Lecture 19  Recursion

Reading assignment: chapters 15, 23 recursion sections.

**Recursive functions**

A function is defined recursively if it is defined in terms of earlier values. \(2^n = 2 \cdot 2 \cdot 2 \cdot \ldots \cdot 2\)  \(n\) times can be defined recursively by

\[2^0 = 1\]

\[2^{n+1} = 2 \cdot 2^n\]

To rewrite this as a single recursive definition note that

\[2^{n+1} = 2 \cdot 2^n\]  for  \(n \geq 0\)  and

\[2^n = 2 \cdot 2^{n-1}\]  for  \(n \geq 1\) say the same thing: the next power is twice the previous power. The above two equations can now be written as one recursive definition:

\[2^n = \begin{cases} 
1 & \text{if } n = 0 \\
2 \cdot 2^{n-1} & \text{if } n > 0 
\end{cases}\]
Write a recursive function `power_2(n)` which calculates \(2^n\) without using the builtin \(2^{**n}\).

Problem: if we use `power_2` as a function in its definition, `power_2` can’t also hold the output value. Solution: use `result(p)`. Thus `p` will hold the output result instead of `power_2` and `power_2` can be used as a function.

```fortran
!c19_1_3power_2.f95
power_of_2(n)=2^n
recursive integer function power_2(n) ____
! p=2^n  In blank, make p the output variable
integer::n
if (n==0) then
  p = _____ !base case  p=2^0
else  !2^n = 2*2^(n-1)
  p = _____________  !don’t use ^ or **
endif
endfunction

program test_power_2
integer::i,power_2
do i=1,6;print *,i,power_2(i);enddo
endprogram
```
A telephone directory is an alphabetically-ordered list of names with a corresponding list of their phone numbers (phone numbers are words, not numbers!). Search problem: given a name, find the phone number. The slow (homework problem) way is to start at the beginning and search for the name one step at a time as in the find subroutine of Lecture 17. In a directory of a million names, finding a number this way can take a million steps. This isn’t how we search a directory.

The binary search algorithm below will take at most 30 steps to find a name in a million-name directory.

**CLASSWORK 19.2(4) binary_search.f95  On your own.** Each blank is one of min, mid-1, mid, mid+1 or max
recursive integer function &
bin_search(min,max,list,n,name)result(index)
binary_search finds the index for name between limits min, max
character(30)::list(n),name
integer::min,max,n
integer,parameter:::NOTFOUND=0
if(max<min) then;
    index = NOTFOUND; return; endif
mid=(min+max)/2
if(name==list(mid)) then
    index = ; return
endif
if(name < list(mid)) then
    index = bin_search( , ,list,n,name)
endif
if(name > list(mid)) then
    index = bin_search( , ,list,n,name)
endif
endfunction

program phone_number
character(30):::list(9), name='john'
character(30):::number(0:9)
integer:: index, bin_search
list(1)='ann'; number(1)='8907'
list(2)='bob'; number(2)='3526'
list(3)='jack'; number(3)='9360'
list(4)='john'; number(4)='6639'
list(5)='mandy'; number(5)='1104'
list(6)='naomi'; number(6)='5928'
list(7)='nick'; number(7)='3995'
list(8)='renee'; number(8)='1253'
list(9)='stacie'; number(9)='0988'
number(0)='not found'

!~ print *, 'Enter a name.'
!~ read *, name  !uncomment these two once program works
index=bin_search(1, 9, list, 9, name)
print*, trim(name), "'s #: ", trim(number(index))
endprogram

Classwork 19.3(5) det_row.f95 Write an external subroutine minor(a, n, j, b) which given an \( n \times n \) matrix \( a \) and a \( j \) between 1 and \( n \), sets \( b \) equal to the minor of \( a \) at position \((1, j)\).
In the same file write an external function \texttt{det}(\textit{a}, \textit{n}) which calculates the determinant of an \(n \times n\) matrix \textit{a} by using minors on the first row.

First get the \texttt{minor} subroutine working. Highlight then press <\texttt{ctrl-q}> to comment out the lines of the determinant function \texttt{det} and the next-to-the-last line which uses \texttt{det}. If

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 0 & 1 & 2 \\
3 & 4 & 5 & 6 \\
\end{array}
\]

minor of \textit{a} at position \((1, 3)\) is \textit{b} =

\[
\begin{array}{cccc}
5 & 6 & 8 \\
9 & 0 & 2 \\
3 & 4 & 6 \\
\end{array}
\]

Press <\texttt{ctrl-q}> again to activate the determinant function lines and the next-to-the-last line.

The recursive formula for the determinant of matrix \(\textit{a} (n, n)\) is 

\[
\texttt{det}(\textit{a}) = \sum_{j=1}^{n} (-1)^{j+1} a(1, j) \text{det}(M_{1,j}) \text{ where } M_{1,j} \text{ is the minor at position } (1,k).
\]

