконати іншу операцію. Якщо так, повернутись назад.
- for: список усіх книг за допомогою циклу.
- for each: перевірити наявність кожної книги.
- goto: якщо користувач вводить неправильний вибір, використовувати goto, щоб перенаправити його до головного меню.

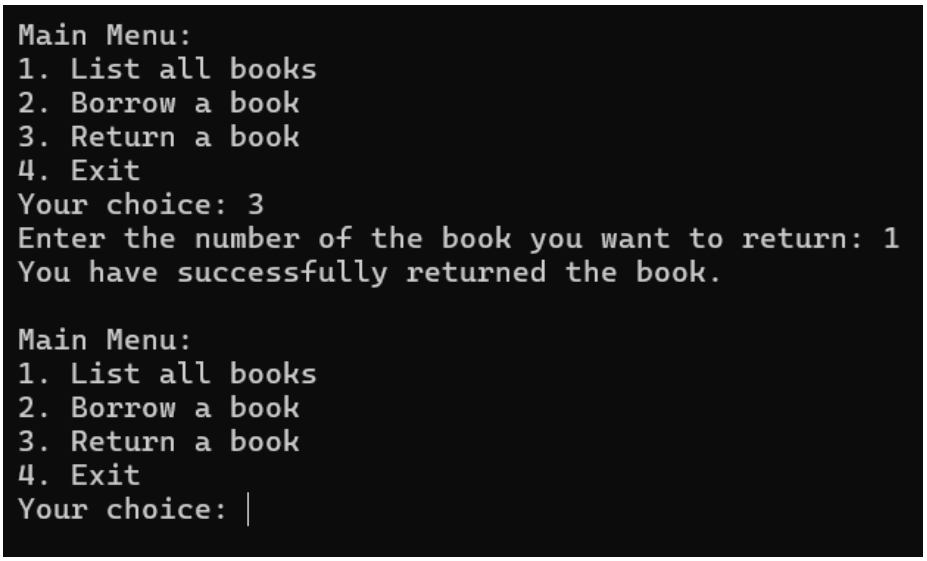
Запланований час: 1 год

Витрачений час: 1,5 год


Start
initialize book arrays and availability
output menu:
view books
borrow books
return books
exit,
input choice
choice == 1
output book list
available[i]
available
unavailable
choice == 2/3
input number
bookNumber valid
no
yes
invalid bookNumber
perform operation
another operation?
no
yes
choice != 1/2/3/4
invalid choice
choice == 4
End

```cpp
#include <iostream>
#include <vector>
#include <string>

void listBooks(const std::vector<std::string>& books, const std::vector<bool>& availability) {  //
оголошення та передача пареметру функції
    std::cout << "List of books:\n";
    for (size_t i = 0; i < books.size(); ++i) {   // цикл for для виведення списку книг
        std::cout << i + 1 << ". " << books[i] << (availability[i] ? " (available)\n" : " (borrowed)\n");
    }
}

void updateBookStatus(std::vector<bool>& availability, bool borrow) {   // оголошення та передача
пареметру функції
    int bookNumber;
    std::cout << "Enter the number of the book you want to " << (borrow ? "borrow" : "return") << ": ";
    std::cin >> bookNumber;

    if (bookNumber > 0 && bookNumber <= static_cast<int>(availability.size())) {   // оператор if else
для перевірки введеного номера книги
        if (availability[bookNumber - 1] == borrow) {
            availability[bookNumber - 1] = !borrow;
            std::cout << "You have successfully " << (borrow ? "borrowed" : "returned") << " the
book.\n";
        }
        else {
            std::cout << "This book is already " << (borrow ? "borrowed" : "available") << ".\n";
        }
    }
    else {
        std::cout << "Invalid book number.\n";
    }
}

int main() {
    std::vector<std::string> books = { "Book 1", "Book 2", "Book 3", "Book 4" };   // оголошення та
ініціалізація вектора
    std::vector<bool> availability(books.size(), true);   // оголошення та ініціалізація вектора

    while (true) {   // цикл while
        int choice;  // оголошення змінної
        std::cout << "\nMain Menu:\n";
        std::cout << "1. List all books\n";
        std::cout << "2. Borrow a book\n";
        std::cout << "3. Return a book\n";
        std::cout << "4. Exit\n";
        std::cout << "Your choice: ";
        std::cin >> choice;

        switch (choice) {
        case 1:
            listBooks(books, availability);  // виклик функції
            break;
        case 2:
            updateBookStatus(availability, true); // виклик функції
            break;
        case 3:
            updateBookStatus(availability, false);  // виклик функції
            break;
        case 4:
            std::cout << "Thank you for using our library!\n";
            return 0;   // фунФункція main повертає значення 0 при завершенні роботи програми
        default:
            std::cout << "Invalid choice, please try again.\n";
            break;   //оператор break для виходу з оператора switch
        }
    }
}
```


C:\Users\38093\source\repos\
Main Menu:
1. List all books
2. Borrow a book
3. Return a book
4. Exit
Your choice: 1
List of books:
1. Book 1 (available)
2. Book 2 (available)
3. Book 3 (available)
4. Book 4 (available)
Main Menu:
1. List all books
2. Borrow a book
3. Return a book
4. Exit
Your choice: 2
Enter the number of the book you want to borrow: 1
You have successfully borrowed the book.
Main Menu:
1. List all books
2. Borrow a book
3. Return a book
4. Exit
Your choice: 1
List of books:
1. Book 1 (borrowed)
2. Book 2 (available)
3. Book 3 (available)
4. Book 4 (available)


Main Menu:
1. List all books
2. Borrow a book
3. Return a book
4. Exit
Your choice: 3
Enter the number of the book you want to return: 1
You have successfully returned the book.
Main Menu:
1. List all books
2. Borrow a book
3. Return a book
4. Exit
Your choice:

