Урок 22

Процедуры и функции

Тема урока: понятие подпрограмм. Механизм реализации подпрограмм с помощью процедур и функций

I. Повторение материала

- Какова структура программы?
- Обязателен ли заголовок программы?
- Какие разделы описаний вы знаете?
- С чего начинается раздел констант?
- Как описать переменные?
- С чего начинается основная часть программы? Как ее закончить?

- При создании программы для решения сложной задачи программисты выполняют разделение этой задачи на подзадачи, подзадачи – на еще меньшие подзадачи и так далее, до легко программируемых элементарных задач. Со временем у каждого программиста через некоторое время появляется большой набор собственных заготовок, неординарных решений и т.д., которые он хотел бы использовать во всех своих творениях.
- Языки программирования Qbasic и Turbo Pascal позволяют разделять программу на отдельные части, которые называются подпрограммами. Сам термин подпрограмма говорит о том, что она подобна и подчинена основной программе.

Подпрограммы решают три важные задачи, значительно облегчающие программирование:

- избавляют от необходимости многократно повторять в тексте программы аналогичные фрагменты, т.е. сократить объем программы;
- улучшают структуру программы, облегчая понимание при разборе;
- уменьшают вероятность появления ошибок, повышают устойчивость к ошибкам программирования и непредвиденным последствиям при модификации.
- Таким образом, **подпрограмма** — это повторяющаяся группа операторов, оформленная в виде самостоятельной программной единицы. Она записывается однократно, а в соответствующих местах программы обеспечивается лишь обращение к ней по имени.

Общие принципы выделения подпрограмм:

- Если в программе необходимо переписывать одни и те же последовательности команд, то стоит эту последовательность команд оформить в виде подпрограммы;
- Стоит перенести в подпрограмму подробности, заслоняющие основной смысл программы;
- Слишком длинную программу полезно разбить на составные части – подобно тому, как книгу разбивают на главы. При этом основная программа становиться похожей на оглавление;

- При решении задачи могут возникать слишком сложные подзадачи. Целесообразней отладить их отдельно в небольших программах. Добавление этих программ в основную задачу будет легким, если они оформлены как подпрограммы.
- Все, что вы сделали хорошо в одной программе, вам захочется перенести в новые программы. Для повторного использования частей кода лучше сразу выделять в программе полезные вам подзадачи в виде отдельных подпрограмм.

- Процедуры предназначены для выполнения некоторой последовательности действий.
- Любая процедура начинается с заголовка, обязательной частью, (в отличие от заголовка программы).
- Заголовок состоит из служебного слова **Procedure** (**Sub** в языке QBasic), за которым следует имя процедуры, а в круглых скобках — список формальных параметров. После заголовка могут идти те же разделы, что и в программе. Таким образом, общий вид будет следующим:

SUB<имя>[(формальные параметры)]
Procedure<*имя*>*[(формальные параметры)];*

Список формальных параметров может отсутствовать.

Объявление переменных	*описательная часть*
	Begin
тело процедуры	*тело процедуры*
EXIT SUB	***досрочный выход*** EXIT SUB
END SUB	***End;***

Формальные и фактические параметры

Результат выполнения процедуры — это одно или несколько значений. Оно (или они) передается в основную программу как значение ее параметра. При вызове процедуры ее формальные параметры заменяются фактическими в порядке их следования.

Фактические параметры — это параметры, которые передаются процедуре при обращении к ней.

Число и тип формальных и фактических параметров должны совпадать с точностью до их следования.

Формальные параметры — это переменные, фиктивно присутствующие в процедуре и определяющие тип и место подстановки фактических параметров, над которыми производятся действия.

Все формальные параметры делятся на два вида:

- параметры-переменные и параметры-значения.

Qbasic

- В Qbasic для передачи по значению переменные берутся в круглые скобки. Т. е. передается только копия значения этих параметров, внутри процедуры можно производить любые действия с данными формальными параметрами (допустимыми для его типа), но их любые изменения никак не отражаются на значениях соответствующих фактических параметров, то есть какими они были до вызова процедуры, то такими же и останутся после завершения ее работы.
- **Без круглых скобок переменные передаются по ссылке.** Это делается для того, чтобы изменения в теле процедуры значений формальных параметров приводило к изменению соответствующих фактических параметров, таким образом, они и получают новое значение.

Turbo Pascal 7.0

- **Параметры-переменные в Turbo Pascal 7.0** — это те формальные параметры, **перед которыми стоит служебное слово Var.** Они передаются по ссылке (передается адрес фактического параметра) тогда, когда необходимо передать некоторые новые значения в точку вызова процедуры из программы, то есть когда нужно, чтобы изменения в теле процедуры значений формальных параметров приводило к изменению соответствующих фактических параметров, таким образом, они и получают новое значение.
- **Параметры-значения — перед ними слово Var не ставится**, и идет передача по значению, т. е. передается только копия значения этих параметров, внутри процедуры можно производить любые действия с данными формальными параметрами (допустимые для его типа), но их любые изменения никак не отражаются на значениях соответствующих фактических параметров, то есть какими они были до вызова процедуры, то такими же и останутся после завершения ее работы.

Локальные или глобальные переменные

- Область действия переменной (идентификатора) - часть программы, где он может быть использован. Область действия переменной определяется местом их объявления. В программе все переменные делятся на глобальные и локальные.
- **Глобальные переменные** — это те переменные, которые объявлены в описании основной части, и ее могут использовать любые процедуры и функции данной программы.
- Переменные, описанные внутри подпрограммы, называются **локальными** и могут быть использованы только внутри данной подпрограммы. Локальные переменные могут быть описаны как в заголовке программы, так и в разделе описания переменных. При совпадении имен глобальных и локальных переменных, локальные определения в пределах своего действия отменяют действия глобальных, и эти переменные никак не связаны между собой.

- Какова роль локальных переменных, нельзя ли все переменные описать как глобальные? Подпрограмма должна быть, по возможности, независима от основной программы, поэтому все переменные, нужные только в пределах подпрограммы, должны описываться как локальные. Общение основной программы с подпрограммой должно, как правило, идти через список параметров подпрограммы, что придает ей необходимую гибкость.
- Локальность или глобальность – понятия относительные. Программа с вложенными в нее подпрограммами - иерархическое дерево. Объект, локальный по отношению к более высокому уровню иерархии, ведет себя как глобальный по отношению к объектам более низкого уровня.
- Программу можно изобразить в виде блоков (подпрограмм), каждый блок это дом с зеркальными стеклами в окнах. Изнутри через них видно все, что находиться снаружи, внутрь заглянуть нельзя.
- Программу можно изобразить в виде блоков (подпрограмм), каждый блок это дом с зеркальными стеклами в окнах. Изнутри через них видно все, что находиться снаружи, внутрь заглянуть нельзя.

