heron (2·x + 3; y/2, 2·sinx, ter);

~~fun~~

function heron (a1; a2; a3: real): real;
var s real
begin
s:=(a1+a2+a3)/2
heron:=sqrt (s·(s-a1)(s-a2)(s-a3));
end.;
ter:=heron (2,3;4)

writeln (x; y; 2; heron (x,y,2));

if heron (2*x+3, y/2,2)>10
then ...
else ...;

A függvényeljárás egy másik eljárás paraméterlistáján, vagy valamilyen feltételes utasítás relációjaként szerepel, vagy értékadó utasítás jobboldalán.

Ha egy nagy programot modulokra bontunk és azok megírásánál lokális változókat használunk,

Ilyen esetben a memóriában csak a működő alprogram változói foglalnak helyet, az újabb indításkor a változók az előzőleg lefoglalt memóriaterületre kerülnek.

## rekurzív eljárások, fgv. ill. eljárások

Egy eljáráshívás rekurzív tekinthető, ha önmagát eggyel kisebb paraméterrel meghívja, ill. ha két azonos hierarchiaszinten elhelyezkedő eljárás egymást kölcsönösen meghívja.

$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$

```
function faktoriális (n : longint) : longint
  var i, fakt : longint
  begin
     fakt := 1
     for i := 1 to n do
     fakt := fakt * i ;
     faktoriális := fakt
  end ;
szám := faktoriális (5);
readln (e);
szam := faktoriális (e);
function faktoriális (n longint) : longint
  begin
   if n = 0  then faktoriális := 1
             else faktoriális := n * faktoriális (n-1)
```

$$fibonacci(n) = \begin{cases} n & \text{ha } n < 2 \\ fibonacci(n-1) + fibonacci(n-2) \end{cases}$$

```
function fibonacci (n: longint): longint ;
begin
if  n < 2  then fibonacci := n
           else fibonacci := fibonacci (n-1) + fibonacci (n-2)

end ;
szám := fibonacci (9)
```

```
function fibonacci (n: longint): longint ;
 var  fib, i, a, b : longint;
 a := 0 ;  b := 1 ;  i := 0;
 fib := 0;
 while     i < n      do
  begin
          a := b;
          b := fib;
          fib := a + b;
          i := i + 1;

  end;
```

~~1.~~ 22

Programozási nyelvek ea. XI. 4.

## ELDÖNTÉS

```
program eldont;
uses crt;
var tomb: array (1..100) of integer;
i, elemszam: integer
van: boolean;
valasz: char;
procedure eldontes(n: byte; var jel: boolean);
begin
if (n<=elemszam) and (tomb(n) mod 2<>0) then
begin n:=n+1; jel:= n<=elemszam; eldontes
(n, jel) end
else jel:=n<=elemszam
end;
BEGIN
repeat
clrscr;
write('Kérem az adatok számát (max. 100)!');
readln(elemszam);
for i:=1 to elemszam do
begin write(i,'. adat!'); readln(tomb[i])
end;
i:=1; eldontes(i, van);
if van then writeln('Van páros eleme a sorozat-
nak!')
writeln('Van-e újabb adatsor (i,n)?')
```

15

HATVÁNYOZÁS (rekurzió) 2

```
program hatvany
var a: integer
    v: char
    x: real

function hatvany(a: real; n: integer): real;
begin
 if n=0 then hatvany:=1
 else if n>0 then hatvany:=a* hatvany(a,n-1);
 else hatvany:=1/hatvany(a,-n);
 end;
 begin
 repeat
 clrscr;
 writeln('A az N-ediken meghatározása');
 write('A='); readln(x);
 write('N='); readln(a);
 writeln(x:6:3,' ',a,'. hatványa=',(x,a):6:3);
 write('Van újabb adat?'); v:=readkey
 until v:='n'
 END.
```

# Unitok

Unit: eljárás- és függvénygyűjtemény

~~b) Standard unitok~~

```
UNIT ␣ név;
interface
uses ...
type ....      } deklaráció
const ....
var ....
```

Felsorolja az összes eljárást és fv.-t, amit használni akarunk, de elegendő csak a fejét megadni.

```
procedure n1(a,b,c: integer, var d: real);
function fg(x: real): real;
  ⋮
implementation
procedure n1; (fölösleges a par. listát megismételni)
  ⋮ } kifejtés
begin
end;
function fg;
  ⋮ } kifejtés

initialization
begin } ha nincs benne utasítás, a begin
end;  } elhagyható!
```

Kimentéskor ugyanazt a nevet kell adni, mint a UNIT neve. ( uncv.pas)
Compiler fordítóval le kell fordítani, így .TPU kiterjesztése lesz (TurboPascal Unit)

program sajat;
~~Unit~~ uses uncv; (a prg. használja a UNIT-ot)
↑
Saját unit

Gyári unitok: crt, system, printer, dos, graph, overlay ...

sak azt a UNIT-ot tudom használni, amelyik ugyanabban a PASCAL verzióban van lefordítva, mint amelyik verzióban írom a programot.

Gyári unitok: a programmal együtt adják

- CRT unit: képernyő- és billentyűzet kezeléshez
- SYSTEM unit: automatikusan hozzáíródik a programhoz, nem kell külön feltüntetni
- GRAPH unit: grafikai eszközöket tartalmaz
- DOS unit: a DOS-t összekapcsolja a PASCAL-la Tudok DOS parancsot használni (pl.: DATE, TIME ...)
- PRINTER unit: ha sonyomtatót használ a PASCAL.
- OVERLAY unit: lehetőséget a prg "felszeleteléséré": a programot rezidens és tranziens részre osztja

rezidens → állandóan a tárban van

tranziens → csak ha hálatkor tölti be

# CRT unit

- billentyűzet és képernyő kezelése
- 0-15-ig a SZÍN kódokat tartalmazza (konstansok)
- clrscr parancs (~~képe~~ képernyőtörlés)
- gotoxy(oszlop, sor) általában 80 oszlop, 25 sor a képernyő felbontás
  képernyő közepe (40;13)
  - az adott oszlop - sor koordinátára ugrik a kurzor
- wherex : hanyadik oszlopban van a kurzor
- wherey : hanyadik sorban van a kurzor
- window $(x_1, y_1, x_2, y_2)$ : külön ablak létrehozása a képernyőn
- window (1,1, 80,25) : visszaadja az eredeti ablak (alapállapot)
- clreol : sortörlés, a kurzortól kezdve a hátralevő részt törli
- delline : a teljes sort törli és megszünteti tehát az alatta levő sorok eggyel feljebb ugranak.
- insline : a kijelölt sort (ahol a kurzor van) eggyel lejjebb tolja (beszúr egy üres sort)

```
Pl.: repeat
     gotoxy(5,10)
     clreol;
     write('a(a<100, a>1)=');
     readln(a);
     until (a>1) and (a<100);
```