PH502: Scientific Programming Concepts

Irish Centre for High End Computing (ICHEC)

September 23, 2020

Overview

- In this lecture will we discuss nested loops and ifs.
- Nesting means that there is one loop inside another one, similarly for ifs.
- You will see that when we come to use variables called arrays nesting is useful.
- A nested loop (or if) must be completely contained inside another.

Nested Ifs

- If statements can be nested.

```c
int i,j;
if (i > 0) {
  if (i*j > 0) {
    printf("i,j +ve\n");
  }
}
```

```fortran
if (i .gt. 0) then
  if (i*j .gt. 0) then
    write(*,*) ' i,j  +ve'
  endif
endif
```

- Or:

```c
if (i>0 && j>0) {
 printf("i,j +ve\n");
}
```

```fortran
if (i.gt.0 .and. j.gt.0) then
  write(6,*) ' i,j  +ve'
endif
```

Nested Ifs

- In the previous example the nested statements could be changed to a single one.
- In the example below we cannot do this because dividing by zero may cause an error.
- If there are multiple statements in a single if statement, we have no control over which is evaluated first.

```c
int i,j;
if (i != 0) {
  if (j/i > 0) {
    printf("i,j same
    sign\n");
  }
}
```

```fortran
integer (kind=4) :: i,j
if (i .ne. 0) then
  if (j/i .gt. 0) then
    write(*,*) ' i,j same sign'
  endif
endif
```

Nested Do

- Loops can also be nested.
- The inner loop completes for each iteration of the outer loop.
- Below we print the loop indices *i* and *j* for each iteration. We can control the printing with careful placement of the print statements.

```c
for (i=0; i<3; i++) {
  printf("i=%d, j=", i);
  for (j=0; j<10; j++) {
    printf("%d,",j);
  }
  printf("\n");
}
```

```fortran
do i = 0,2,1
  write(6,'(a,i1,a)', &
&   advance='NO') 'i=',i,', j='
  do j = 0,9,1
    write(6,'(i1,a)', &
&     advance='NO') j,','
  end do
  write(6,*)
end do
```

```
i=0, j=0,1,2,3,4,5,6,7,8,9,
i=1, j=0,1,2,3,4,5,6,7,8,9,
i=2, j=0,1,2,3,4,5,6,7,8,9,
```

Loop Control

- Loop control statements apply to the current loop.
- Let's say we want to add only positive numbers in a sum.

```
sum = 0.0; flg = 0;
for (i=1; i<=100; i++) {
   sumy = 0.0;
   for (j=1; j<=100; j++) {
      x = funcx(); y = funcy();
      if (x .lt. 0) {
        flg = 1;
        break;
      }
      if (y < 0) {
         continue;
      } else {
         sumy = sumy + x*y;
      }
   }
   if (flg == 0) {
     sum = sum + sumy;
   } else {
     break;
   }
}
```

Loop Control

- From the above program you can see;
- when $x < 0$ the whole loop terminates and
- if $y < 0$ we skip to the next values of x, y.
- But there are two breaks because we need to break out of each loop separately.
- The code is becoming more complex, notice how the code is indented to indicate which lines are part of which loop.

Labelled Loops

FORTRAN only

- FORTRAN allows labelled loops. This has two advantages:
  1. make the code more interpretable,
  2. allows cycle and exit to work throughout a nest.
- Below is the equivalent code using labels

```
sumxy = 0.0
loop1 : do i = 1,100
          sumy = 0.0
loop2 :   do j = 1,100
            x = funcx(); y = funcy();
            if (x .lt. 0) exit loop1
            if (y .lt. 0) cycle loop2
            sumy = sumy + x*y
          end do
          sumxy = sumxy + sumy
        end do
```