Составить процедуру сложения двух чисел, вводимых с клавиатуры

```
DIM A, B, S AS SINGLE
SUB summa (X, Y, S)

S = X + Y
PRINT "S="; S, A, B
END SUB

CLS
INPUT "A=,B="; A, B
CALL summa(A, B, S)
PRINT "summa="; S, A,B

END
```

```
uses crt;
var a,b,s:real;
procedure  summa (x,y:real;var s:real);
begin
s:=x+y;
 writeln('s=',s:3:1,'   ',a:3:1,'    ',b:3:1);
   end;
begin
clrscr;
writeln('a=b=');readln(a,b);
    summa(a,b,s);
writeln('s=',s:3:1,'   ',a:3:1,'  ',b:3:1);
readln;
end.
```

QB TP

Процедура вызывается как оператор, состоящий из имени процедуры. В круглых скобках передаются фактические параметры. В нашем примере, фактические параметры *a, в* и *s* передают свои значения соответственно формальным параметрам *X, Y* и S. После завершения работы процедуры переменные *A* и B имеют те же значения, что и при вызове, а S получает новое значение.

Передача параметров очень важна для процедур и функций. Это процесс, благодаря которому передается информация.

Пусть a=3 и в=4. Когда в программе встречается оператор

summa((a), (b),S) или summa (a,b,s);, то ЭВМ выполняет следующие действия:

- выделяет память для переменных, описанных в процедуре*;*
- присваивает формальным параметрам значения фактических: x:=a (x=3), y:=в (y=4), s:=s;
- выполняет операторы процедуры, то есть найдет сумму
- полученное значение присвоит переменной **S**, а переменные **A** и **B** остаются прежними, после этого переходит к выполнению следующих действий программы в точке вызова, то есть выполняется следующий оператор, стоящий за обращением процедуры.

Составить процедуру нахождения максимального из двух действительных чисел, вводимых с клавиатуры. Используйте процедуру для нахождения максимального значения для четырех чисел.

```
DIM A, B, S,C,D AS SINGLE
SUB maxim (X, Y, S)


IF X < Y THEN S = Y ELSE S = X
END SUB

CLS
INPUT "a=,b="; A, B
INPUT "c=,d="; C, D
```

```
uses crt;
var a, b, s, c, d:real;
procedure  maxim x, y:real;var
    s:real);
Begin
if x<y then s:=y else s:=x;
end;
Begin
clrscr;
writeln('a=b=');readln(a,b);
writeln('c=d='); readln(c,d);
```

В перменную S заносим большее из	двух чисел A и B
CALL maxim(A, B, S)	maxim(a,b,s);
В перменную S заносим большее из	двух чисел C и S
CALL maxim(C, S,S)	maxim(c,s,s);
В перменную S заносим большее из	двух чисел D и S
CALL maxim(D, S,S)	maxim(d,s,s);}
PRINT "max="; S	writeln('max=',s:3:1);
	readln;
END	end.

QB TP

Составить программу, которая будет находить a^b, то есть b-ую степень числа A, где A и B - это целые числа и B>0, вводимые с клавиатуры.

```
DIM A, B  AS  INTEGER          uses crt;
DIM S  AS  LONGSUB             var a,b:integer;
stepen (X, Y AS INTEGER,       s:longint;
S AS LONG)                     procedure  stepen (x,y:integer;var
                               s:longint);

DEFINT I                       var i:integer;
                               begin
 S = 1                          s:=1;
FOR I = 1 TO Y                  for i:=1 to y do
 S = S * X                      s:=s*x;
NEXT
END SUB                         end;
```

	begin
CLS	*clrscr;*
INPUT "a=,b="; A, B	*writeln('a=b=');readln(a,b);*
Вызов процедуры нахождения степени числа A.	
CALL stepen(A, B, S)	*stepen(a,b,s);*
PRINT "s="; S	*writeln('s=',s);*
	readln;
END	*end.*

QB TP

```
uses crt;
var a,b:integer;
procedure  swap (var x,y:integer);
var z:integer;
Begin
z:=x;x:=y;y:=z;
end;
Begin
clrscr;
write('a=,b=');readln(a,b);
swap(a,b);
writeln('a=,b=',a,' ',b);
readln;
end.
```

TP

Домашнее задание

Используя процедуру упорядочить значения трех переменных a, b, и c в порядке их возрастания.

Урок 23

Тема урока: **Понятие подпрограмм. Механизм реализации подпрограмм с помощью процедур и функций**

Проверка домашнего задания.

1. Ответить на вопросы:

а) что такое подпрограмма?

б) какие преимущества она дает?

в) какие бывают параметры?

г) чем отличаются друг от друга формальные и фактические параметры?

д) что такое область действия переменной?

е) чем отличаются друг от друга глобальные и локальные параметры?

2. Используя процедуру упорядочить значения трех переменных *a, b,* и *c* в порядке их возрастания.

Решение.

В основной программе появятся строки:

IF A<B THEN SWAP A,B *IF A<B THEN SWAP(A,B)*

IF A<C THEN SWAP A,C *IF A<C THEN SWAP(A,C)*

IF B<C THEN SWAP B,C *IF B<C THEN SWAP(B,C)*

Описание функции

- Функции предназначены для того, чтобы вычислять только одно значение, поэтому ее первое отличие состоит в том, что процедура может иметь новые значения у нескольких параметров, а функция только одно (оно и будет ее результатом).
- Второе отличие заключается в заголовке функции. Он состоит из слова ***Function,*** за которым идет имя функции, затем в круглых скобках идет список формальных параметров, после чего через двоеточие записывается тип результата функции. Остальное как в процедуре. **В Бейсике тип результата не записывается.**

Кроме того:

В теле функции обязательно должен быть хотя бы один оператор присвоения, где в левой части стоит имя функции, а в правой — ее значение. Иначе, значение не будет определено.

Таким образом, общий вид такой:

Описание функции

Function <имя>
[(<список форм. параметр.>)]

В Бейсике тип результата не записывается.

<тип результата>

Эта часть заголовка может отсутствовать.

Описательная часть

Тело функции
<имя>=<значение>

End function

Function <имя>
[(<список форм.параметров>)]:

Описательная часть

Begin

Тело функции
<имя>:=<значение>;

End;

Пример. **Составить программу, подсчитывающую число сочетаний без повторения из N элементов по К элементов.**

Число сочетаний без повторения считается по формуле:

$$C_n^k = \frac{n!}{k!(n-k)!}$$

Обозначим:

n, k — переменные для хранения введенных чисел;

C — переменная для хранения результата. Чтобы подсчитать количество сочетаний без повторения, необходимо вычислить n!, (n—k)!, k!.

Опишем функцию, вычисляющую факториал числа n (n!=1*2*...*n).

Опишем функцию, вычисляющую факториал числа n (n!=1*2*...*n).

