Лабораторная работа №3.

РЕКУРСИИ: ОПРЕДЕЛИТЕЛЬ МАТРИЦЫ

1.Основные сведения

Если имя функции появляется где-либо в выражении внутри самой функции, то говорят о *рекурсивном* выполнении функции.

Классическим примером рекурсивного алгоритма служит вычисление факториала

N! = 1, если N = 0,

(N-1)!*N, если N > 0.

Этой математической формуле соответствует текст рекурсивной функции Fact:

```
long long Fact(long long N){
   if (N==0) return 1;
   else return Fact(N-1)*N;
}
```

В операторе возврата `return Fact(N-1)*N` имя Fact есть вызов функции Fact (т.е. самой себя) от предыдущего значения N, а результат вычисления выражения является результатом функции. Таким образом, функция вызывает сама себя с последовательно уменьшающимися значениями аргумента от (N-1) до N = 0. Вызовы функций последовательно сохраняются в стеке вызовов. Наконец, вызов Fact(0) возвращает единицу, которая умножается на N=1. Затем выполняется возврат из предыдущего вызова Fact(1) с результатом 1, который умножается на 2. Эти возвраты выполняются в порядке, обратном вызовам (за счет стека) для N = 0,1,2,3,4, …, N. Самый первый вызов функции Fact(N) завершается последним, возвращая Fact(N-1)*N, т.е. произведение 1*1*2*3*4*5*…*(N-1)*N или N!

При использовании этой функции необходимо значение N проверить на допустимость, как показано в программе Factorial:

```
// Factorial
#include <iostream>
long long N;
   // здесь описание функции Fact
int main()
{
  std::cout << "Вычисление факториала, введите целое: ";
   std::cin >> N;
   if (N<0)
        std::cout << "Число должно быть >=0 !";
   else std::cout << N << " != " << Fact(N);
   return 0;
}
```

Задачи, формулируемые естественным образом как рекурсивные, часто приводят и к рекурсивным решениям (алгоритмам). Такими задачами являются, например, нахождение суммы бесконечных рядов (следующее слагаемое определено через предыдущее), задачи комбинаторики, преобразование арифметических выражений в постфиксную нотацию («польская запись»), трансляция языков программирования, обработка двоичных деревьев, вывод связных списков.

Однако надо предостерегаться от использования рекурсивных методов без должного анализа. Хотя они и бывают «прозрачными», но не всегда приводят к самым эффективным решениям.

2. Индивидуальное задание

Разработать программу вычисления определителя квадратной матрицы с максимальным размером 10*10. Рекомендуется использовать принцип нисходящего проектирования с постепенной детализацией управляющей (алгоритм) и информационной (данные) структуры программы. Для

этого необходимо представить основную программу (функция main()) в виде указанной последовательности вызовов следующих функций:

1. Ввод требуемого размера и номера строки или столбца разложения. Предусмотреть контроль входных данных.

2. Заполнение матрицы данными с клавиатуры, генератором случайных вещественных чисел или натуральным рядом чисел (заполнение матрицы по столбцам для нечётных вариантов и по строкам – для чётных) по выбору пользователя. Получить случайные числа можно с помощью функции rand() библиотеки <cstdlib>, которая генерирует числа в диапазоне от 0 до RAND_MAX.

3. Вывод полученной матрицы с нумерацией строк и столбцов.

4. Рекурсивное вычисление определителя матрицы разложением по указанной строке (для нечетного варианта) или по указанному столбцу (для четного варианта) по формулам (1) или (2) соответственно:

$$\text{Det } A = \sum_{j=1}^{n} A_{ij} * D_{ij} \text{ – разложение по строке } i = 1, 2, \ldots, n \qquad (1)$$

или

$$\text{Det } A = \sum_{i=1}^{n} A_{ij} * D_{ij} \text{ – разложение по столбцу } j = 1, 2, \ldots, n \qquad (2)$$

где

A	– исходная матрица,
i	– номер строки,
j	– номер столбца,
A_{ij}	– элемент матрицы,
D_{ij}	– алгебраическое дополнение,
Det A	– определитель.

5. Преобразование исходной матрицы в треугольную матрицу, у которой все элементы ниже главной диагонали равны нулю. Использовать метод Гаусса с выбором главного элемента по строке (для чётных вариантов) и по столбцу (для нечётных вариантов).

6. Вывод треугольной матрицы с нумерацией строк и столбцов.

7. Вывод значений определителей входной и треугольной матриц и разности между ними с целью контроля правильности программы.

8. Сравнить время вычисления определителя разными способами (табл.) с использованием функции clock() библиотеки <ctime>.

Обмен данными между функциями должен осуществляться только через заголовки; функция п. 2 должна использовать три вспомогательные подпрограммы; вычисление определителей матриц реализовать в виде отдельных функций. Максимальный размер матрицы задать константой.

3. Подготовка

1. Нарисовать блок-схему алгоритма главной программы.

2. Привести математическое обоснование (формулы с комментариями, особенности алгоритма):

   а). вычисления определителя матрицы через разложение по указанной строке/столбцу;

   б). приведения матрицы к треугольному виду (метод Гаусса).

3. Записать тексты сложных подпрограмм разрабатываемой программы (6 обязательных подпрограмм) с подробными комментариями по каждой строке кода:

   а). три подпрограммы заполнения исходной матрицы;

   б). вывод полученной матрицы с нумерацией строк и столбцов;

   в). рекурсивное вычисление определителя матрицы;

   г). реализация метода Гаусса.

4. Заготовить тестовый пример № 1: матрица размером 2 на 2 и значение определителя.

5. Заготовить тестовый пример № 2: матрица размером 3 на 3 и значение определителя.

6. Заготовить сравнительную таблицу выполнения программы для разных тестовых данных (табл.)

Таблица

Оценка результатов работы программ

размер матрицы	*способ заполнения матрицы*	*результаты выполнения программы*		*временя выполнения программы, мс*	
		рекурсивное вычисление	*метод Гаусса*	*рекурсивное вычисление*	*метод Гаусса*
6	случайные числа				
	натуральный ряд чисел				
7	…				
8					
10	…				

4. Состав отчета

1. Индивидуальное задание.

2. Информация в соответствии с подготовкой.

3. Результаты прогона тестов.

4. Результаты выполнения программы с указанием времени выполнения программы для двух случаев: заполнение генератором случайных чисел, заполнение натуральным рядом чисел (табл.).