190 Lecture 15  strings

Reading assignment: chapters 12,13,14,22,23.

**Character declarations, trimming, reading, writing**

Assigning, reading, printing or writing a string to a smaller field truncates the string, the tail is lost.

```character(2): s='abcd'  gives  s='ab'
print '(a2)', 'abcd'  prints  "ab"
```

Assigning or reading to a longer field adds blank spaces on the right. Short strings are *left-justified* in long fields.

```character(6): s='abcd'  gives  s='abcd  '
print '(a6)', 'abcd' prints  "  abcd"
```

Printing or writing to a longer field adds blank spaces on the left. Short strings are *right-justified* in long fields.

```character(6): s='abcd'  gives  s='abcd  '
print '(a6)', 'abcd' prints  "  abcd"
```

**Character string operations**

```character(5): s='abc', t='xyz'
character(10): r
s/t is the concatenation of s and t = s followed by t.
s//t = 'abc  xyz'
trim(s)///trim(t)='abcxyz'
trim(s)///'//trim(t)='abc, xyz'
print'(a,"",a)', trim(s),trim(t)
write(*,'(a,"",a)'),trim(s),trim(t)
gives r='abc, xyz'
```

```print '(a6)', 'abcd'  gives  s='abcd  '
print '(a6)', 'abcd'
```

```print '(a6)', 'abcd'  gives  s='abcd  '
print '(a6)', 'abcd'
```

```trim(s) trims off any blank spaces at the right end.
trim(" abcd  ") = " abcd"
adjustl(s) left-justifies the string.
adjustl(" abcd") = " abcd  
trim(adjustl(" abcd "))="abcd"
```

```read '(a)', t  reads one line from keyboard to t
read *,r  reads one word from keyboard to r
```

```read(s,*), r,t  !reads words r,t from string s
write(s,*), r,t  !writes words r,t to string s
```

```If line is 'Good day'
read(line,*), s,t  gives s = 'Good' and t = 'day'
read(line,'(a)'), r  gives r = 'Good day'
```

**Classwork 15.1(2)** namel.f95

```!c15_1_2name1.f95
program namel
character(20): first='John', last='Doe'
print*,'Hello,first,last,'!
endprogram
```

What went wrong? Correct the two *pinks*. Add two spaces.

Want  **Hello John Doe!**

not  HelloJohn Doe !
not  HelloJohnDoe!
COMMON ERROR If you don’t see the last !, it might be off screen due to untrimmed blank spaces.

CLASSWORK 15.2(2) name2.f95
!c15_2_2name2.f95
program name2
character(20) :: fullname
print *, 'Enter first name, space, last name'
read *, fullname
print *, 'Aloha ', trim(fullname), '!'
endprogram

Correct the pink line to read in all of the fullname.
Want Aloha John Doe!
not Aloha John

CLASSWORK 15.3(2) name3.f95
Write a program which reads a first name first, a last name last, then writes the full name = last name, comma, space, first name. to a strings full, full2.
E.g. full = Smith, Tom
!c15_3_2name3.f95
program name3
character(20) :: first, last, full, full2, full3
full = 'Doe, John'
i = index(________) ! the index of ', in full.
first = _______ ! full(i) ? full(i) ?
last = _______ ! full(i) ? full(i) ?
print *, first, last !fix three things:
!Want: John Doe no comma
endprogram

CLASSWORK 15.4(2) index.f95
Write a program which, reads a full name full such as full = “Smith, John” consisting of a last name, comma, space, first name, then prints the first name, a space, then the last name (with no commas).
Hint: use the substring functions, e.g., s(3), full(3), and the index function i=index(full,'').

Change the 4 pink lines. Start with reads commented out.
!c15_4_2name4.f95
program name4
character(20) :: first, last, full
full = 'Doe, John'
i = index(_______) ! the index of ', in full.
first = _______ ! full(i) ? full(i) ?
last = _______ ! full(i) ? full(i) ?
print *, first, last !fix three things:
!Want: John Doe no comma
endprogram
**COMMON ERROR** Don’t set `first=index(full, ',').first` is a character string, `i=index(...)` is an integer. Set `first =` to some substring of `full`, maybe `full(i:)`? -- actually not.