FUNCTION Factorial (N AS INTEGER)	*function factorial (n:integer): longint;*
DEFINT I	*var i: integer;*
DEFLNG REZ	*rez:longint;*
	begin
REZ = 1	*rez:=1;*
FOR I = 1 TO N	*for i:=1 to n do*
REZ = REZ * I	*rez:=rez*i;*
NEXT	
Factorial =REZ	*factorial :=rez;*
END FUNCTION	*end;*

Первая строчка в описании функции — это заголовок функции. Служебное слово ***function*** (функция) указывает на то, что *именем factorial* названа функция. В скобках перечислен список формальных параметров функции, указаны их имена и задан их тип.

Функция *factorial* имеет один параметр *n* (число, факториал которого мы будем находить), который является целым числом.

Далее в Turbo Pascal 7.0 в заголовке указывается тип значения функции, ее результата. В данном примере результат функции *factorial— целое* число.

За заголовком функции следует описательная часть функции, которая, как и у программы, может состоять из раздела описаний переменных, констант, типов и т.д. В данном примере нам понадобится только раздел переменных. Опишем переменные *i* (переменная для управления циклом) и *rez* (для накопления значения факториала).

Далее идет раздел операторов (тело функции), в котором подсчитывается значение факториала числа. Результат этого вычисления присваивается имени функции.

В тексте программы на Turbo Pascal 7.0 описание функции всегда следует за описанием переменных и до начала основной части, как и описание процедур. После того как функция описана, ее можно использовать в программе.

На Quick Basic через главное меню процедуры и функции записываются отдельно. При распечатке программы, они печатаются после основной программы.

Мы, для удобства сравнения, печатаем их перед основной программой, как на Паскале. Хотя это не принципиально.

Итак, составим программу:

```
DEFINT K,N
DEFLNG A,C
FUCTION factorial (N AS
INTEGER)
DEFINT I
DEFLNG REZ

REZ = 1
FOR I = 1 TO N
REZ = REZ * 1
NEXT
Factorial = REZ
END FUNCTION

CLS
```

```
uses crt;
var n,k: integer;
a1, a2, a3, c:longint;
Function factorial
(n:integer):longint;
var i: integer;
rez:longint;
begin
rez = 1
for i: = 1 to n do
rez: = rez* i

factorial: = rez
end;
begin
clrscr;
```

```
INPUT "n=,k="; N, K          writeln('введите n>=k  ');
                                  readln(n,k);
A1 = factorial (N)                a1:=factorial (n);
A2 = factorial (K)           a2:=factorial (k);
A3 = factorial (N – K)            a3:=factorial (n-k);
C = A1 \ (A2 * A3)           c:=a1 div (a2*a3);
PRINT "C="; C, A1, A2, A3    writeln('c=',c,'   ', a1,'   ',a2,'
',a3);                            readln;
END                               end.
```

QB TP

- При выполнении программы описание функции хранится в памяти ЭВМ. Действия функции выполняются тогда, когда в основной части программы необходимо найти значение функции, то есть в момент обращения к этой функции. Ее вызов осуществляется внутри некоторого выражения. Обращение записывается в виде имени функции, за которым следует в круглых скобки список параметров, отделенных друг от друга запятыми.

Например:

A1 = **factorial(n)** — *есть обращение к функции* **factorial** в QBasic.

a1:= factorial(n) — есть обращение к функции *factorial* в TPascal

- Параметры, записываемые в обращении к функции, называются фактическими, а параметры, указанные в ее описании, называют формальными. Тип фактических параметров определяется типом формальных параметров. Перед вычислением функции формальным параметрам присваиваются значения фактических параметров. Фактическими параметрами могут быть не только константы и переменные, но также и выражения.

- Еще раз подчеркнем, что описание функции - это самостоятельная часть программы, имеющая собственные переменные, которым отводится отдельное, не зависящее от основной программы место в памяти ЭВМ. Этим объясняется тот факт, что переменные, именуемые одним именем и используемые как в описании функции, так и в основной программе, фактически являются разными переменными (в примере — переменная *n* основной части программы и параметр *n* в описании функции). При выполнении программы машина не путает имена этих переменных, так как области их действия не совпадают.
- Таким образом, программист может вводить в описание функции различные имена, не заглядывая в другие части программы, что особенно важно при написании больших программ.

Написать функцию, подсчитывающую количество цифр целого числа. Используя ее, определить, в каком из двух данных чисел больше цифр.

Для решения задачи вспомним, как подсчитать количество цифр? Для этого можно выделять последнюю цифру до тех пор, пока число не станет равным нулю. При этом каждый раз надо увеличивать счетчик на 1 (начальное значение счетчика — 0). Тогда можно описать такую функцию:

```
FUNCTION zifr (x AS INTEGER)
DEFINT K

K = 0
WHILE X <> 0

K = K + 1
X = X \ 10
WEND
zifr = K
END FUNCTION
```

```
function zifr (x:longint):integer;
var k:integer;
begin
k:=0;
while x<>0 do
begin
inc(k);
x:=x div 10
end;
zifr:=k;
end;
```

- В заголовке функции указано ее имя — Zifr. Ей передается только один параметр — число, количество цифр которого надо найти. Результат — то же целое число. В разделе переменных описана переменная *k* — это счетчик цифр. В теле функции с помощью цикла *While* и выполняются указанные выше действия (увеличение значения счетчика и "удаление" последней цифры).

- Еще раз заметим, что память для переменной *k*, которая является локальной, выделяется только тогда, когда начинает свою работу функция. После завершения ее работы эта часть памяти снова освобождается и значение *k* будет не определено.

```
DEFLNG K, N

FUNCTION zifr (x AS INTEGER)
DEFINT K

K = 0
WHILE X <> 0
K = K + 1
X = X \ 10
WEND
Zifr = K
END FUNCTION

CLS
 INPUT "n=,k="; N1, N2
 K1 = zifr(N1):  K2 = zifr(N2)
 IF K1 = K2 THEN PRINT "Одинаково ": END


 IF K1 > K2 THEN  PRINT "В N1 больше чем в
      N2"
              ELSE PRINT "в N2 больше чем в N1"

 END IF
END
```

```
Uses crt;
var n1,n2:longint;
    k1,k2:byte;
function  zifr (x:longint):integer;
 var k:integer;
    begin
     k:=0;
    while x<>0 do    begin
       inc(k);
       x:=x div 10;
     end;
  zifr:=k;
end;
 begin
   clrscr;
    writeln('n1=n2=');readln(n1,n2);
     k1:= zifr(n1);   k2:= zifr(n2);
     if k1=k2 then writeln('Одинаково ')

          else
             if k1>k2 then writeln('B n1>n2')

                        else writeln('B n2>n1');

  readln;
end.
```

QB TP

Домашнее задание

Ответить на вопросы:

1. Как описывается функция?
2. Каковы отличия функции от процедуры?
3. Указывается ли тип результата в Quick Basic?
4. Чем отличаются друг от друга формальные и фактические параметры?
5. Что такое область действия переменной?
6. Чем отличаются друг от друга глобальные и локальные параметры?

Урок 24

Тема урока: Рекурсия. Примеры рекурсивного программирования

Проверка домашнего задания.