Завдання №2: VNS Lab 2 Task 1 Variant 12

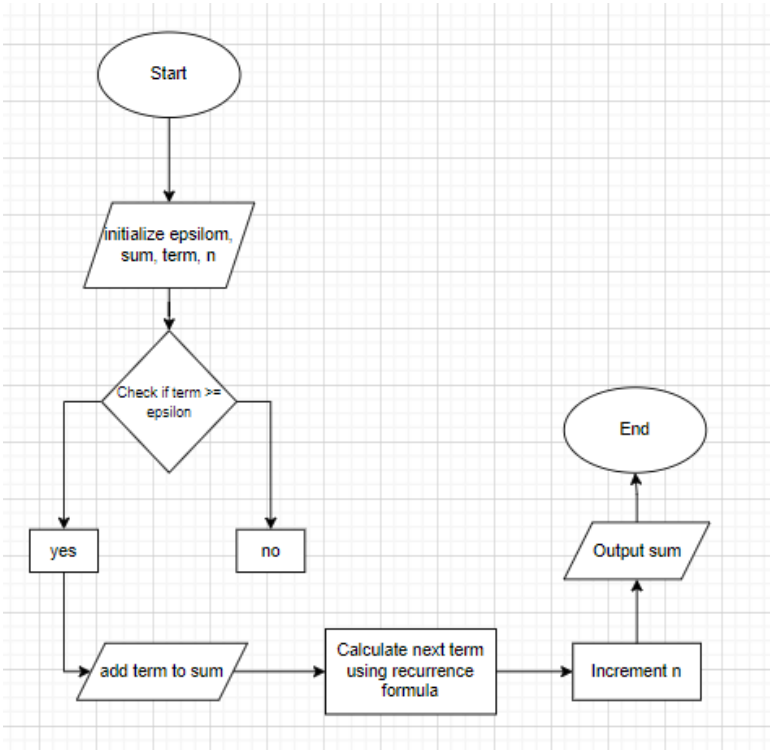
Використовуючи оператор циклу, знайти суму елементів, зазначених у конкретному варіанті. Результат надрукувати, надавши відповідний заголовком.

12) Знайти суму ряду з точністю ε=0.0001, загальний член якого

$$a_n = \frac{2^n n!}{n^n}$$

Запланований час: 40 хв

Витрачений час: 45 хв



```cpp
#include <iostream>
#include <cmath>

// Рекурсивна функція для обчислення факторіала числа
unsigned long long factorial(int n) {
    if (n <= 1) {
        return 1;
    }
    return n * factorial(n - 1);  // Рекурсивний виклик
}

// Функція для обчислення n-го члена ряду
double computeTerm(int n) {
    return (pow(2, n) * factorial(n)) / pow(n, n);  // повернення значення n-го члена ряду
}

int main() {
    const double epsilon = 0.0001; // Точність
    double sum = 0.0; // Сума ряду
    double term; // Поточний член ряду
    int n = 1; // Індекс члена ряду

    // Цикл do while для обчислення суми ряду
    do {
        term = computeTerm(n);
        sum += term;
        ++n;
    } while (std::abs(term) >= epsilon);

    std::cout << "Row sum with accuracy epsilon = " << epsilon << " equals: " << sum << std::endl;

    return 0;  // Повернення 0
}
```

Завдання №3: VNS Lab 3 Task 1

Для x, що змінюється від a до b з кроком (b-a)/k, де (k=10), обчислити функцію f(x), використовуючи її розклад в степеневий ряд у двох випадках:

а) для заданого n;

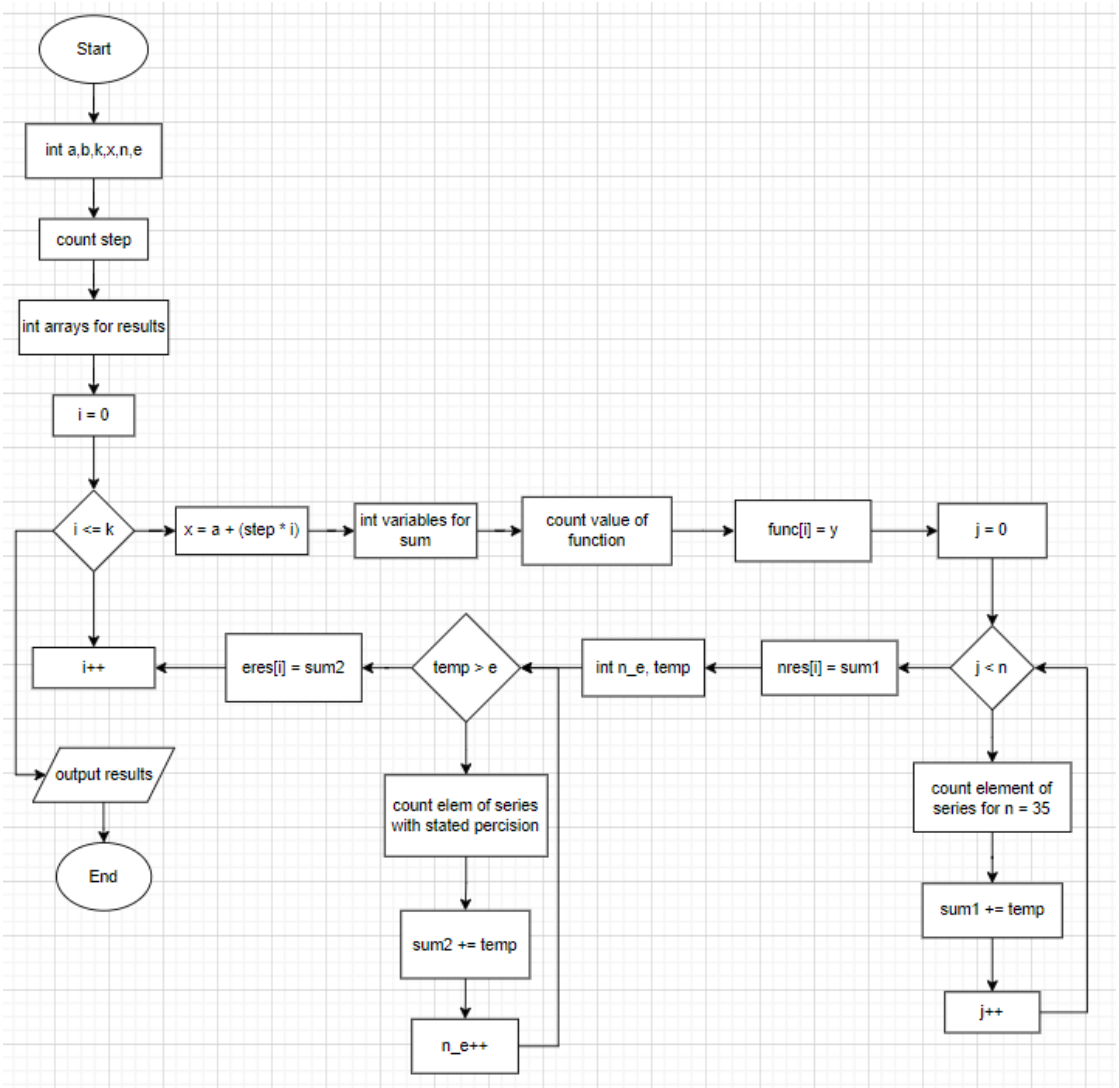
б) для заданої точності ε (ε=0.0001).

