CS16, 10S, **Handout to go with H17** (even more on structs) (printable PDF)

Available online at: http://www.cs.ucsb.edu/~pconrad/cs16/10S/homework/H17/handout
The assignment is available at http://www.cs.ucsb.edu/~pconrad/cs16/10S/homework/H17

This handout—which is your reading assignment for H17—covers more about structs, which are not covered in the textbook until Chapter 7. We are covering them a bit earlier than the textbook coverage, because they will open up some more interesting problem solving opportunities. This builds on the handout from H13 and the handout from H14

If you want to read some additional material about structs, you may read Section 7.1 and 7.2 in the Etter text, or Section 12.1 of the online Oualline text (on-campus link, off-campus link)

---

structs and the idea of abstraction

One of the useful aspects of structs is that they allow us to work with more data in a single operation. This is much more convenient than having to manipulate each individual number separately. For instance, with the following definition...

```
struct Point
{
  double x;
  double y;
};
```

... we can think in terms of a "point" rather than having to work with the x and the y as separate items.

And with the definition...

```
struct Time
{
  int hours;
  int minutes;
};
```

... we can think in terms of working with a time of day as a single item, rather than working with the hours and minutes separately.

Thus structs are a basic tool of *abstraction*, which is one of the most important ideas in Computer Science.

Our study of structs is also laying a foundation here for studying the ideas of Object-Oriented Programming in later courses, including CS24,32,48, and 56. Even if you don't go on to take further CS courses formally at UCSB, if you have any involvement in programming, you are bound to encounter object-oriented programming. So this is important foundational material.

In this homework assignment, we build on H13 and H14 by covering: (a) using assignment statements with entire structs, or members of structs (b) using structs with pointers, (c) arrays of structs, (d) arrays inside structs and (e) structs inside other structs

Please turn over for more...

...continued from other side

(a) Using assignment statements with structs

Suppose we declare three struct Time variables and initialize them as follows:

```
  struct Time t1 = {8,15};
  struct Time t2 = {12,30};
  struct Time temp;
```

We can then use assignment statements with these structs just like we would with any other variable. Here, for example are three assignment statements that cause `t1` and `t2` to swap their values:

```
  temp = t1;
  t1 = t2;
  t2 = temp;
```

The file `assignment.c` in the `code` directory associated with this assignment contains a full C program showing this code in context—you are encouraged to go online where you can follow the links and look at the entire program. You can also copy this code into your own `~/cs16/H17` directory with this command:

```
mkdir ~/cs16/H17
cp -r /cs/faculty/pconrad/public_html/cs16/10S/homework/H17/handout/code/* ~/cs16/H17
```

After you copy the code, you can compile and run the code for yourself to see how this works. You may like to experiment with the code to investigate further how struct assignment works—this may be helpful as you work on the H17 problems.

(c) arrays of structs

Once we have a struct definition like struct Time (see box to the right) we can make an array of struct Time. For example, if we wanted to store the typical starting times Tuesday/Thursday classes at UCSB, we could create an array like this:

As a reminder, here's the definition:

```
struct Time
{
  int hours;
  int minutes;
};
```

```
// make an array of 6 struct Time variables
struct Time classTimesTR[6];
```

And initialize the elements of the array with assignment statements like this:

```
classTimesTR[0].hours = 8; classTimesTR[0].minutes=0;
classTimesTR[1].hours = 9; classTimesTR[1].minutes=30;
classTimesTR[2].hours = 11; classTimesTR[2].minutes=0;
classTimesTR[3].hours = 12; classTimesTR[3].minutes=30;
classTimesTR[4].hours = 14; classTimesTR[4].minutes=0;
classTimesTR[5].hours = 15; classTimesTR[5].minutes=30;
```

We could then call a printTime function in a for loop to print each of these times:

```
  int i;
  for (i=0; i<6; i++)
    {
      printf(" classTimesTR[%i]=",i); printTime(classTimesTR[i]); printf("\n");
    }
```

The output would look like this:

```
-bash-4.1$ ./arrayOfStruct
 classTimesTR[0]=08:00
 classTimesTR[1]=09:30
 classTimesTR[2]=11:00
 classTimesTR[3]=12:30
 classTimesTR[4]=14:00
 classTimesTR[5]=15:30
-bash-4.1$
```

To see this code in the context of a complete program, look at arrayOfStruct.c in the code directory. Experimenting with this program may help you answer the questions on this homework assignment.

End of handout that goes with H17