- Как описывается функция?
- Каковы отличия функции от процедуры?
- Указывается ли тип результата в Quick Basic?
- Чем отличаются друг от друга формальные и фактические параметры?
- Что такое область действия переменной?
- Чем отличаются друг от друга глобальные и локальные параметры?

Рекурсия

- Подпрограммы в Turbo Pascal и Qbasic могут обращаться к самим к себе. Такое обращение называется **рекурсией**. **Объект**, который частично определяется через самого себя, называется **рекурсивным**.
- Рекурсивные определения как мощный аналитический аппарат используются во многих областях науки, особенно в математике. Для того чтобы не было бесконечного обращения подпрограммы к самой себе, требуется наличие некоторого условия (условного оператора) в тексте программы, по достижении которого дальнейшее обращение не происходит.
- Таким образом, рекурсивное программирование может включаться только в одну из ветвей условного оператора, присутствующего в подпрограмме.
- Некоторые задачи являются рекурсивными по своему определению, поэтому рекурсивные алгоритмы – это точные копии с соответствующего определения.

Вычисление факториала натурального числа

- Как правило, N! факториал определяют как произведение первых N чисел: N!=1·2·3· … · N, (так называемое итеративное определение). Такое произведение конечно можно легко вычислить с помощью оператора цикла, однако существует и другое определение факториала (рекурсивное):
- Для того, чтобы вычислить N!, надо знать значение (N—1)! и умножить его на N, при этом 1!=1. В общем виде это можно записать так:

$$N! = \begin{cases} 1, при\ N = 1 \\ N * (N-1)!, при\ N > 1 \end{cases}$$

- Для вычисления факториала опишем функцию, которая будет вычислять его значение. Ей будет передаваться целое число, поэтому у нее есть только один параметр. Результат выполнения — тоже целое число. Например, можно описать такую функцию:

Вычисление факториала натурального числа

```
FUNCTION factorial          Function   factorial
(N AS INTEGER)              (n:integer):longint;
                    Begin
IF N = 1 THEN factorial = 1    if n=1 then factorial: =1
ELSE factorial  = N * factorial  else
(N - 1)                  factorial:=n*factorial
                    (n-1);
END IF
END FUNCTION                end;
```

- Найдем 3!. Как же будет вычисляться факториал этого числа? Первый вызов этой функции будет из основной программы (Например, a:= factorial(3)), где переменной **a** присваиваем значение 3!).
- При каждом обращении к функции будет появляться свой набор локальных переменных со своими значениями, в данном случае — переменная **n**, для которых выделяется память. А после завершения работы эта часть памяти освобождается и переменные удаляются.
- Так как N<>1, то пойдем по ветке Else и функции Factorial присваиваем значение n*Factorial(n—l), т. е. надо умножить 3 на значение функции Factorial(2). Поэтому обращаемся второй раз к этой же функции, но передаем ее новое значение параметра -2.

- Так делаем до тех пор, пока не передадим значение, равное 1. Тогда N=1, а поэтому значение функции Factorial:=1. Таким образом, N=1 — это условие, по которому процесс входа в следующую рекурсию заканчивается. Идет возвращение в точку вызова и подстановка в оператор присвоения значения вычисленной функции. То есть возвращаемся в предыдущую функцию для n=2:Factorial:=n*Factorial(n-l), значит, Factorial:=2*1, следовательно, Factorial(2)=2. И возвращаемся дальше n=3: Factorial:=n*Factorial(n-l), значит, Factorial:=3*2*1, следовательно, Factorial(3)=6.. Таким образом, получаем значение *Factorial(3)=6,* это значение и присвоим переменной *a.*

Программа вычисления факториала натурального числа

```
DEFINT N                    Uses crt;
DEFLNG A                        Var n:integer;
FUNCTION factorial  (N AS INTEGER)      a:longint;
                                function    factorial
                            (n:integer):longint;
                                begin
IF N = 1 THEN factorial  = 1            if n=1 then factorial :=1
 ELSE factorial = N * factorial (N - 1)     else factorial :=n*factorial
                            (n-1);
END IF
END FUNCTION                                end;
                            begin
CLS                             clrscr;
INPUT "n="; N                       writeln('n='); readln(n);
A= factorial (N)                    a:=factorial (n);
PRINT"A=";A                         writeln('a=',a);
                            readln;
END                             end.
```

QB TP

- Если для факториала, на первый взгляд, рекурсивное определение выглядит сложнее чем итеративное, то для чисел Фибоначчи рекурсивное определение выглядит для вычисления лучше чем прямая формула.

$$F(n)=\begin{cases}1, при\ n=1\\1, при\ n=2\\F(n-1)+F(n-2), при\ n>2\end{cases}$$

рекурсивное определение

$$F(n)=\frac{\left(\frac{1+\sqrt{5}}{2}\right)^n+\left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$$

прямое определение

Похожие рекурсивные формулы можно вывести также и для многих других математических определений. Можно организовать вычисления и без рекурсивных соотношений, но использование рекурсии позволяет легко запрограммировать вычисления по рекурсивным формулам. К достоинствам рекурсии можно отнести то, что она позволяет с помощью конечного выражения задать бесконечное вычисление, причем алгоритм вычисления не будет содержать повторений.

Многие задачи не содержат явно в себе рекурсию, однако некоторые из них можно свести к рекурсивным.

Нахождение НОД (наибольшего общего делителя) двух натуральных чисел

- Способ нахождения этого значения - алгоритм Евклида.
- Пусть есть два целых числа *a* и *b.*

Если a=b, то НОД(a,b)=a.
Если a>b, то НОД(a,b)=НОД(a-b,b).
Если a<b, то НОД(a,b)=НОД(a,b-a).
Например. Найдем наибольший общий делитель чисел 22 и 16.

A	B	Примечание
22	16	так как a>b, то a = a – b
6	16	так как b>a, то b = b – a
6	10	b = b – a
6	4	так как a>b, то a =a - b
2	4	b = b – a
2	2	Так как a = b, то НОД = a

Составим следующую функцию и программу:

```
DEFLNG A-B
FUNCTION nod (a AS LONG,
b AS LONG)
IF A = B THEN nod = A
ELSE
IF A > B THEN
nod = nod (A - B, B):EXIT FUNCTION
ELSE nod = nod(A, B – A)
END IF
END FUNCTION

 CLS
INPUT "a,b="; A, B
A = nod(A, B)
PRINT "nod="; A

END
```

```
Uses crt;
var a,b:longint;
function nod (a,b:longint): longint;
begin
if a =b  then nod:=a
else
if a>b then nod:=nod(a - b,b)
else nod:=nod (a, b- a)
end;
begin
clrscr;
writeln('a=b=');readln(a,b);
a:=nod(a,b);
writeln('nod= ',a);
readln;
end
```

QB TP

Перевод натурального числа из десятичной системы счисления в двоичную

Для решения этой задачи рассмотрим сначала, как перевести число из десятичной системы счисления в двоичную. Пусть есть число 39, которое и надо представить в двоичной системе. Для этого разделим его на 2, получим целую часть и остаток от деления. Целую часть снова делим на 2 и получаем целую часть и остаток. Так делаем до тех пор, пока целую часть можно делить на 2 (то есть пока она не станет, равной 1).