Для порівняння знайти точне значення функції.

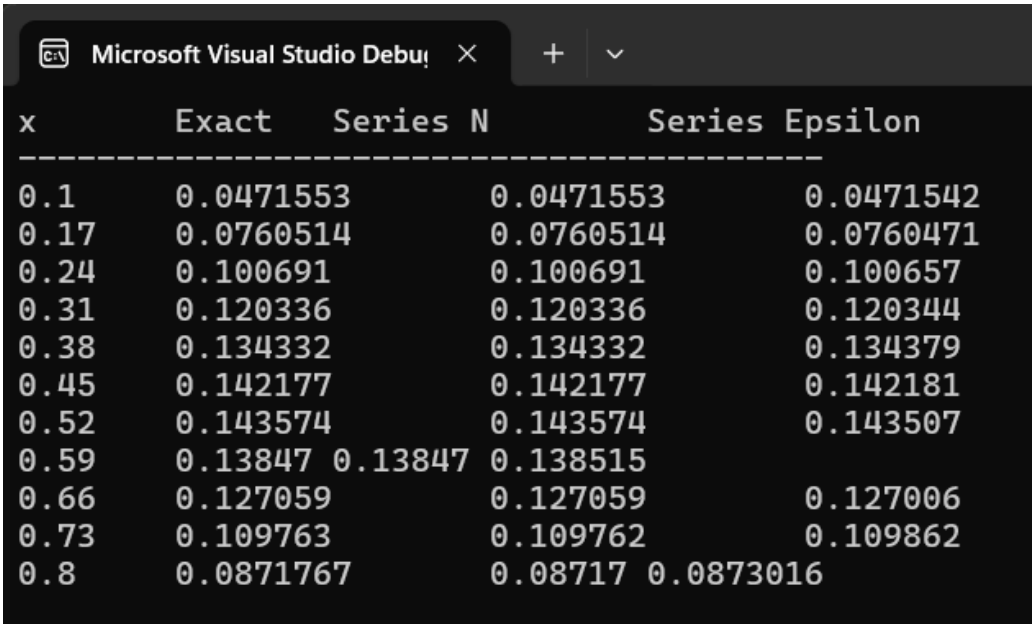
12	$y = -\frac{1}{2}\ln(1 - 2x\cos\frac{\pi}{3} + x^2)$	$0{,}1 \le x \le 0{,}8$	35	$S = \frac{x\cos\frac{\pi}{3}}{1} + \frac{x^2\cos 2\frac{\pi}{3}}{2} + \ldots + \frac{x^n\cos n\frac{\pi}{3}}{n}$

Запланований час: 40 хв

Витрачений час: 1,5 год



```cpp
#include <iostream>
#include <cmath>
#include <vector>

const double PI = 3.14159265358979323846; // Визначення числа пі

// Функція для обчислення точного значення функції y
double exactFunction(double x) {
    return -0.5 * std::log(1 - 2 * x * std::cos(PI / 3) + x * x);
}

// Функція для обчислення значення функції через степеневий ряд для заданого n
double seriesFunctionN(double x, int n) {
    double sum = 0.0;
    for (int i = 1; i <= n; ++i) {
        sum += (std::pow(x, i) * std::cos(i * PI / 3)) / i;
    }
    return sum;
}

// Функція для обчислення значення функції через степеневий ряд з точністю epsilon
double seriesFunctionEpsilon(double x, double epsilon) {
    double sum = 0.0;
    double term;
    int i = 1;
    do {
        term = (std::pow(x, i) * std::cos(i * PI / 3)) / i;
        sum += term;
        ++i;
    } while (std::abs(term) >= epsilon);
    return sum;    // повернення результату функції
}

int main() {
    double a = 0.1; // Початок діапазону
    double b = 0.8; // Кінець діапазону
    int k = 10; // Кількість кроків
    int n = 35; // Задане значення n
    double epsilon = 0.0001; // Точність

    double step = (b - a) / k;
    std::vector<double> x_values;  // Вектор для зберігання значень x

    // Заповнення вектора значеннями x
    for (int i = 0; i <= k; ++i) {
        x_values.push_back(a + i * step);
    }

    std::cout << "x\tExact\tSeries N\tSeries Epsilon\n";
    std::cout << "----------------------------------------\n";

    // Обчислення значень функції для кожного x
    for (double x : x_values) {
        double exact = exactFunction(x);  // виклик функції
        double series_n = seriesFunctionN(x, n); // передача параметру функції (x і n як параметри)
        double series_epsilon = seriesFunctionEpsilon(x, epsilon);

        std::cout << x << "\t" << exact << "\t" << series_n << "\t" << series_epsilon << "\n";
    }

    return 0;
}
```


