

Announcements

Office Hours this week (Telephone interviews continue)

Monday: 2:30 – 4:00 pm

Tuesday: 3:15 – 4:00 pm

Wednesday: 2:30 – 4:00 pm

Thursday: 2:00 – 2:45 pm

Friday: 2:30 – 4:00 pm

Next week: No Office Hours Monday or Tuesday

Thursday Extra this week uncertain

Forthcoming topics for quick discussion this week

char [??] versus char * (Today)

while (1) versus other looping (Today)

white box versus black box testing (Wednesday)

Insertion Sort

Questions

Clicker Questions

Lab

New Lab Partners

char str [??] versus char * str

Consider the following code segment:

```
char str1 [10] = "computer";  
char * str2 = str1;  
printf ("str2:  %s\n", str2);  
char str3 [10] = "science";  
str2 = str3;  
printf ("str2:  %s\n", str2);
```

What happens when one attempts to compile and run this code?

- | | |
|--|-------------------------|
| A. Compile error | D. computer
computer |
| B. Likely run-time error
(e.g., segmentation fault) | E. science
science |
| C. computer
science | |

char str [??] versus char * str

Consider the following code segment:

```
char str1 [10] = "computer";  
char str2 [10] = str1;  
printf ("str2:  %s\n", str2);  
char str3 [10] = "science";  
str2 = str3;  
printf ("str2:  %s\n", str2);
```

What happens when one attempts to compile and run this code?

- | | |
|--|-------------------------|
| A. Compile error | D. computer
computer |
| B. Likely run-time error
(e.g., segmentation fault) | E. science
science |
| C. computer
science | |

char str [??] versus char * str

Consider the following code segment:

```
char str1 [10];  
char * str2;  
scanf ("%9s", str1);  
scanf ("%9s", str2);  
printf ("%s %s", str1, str2);
```

What happens when one attempts to compile and run this code, entering the text

computer science

on a single line:?

- | | |
|--|----------------------|
| A. Compile error | C. computer science |
| B. Likely run-time error
(e.g., segmentation fault) | D. computer computer |
| | E. science science |

Suppose `str` is defined as follows:

```
char str [30];
```

Now consider the following two statements to read a string:

```
1. scanf ("%s", str);
```

```
2. scanf ("%29s", str);
```

Vote 1: Which, if any, of these statements is safer?

A. Both are
unsafe

B. Both are
safe

C. 1 is safe;
2 is not

D. 2 is safe;
1 is not

Vote 2: Is

```
scanf ("%30s", str);
```

safe?

A. Yes

B. No

Looping conditions

Suppose a user is to read numbers and add them until the user enters a negative number. Two solutions are proposed:

Approach 1

```
int value = 0;
int sum = 0;
while (1)
{ printf ("enter value: ");
  scanf ("%d", &value);
  if (value < 0)
    break;
  sum += value;
}
```

Approach 2

```
int value;
int sum = 0;
printf ("enter value");
scanf ("%d", &value);
while (value >= 0)
{ sum += value;
  printf ("enter value");
  scanf ("%d", &value);
}
```

Which solution is more readable? (we'll talk about why shortly)

- A. both are equally good
- B. 1 is better

- C. 2 is better
- D. both are bad

In reordering elements in an array so they will be in ascending order, what is the underlying idea of an insertion sort?

1. Find the largest array element and place it last;
Find the second largest array element and place it next to last;
Find the third largest array element and place it third to last;
Etc.
2. Successively insert elements to an already ordered array segment, so that the array segment remains ordered after the insertion.
3. Repeated move through the array, checking adjacent elements and swapping them if the latter element is smaller than the first.
4. Create a new array of the same size as the original;
For each element in the original array, count how many elements are smaller (and suppose the count is i). Then, place the element in the i th position in the new array.

```
void insertionSort (int data[], int length) {  
    for (int boundary = 1; boundary < length; boundary++) {  
        int item = data[boundary];  
        int position = boundary-1;  
        while ((position >= 0) && data[position] > item) {  
            data[position+1] = data[position];  
            position--;  
        } // while  
        data[position+1] = item;  
    } // for  
} // insertionSort
```

How many times is the expression `data[position] > item` run when called with the array {10,11,12,13} ?

- A.1
- B.3
- C.4
- D.6
- E.None of the above.


```
void insertionSort (int data[], int length) {  
    for (int boundary = 1; boundary < length; boundary++) {  
        int item = data[boundary];  
        int position = boundary-1;  
        while ((position >= 0) && data[position] > item) {  
            data[position+1] = data[position];  
            position--;  
        } // while  
        data[position+1] = item;  
    } // for  
} // insertionSort
```

How many times is the expression `data[position] > item` run called with the array {13,12,11,10} ?

- A.1
- B.3
- C.4
- D.6
- E.None of the above.