THE MOST BASIC CONCEPTS

syntax: the form of a program

semantics: the meaning of a program

```
/*  The World of Containers  */

class  Container
{           int  capacity;
            void  *contents[MAX];

public:
            addObject(void *anObject);
            removeObject(void *anObject);
};

class  Vehicle :  Public  Container
{           protected : int  position, velocity;
}
class  Train  :  Public  Vehicle
{           train();
            private:  int  maxObtainableSpeed;
                      int  maxNumberOfPassengers;
            public:
                      setSpeed(int  speed);
                      addPassenger(void  *passenger);
}
```

FORTRAN

```
C     FORM PARTIAL SUMS
C
      SUBROUTINE PARSUM(A, B, N)
      REAL A(N), B(N)
      SUM = 0
      DO  10  I  =  1 ,  N
      SUM  =  SUM  +  A(I)
      B(I)  =  SUM
10    CONTINUE
      RETURN
      END
```

- scientific computation, real and complex
- easy-to-use, but powerful I/O facilities
- compilers for efficient parallel code
- lots of existing programs and libraries

C.10

ALGOL 60

```
begin
   integer m, n;
   n  :=  10;
   begin
      array  a[1:n];
      procedure  f(r,s);
        array  r;  integer s;
         begin
             for  m = 1 step  s  until  n  do
                 s  :=  r[m]  :=  s/2;
         end;
       f(a,n)
   end
end
```

- block structuring
- variables local to blocks; deleted on exit
- recursion
- terrible I/O (at least in the IBM version)

C.11

SNOBOL4

```
      PAT = ('SUBROUTINE' | 'FUNCTION')  ARBNO(' ')
         SPAN('ABCDEFGHIJKLMNOPQRSTUVWXYZ
         0123456789')  .  NAME

IN   LINE = INPUT                                          :F(END)
      LINE  'C'                                            :S(IN)
      LINE  PAT                                            :S(NEW)
CONT
      LINE LEN(65) .  LINE2
      OUTPUT  =  LINE2   NAME   N
      N  =  N  +  10                                        :(IN)
NEW
      (NAME  '000000')   LEN(6)  .  NAME
       OUTPUT =
       OUTPUT = 'STARTING NEW ROUTINE'
       N  =  0                                              :(CONT)
END
```

- excellent string manipulation facilities
- automatic pattern matching, built-in functions
- user-defined structures
- recursion

C.12

PASCAL

```
TYPE  RECPOINTER  =  ^SPACEREC;
          SPACEREC  =  RECORD
                              DATA: INTEGER;
                              LINK:   RECPOINTER
                             END;
VAR  HEAD,  TAIL:  RECPOINTER;

PROCEDURE  ADD(P : RECPOINTER);
  IF  HEAD  =  NIL
  THEN  BEGIN
             HEAD := P:     TAIL := P
           END
  ELSE   BEGIN
             TAIL^.LINK  :=  P;   TAIL := P
           END
END
```

- simple syntax
- user-defined types / dynamic allocation
- recursion
- limited I/O
- no string handling, must use arrays

C.13

LISP

```
DEFUN  MYFUNC  ( N  M )
  (COND
  (( AND  ( NUMBERP  N)  (NUMBERP  M))  (+  N  M))
  (T   NIL)    ))
```

- programs made up of functions
- symbolic expressions
- list and tree handling
- untyped variables
- recursion

PROLOG

```
ON(REDBLOCK,  BLUEBLOCK).
ON(BLUEBLOCK, GREENBLOCK).
ON(GREENBLOCK, YELLOWBLOCK).

ON(X,Y)  :-  ON(X,TEMP)  ON(TEMP,Y).

?- ON(REDBLOCK, YELLOWBLOCK).
```

- symbolic expressions
- built-in logic proving mechanism
- recursion

JAVA

```
Import  java.awt.Graphics;
Import java.awt.Color;

public class Hello extends java.applet.Applet  {

    public  void  paint(Graphics g)  {
      g.setColor(Color.red);
      g.drawString("Hello World!", 5, 25);

      g.setColor(Color.blue);
      g.drawString("More next week!",5,50);

}
```

MORE NEXT WEEK!

MORE BASIC CONCEPTS

- variables and constants
  - datatypes
  - values and references
  - memory allocation and deallocation
- expressions and assignments
- control structures
  - blocks
  - branching statements
  - if-then-else statements
  - loops
  - procedure / function calls