Microsoft Visual Studio Debu
x Exact Series N Series Epsilon
0.1 0.0471553 0.0471553 0.0471542
0.17 0.0760514 0.0760514 0.0760471
0.24 0.100691 0.100691 0.100657
0.31 0.120336 0.120336 0.120344
0.38 0.134332 0.134332 0.134379
0.45 0.142177 0.142177 0.142181
0.52 0.143574 0.143574 0.143507
0.59 0.13847 0.13847 0.138515
0.66 0.127059 0.127059 0.127006
0.73 0.109763 0.109762 0.109862
0.8 0.0871767 0.08717 0.0873016

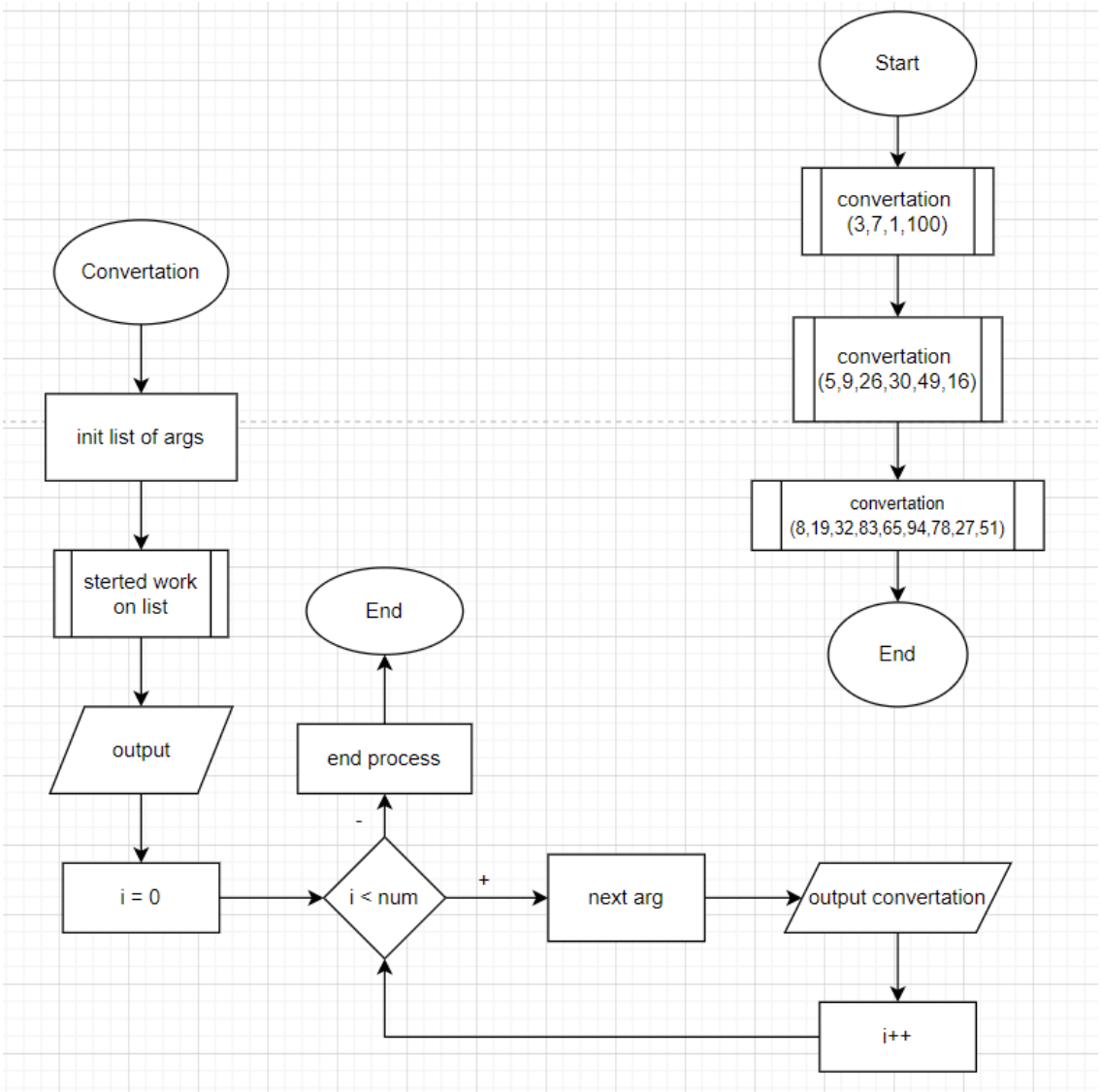
Завдання №4: VNS Lab 7 Task 1 Variant 12

Розв’язати зазначене у варіанті завдання, використовуючи функції зі змінною кількістю параметрів.

Написати функцію зі змінною кількістю параметрів для перетворення чисел з десяткової системи числення у вісімкову. Написати викликаючу функцію main, що звертається до цієї функції не менше трьох разів з кількістю параметрів 3, 5, 8.