Теперь, начиная с этой единицы, выписываем в обратном порядке все остатки от деления, это и будет запись числа 39 в двоичной системе счисления: $3910=100111_2$.

Таким образом можно переводить любое натуральное число из десятичной системы счисления в двоичную, а также и другие системы (например, восьмеричную или шестеричную).

```
DEFLNG N
SUB rec (N AS LONG)
IF N > 1 THEN rec (N \ 2)
PRINT N MOD 2;
END SUB

CLS
INPUT "n="; N
rec (N)

END
```

```
uses crt;
    var n:longint;
        procedure  REC (n:longint);
        begin
        if n>1  then rec(n div 2);
            write(n mod 2);
        end;
begin
    clrscr;
        writeln('n=');readln(n);
    rec(n);
readln;
    end.
```

Результат работы программы при N=39 выводится на экран: ***100111***
Первая цифра (1) выводится на экран из последнего вызова, следующая цифра (0) из предпоследнего и так далее, последняя (1) — из первого. Таким образом, вывод очередной цифры происходит перед тем, как выйти из данной функции.

QB TP

Определить, является ли заданное натуральное число простым

Суть заключается в том, что если нельзя извлечь рекурсию из постановки задачи, то можно расширить задачу, обобщить ее (например, введя дополнительный параметр). Удачное обобщение дает возможность увидеть рекурсию. После этого возвращаемся к частному случаю и решаем исходную задачу.

Данную задачу можно обобщить, например, так: определить, верно ли, что заданное натуральное число *N* не делится ни на одно число, большее или равное *M,* но меньшее *N.* При этом значения числа M находятся в промежутке от *2* до *N.* Истина может быть в двух случаях:

- Если *M=N*;
- *N* не делится на *M* и истина для чисел *M+1* и *N*.

N не делится на *M* — это значит, что остаток от деления *N* на M не равен 0.

Программа определения, является ли заданное натуральное число простым.

```
DEFINT M-N                          uses crt;
FUNCTION                            var m,n,S:integer;
Prost (M AS INTEGER, N AS INTEGER)  function prost
                                    (m,n:integer):boolean;
IF N = M THEN prost = -1: EXIT      begin
FUNCTION                            if n=m  then prost :=true
ELSE prost = (N MOD M <> 0)         else prost:=(n mod m<>0) AND
                                    and
prost(M + 1, N)                     prost(m+1,n);
END IF
END FUNCTION                        end;
                                    begin
CLS                                 clrscr;
```

```
INPUT "n="; N                    write('n=');readln(n);
 M = 2                           m:=2;
                                 prost(m,n);
IF prost(M, N) <> 0 THEN         if prost(m,n)  then write(n,'-
PRINT N; "-простое”              простое ')
ELSE   PRINT N; “-составное”          else write(n,'-состовное ');
END IF
                                 readln;
END                               end.
```

Домашнее задание.
Найдите все простые числа от 2 до 50.

QB TP

Урок 25

Тема урока: **Применение подпрограмм при решении задач**

Ход урока.

I. Проверка домашнего задания.

Ответить на вопросы:

- Чем отличаются друг от друга глобальные и локальные параметры?
- Что такое рекурсия?
- Как избавиться от бесконечного обращения подпрограммы к самой себе?
- Все ли задачи можно свести к рекурсивным?

Проверка домашней задачи.

Найдите все простые числа от 2 до 50.

В предыдущей задаче заменить INPUT "n="; N на цикл FOR N=2 to 50

Составить программу нахождения суммы элементов целочисленного массива, состоящего из 20 элементов.

Опишем две процедуры:

- формирования массива;
- вывода массива;
- функцию нахождения суммы элементов

Эти процедуры и функции будем использовать в основной части.

Значения элементов массива вводим с помощью генератора случайных чисел.

В приведенной ниже программе процедура Init отвечает за заполнение массива случайными числами, процедура Print отвечает за вывод элементов массива.

	uses crt;
CONST c = 20	*const c=20;*
	type mas=array[1..c] of integer;
DIM A(C) AS INTEGER	*var a: mas;*
DEFINT P	*p:integer;*
SUB init (M() AS INTEGER)	*procedure init (var m: mas);*
DEFINT I	*var i:integer;*
	begin
RANDOMIZE TIMER	*randomize;*

```
FOR I = 1 TO C              for i:=1 to c do
M(i) = RND * 45 – 22            m[i]:=trunc(random(45))-22;
NEXT
END SUB                 end;

SUB print (M() AS INTEGER)      procedure print (var m:mas);
 DEFINT I               var i:integer;
                    begin
 FOR I = 1 TO C             for i:=1 to c do
 PRINT M(I);" ";             write(m[i],' ');
 NEXT
END SUB                 end;
```

Функция Summa находит сумму элементов массива, процедура upor упорядочивает элементы массива.

Для упорядочения элементов массива воспользуемся следующим алгоритмом: Берем первый элемент и сравниваем его со всеми последующими. Если они не равны, то меняем их местами и продолжаем сравнивать.

```
FUNCTION summa (M() AS INTEGER)     function summa (var m:
                                        mas):integer;
DEFINT I, S                             var i,s:integer;
                                        begin
 S = 0                                      s:=0;
 FOR I = 1 TO C                             for i:=1 to c do
 S = S + M(I)                                   s:=s+m[i];
 NEXT
Summa = S                               summa:=s;
END FUNCTION                            end;
```

```
SUB upor (M() AS INTEGER)          procedure upor (var m:
                                        mas);
DEFINT I-J                              var i,j,z:integer;
                                   begin
FOR I = 1 TO C – 1                      for i:=1 to c-1 do
FOR J = I + 1 TO C                      for j:=i+1 to c do
IF M(I) < M(J) THEN SWAP                if m[j]>m[i] then
M(I), M(J)                         begin
NEXT J, I                          z:=m[j];m[j]:=m[i];m[i]:=z;
                                        end;
END SUB                            end;
```

Вид основной программы

```
                                    begin
CLS                                 clrscr;
CALL init(A())                          init(a);
CALL print(A())                         print(a);
PRINT    CALL upor(A())                     upor(a);
CALL print(A())                         print(a);
PRINT                               writeln;
 p = summa(A())                         p:=summa(a);
PRINT "summa="; P                           writeln('s=',p);
                                    readln;
 END                                end.
```

QB TP

Пример 2

Составить процедуру упорядочения массива.

Для упорядочения элементов массива воспользуемся следующим алгоритмом:

Берем первый элемент и сравниваем его со всеми последующими. Если они не равны, то меняем их местами и продолжаем сравнивать. В итоге на последнем месте окажется самый маленький (или самый большой) в зависимости от знака неравенства.

