Практическая работа № 3.
Понятие рекурсии. Рекурсивные функции

Цель. Изучить понятие рекурсии, *рекурсивные функции* в программировании, приемы построения рекурсивной функции при решении задач, научиться применять *рекурсивные методы* в решении задач на языке С++.

Рекурсивный алгоритм – это *алгоритм*, в определении которого содержится *прямой* иликосвенный вызов этого же алгоритма. *Функция* называется **рекурсивной**, если в своем теле она содержит обращение к самой себе с измененным набором параметров. При этомколичество обращений конечно, так как в итоге решение сводится к базовому случаю, когда ответ очевиден.

Пример1.В арифметической прогрессии найдите a_n, еслиизвестны a_1=-2.5, **d**=0.4, не используя формулу **n**-го члена прогрессии.

По определению арифметической прогрессии, $a_n=a_{n-1}+d$, при этом $a_{n-1}=a_{n-2}+d$, $a_{n-2}=a_{n-3}+d$, $...a_2=a_1+d$.

Таким образом, нахождение a_n для номера **n** сводится к решению аналогичной задачи, но только для номера **n-1**, что всвою *очередь* сводится к решению для номера **n-2**, и так далее, пока не будет достигнут номер **1**(*значение* a_1 дано *по* условию задачи).

```
Float arifm(int n,float a,float d)
{
if(n<1)
return 0;//для неположительных номеров
if(n==1)
return a;//базовый случай:n=1
return arifm(n-1,a,d)+d;//общий случай
}
```

В рекурсивных функциях несколько раз используется return. В базовом случае возвращается конкретный результат (в примере – *значение* **a**), а общий случай предусматриваетвызов функцией себяже, но сменяющимися значениями отдельных параметров(в примере изменяется только номер члена последовательности, при этом не меняются *разность* и первый член прогрессии). Для решения задач *рекурсивными методами* разрабатывают следующие этапы, образующие **рекурсивную триаду**:

- *Параметризация* – выделяют параметры, которые используются для описания условиязадачи, а затем в решении;
- *База рекурсии* – определяет тривиальный случай, при котором решение очевидно,то есть не требуется обращение функции к себе;
- *Декомпозиция* – выражает общий случай через более простые подзадачи с измененными параметрами.

Пример2. Для целого неотрицательного числа **n** найдите его *факториал*.

Разработаем рекурсивную триаду.
Параметризация: n–неотрицательное *целое число*.
База рекурсии: для n=0 *факториал* равен 1.
Декомпозиция: **n!=(n-1)!*n**.

```
int Fact (int n)
{  if (n<0)
return 0;        // для отрицательных чисел
if (n==0)               // базовый случай: n=0
return 1;
return n*Fact(n-1);  // (общий случай (декомпозиция)
}
```

Эффективность рекурсивного или итерационного способов решения одной и той же задачи определяется в ходе анализа работоспособности программы на различных наборах даных. Таким образом, *рекурсия* не является универсальным способом в программировании. Ее следует рассматривать как альтернативный вариант при разработке алгоритмов решения задач.

Область памяти, предназначенная для хранения всех промежуточных значений *локальных переменных* при каждом следующем рекурсивном обращении, образует *рекурсивный стек*. Для каждого текущего обращения формируется локальный слой данных стека (при этом совпадающие идентификаторы разных слоев стека независимы друг от друга и неотождествляются). Завершение вычислений происходит посредством восстановления значений данных каждого слоя в порядке, обратном рекурсивным обращениям. В силу подобной организации количество рекурсивных обращений ограничено размером области памяти, выделяемой подпрограммный код. При заполнении всей предоставленной области памяти попытка вызова следующего рекурсивного обращения приводит к ошибке *переполнения* стека.

Указания

Следующие задачи необходимо решить в соответствии с *рекурсивными методами* решения задачи методами обработки числовых данных в языке С++. Передреализацией кода каждой задачи необходимо разработать рекурсивную триаду в соответствии с постановкой задачи. Программу для решения каждого задания необходимо разработать методом процедурной абстракции, используя *рекурсивные функции*. Этапы сопроводить комментариями в коде.

Задача1.

Определите закономерность формирования членов последовательности. Найдите n-ый членпоследовательности: 1,1,2,3,5,8,13,…

Задача2.

Разработать алгоритм и программу вычисления числа сочетаний из *n* элементов по m:$C_n^m = \frac{n(n-1)(n-2)\ldots(n-m-1)}{1\cdot 2\cdot 3\cdot\ldots\cdot m} = \frac{n!}{m!(n-m)!}$.