**ASCII numbers**

Letters have numbers, ASCII numbers.

Letters: “A”, “B”, “C”, ... , “a”, “b”, “c”, ...

ASCII: 32, 33, 34, ..., 64, 65, 66, ...

If `let` is a letter, `iachar(let)` is its ASCII number.

If `n` is a number, `achar(n)` is its ASCII character.

Hence `iachar("A")=32` and `achar(32)="A"

**CLASSWORK 15.5(2)** lower_case.f95

Write a subroutine `lower_case(u)` which converts a phrase `Hello There` to all lowercase letters `hello there`. Fill the pink blank.

```fortran
!c15_5_2lower_case.f95
subroutine lower_case(word)
character (*) :: word

do i = 1, len(word)
   select case(word(i:i))
   case("A":"Z")
      word(i:i)=________________
   endselect
endo
endsubroutine

!reverse_letters2a.f95
subroutine reverse_letters(word,n,backword)
character(n) :: word, backword

backword="" word=n
do i=1,n
   backword=trim(backword)//word(i:i)
endo
endsubroutine
```

Do one of **Homework 15.1a** or **Homework 15.1b** not both.

**CLASSWORK 15.1(2)** name1 15.2(2) name2 15.3(2) name3 15.4(2) index 15.5(2) lower_case

- email: dale@math.hawaii.edu  subject line: 190 c15(12)

- In the next homework problem you write a subroutine `reverse_letters(word,n,backword)` word of `n` letters, makes a word `backword` with the letters in the reverse order. If `word` is "abcde" then `backword` is "edcba". The two example subroutines below work only for a word of length `n=3`, your subroutine must work with words of any length `n`. The two alternatives have equal points.

Recall `word(i:i)` is the `i`th letter.

```fortran
!reverse_letters2a.f95
subroutine reverse_letters(word,n,backword)
character(n) :: word, backword

write(backword,'(200a1)') word(3:3), word(2:2), word(1:1)
endsubroutine
```

**CLASSWORK 15.1(2)** reverse_lettersa.f95

email: dale@math.hawaii.edu  subject line: 190 h15.1a(5)
To generate a sum, you start with 0 and add items one at a time, \( S = S + ... \). To generate a word, start with "" and concatenate one letter at a time word = trim(word) // ... .

In the next homework problem you write a subroutine `line_writer(words, numwords, line)` which writes a vector of words, e.g., `words = ["one", "two", "three"]` of length `numwords` into a single line `line = "one two three"` with the words separated with spaces. The subroutines below work only for vectors of length 2, your subroutine must work with vectors of any length. The two alternatives have equal points. All strings have at most 200 characters.

Do one of HOMEWORK 15.1A or HOMEWORK 15.1B not both.

email: dale@math.hawaii.edu subject line: 190 h15.1a(5)

Do one of HOMEWORK 15.1B or HOMEWORK 15.1A not both.

email: dale@math.hawaii.edu subject line: 190 h15.1b(5)
Do one of Homework 15.2a or Homework 15.2b not both.

Homework 15.2a(6) line_writera.f95
dale@math.hawaii.edu  subject line: 190 h15.2a(6)

To generate a sum, you start with 0 and add items one at a time, \( S = S + \ldots \). To generate a word, you start with "" and concatenate items one at a time word=trim(word)//\ldots.

Do one of Homework 15.2a or Homework 15.2b not both.

Homework 15.2b(6) line_writerb.f95
dale@math.hawaii.edu  subject line: 190 h15.2b(6)

Replace the deleted line with one write statement which writes the vector words into the line line. You’ll need a format of '(200(a,""))'. Use a implicit do statement like print*(trim(words(i))," ",i=1,numwords)