Запланований час: 40 хв

Витрачений час: 35 хв



```cpp
#include <iostream>
#include <cstdarg>
#include <sstream> // включення бібліотеки для роботи зі стрічками

std::string toOctal(int number) {  // функція для переведення числа в вісімкову систему числення
    std::ostringstream result;
    do {
        int remainder = number % 8;
        result << remainder;
        number /= 8;
    } while (number > 0);

    std::string octal = result.str();
    std::reverse(octal.begin(), octal.end());
    return octal;
}
void convertToOctal(int count, ...) {   // Функції зі змінною кількістю параметрів (еліпсис)
    va_list args;  // тип для зберігання аргументів
    va_start(args, count);  // макрос для отримання поточного аргументу
    for (int i = 0; i < count; ++i) {
        int number = va_arg(args, int);
        std::string octal = toOctal(number);
        std::cout << "Decimal: " << number << " -> Octal: " << octal << std::endl;
    }
    va_end(args);  // макрос для очищення пам'яті
}
int main() {
    std::cout << "Conversion with 3 parameters:" << std::endl;
    convertToOctal(3, 10, 20, 30);  // виклик функції зі змінною кількістю параметрів

    std::cout << "\nConversion with 5 parameters:" << std::endl;
    convertToOctal(5, 15, 25, 35, 45, 55);

    std::cout << "\nConversion with 8 parameters:" << std::endl;
    convertToOctal(8, 8, 16, 24, 32, 40, 48, 56, 64);

    return 0;
}
```

```
Microsoft Visual Studio Debu
Conversion with 3 parameters:
Decimal: 10 -> Octal: 12
Decimal: 20 -> Octal: 24
Decimal: 30 -> Octal: 36

Conversion with 5 parameters:
Decimal: 15 -> Octal: 17
Decimal: 25 -> Octal: 31
Decimal: 35 -> Octal: 43
Decimal: 45 -> Octal: 55
Decimal: 55 -> Octal: 67

Conversion with 8 parameters:
Decimal: 8 -> Octal: 10
Decimal: 16 -> Octal: 20
Decimal: 24 -> Octal: 30
Decimal: 32 -> Octal: 40
Decimal: 40 -> Octal: 50
Decimal: 48 -> Octal: 60
Decimal: 56 -> Octal: 70
Decimal: 64 -> Octal: 100
```

Завдання №5: VNS Lab 7 Task 2 Variant 12

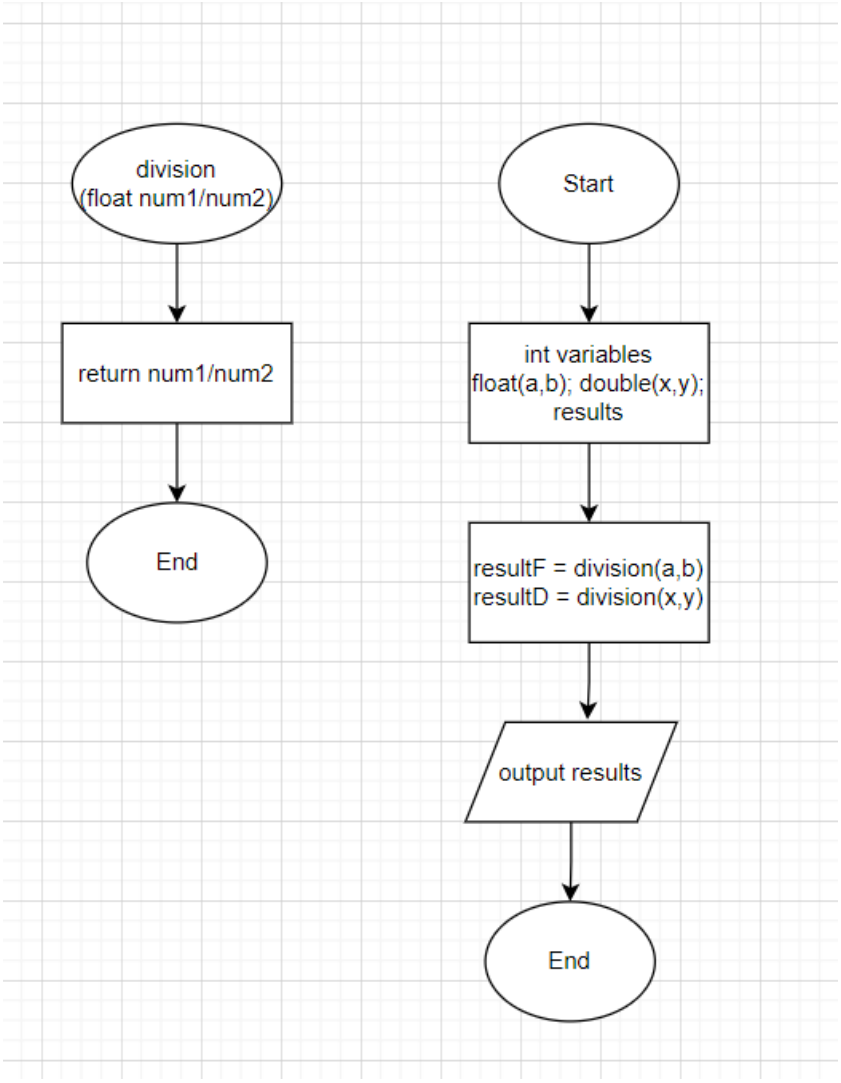
Написати перевантажені функції й основну програму, що їх викликає.

12. а) для ділення десяткових дробів;

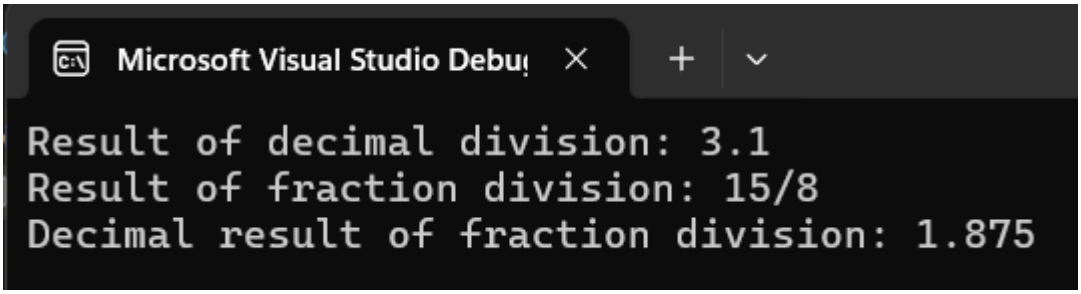
б) для ділення звичайних дробів.