Далее, берем второй элемент, третий и так до предпоследнего.

```
SUB upor (M() AS INTEGER)
    DEFINT I-J

 FOR I = 1 TO C – 1
 FOR J = I + 1 TO C
IF M(I) < M(J) THEN SWAP
M(I), M(J)
NEXT J,I
END SUB
```

```
Procedure upor (var m: mas);
var i,j,z:integer;
begin
for i:=1 to c-1 do
for j:=i+1 to c do
if m[j]>m[i] then
begin
z:=m[j];m[j]:=m[i];m[i]:=z;end;
end;
```

Составьте процедуру нахождения максимального элемента и его индекса в одномерном целочисленном массиве.

Для решения задачи в процедуре задаем начальный максимум и его индекс. Например, первый элемент массива. Затем сравниваем максимум с каждым элементом массива и если найдем элемент, больший максимума, то перезапоминаем максимум и идем далее.

```
SUB MINMAX (M() AS INTEGER          Procedure minmax
MAX AS INTEGER, IND AS INTEGER)      (m:mas; var
                                     max, ind:integer);
DEFINT I                                var i:integer;
                                     begin
MAX = M(1): IND = 1                        max:=m[1];ind:=1;
FOR I = 1 TO C                          for i:=1 to c do
IF MAX <= M(I) THEN                        if max<=m[i] then
MAX = M(I): IND = I                        begin max:=m[i]; ind:=i; end;
NEXT
END SUB                                 end;
```

В основной программе необходимо описать переменные MAX, IND.

Составьте процедуры:
- заполнения целочисленного массива A(20) датчиком случайных чисел;
-печать массива;
- нахождение одинаковых элементов в массиве с указанием их индексов.

Первые две процедуры возьмем из предыдущих примеров. Алгоритм нахождения одинаковых элементов состоит в следующем:

1. Каждый элемент массива, кроме последнего, (докажите почему) сравниваем со всеми идущими после него.
2. В случае нахождения одинаковых, печатаем их индексы.

```
SUB RAVN (M() AS INTEGER)
DEFINT I-J
FOR I = 1 TO C - 1
FOR J = I + 1 TO C
IF M(I) = M(J) THEN PRINT I, J
NEXT NEXT
END SUB
```

```
procedure RAVN (var m: mas);
var i,j:integer;
begin
for i:=1 to c-1 do
for j:=i+1 to c do
if m[i]=m[j] then writeln (i,' ',j);
end;
```

QB TP

Домашнее задание.

Ответить на вопросы:

- Чем отличаются друг от друга глобальные и локальные параметры?
- Что такое рекурсия?
- Как избавиться от бесконечного обращения подпрограммы к самой себе?
- Все ли задачи можно свести к рекурсивным?
- Как описывается функция?
- Каковы отличия функции от процедуры?
- Указывается ли тип результата в Quick Basic?
- Чем отличаются друг от друга формальные и фактические параметры?
- Что такое область действия переменной?

Урок 26

Тема урока: **Применение подпрограмм при решении задач**

Ход урока.

I. Проверка домашнего задания.

Ответить на вопросы:

- Чем отличаются друг от друга глобальные и локальные параметры?
- Что такое рекурсия?
- Как избавиться от бесконечного обращения подпрограммы к самой себе?
- Все ли задачи можно свести к рекурсивным?
- Как описывается функция?
- Каковы отличия функции от процедуры?
- Указывается ли тип результата в Quick Basic?
- Чем отличаются друг от друга формальные и фактические параметры?
- Что такое область действия переменной?

Дана строка, состоящая из слов разделенными одним пробелом.

Составить процедуру:

А) определяющую количество слов в строке;

Б) вывода на экран этих слов;

В) печатающие слова по возрастанию их длин;

Г) проверяющие слова, являются ли они палиндромами.

Составляем процедуру определяющую количество слов в строке.

```
                    Uses crt;
CONST N = 30                const n=30;
DIM B(N) AS STRING              type mas=array[1..n] of string;
                    var  b:mas;
DIM ST AS STRING            st:string;
DEFINT  K                   k:integer;
```

В процедуре разбиваем предложения на слова и считаем их количество.

```
SUB slowa (ST AS STRING, B()        procedure slowa (st:string;var
AS STRING,  K AS INTEGER)           b:mas;var k:integer);
DEFINT I                    var i:integer;
                    begin;
K = 1                       k:=1;
FOR I = 1 TO LEN(ST) – 1        for i:=1 to length(st)-1 do  begin
```

Пока не встретился пробел

```
IF MID$(ST, I, 1) <> " " THEN
B(K) = B(K) + MID$(ST, I, 1)
ELSE
IF I <> LEN(ST) - 1 THEN
K = K + 1: B(K) = “”
END IF
NEXT
END SUB
```

```
begin
if st[i]<>  ' ' then
        b[k]:=b[k]+st[i]
else
if i<>length(st)-1 then
begin  k:=K+1;b[k]:='';end;
end;
```

Печатаем массив слов

```
SUB prin (B() AS STRING,          procedure print (var b:mas);
K AS INTEGER)                     var i:integer;
                                  begin
DEFINT I
FOR I = 1 TO K                    for i:=1 to k do
PRINT B(I); " ";                  write (b[i],' ');
NEXT
END SUB                           end;
```

Упорядочиваем массив слов по возрастанию их длин

```
SUB upor (B() AS STRING,        Procedure upor (var b:mas);
K AS INTEGER)   DEFINT I-J            var i,j:integer;
                          z:string;
                          begin
FOR I = 1 TO K-1                for i:=1 to k-1 do
FOR J = I + 1 TO K              for j:=i+1 to k do
IF LEN(B(I))<=LEN(B(J))THEN SWAP B(I),B(J)
                          if length(b[i])<=length(b[j])
    then
                    begin z:=b[i];b[i]:=b[j];b[j]:=z;end;
NEXT J, I
END SUB                       end;
```

Проверяем слова, являются ли они палиндромами.

```
SUB palindrom (B() AS STRING,       procedure palindrom
K AS INTEGER)                       (var b:mas;var k:integer);
DEFINT I-J                          var i,j:integer;
DEFSTR Z                            z:string;
                                    label 10;
                                    begin
FOR I = 1 TO K                      for i:=1 to k do begin
Z = B(I)                            z:=b[i];
FOR J = 1 TO LEN(B(I))              for j:=1 to length (b[i])  do
IF MID$(Z, J, 1) <> MID$            if z[j]<>z[length(b[i])-j+1]
(Z, LEN(B()) - J + 1, 1)            then
THEN PRINT z; " не палиндром":      begin writeln(z,
GOTO 10                             'не палиндром'); goto 10;end;
NEXT
PRINT Z; " палиндром"               writeln(z,'  палиндром');
10 NEXT                             10:   end;
END SUB                             end;
                                    begin
CLS                                 clrscr;
INPUT "предложение"; st             write('st=');readln(st);
```

Основная программа. Добавляем пробел в конце предложения.

```
ST = ST + " "                          st:=st+' ';
CALL slowa(ST, B(), K)                      slowa(st,b,k);
PRINT "slow="; K                       writeln( 'слов =' ,k);
CALL print(B(), K)                     print(b);
CALL upor(B(), K)                      upor(b);
PRINT                                  writeln;
CALL print(B(), K)                     print(b);
PRINT                                  writeln;
CALL palindrom(B(), K)                      palindrom(b);
                                  readln;
END                                    end.
```

QB TP

Составить процедуру для заполнения двумерного массива целыми случайными числами.

