Relational Operators.

Relational operators are used in C for testing conditions of certain variables. For example, we might want to test if a variable equals some number, such as zero. Tests are often made within repetition structures (**for**, **while** and **do** loops), as well as within selection structures (**if**, **else**, **else if**, **switch**). We need a way to represent to the computer that we want to test the contents of a variable. For this reason, we have the relational operators.

When a test is made using a relational operator, C responds with either a *true* or *false* response. In C, the value *true* is represented by a **1**, and the value *false* is represented by a **0**. This concept of *true* being equal to **1** and *false* being equal to **0** is not important at this time. In fact, you can use relational operators without ever having to consider the "values" of true and false. More on this later.

There are 6 relational operators. They are shown in the table below, and later described in more detail.

Relational Operator Symbol	Test being made
==	is equal to
!=	is not equal to
<	is less than
>	is greater than
<=	is less than or equal
>=	is greater than or equal

1. "Is Equal To" Relational Operator: ==

This relational operator tests for equality, and is represented by 2 equal signs together with no spaces between them. This relational operator is not commonly used as the loop condition in a **for** loop, because it is not very practical. It is however commonly used in **while** and **do** loops, as well as in the **if**, **else**, **else if** and **switch** statements. Below is a segment of code which uses the equality relational operator in a **for** loop (since it is the only loop covered so far). You can see how the body of this particular loop will not execute at all because the loop condition (**x == 5**) is false to begin with:

```
for (x = 1;  x == 5; x = x + 1)
{
    printf ("I am in the loop!\n")
}
printf ("Goodbye.\n");
```

In this example the following sequence takes place:

1. variable **x** is assigned the value **1** (**x = 1**).
2. variable **x** is tested against the value **5** (**x == 5**).
3. this test is false, so the body of the **for** loop (the **printf** statement) is not executed.
4. the program continues with the "goodbye" **printf** statement.

The output of the above program would be:

```
Goodbye!
```

*Please be sure that you do not confuse the equality relational operator (**==**) with the assignment operator (**=**).* This is a very common programming error, and can lead to many hours spent debugging your program. This type of error is difficult to detect visually. (Particularly at 2:00 a.m. when most programmers are debugging their code! :-))

2. "Is Not Equal To" Relational Operator: !=

This relational operator tests for in-equality, and is represented by an exclamation point followed by an equal sign, with no spaces between them. This relational operator is not usually used as the loop condition in a **for** loop because for loops usually run from some initial value, until a final value. It is however commonly used in **while** and **do** loops, as well as in the **if, else** and **else if** statements. Below is a segment of code which uses the in-equality relational operator in a **for** loop (again, since it is the only loop covered so far). This particular loop will execute 4 times.

```
for (x = 1;  x != 5; x = x + 1)
{
    printf ("I am in the loop!\n")
}
printf ("Goodbye.\n");
```

The first time through the loop, **x** is assigned **1**, and the body of the loop is executed (because the loop condition (**x != 5**) is true.

The second iteration, **x** is incremented to **2** and again the loop condition (**x != 5**) is true, so the body of the loop is executed.

The third iteration, **x** is incremented to **3** and again the loop condition (**x != 5**) is true, so the body of the loop is executed.

The fourth iteration, **x** is incremented to **4** and again the loop condition (**x != 5**) is true, so the body of the loop is executed.

The fifth iteration **x** is incremented to **5** and again the loop condition (**x != 5**) is false, so the body of the loop is NOT executed.

The output of the above program is as follows:

```
I am in the loop!
I am in the loop!
I am in the loop!
I am in the loop!
Goodbye!
```

3. "Is Less Than" Relational Operator: <

This relational operator tests that a value is *less than* another value, and is represented by a *less than* sign. This relational operator is commonly used in all the looping structures (**for**, **while**, **do**), as well as in the **if**, **else**, and **else if** selection structures. Below is a segment of code which uses the less-than relational operator in a **for** loop. Try to figure out how many times this the body of this loop will execute.

```
for (x = 1;  x < 5; x = x + 1)
{
    printf ("I am in the loop!\n")
}
printf ("Goodbye.\n");
```

As you might have guessed, the body of this loop will execute 4 times, and the output of the above program is as follows:

```
I am in the loop!
I am in the loop!
I am in the loop!
I am in the loop!
Goodbye!
```

4. "Is Less Than Or Equal To" Relational Operator: <=

This relational operator tests that a value is *less than or equal to* another value, and is represented by a less than sign followed by an equal sign, with no spaces between them. This relational operator is commonly used in all the looping structures (**for**,

while, **do**), as well as in the **if**, **else**, and **else if** selection structures. Below is a segment of code which uses the less-than relational operator in a **for** loop. Try to figure out how many times the body of this loop will execute.

```
for (x = 1;  x <= 5; x = x + 1)
{
    printf ("I am in the loop!\n")
}
printf ("Goodbye.\n");
```

The body of this loop will execute 5 times. The output of the above program is as follows:

```
I am in the loop!
I am in the loop!
I am in the loop!
I am in the loop!
I am in the loop!
Goodbye!
```

5. "Is Greater Than" Relational Operator: >

This relational operator tests that a value is *greater than* another value, and is represented by a *greater than* sign. This relational operator is commonly used in all the looping structures (**for**, **while**, **do**), as well as in the **if**, **else**, and **else if** selection structures. Below is a segment of code which uses the greater-than relational operator in a **for** loop. Try to figure out how many times this the body of this loop will execute.

```
for (x = 1;  x > 5; x = x + 1)
{
    printf ("I am in the loop!\n")
}
printf ("Goodbye.\n");
```

As you might have guessed, the body of this loop will not execute at all since the initial condition (**x > 5**) is false to begin with. The output of the above program is as follows:

```
Goodbye!
```

6. "Is Greater Than Or Equal To" Relational Operator: >=

This relational operator tests that a value is *greater than or equal to* another value, and is represented by a greater than sign followed by an equal sign, with no spaces between them. This relational operator is commonly used in all the looping structures (**for**, **while**, **do**), as well as in the **if**, **else**, and **else if** selection structures. Below is a segment of code which uses the greater-than relational operator in a **for** loop. Try to figure out how many times the body of this loop will execute.

```
for (x = 1;  x >= 5; x = x + 1)
{
     printf ("I am in the loop!\n")
}
printf ("Goodbye.\n");
```

Again, the body of this loop will not execute at all since the initial condition (**x >= 5**) is false to begin with. The output of the above program is as follows:

```
Goodbye!
```

I guess the above output is fitting at this time because this ends the discussion of relational operators. (I see those smiles!) We will be using relational operators throughout the remainder of this course, so you are sure to understand them fully soon enough!

The next topic discusses the increment and decrement operators. These operators are very much used in looping, as well as other C statements.