Запланований час: 40 хв

Витрачений час: 55 хв



```cpp
#include <iostream>

// Перевантажена функція для ділення десяткових дробів
double divide(double a, double b) {
    if (b == 0) {
        std::cerr << "Error: Division by zero!" << std::endl;
        return 0;
    }
    return a / b;
}

// Структура для звичайних дробів
struct Fraction {
    int numerator;   // Чисельник
    int denominator; // Знаменник

    // Конструктор для ініціалізації дробу
    Fraction(int num, int den) : numerator(num), denominator(den) {
        if (den == 0) {
            std::cerr << "Error: Division by zero!" << std::endl;
            numerator = 0;
            denominator = 1; // За замовчуванням
        }
    }

    // Перевантаження оператора ділення для звичайних дробів
    Fraction operator/(const Fraction& other) const {
        if (other.numerator == 0) {
            std::cerr << "Error: Division by zero!" << std::endl;
            return Fraction(0, 1); // Повертаємо невизначений дріб
        }
        return Fraction(numerator * other.denominator, denominator * other.numerator);
    }

    // Перетворення дробу в десятковий дріб
    double toDecimal() const {
        return static_cast<double>(numerator) / denominator;
    }
};

int main() {
    // Виклик перевантаженої функції для ділення десяткових дробів
    double a = 15.5, b = 5.0;
    std::cout << "Result of decimal division: " << divide(a, b) << std::endl;

    // Виклик перевантаженого оператора для ділення звичайних дробів
    Fraction f1(3, 4); // 3/4
    Fraction f2(2, 5); // 2/5
    Fraction result_fraction = f1 / f2;  // Результат ділення звичайних дробів

    std::cout << "Result of fraction division: " << result_fraction.numerator << "/" <<
result_fraction.denominator << std::endl;
    std::cout << "Decimal result of fraction division: " << result_fraction.toDecimal() << std::endl;

    return 0;
}
```



Завдання №6: Self Practice “Спекотні дні пінгвінів”

Спекотні дні пінгвінів

Обмеження: 2 сек., 256 МіБ

Ви собі навіть уявити не можете, як же спекотно пінгвінам на Мадагаскарі. Щоб хоч трішки охолодитись, вони випивають безалкогольні коктейлі, однак і з цим проблемно — руки не пристосовані до такого способу життя. Вам потрібно допомогти визначити пінгвінам, чи зможуть вони випити коктейль, що лежить на столі.

Для простоти будемо вважати, що коктейль на столі — круг із діаметром l, в той час, як рот пінгвіна в будь-який момент часу — прямокутник із шириною w, сторони якого паралельні осям координат. В початковий момент часу рот пінгвіна закритий, тому прямокутник вироджений — його висота 0 (інакше кажучи, він є горизонтальним відрізком). Верхня щелепа пінгвіна може розкритись не більше ніж на u дюймів відносно початкової позиції, в той час, як нижня — на d дюймів.

Будемо вважати, що пінгвін може випити коктейль, якщо той повністю впишеться в його рот. Допоможіть пінгвіну за всіма заданими параметрами визначити, чи зможе він випити коктейль.

Вхідні дані

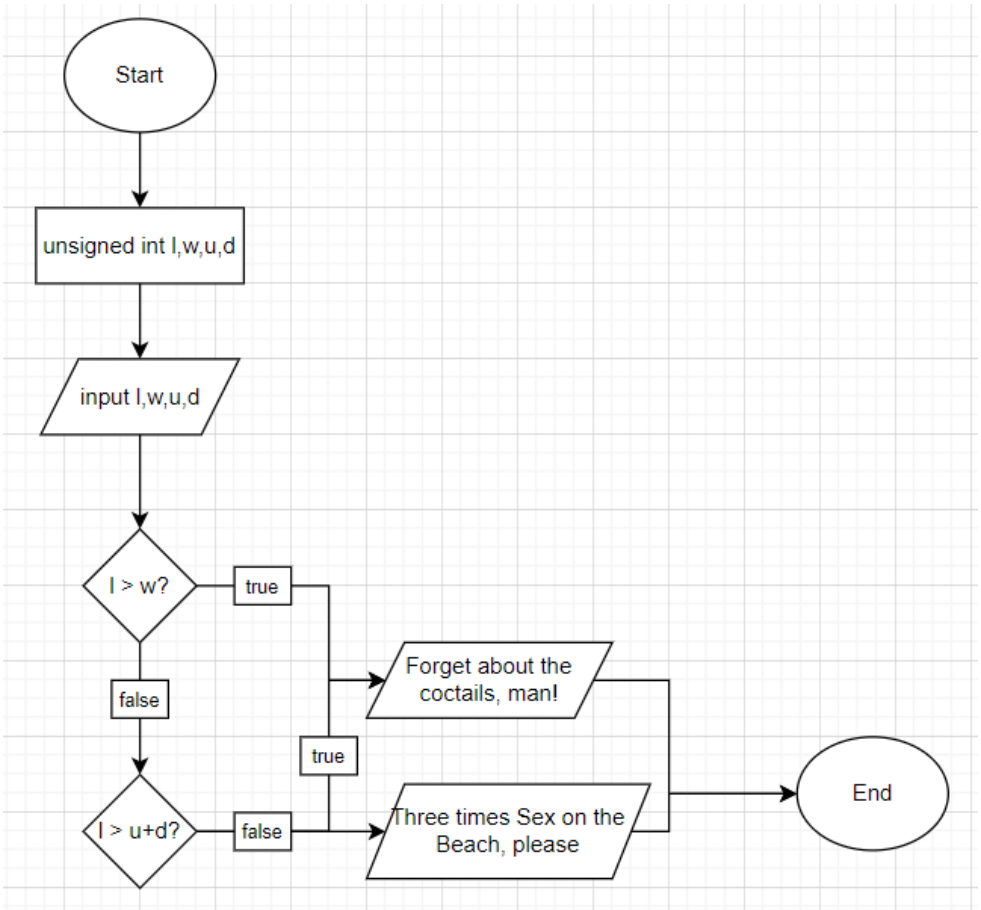
У єдиному рядку задано 4 натуральні числа l, w, u та d — діаметр коктейлю, ширина роту і на скільки дюймів щелепи можуть розкритися відповідно.