Найти:

- первый отрицательный элемент;
- максимальный элемент массива;
- одинаковые элементы массива;
- поменять местами первый отрицательный и максимальный элемент.
- Для решения этой задачи составим соответствующие процедуры.

Процедура заполнения массива целыми случайными числами

Начало основной программы

```
                        uses crt;
CONST C = 3                         const c=3;
DIM A(C, C) AS INTEGER              type mas=array[1..c,1..c]
              of integer;
                        var a:mas;
DEFINT D, F, I-J, M                 max,im,jm:integer;
                        io,jo,f:integer;
```

Процедура заполнения массива случайными числами

```
SUB init (M() AS INTEGER)
DEFINT I-J
RANDOMIZE TIMER
FOR I = 1 TO C
FOR J = 1 TO C
M(I, J) = RND * 5 – 2
NEXT
NEXT
END SUB
```

```
begin
procedure init (var m:mas);
var  I,J:integer;
begin
RANDOMIZE;
FOR i:= 1 TO c do
FOR j:= 1 TO c do
m[i, j]:=trunc(Random
( 5)) - 2
end;
```

Процедура вывода массива на экран

```
SUB PRIN (M() AS INTEGER)   procedure PRINT(m:mas);
DEFINT I-J                  var I,J:integer;
                        begin
FOR I = 1 TO C              FOR i:= 1 TO c do begin
FOR J = 1 TO C                  FOR j:= 1 TO c   do
PRINT M(I, J); " ";             write( m[i, j]:4);
NEXT
PRINT                   writeln;
NEXT                    end;
END SUB                     end;
```

Процедура нахождения одинаковых элементов в массиве

```
SUB odinak (M() AS INTEGER)     procedure odinak (m :mas);
DEFINT I-L                      var I,j,k,l:integer;
                                begin
FOR I = 1 TO C                  FOR i:= 1 TO c do
FOR J = 1 TO c                  FOR j:= 1 TO c    do
‘PRINT I; J; "=";               {PRINT i; j; "=";}
FOR K = 1 TO C                  FOR k:= 1 TO c   do
FOR L = 1 TO C                  FOR l:= 1 TO c do
IF ((I <> K) OR (J <> L)) AND                IF ((i <> k) OR (j <> l)) AND
(M(I, J) = M(K, L))                     (m[i, j] = m[k, l])
THEN PRINT K; L;                        THEN write (i,j,' ', k, l,'  ');
NEXT L, K
NEXT J, I
END SUB                         END;
```

Процедура нахождения первого отрицательного элемента

```
SUB otr (M() AS INTEGER, IO,          procedure otr (var m:mas;
JO AS INTEGER)                        var io,jo:integer);
DEFINT I-J                          var I,J:integer;
                                  begin
FOR I = 1 TO C                        FOR i:= 1 TO c do begin
FOR J = 1 TO C                        FOR j:= 1 TO c do
IF M(I, J) < 0 THEN IO = I:           IF m[i, j] < 0 THEN begin io:= i;
JO = J: EXIT SUB                      jo:= j; EXIT;end;
NEXT
NEXT                                End;
END SUB                             end;
```

Процедура нахождения максимального элемента массива

```
SUB maxmin (M() AS INTEGER,        procedure maxmin (var m:mas;
MAX, IM, JM AS INTEGER)            var max,im,jm:integer);
DEFINT I-J                         var  I,J:integer;
                                   begin
Max = N(1, 1): IM = 1: JM = 1      max:= m[1,1]; im:=1; jm:=1;
FOR I = 1 TO C                     FOR i:= 1 TO c do begin
FOR J = 1 TO C                     FOR j:= 1 TO c do
IF max <= M(I, J) THEN             IF max <= m[i, j] THEN begin
max = M(I, J): IM = I: JM = J:     max:= m[i, j]; im:= i; jm:= j;
'exit do                           {exit;};end;
NEXT
NEXT                               End;
END SUB                            End;
```

Основная программа имеет вид.
Она становится компактной и легко читаемой

```
                    begin
CLS                 clrscr;
CALL init(A())              init(a);
CALL PRIN(A())              PRINT(a);
CALL odinak(A())            odinak(a);
PRINT                   writeln;
CALL otr(A(), IO, JO)           otr(a, io, jo);
PRINT "1-ый отр = "; A(IO, JO),     writeln('1-ый отр = ', a[io, jo],
IO; JO                  ``, io, jo);
CALL maxmin(A(), max, IM, JM)       maxmin(a, max, im, jm);
PRINT "max="; max, im; jm           writeln('max=', max,'  ', im, jm);
```

Меняем местами первый отрицательный и максимальный элемент.

```
f = A(io, jo): A(io, jo) = A(im, jm):   f:= a[io, jo]; a[io, jo]:=a[im,
A(im, jm) = f jm];                      a[im, jm]:= f;
PRINT                                   writeln;
CALL PRIN(A())                                  PRINT(a);
                                        Readln;
 END                                    end.
```

QB TP

Контрольная работа

Вариант 1

1. Найти сумму цифр числа.
2. Среди чисел из интервала от *A* до *B* найти все простые.
3. Каждый элемент одномерного массива *A(K)*, заполненного натуральными числами, замените наибольшим простым делителем этого элемента.

Вариант 2

1. Составьте программу, проверяющую, является ли число палиндромом.
2. Определить, является ли число совершенным, то есть равно ли оно сумме своих делителей, кроме самого себя.
3. Найдите наибольший простой делитель среди всех делителей каждого элемента данного натурального массива A(K).

Вариант 3

1. Дано N чисел. Вывести на экран наибольшую из первых цифр заданных чисел.
2. Составить программу разложения данного числа на простые множители.
3. В массиве A(K), заполненным натуральными числами, определите количество элементов, являющихся простыми числами, индексы которых также простые числа.

Вариант 4

1. Имеется трехзначное число, цифры которого различны. После зачеркивания в нем средней цифры остается двузначное число, являющееся делителем данного. Найдите N-первых таких трехзначных чисел.
2. Даны координаты трех вершин треугольника. Найти длины всех его сторон.
3. В массиве A(K), заполненным натуральными числами, замените значения элементов, индексы у которых простые числа, на значения соответствующих индексов.

