Pascal

looping

FOR

```
for var:= start to end do
begin
      code
end;

for var:= end downto start do
begin
      code
end;
```

WHILE/DO

```
while condition do
begin
      code
end;
```

REPEAT/UNTIL

```
repeat
      code
until condition;
```

conditional statements

if/then

```
if condition then
begin
      code
end;
```

if/then/else

```
if condition then
begin
      code
end
else if condition then
begin
      code
end
else
begin
      code
end
```

output

WRITELN/WRITE

```
writeln('blah',var,'blah');

blah var blah
_

write('blah',var,'blah');

blah var blah_
```

FIELDS

```
writeln(var:field:places);

writeln(2.34:7:3);

_ _ 2 . 3 4 0

*note*
zeros are placed for lack of
placeholder

a decimal counts as a field
```

variables

DECLARATION

```
var
      label :type

type
Integer     (-32768 to +32767)
Real
String      (up to 255 letters)
Char        (1 letter)
```

CONSTANTS

```
const
      label = anything
```

abstraction

PROCEDURE

```
procedure name;
var declarations;
begin
      code
end;
```

FUNCTIONS

```
function name : type;
var declarations;
begin
      code
      name:=return;
end;
```

INPUT

```
procedure name (name1 : type;
name2 : type; . . .);
```

abstract data types

ARRAYS

```
var
      arrayname : array[x..y] of
vartype;

*note*
pascal is NOT zero referenced
```

MULTIDIMENSIONAL ARRAYS

```
var
      arrayname : array[x..y,
x..y] of vartype;
      *2 dimensional*

      arrayname : array[x..y,
x..y, x..y] of vartype;
      *3 dimensional*

708-9320
```

pointers

DECLARATION AND USE

```
var
label : ^dataType;

@ (address)
^ (pointing to)

m:=7;
p:=@m;
p^:=8; {or m:=8}

writeln(p^); {or writeln(m);}
```

abstract data types

DECLARATION

```
*note* equivalent to structures

type
      label = record
                  label1 : type;
                  label2 : type;
                        .
                        .
                  labeln : type;
            end;

var
      x : label;
```

ACCESS

```
x.label1:=[illegible]type;
x.label2:=[illegible]type;
      .
      .
x.label1:=[illegible]type;
```

types

```
type
      label = {anything,
anything}

var
      x : label;
```

---

random things

COMMENTS

```
{ things to be comments }
```

MODULUS

```
x mod y;
```