The code for \(k\)th item of the sum is 

\[
(-1)^{j+1} a(1, j) \text{det}(\textit{b}) \text{ where } \textit{b}=M_{1,j}.
\]
subroutine minor(a,n,j,b)
integer::j,n
real::a(n,n),b(n-1,n-1)
!delete this line, complete the minor function
endsubroutine

recursive real function det(a,n) result(d)
real::a(n,n),b(n-1,n-1)
if(n==1)then; d=a(1,1); return; endif
d = ___ ; ! d is a sum
do _________
call minor(a,n,j,b) ! b = minor of a at (1,j)
d = ____________________________
enddo
endfunction

program test_det
real::a(3,3),b(2,2)=0,det
a(1,:)=(/0.,2.,3./)
a(2,:)=(/4.,5.,6./)
a(3,:)=(/7.,8.,9./)
call minor(a,3,2,b)
```fortran
3 format(200f7.2)
print*, 'a='; do i=1,3; print 3,a(i,:); enddo
print*, 'b='; do i=1,2; print 3,b(i,:); enddo
!~print*, 'determinant=', det(a,3)
endprogram !Answer should be det(a)=3

If you get a “Missing actual argument...” error, it is because you wrote det(b). Should be two arguments det(b, n?), the matrix and its dimension. If the dimension of a is n, what is the dimension of b?

CLASSWORK
19.1(3) power_2 19.2(4) binary_search 19.3(5) det_row
email: dale@math.hawaii.edu subject line: 190 c19(12)

Quiz on factorial and fibonacci (a or b, your choice) functions, not their programs, not on HOMEWORK19.3(6) det_col.

\[ n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1 \]
The two-equation recursive definition of \( n! \) is
\[ 0! = 1 \]
\[ (n+1)! = (n+1) \cdot n! \]
The one-equation recursive definition of \( n! \) is
\[ n! = \begin{cases} 
1 & n = 0 \\
n \cdot (n - 1)! & n > 0 
\end{cases} \]

**Homework 19.1(3) factorial.f95**

*email: dale@math.hawaii.edu subject line: 190 h19.1(3)*

Write a recursive function `factorial(n)` which calculates the \( n! \). Include test program. See **Homework 11.1(2)** `myfactorial` for nonrecursive version.

```fortran
!factorial.f95     subject line: 190 h19.1(3)
!delete this line, write the function

program test_factorial
integer::factorial,i
3 format(200i5)

print 3,(factorial(i),i=0,6)
endprogram  !ans: 1 1 2 6 24 120 720
```
The *Fibonacci* numbers are
\[ F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, F_5 = 5, \ldots \]
For \( i > 1 \),
the \( i \)th Fibonacci number is the sum of the previous 2.

Do **19.2a(5)** or **19.2b(7)** --latter requires creativity. Don’t do both.

**Homework 19.2a(5)** fibonacci_a.f95
email: dale@math.hawaii.edu subject line: 190 h19.2a(5)
Write a recursive function `fibonacci(n)` which calculates the \( n \)th Fibonacci number. Include the following test program

```
!fibonacci_a.f95    subject line: 190 h19.2a(5)

!delete this line, write the function
program test_fib
integer::fibonacci,i
3 format(200i5)
print 3,(fibonacci(i),i=0,10)
endprogram !ans: 0 1 1 2 3 5 8 13 21 34 55
```
Do 19.2A(5) or 19.2B(8) --latter requires creativity. Don’t do both. 

**Homework 19.2B(8)** fibonacci_b.f95 
email: dale@math.hawaii.edu subject line: 190 h19.2b(8)

Write a **nonrecursive** function `fibonacci(n)` which calculates the *n*th Fibonacci number. Don’t do both A, B 

Hint, you need three variables, *f*1, *f*2, *f*3 to hold the current and two previous values of the Fibonacci sequence. At each stage of the do-loop, you must update all three values.

```plaintext
!fibonacci_b.f95 subject line: 190 h19.2b(8)

!delete this line, write the function

program test_fib
integer::fibonacci,i
3 format(200i5)
print 3,(fibonacci(i),i=0,10)
endprogram !ans: 0 1 1 2 3 5 8 13 21 34 55
```
Write an external subroutine \texttt{minor(a,n,i,b)} which given an \( n \times n \) matrix \( a \) and a \( i \) between 1 and \( n \), sets \( b \) equal to the minor of \( a \) at position \( (k,1) \). These minors are along the first column, not the classwork minors on the first row. In the same file write an external function \texttt{det(a,n)} which calculates the determinant of an \( n \times n \) matrix \( a \) by using minors on the first \textbf{column} (Not first row as in classwork).

Not quizzed on this homework.

If \( a = 
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 0 & 1 & 2 \\
3 & 4 & 5 & 6 \\
\end{array} \\
\) minor of \( a \) at position \( (3,1) \) is \( b = 
\begin{array}{cccc}
2 & 3 & 4 \\
6 & 7 & 8 \\
4 & 5 & 6 \\
\end{array} \\
\) The recursive formula for the determinant of matrix \( a(n,n) \) expanded along the first column (no credit for expanding on first row) is
\[ \text{det}(a) = \sum_{i=1}^{n} (-1)^{i+1} a(i,1) \text{det}(M_{i,1}) \]

!det_col.f95 subject line: 190 h19.3(6)

subroutine minor(a,n,i,b)
integer::j,n
real::a(n,n),b(n-1,n-1)
!delete this line, write the minor subroutine, test it carefully
endsubroutine

recursive real function det(a,n) result(d)
real::a(n,n),b(n-1,n-1)
!delete this line, write the first-column-expanding determinant function
endfunction

program test_det
real::a(3,3),b(2,2)=0,det
a(1,:)=(/0.,2.,3./)
a(2,:)=(/4.,5.,6./)
a(3,:)=(/7.,8.,9./)
call minor(a,3,2,b)
3 format(200f7.2)
print*,'a='; do i=1,3; print 3,a(i,:); enddo
print*,'b='; do i=1,2; print 3,b(i,:); enddo
print*,'determinant=',det(a,3)
endprogram !Answer should be det(a)=3