Вихідні дані

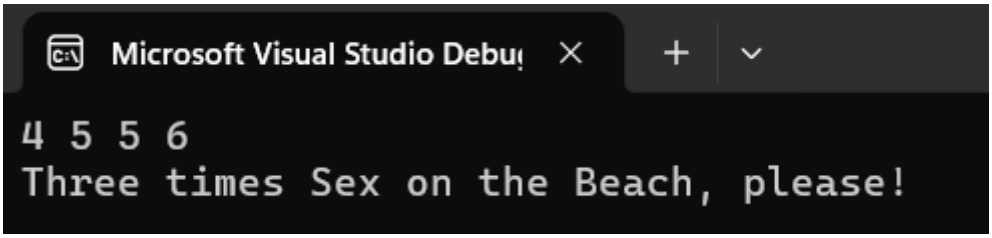
Якщо пінгвін зможе випити коктейль, виведіть рядок `Three times Sex on the Beach, please!`, в протилежному випадку — `Forget about the cocktails, man!`.

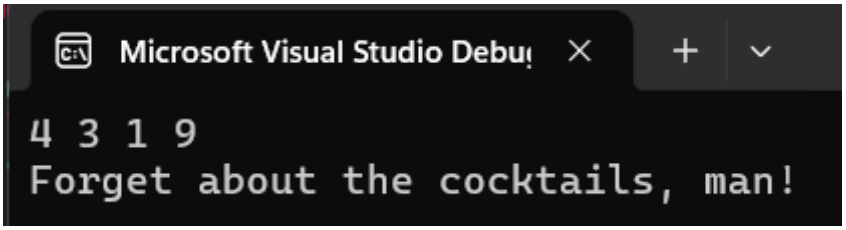
Запланований час: 40 хв

Витрачений час: 25 хв



```cpp
int main() {
    long long l, w, u, d;  // Довжина, ширина, висота рота, діаметр коктейлю
    cin >> l >> w >> u >> d;

    // Якщо діаметр коктейлю менший або рівний ширині та висоті рота, пінгвін може випити коктейль
    if (l <= w && l <= u + d) {
        cout << "Three times Sex on the Beach, please!" << endl;
    }
    else {
        cout << "Forget about the cocktails, man!" << endl;
    }

    return 0;  // повернення 0 (вказує на успішне завершення програми)
}
```





Завдання №7: Self Practice “Зуби”

Зуби

Обмеження: 2 сек., 256 МіБ

Мале Бісеня любить гострити зуби. А Зла Тітонька любить до нього підходити і питатися: «Що, зуби гостриш?». Бісеняті таке не дуже подобається, тому воно придумало робити таке.

У Малого Бісеняти є n зубів. Кожен зуб має коефіцієнт загостреності a_i. Також існує межа загостреності k. Якщо коефіцієнт загостреності певного зуба є більшим чи рівним межі загостреності, то такий зуб вважається загостреним.

Мале Бісеня хоче наступного разу, коли Зла Тітонька його щось запитає, показати їй якнайбільше загострених зубів, що розташовані поспіль.

Допоможіть Малому Бісеняті дізнатися, скільки найбільше зубів воно зможе показати.

Вхідні дані

У першому рядку задані два цілих числа n та k — кількість зубів та межа загостреності відповідно.

В другому рядку задано n цілих чисел a_i — коефіцієнти загостреності зубів.

Вихідні дані

Єдине ціле число — відповідь на задачу.

