

C: A Review

Minimum program structure

```
#include <stdio.h>
main()
{
    printf("Hello\n");
    exit(0);
}
```

Comments

```
/* These two lines are one comment. It
   continues until a matching star-slash
*/
```

Parameters & definitions

```
#define ParameterName value
```

Declarations

```
int    name_of_variable
float  name_of_variable
char   name_of_variable
int    IntArray_name[100]
int
IntArray2d_name[10][10]
float  RealArray_name[100]
float
RealArray2d_name[10][10]
char   name_of_string[10]
int    *name_of_pointer
float  *name_of_pointer
char   *name_of_pointer
```

Logical expressions

```
a == b    equal
a > b     greater (>= is > or =)
a < b     less    (<= is < or =)
a != b    not equal
```

Details ...

1. All variables must be declared.
2. Statements can span many lines. Except #define, all end with ";"
3. Like Fortran, you can set array dimensions with a parameter (#define) statement, and then use that parameter to declare an array:

```
#define NX 50
float array[NX];
```
4. You can define a variable & assign its initial value in one statement:

```
float dx = 0.1;
```

Assignments

```
a = 4.8*(3.0+4.0);      "a" =
33.6
i = 7;                  "i" now has value
7
name = "hello";         name is of type char *
```

Logic

```
if (logical expression) {
    ... do something... ;
} else if (expression) {
    ... do something else ... ;
} else {
    ... something different ... ;
}
```

Loops

```
for (var=start; var<=end; var++)
{
    ....do something.... ;
}
while (logical expression) {
    ... do something until expression is false ... ;
}
```

Stop!

```
exit(0);                halts entire program
break;                  exits from a for-loop
```

I/O

```
printf("Integers =
%d,%d\n",i,j);
printf("Float = %f\n",a);
printf("Formatted = %7.3f\n",b);
printf("Start of
s1=%f\n",s1[0]);
scanf("%d",&IntVariableToRead);
scanf("%f",&RealVariableToRead);
```

Subroutines

```
subr_name(a,&b,"Hello");
....
subr_name(a,bptr,string)
float a; float *bptr;
char *string;
{
    *bptr = 7.12;        can only set value of
ptr
}
```

Working with arrays

```
#define NX 75
int i,j; float s1[NX][NX];
```

5. Math shorthand:
 i=i+1 => i++; i=i+2 =>
 i+=2;
 i=i-1 => i--; i=i-2 => i-
 =2;

```
for (j=0; j<NX; j++) {      j = 0,1,...,NX-1
  for (i=0; i<NX; i+=2) {    i=0,2,...
    sl[i][j] =
      4.61*otherarray[j] + 7.0/3.0;
  }
}
```

Fortran: A Review

Minimum program structure

```
program BasicFortran
implicit none          always do
this!!
print*, 'Hello'
stop
end
```

Comments

“!” in first column (for free-format style;
 compile with “-free” if file suffix not
 .f90)
 “c” in first f77 column: line = comment

Parameters & definitions

```
parameter (name=value)
data VariableName /value/
```

Declarations

```
integer    name_of_variable
real       name_of_variable
character  name_of_variable
logical    name_of_variable
integer    array_name(100)
integer    array2d_name(10,10)
real       array_name(100)
real       array_name(0:101)
           102 values, from 0..101
real       array2d_name(10,10)
character*10 name_of_string
```

Logical expressions

```
a.eq.b      equal
a.gt.b      a > b (.ge. : > or =)
a.lt.b      a < b (.le. : < or =)
a.ne.b      not equal
```

Details ...

1. Always use *implicit none* in your code! Implicit typing is dangerous.
2. F77 statements usually begin with a tab or 6 spaces at the start of each line. Statements may not exceed 72 columns in length

Assignments

```
a = 4.8*(3.0+4.0)      “a” =
33.6
i = 7                  “i” now has value
7
name = 'hello'         name is char
variable
```

Logic

```
if (logical expression) then
  ... do something...
else if (expression) then
  ... do something else ...
else
  ... something different ...
endif
```

Loops

```
do variable = start,stop,step
  ....do something....
end do
do while (logical expression)
  ... do something ...
end do
```

Stop!

```
stop      halts entire program
exit      exits from a do-loop
```

I/O

```
print*, 'Variable =', VarName
read*, Variable1, Variable2
write(6,10) IntVar, RealVar
10 format('I=', i4, ', A=', f10.2)
```

Subroutines

```
call some_name(a, 'hello')
....
subroutine some_name(a, word)
implicit none
real a
character*5 word
print*, 'Subroutine: hello'
return
end
```

Working with arrays

```
real array1(nx, ny, nz)
```

unless you compile with *-I32*.

Exceptions: if free-format styling;
also comments (see above); and
I/O or *statement numbers* in
columns 1 through 5.

3. Free-format: use “&” at end of a
line to continue it on the *next* line.
f77 continuation: non-blank letter
in column 6 continues *previous*
line.

```
      real
array2(nx,ny,nz),a,b,d(10)
      array1 = 0.0           entire
array=0
      array2 = array1       full array
copy
      do k = 1,nz           k = 1,2,...,value of
nz
          do j = 1,ny,2      j = 1,3,5,...,ny
              array1(i,j,k) = 12.7
              array2(i,j,k) =
+              4.61*array1(i,j,k)
          end do
      end do
```