CSC 180 lab 3: Boolean expressions

Mon Sept 24 or Thurs Sept 27, 2001

In writing 'if' statements (and also in writing loops), we are examining the truth value of a proposition, such as "*i* is less than ten". So we need formulas of this type. This data type is called "Boolean".

In C, boolean expressions are actually of type *int*. The operators which produce boolean values have result-type int, and the situations in which a boolean value is required allow int values.

1. If you didn't finish lab 2 last week, I suggest you finish it today (whether before or after the lab material below). Lab 2 was a bit long and will probably take some people longer than the two-hour period. This lab, on the other hand, is shorter than two hours, I think; and if you're pressed for time you can do only one of questions 6 and 7 below.

(In general, though, you should try not to get too far behind in the labs—some people will routinely take more than the two hours, and it will probably be worth it for them to put in the extra time on a weekly basis if possible.)

2. What is the value of each of the following expressions? What is the type of this data?

```
1 + 1 == 3
1 + 2 == 4 || 1 + 1 == 2
1 + 2 == 4 || 1 + 1 == 2 && 2 + 2 == 4
```

3. How would you write a little C program to find out the answers to the previous question? (An answer to this question appears at the end of this document.)

4. Instead of compiling your C program simply with "cc file.c", try using some options which activate additional compiler "warning" messages (error messages):

```
gcc -Wall -ansi -pedantic file.c
```

The most important of these is the `-Wall`. These compiler options are specific to a particular C compiler called "gcc"; gcc and cc are the same on linux (that is, cc gets you gcc), but different on skule.ecf.

Normally, your programs should all compile with no error messages **or** warning messages.

5. If we haven't got to for-loops in lecture by the time you are doing this lab, you can leave this question for next week, or skip it.

Write a loop whose output is eleven lines long: It outputs the numbers 1 through 10, one line each, except between lines 7 and 8 it says "hello". This can be accomplished by an 'if' statement within the 'for' body which does something special for the i==7 case. (The 'for' loop header will probably look much like ones discussed in class ("`for (i = 0; i < 10; i++) {`"); the *next* lab is about loops.)

6. Write a C program which inputs an integer grade value, 0 to 100, and outputs the letter grade for that numeric grade. (To simplify the problem, we could say that the grades are only A, B, C, D, and F; A is 80 to 100, B is 70 to 79, and so on.)

Be sure to use some good test cases in testing your program, including the borderline values (79, 80, 69, etc). What happens if a value greater than 100 or less than 0 is input? Is your program's behaviour in these cases reasonable?

7. Write a C program which prompts for and reads three pairs of Cartesian x,y coordinates, and outputs which two of them are closest together. The distance between points (x0,y0) and (x1,y1) is sqrt((x0−x1) * (x0−x1) + (y0−y1) * (y0−y1)). Remember to put anything which is a separate conceptual unit (such as this distance formula) into a separate C function, especially if the code would otherwise appear more than once in your program. Again, and for every program you write, to gain some assurance that your program is working you will need to try it with a variety of different test cases.

(continued)

8. On a separate note: If you have time today (after completing the above and also completing lab 2), you might want to test the assignment submission mechanism.

For this trial, you can simply submit your hello.c file from lab 1. If you don't have it any more, just create a file called hello.c with any contents at all.

You submit the source code files, not the compiled files. Your files ***must*** have the correct names (in this case, just one file, named hello.c). Unfortunately, the file names are not checked, so be careful. If you submit a file with the wrong name, just submit the right file (name). It's okay if there are some superfluous, incorrect files submitted, so long as you also submit the right ones. And if you submit more than one file with the same name, only the last one counts.

Submit your files with the command

```
submitcsc180f 0 hello.c
```

That 0 is the assignment number, and will be 1, 2, and 3 for the actual assignments. ''Assignment zero'' has been allocated for you to experiment with.

You may still change your files and resubmit them with the same command any time up to the due time. You can check that your assignment has been submitted with the command

```
submitcsc180f -l 0
```

(Note that the option is ''−l'' with the letter L, for ''list'', but lower-case.)

To compare, type ''ls −l hello.c''. Notice, in particular, that the file sizes are the same (the size in characters appears on the line just before the date).

This is the only submission method; you do not turn in any paper.

An answer to the question #3 above (towards the start of this lab):

```
#include <stdio.h>

int main()
{
    printf("%d\n", 1 + 1 == 3);
    return 0;
}
```

When you run this program, it will output ''0'', meaning that the expression ''1 + 1 == 3'' evaluates to 0. The other expressions can be substituted in here, or put into multiple printf statements.

A program which says ''if (1 + 1 == 3) ...'' is not helpful to answer question #3 because you will find out whether the result of that expression is considered to be true or false in a boolean context, but you will not get the actual datum. And you do know that 1+1 is not equal to 3, so it wouldn't help.

And by the way, the answer to the other part of the previous question (#2) (what is the type of this data) is *int*. Boolean operators in C return values of type *int*.