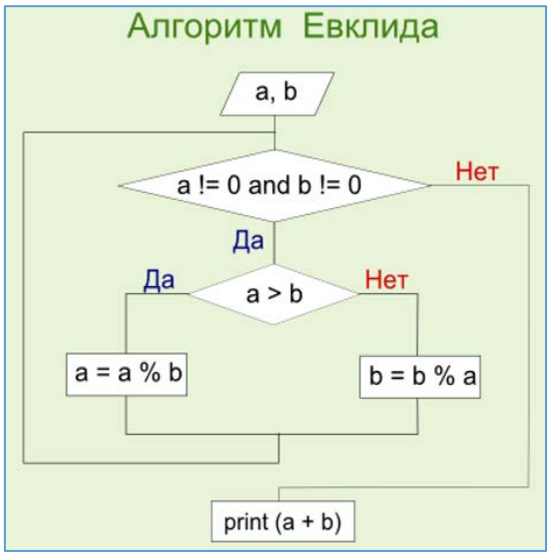
Числа *n* и *m* вводятся с клавиатуры (*n*≥*m*). Решить задачу рекурсивно, выразив вычисление C_n^m через C_{n-1}^{m-1}. Кроме того, решить задачу итерационным методом. Сравнить результаты.

Задача3.

Найдите наибольший общий *делитель* двух натуральных чисел с помощью алгоритма-Евклида, используя рекурсивный и не рекурсивный(итерационный) алгоритмы. Определить оценку сложности и провести сравнительный анализ двух алгоритмов

Справочный материал к задаче 3.

В самом простом случае алгоритм Евклида применяется к паре положительных целых чисел и формирует новую пару ,которая состоит из меньшего числа и разницы между большим и меньшим числом. Процесс повторяется, пока числа не станут равными. Найденное число и есть наибольший общий делитель исходной пары. Для итерационного алгоритма:



Задача 4.

Дано натуральное число, кратное 3.Получите сумму кубов цифр этого числа, затем сумму кубов получившегося числа и т.д. Проверьте на нескольких примерах ,что любая такая последовательность чисел сходится к числу 153. Определить зависимость между вводимыми числами и количеством итераций.

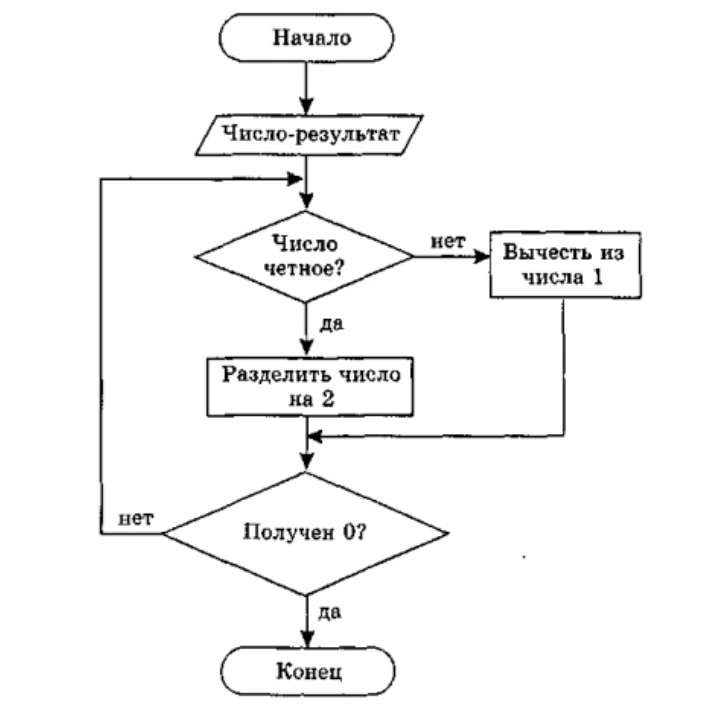
21=>8+1=9=>729=>343+8+729=1080=1+512=513=>153

Задача 5.

Исполнитель умеет выполнять два действия:"+1","*2" . Составьте рекурсивную функцию для программы получения из числа 1 числа 100 за наименьшее количество операций. Определить количество операций.

Справочный материал к задаче 5.

Разрабатывать алгоритмы будет проще, если воспользоваться следующей блок-схемой:



Задания 2 и 3 выполнить используя рекурсивный и итерационный методы. Сравнить их и вывести на экран результаты сравнения Время счета и количество итераций для каждого метода. Результаты проверяются на одних и тех же данных.

Формула для рекурсии $C_n^m = C_{n-1}^{m-1} + C_{n-1}^m$