# Couple Notes about Boolean Expressions in C

## Not(!) operator

**Although the boolean operator "not" (!) is supposed to simply operate on boolean expressions, technically speaking, C does not have a type "boolean." In C, a boolean expression is actually an integer with a value of 1 or 0.**

**Thus, even though it seems strange, the statement**

```
x = (x > y);
```

**would be perfectly valid in C (assuming that x and y were integer variables.)**

**My advice would be to avoid using boolean expressions in places where one would typically expect arithmetic expressions. Similarly, avoid using arithmetic expressions in places where one would typically expect boolean expressions.**

**An example of this is as follows:**

```
if (x)
  x++;
else
  x--;
```

**Here, if x were 1 before the code started, x would change to 2. If x were 0 before the code start, it would change to -1. The expression !!5 evaluates to 1. This is quite confusing and code that involves such an expression should be avoided.**

## Short-Circuit Evaluation

**When a C compiler encounters complex boolean expressions involving either && or ||, it attempts to do as little work as possible. For example, consider the following sequence of statements:**

```
int age, fakeid;
age = 21;
fakeid = 0;
if ((age >= 21) || (fakeid == 1))
  printf("You can drink at the bar!\n");
```

**The compiler evaluates the expression inside of the if statement from left to right and first determines that (age>=21) is true. Then it sees that it is supposed to calculate the value of true "or" another expression. Regardless of whether that other expression is true or not, it's clear that the overall expression is true.**

**Since this is the case, the compiler never even TRIES to evaluate the second expression, it simply skips doing so and moves onto the printf.**

**If you think about it, in real life, if you were trying to figure out if someone could drink in a bar, once you knew they were 21, you wouldn't need to know if they had a fake ID or not!**

**Similarly, if a boolean expression is the and of two smaller expressions, if the first is false, the second is never even evaluated. Although this seems like a trivial detail, in some situations (in later examples), it will help us to shorten our code and avoid run-time errors.**

# Compound and Empty Statements

**I mentioned a block of statements in reference to the if-statement in the last lecture. A block of statements is simply denoted by matching curly braces:**

```
{
    stmt1;
    stmt2;
        ...
    stmtn;
}
```

**Typically, we will indent all the statements that are part of a block of statements. Syntactically, C treats a block of statements as a single statement. This is how we can get more than one line of code executed if the boolean expression in an if statement is true.**

**The empty statement is simply denoted by a semicolon and does nothing:**

```
;
```

**A valid example of the use of this statement is as follows:**

```
if (done == 1)
    ;
else
    printf("Hello World.\n");
```

**The empty statement is hardly ever necessary. (You can write code without it that does the same thing.)**

# If Statement Example

**Write an short C program (without comments) that reads in the dimensions of a room (length x width) and also reads in the size of a desk (length x width) and determines if the desk can fit in the room with each of it's sides parallel to a wall in the room. (This simply precludes the possibility of placing the desk diagonally. The problem becomes much more difficult if you allow this.) Assume the user enters positive numbers for each of the four dimensions.**

```
#include <stdio.h>
int main() {

   int roomlen, roomwid;
   int desklen, deskwid;

   printf("Enter the length and width of the room.\n");
   scanf("%d%d",&roomlen, &roomwid);
   printf("Enter the length and width of the desk.\n");
   scanf("%d%d",&desklen, &deskwid);

    if ((deskwid <= roomwid) && (desklen <= roomlen))
        printf("The desk will fit in the room.\n");
    else
        printf("The desk will not fit in the room.\n");

    return 0;

}
```

**Will this work in every situation? Why or why not?**

**Okay, here is my new solution. Will this work?**

```
#include <stdio.h>
int main() {

    int roomlen, roomwid;
    int desklen, deskwid;

    // Read in room and desk dimensions.
    printf("Enter the length and width of the room.\n");
    scanf("%d%d",&roomlen, &roomwid);
    printf("Enter the length and width of the desk.\n");
    scanf("%d%d",&desklen, &deskwid);

     //Adjust lengths and widths, if necessary
     if (roomlen < roomwid) {
         roomlen = roomwid;
         roomwid = roomlen;
     }
     if (desklen < deskwid) {
         desklen = deskwid;
         deskwid = desklen;
     }

     // Compare corresponding dimensions and output result.
     if ((deskwid <= roomwid) && (desklen <= roomlen))
         printf("The desk will fit in the room.\n");
     else
         printf("The desk will not fit in the room.\n");

     return 0;
}
```

**Here is my final version, taking into account the correct way to swap the value of two variables.**

```
#include <stdio.h>
int main() {
    int roomlen, roomwid;
    int desklen, deskwid;
    int temp;

    // Read in room and desk dimensions.
    printf("Enter the length and width of the room.\n");
    scanf("%d%d",&roomlen, &roomwid);
    printf("Enter the length and width of the desk.\n");
    scanf("%d%d",&desklen, &deskwid);

    //Adjust lengths and widths, if necessary
    if (roomlen < roomwid) {
        temp = roomlen;
        roomlen = roomwid;
        roomwid = temp;
    }
    if (desklen < deskwid) {
         temp = desklen;
        desklen = deskwid;
        deskwid = temp;
    }

    // Compare corresponding dimensions and output result.
    if ((deskwid <= roomwid) && (desklen <= roomlen))
        printf("The desk will fit in the room.\n");
    else
        printf("The desk will not fit in the room.\n");

    return 0;
}
```

**Here is another way to solve the same problem:**

```
#include <stdio.h>
int main() {

    int roomlen, roomwid;
    int desklen, deskwid;
    int temp;

    // Read in room and desk dimensions.
    printf("Enter the length and width of the room.\n");
    scanf("%d%d",&roomlen, &roomwid);
    printf("Enter the length and width of the desk.\n");
    scanf("%d%d",&desklen, &deskwid);

     // Check both ways of putting the desk in the room and
     // output the result.
     if ((deskwid <= roomwid) && (desklen <= roomlen))
         printf("The desk will fit in the room.\n");
     else if ((deskwid<=roomlen) && (desklen<=roomwid))
         printf("The desk will fit in the room.\n");
     else
         printf("The desk will not fit in the room.\n");

     return 0;

}
```

# Switch Statement

**This statement, just like the if statement, allows for conditional execution. The general construct is as follows:**

```
switch (<integer expression>) {
    case <value1>:
        <stmts1>
        break;
    case <value2>:
        <stmts2>
        break;
    ...
    default:
        <stmtsn>
}
stmtA
```

**The manner in which this expression is evaluated is as follows:**

**1) The integer expression is evaluated.**
**2) If this expression is equal to value1, then <stmts1> are executed and the break statement carries the execution to stmtA.**
**3) Otherwise, each value is compared th the integer expression value until one is found to match. At that point, the corresponding set of statements is executed and then the break carries execution to stmtA.**
**4) If no listed value is equal to the value of the expression, then the default case is executed.**

**Couple notes: The lists of values may not include ranges of values, but only single values. Also, the break statements are not required, but without them, then several cases may get executed.**

**This generally limits the use of case statements to situations where you know an expression will equal one of a few integer values and based on that want to execute a certain segment of code. You can emulate a case statement with an appropriate if statement.**

**Here is an example of a case statement:**

```
switch (answer) {
    case 1:
        printf("You have selected #1.\n");
        break;
    case 2:
        printf("You have selected #2.\n");
        break;
    case 3:
        printf("You have selected #3.\n");
        break;
    default:
        printf("Invalid selection.\n");
}
```