Computer Science 1510

Lecture 34

April 4, 2016

Lecture Outline

- Fortran example: sorted linked list
- Fortran command-line arguments

Final exam:
Wednesday, April 13
12 – 2 pm
EN-1001

Example: Sorted Linked List

- The following example builds a sorted linked list by inserting each value in the appropriate location in the list when it is read (insertion sort).

- The list is sorted in ascending order when the user stops entering values.

Example: Sorted Linked List

```
MODULE LL
   TYPE Node
      INTEGER::value
      TYPE(Node),POINTER::next
   END TYPE

CONTAINS

SUBROUTINE Find_insert_point(top,ans,i)
   IMPLICIT NONE
   TYPE(Node),POINTER::top,cur,ans
   INTEGER,INTENT(IN)::i

   IF (.NOT.ASSOCIATED(top)) THEN  ! Insert at beginning of list
      NULLIFY(ans)
      RETURN
   ELSE IF (top%value>i) THEN ! Insert at beginning of list
      NULLIFY(ans)
      RETURN
   END IF
   cur=>top ! Otherwise start at top of list
   DO WHILE (ASSOCIATED(cur))
      IF (.NOT.ASSOCIATED(cur%next)) THEN ! At end of list
         ans=>cur
         RETURN
      END IF
      IF (cur%next%value>i) THEN
         ans=>cur
         RETURN
      END IF
      cur=>cur%next
   END DO
   RETURN
END SUBROUTINE Find_insert_point
END MODULE LL
```

```
PROGRAM SortedLL
   USE LL
   IMPLICIT NONE
   INTEGER::i,InputStatus,AllocateStatus
   TYPE(Node),POINTER::top,newnode,place,cur
   NULLIFY(top) ! Initialize top to point to no target

   WRITE(*,*) 'Please enter a value (CTRL^D to stop)'
   READ(*,*,IOSTAT=InputStatus) i
   DO WHILE(InputStatus==0) ! Read from a file until END OF FILE
      IF (.NOT.ASSOCIATED(top)) THEN
         ALLOCATE(top,STAT=AllocateStatus)  ! Allocate first node
         IF (AllocateStatus /= 0) STOP
         NULLIFY(top%next) ! next doesn't yet point to anything
         top%value=i ! set the first element's value to i
      ELSE
      ! Find where in the list we should insert a new node
         CALL Find_insert_point(top,place,i)
         ! Returns a pointer to the node prior to where i should be placed
         IF (.NOT.ASSOCIATED(place)) THEN
            ! There is no existing node with value less than i, so
            ! we need to insert at the front of the list
            ALLOCATE(newnode,STAT=AllocateStatus)  ! Allocate a new node
            IF (AllocateStatus /= 0) STOP
            newnode%value=i
            newnode%next=>top
            top=>newnode
         ELSE ! Insert somewhere else in the list
            ALLOCATE(newnode,STAT=AllocateStatus)  ! Allocate a new node
            IF (AllocateStatus /= 0) STOP
            newnode%value=i
            newnode%next=>place%next
            place%next=>newnode
         END IF
      END IF
      WRITE(*,*) 'Please enter a value (CTRL^D to stop)'
      READ(*,*,IOSTAT=InputStatus) i
   END DO
```

```
   IF (.NOT.ASSOCIATED(top)) THEN
      WRITE(*,*) 'No data read'
      STOP
   END IF

   WRITE(*,*) 'The sorted values are:'
   cur=>top ! Start at the first node in the list
   DO WHILE (ASSOCIATED(cur))
      WRITE (*,*) cur%value
      cur=>cur%next ! Move to the next node in the list
   END DO
END PROGRAM SortedLL
```

Fortran: command-line arguments

- In C, we saw how to obtain arguments from the command line using `argc` and `argv`.

- Fortran has a function and subroutines that allow us to obtain command-line arguments.

- The `COMMAND_ARGUMENT_COUNT()` function returns the number of command-line arguments (NOT including the executable name).

- The subroutine

  ```
  GET_COMMAND(string_var)
  ```

  will place the entire command-line in `string_var`.

- The subroutine

  ```
  GET_COMMAND_ARGUMENT(number,string_var)
  ```

  will place the `number`-th command-line argument in `string_var`.

Example: Fortran command-line args

```
PROGRAM Test_args
   INTEGER :: i=1, nargs, carg_status, index1
   CHARACTER(Len=256) :: value, buff
   CHARACTER :: delim

   nargs = COMMAND_ARGUMENT_COUNT()
   WRITE(*,*) 'There were ', nargs, ' command-line arguments.'
   CALL GET_COMMAND_ARGUMENT(i,value)
   WRITE(*,*) 'Argument ', i, ' = ',TRIM(value)
   ! TRIM removes trailing blank characters of a string
   CALL GET_COMMAND(buff)
   WRITE(*,*) 'Entire command line: ', TRIM(buff)
   delim = ' '
   index1 = SCAN(buff,delim)
   ! SCAN returns location of first instance of delimiter
   WRITE(*,*) 'index1 = ', index1
   WRITE(*,*) buff(1:index1-1)
   WRITE(*,*) TRIM(buff(index1+1:))
END PROGRAM Test_args
```

Output:

```
$ ./a.out hello
 There were            1  command-line arguments.
 Argument            1  = hello
 Entire command line: ./a.out hello
 index1 =            8
 ./a.out
 hello
```