**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

\[
2^{n+1} = 2 \cdot 2^n \quad \text{for } n \geq 0 \text{ and } 2^n = 2 \cdot 2^{n-1} \quad \text{for } n \geq 1
\]
say the same thing: the next power is twice the previous power. The above two equations can now be written as one recursive definition:

\[
2^n = \begin{cases} 
1 & \text{if } n = 0 \\
2 \cdot 2^{n-1} & \text{if } n > 0 
\end{cases}
\]

**Classwork 19.1(3) power_2.f95** Write a recursive function \(\text{power}_2(n)\) which calculates \(2^n\) **without** using the builtin \(2**n\).

Problem: if we use \(\text{power}_2\) as a function in its definition, \(\text{power}_2\) can’t also hold the output value.

Solution: use \(\text{result}(p)\). Thus \(p\) will hold the output result instead of \(\text{power}_2\) and \(\text{power}_2\) can be used as a function.

**Searching**

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 \textit{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

```fortran
!c19_2_4binary_search.f95
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
```

```fortran
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) 
!uncomment these two once program works
print*,trim(number(index))
endprogram
```

**CLASSWORK 19.3(5) det_row.f95** Write an external
subroutine minor(a,n,j,b) which given an n×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 det(a,n) which calculates the determinant of an n×n matrix a by using
minors on the first row.

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

```
a =
1 2 3 4
5 6 7 8
9 0 1 2
3 4 5 6
```
Press <ctrl-q> again to activate the determinant function lines and the next-to-the-last line.

The recursive formula for the determinant of matrix

\[ a(n,n) \text{ is } \det(a) = \sum_{j=1}^{n} (-1)^{j+1} a(1,j) \det(M_{1,j}) \]

where \( M_{1,j} \) is the minor at position \((1,k)\). The code for kth item of the sum is

\[ (-1)^{j+1} a(1,j) \det(b) \]

where \( b = M_{1,j} \).

**Polynomials**

As with Matlab, polynomials are written in the order of increasing powers. Instead of

\[ x^2 - 2x + 3 \]

write \( 3 - 2x + x^2 \).

Represent \( (3)x^0 + (-2)x^1 + (1)x^2 \) with the vector \( p = [3, -2, 1] \) of its coefficients. Declare it as \( real::p(0:2) \), since the degrees range from 0 to 2. Thus \( p(0)=3, p(1)=-2, p(2)=1 \). Write 0 for any missing coefficient: write \( x^2 - 1 \) as

\[ -1 + 0x + 1x^2 \]

with vector \( p = [-1, 0, 1] \) of degree \( n=2 \).

Evaluating a polynomial \( p \) means the result of substituting in a value for \( x \). \( x^2 - 2x + 3 \) evaluated at \( x = 10 \) is

\[ 10^2 - 2(10) + 3 = 100 - 20 + 3 = 83 \]
**Numbers in different bases**

The digits of a base-\(b\) number will be written as a sequence \text{seq} of base \(b\) digits declared with \texttt{integer::seq(200)}.

If the base-5 sequence is \([1,3,4,2]\), the decimal value is \(d=1 \cdot 5^3 + 3 \cdot 5^2 + 4 \cdot 5^1 + 2 \cdot 5^0 = 222\).

For \(d=2345\), the base-10 sequence is \([2,3,4,5]\). Note that the remainder \(\text{mod}(3,2)=2\) and the quotient \(\text{quo}(3,2)=0\).

For \(d=2345\), the base-10 sequence of digits is \text{seq} = [2,3,4,5]. \(\text{mod}(2345,10)=5\) and \(\text{quo}(2345,10)=234\). The last digit 5 is \(\text{mod}(d,10)\), the previous digits \([2,3,4]\) come from \(\text{quo}(d,10)\). For any base \(b\), the last digit is \(r=\text{mod}(d,b)\), the previous digits come from \(q=\text{quo}(d,b)=d/b\). If \(\text{quo}(d,b)=0\), the sequence is just \([r]\).

**Classwork**

19.1(3) \texttt{power_2} 19.2(4) \texttt{binary_search} 19.3(5) \texttt{det_row}

email: dale@math.hawaii.edu subject line: 190 c19.1(3)

Quiz on factorial and fibonacci (a or b, your choice) functions, not their programs, not on HOMEWORK 19.3(6) \texttt{det_col}.

\[ n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 3 \cdot 2 \cdot 1 \]

The two-equation recursive definition of \(n!\) is

\[ 0! = 1 \]

\[ (n+1)! = (n+1) \cdot n! \]

The one-equation recursive definition of \(n!\) is

\[
\begin{array}{c}
  n! = \begin{cases} 
    1 & n = 0 \\
    n \cdot (n-1)! & n > 0 
  \end{cases}
\end{array}
\]

**Homework 19.1(3) \texttt{factorial.f95}**

email: dale@math.hawaii.edu subject line: 190 h19.1(3)
Write a recursive function \texttt{factorial(n)} which calculates the \(n!\). Cf. nonrecursive HOMEWORK 11.1(2) \texttt{myfactorial}.

19.2A(5) or 19.2B(7) --latter requires creativity. Don’t do both.

**Homework 19.2A(5) \texttt{fibonacci_a.f95}**

Write a recursive function \texttt{fibonacci(n)} which calculates the \(n\)th Fibonacci number.

\[ F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, F_5 = 5, \ldots \]

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) \texttt{fibonacci_a.f95}**

Write a recursive function \texttt{fibonacci(n)} which calculates the \(n\)th Fibonacci number.

email: dale@math.hawaii.edu subject line: 190 h19.2a(5)

Quiz on factorial and fibonacci (a or b, your choice) functions, not their programs, not on HOMEWORK 19.3(6) \texttt{det_col}.

\[ n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 3 \cdot 2 \cdot 1 \]

The two-equation recursive definition of \(n!\) is \(0! = 1\) and \((n+1)! = (n+1) \cdot n!\)

The one-equation recursive definition of \(n!\) is

\[
\begin{array}{c}
  n! = \begin{cases} 
    1 & n = 0 \\
    n \cdot (n-1)! & n > 0 
  \end{cases}
\end{array}
\]
Do 19.2A(5) or 19.2B(8) --latter requires creativity. Don’t do both.

**HOMEWORK 19.2B(8)** fibonacci_b.f95

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.

```fortran
!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

**HOMEWORK 19.3(6)** det_col.f95

email: dale@math.hawaii.edu subject line: 190 h19.3(6)

Write an external subroutine minor(a,n,i,b) which given an $n \times n$ matrix $a$ and an $i$ 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 det(a,n) which calculates the determinant of an $n \times n$ matrix $a$ by using minors on the first column (Not first row as in classwork).

Not quizzed on this homework.

If $a =
\begin{bmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 0 & 1 & 2 \\
3 & 4 & 5 & 6
\end{bmatrix}$

minor of $a$ at position $(3,1)$ is $b =
\begin{bmatrix}
2 & 3 & 4 \\
6 & 7 & 8 \\
4 & 5 & 6
\end{bmatrix}$

The recursive formula for the determinant of matrix $a(n,n)$ expanded along the first column (no credit for expanding on first row) is

$$det(a) = \sum_{i=1}^{n} (-1)^{i+1} a(i,1) det(M_{i,1})$$
!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