Ответы к контрольной работе:

Вариант 1

1. Тест: 12 Ответ 3
   Тест: 347 Ответ 14
2. Тест: A=2 B=10 Ответ 3,5,7
3. Тест: K=5 6, 7, 10, 32, 5 Ответ: 3, 7, 5, 2, 5

Вариант 2

1. ***Тест***: 12721 Ответ: палиндром
   Тест: 127 Ответ: не палиндром
2. Тест: 6, 28, 496 Ответ: совершенное
   Тест: 7 Ответ: не совершенное
3. Тест: K=5 34, 64, 295, 24, 100 Ответ: 19
   Тест: K=3 6, 7, 10 Ответ: 7

Вариант 3

1. Тест: N=3 25, 730, 1995 Ответ: 7
2. Тест: 6 Ответ: 1 2 3
   Тест: 36 Ответ: 2 2 3 3
3. Тест: K=7 5, 6, 7, 10, 1, 3, 2 Ответ: 3

Вариант 4

1. Тест: N=3 Ответ: 105, 108, 110
2. Тест: (0, 0), (0,4), (3,0) Ответ: 3, 4, 5
3. Тест: K=7 5, 6, 7, 10, 1, 3, 2 Ответ: 1, 6, 3,10, 5, 3, 7

Задачи для самостоятельного решения

Вариант 1

1) Используя процедуру, вычислить значение выражения: y=a1x4+a2x3+a3x2+a4x+a5, где коэффициенты a1, a2, a3, a4, *a5* и x — это числа, вводимые с клавиатуры.
2) Определить, является ли число простым.
3) Даны действительные числа S и T. Получить F(t, -2*s, 1.17) + F(2.2, t, t-s), где F(A,B,C) = (2*A – B - SIN(C))/(5+ABS(C)).

Вариант 2

1. Найти сумму цифр числа.
2. Среди чисел из интервала от A до B найти все простые.
3. Даны действительные числа S и T. Получить

*G(1.2, s) + G(t, s) + G(2*s-1, s*t), где G(A,B) = (A2 + b2)/(A2 + 2*A*b + B2 + 3*B2 + 4)*

Вариант 3

1. Найти первую цифру числа.
2. Составьте программу, проверяющую, является ли число палиндромом (например, число 12721 — палиндром).
3. Каждый элемент одномерного массива, заполненного натуральными числами, замените наибольшим простым делителем этого элемента.

Вариант 4

1. Определить, является ли число совершенным, то есть равно ли оно сумме своих делителей, кроме самого себя (используя преобразованную функцию из предыдущего примера).
2. Составить программу разложения данного числа на простые множители.
3. Даны действительные числа A, B, C. Получить
   *max(a, a+b) + max(a, b+c)/ (1 + max(a + b*c, 1.15)*

Вариант 5

1. Составить программу, вычисляющую наименьшее общее кратное четырех заданных с клавиатуры чисел.
2. Определить, является ли число совершенным, то есть равно ли оно сумме своих делителей, кроме самого себя
3. Даны действительные числа S,T. Получить H(s,t) + max(H(s-t, s*t)2, H(s-t, s+t)4) + H(1,1),

   где H(A,B) = A/(1+A2) + B/(1+B2)-(A2-B2)

Вариант 6

1. Составить программу нахождения наибольшего общего делителя нескольких чисел, используя функцию нахождения НОД двух чисел.
2. Сколько натуральных чисел от 20 до 120 не взаимно простых с 30?
3. Даны натуральное число N. Среди чисел 1,2, ..., N найти все, которые можно представить в виде суммы квадратов двух натуральных чисел. (Определить процедуру, позволяющую распознавать полные квадраты.)

Вариант 7

1. 1. Составить программу, вычисляющую наименьшее общее кратное четырех заданных с клавиатуры чисел.
2. 2. Найти натуральное число, произведение всех делителей которого равно 5832.
3. 3. Даны действительные числа X(1),Y (1), ..., X(10),Y(10). Найти периметр 10-угольника, вершины которого имеют соответствующие координаты. (Определить процедуру нахождения расстояния между двумя точками, заданными своими координатами.

Вариант 8

1. Дано четыре числа. Вывести на экран наибольшую из первых цифр заданных чисел. Например, если a=25, b=730, c=127, d=1995, то должна напечататься цифра 7.
2. Проверьте утверждение, что если 2(N-1)-1 число простое, то число вида 2N*{2(N+1)-1} является совершенным. (т. Эйлера)
3. Даны натуральное число N, действительные числа X(1),Y(1), ...,X(N),Y(N). Найти площадь многоугольника, вершины которого при некотором обходе имеют координаты X(1,)Y(1), ... ,X(N),Y(N). (Определить процедуру вычисления площади треугольника по координатам его вершин.)

Вариант 9

1. Даны три натуральных числа. Найдите НОД наибольшего и наименьшего из этих чисел.
2. Найти натуральные числа с 2 до 1000 такие, что числа M,M+10, M+14 все были простыми.
3. Дано четное число N. Проверить для этого числа гипотезу Гольдбаха. Эта гипотеза (по сегодняшний день не опровергнутая и полностью недоказанная) заключается в том, что каждое четное N>=2, представляется в виде суммы двух простых чисел. (Определить процедуру распознавания простых чисел)

Вариант 10

1. Имеется трехзначное число, цифры которого различны. После зачеркивания в нем средней цифры остается двузначное число, являющееся делителем данного. Найдите все такие трехзначные числа.
2. Найти все простые натуральные числа от 2 до 1000, чтобы числа $2*P^2+1$ также были простые.
3. Дано натуральное число N. Выяснить имеются ли среди чисел N, N+1, ...2*N близнецы, т.е. простые числа, разность между которыми равна двум. (Определить процедуру распознавания простых чисел)

Вариант 11

1. Найдите все трехзначные числа, кратные семи, у которых сумма цифр тоже кратна семи, Если найденное число оканчивается нечетной цифрой, то определите, будет ли оно простым числом.
2. Проверьте утверждение: если числа P и $8*P^2+1$ простые, то число $8*P^2+2*P+1$ также простое. (2 <= P<=100).
3. Для данного простого числа P найдите следующее простое число.

Вариант 12

1. Даны координаты трех вершин треугольника. Найти длины всех его сторон.
2. Найти число натуральных чисел меньших 100 и взаимно простых с 36.
3. В массиве A(K), заполненным натуральными числами, определите количество элементов, являющихся простыми числами, индексы которых также простые числа.

Вариант 13

1. Составьте программу, проверяющую, является ли число палиндромом (например, число 12721 — палиндром).
2. У числа 600 найдите сумму делителей и их число.
3. Найдите наибольший простой делитель среди всех делителей каждого элемента данного натурального массива. (34, 64, 225, 24, 100 НД=19)

Вариант 14

1. Сколько натуральных чисел меньших 1000 не делится ни на 5, ни на 7?
2. Найти натуральное число, зная, что оно имеет два простых делителя, а всего 6 делителей, сумма которых равна 28.
3. Каждый элемент одномерного массива, заполненного натуральными числами, замените наибольшим простым делителем этого элемента.