Обмеження

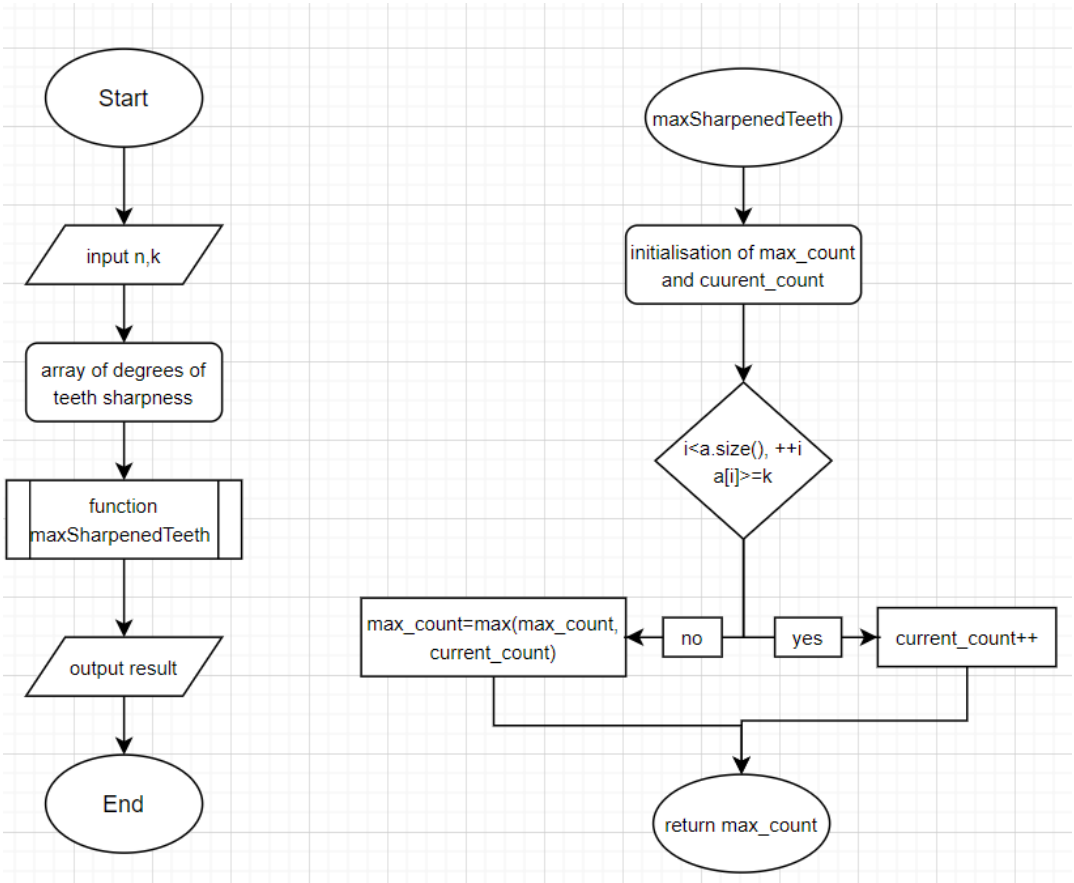
$1 \le n \le 10^5$,

$1 \le k \le 10^9$,

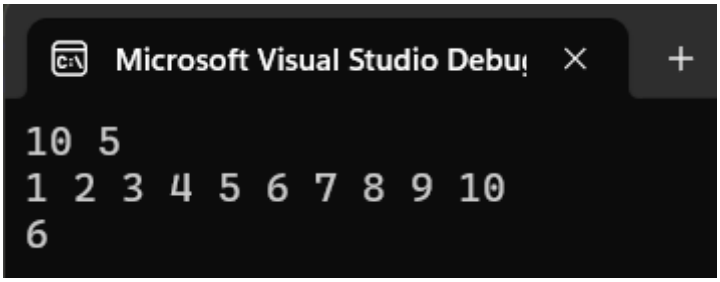
$1 \le a_i \le 10^9$.

Запланований час: 40 хв

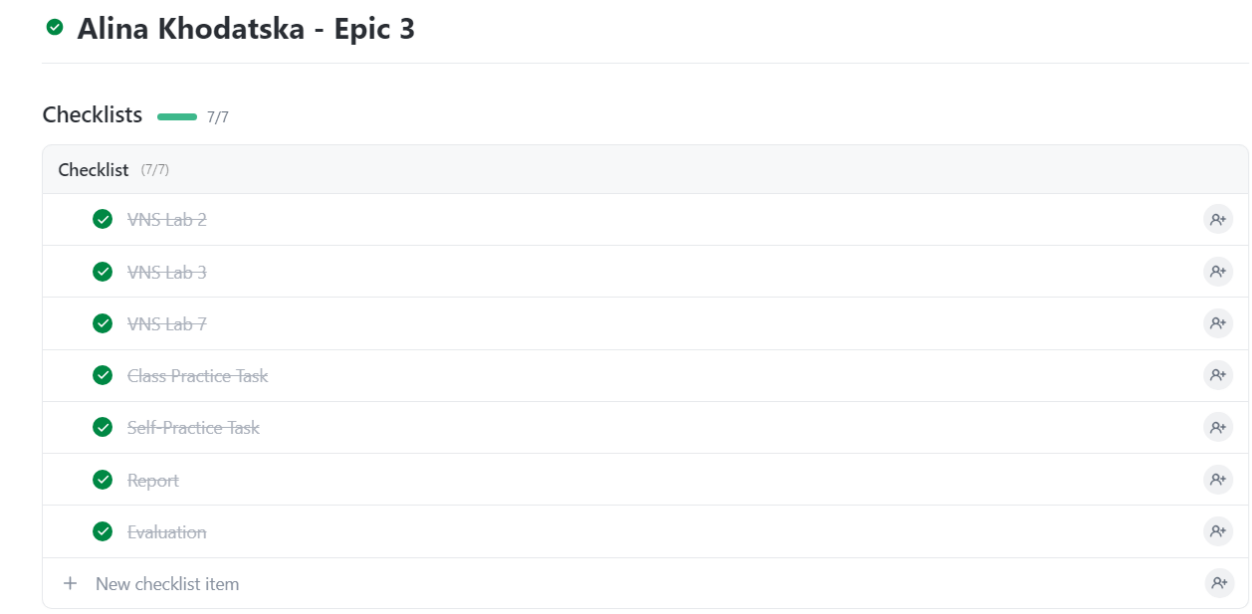
Витрачений час: 40 хв



```cpp
#include <iostream>
using namespace std;

int main() {
    int n, k;
    cin >> n >> k;

    int* a = new int[n];  // Динамічне виділення пам'яті
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }

    int max_len = 0; // Максимальна довжина підрядка з загостреними зубами

    // Вкладений цикл для перевірки всіх підрядків
    for (int i = 0; i < n; i++) {
        int current_len = 0;
        for (int j = i; j < n; j++) {
            if (a[j] >= k) { // Якщо зуб загострений
                current_len++;
            }
            else {
                break; // Якщо знайдено елемент, який менший за k, припиняємо перевірку цього підрядка
            }
        }
        max_len = max(max_len, current_len); // Оновлюємо максимальну довжину підрядка
    }

    cout << max_len << endl;

    delete[] a;  // Звільняємо пам'ять
    return 0;
}
```



Дошка в Click up та зустріч з командою



Висновок: Під час виконання лабораторної роботи були засвоєні основні принципи роботи з одновимірними та двовимірними масивами, а також з динамічними масивами, вказівниками та посиланнями. Окрім того, було вивчено основні методи створення та обробки масивів, а також алгоритми їх обробки для вирішення різноманітних завдань. Отримані навички роботи з динамічними масивами і вказівниками дозволяють ефективно управляти пам'яттю під час виконання програм.

Посилання на Pull Request: https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/500