Conditionals

Darin Brezeale

The University of Texas at Arlington

Basic Concepts – Conditionals

Conditionals are used for making decisions.

The conditionals available in C are

- `if` and `if-else` statements
- the `switch` statement
- the conditional expression operator, `?:`

Operators – Relational

The following relational operators are available in C:

==	equality
!=	not equal
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

Conditionals – `if`

The basic format of the `if` statement is

```
if (condition_is_true)
    do_something;
```

Examples

```
if (x >= 2)
    y = 10;

if (y == 10)
    printf("y is now 10");
```

Conditionals – `if` cont.

Question: What if we want to do more than one thing in response to a condition being true?

Answer: Create a block of statements.

Example

```
if (x >= 2)
{
    y = 10;
    printf("y is now 10");
}
```

Conditionals – `if` cont.

What if we wish to have a second conditional statement in response to the first condition being true? We can nest the `if` statements.

Example

```
if (count > 23)
    if(time < 45)
        z = 56;
```

This assigns a value of 56 to `z` only if `count > 23` and `time < 45`.

Conditionals – `if-else`

Sometimes we wish to do one thing if a condition is true but another if the condition is false. For this we can use an `if-else` statement:

```
if (condition_is_true)
    do_something;
else
    do_something_else;
```

Example

```
if (x >= 2)
    printf("x is greater than or equal to 2");
else
    printf("x is less than 2");
```

Conditionals – `if-else`

What if we have nested `if` statements and an `else` statement; which `if` does the `else` belong to?

```
/*  indented to trick you  */
if (amount == 13)
    if (cost == 52)
        printf("cost is %d\n", cost);
else
    printf("got here\n");


/*  indented to show correct logic; the else goes with
    the nearest unmatched if statement within a block  */
if (amount == 13)
    if (cost == 52)
        printf("cost is %d\n", cost);
    else
        printf("got here\n");
```

Conditionals Notes

Keep in mind that any condition that evaluates to a nonzero value is considered true.

```
if (8)
    printf("non-zero values are true\n");
else
    printf("this never prints\n");

/* another example */
if (-3.4)
    printf("-3.4 is considered true\n");
else
    printf("this never prints\n");

/* another example */
if (0)
    printf("zero is false\n");
else
    printf("this is always false\n");
```

Conditionals Notes cont.

WARNING: Don't use `=` when you really mean `==`

`=` is used for assigning values

Example: `a = 5;`

`==` is used for determining if two values are equal

Example: `if (a == 5)`

Conditionals – `switch`

An `if-else` statement is used for binary decisions–those with two choices. Sometimes there are more than two choices.

```
switch (expression)
{
    case label_1: do_this
    case label_2: do_this
        .
        .
        .
    case label_n: do_this
    default: do_this
}
```

Conditionals – `switch` cont.

Notes on the use of the `switch` statement

- The labels must be integers (or at least evaluate to an integer).
- The `default` line is optional.
- Once the matching label is found, that statement and each following statement will be executed (unless we use a `break`).

`switch` Example

Example:

```
#include <stdio.h>

int main( void )
{
    int a = 3;
    int b = 5;

    switch(b - a)
    {
        case 5: printf("countdown from 5\n");
        case 4: printf("countdown from 4\n");
        case 3: printf("countdown from 3\n");
        case 2: printf("countdown from 2\n");
        case 1: printf("countdown from 1\n");
    }
}
```

`switch` Example cont.

Output

```
countdown from 2
countdown from 1
```

`switch` and `break`

On many occasions we may only wish to do a single thing in response to what the `switch` evaluates to. For this, we can use a `break` statement.

Example:

```
#include <stdio.h>

int main(void)
{
    int cost = 18;

    switch(cost)
    {
        case 23:  printf("cost is 23\n"); break;
        case 18:  printf("cost is 18\n"); break;
        case 75:  printf("cost is 75\n"); break;
    }
}
```

`switch` and `break` cont.

Output

```
cost is 18
```

Conditionals – `? :`

Instead of an `if-else` statement, we can use the *conditional expression operator*:

(condition_is_true) `?` do_if_true `:` do_if_false;

Example

```
(w < 14) ? (x = 10) : (y = 19);
```

Conditionals – `?:` cont.

A difference between this and an `if-else` statement is that the conditional expression can be used like other expressions:

Example

```
answer = ((w > 14) ? 28 : 16);
```

This is equivalent to

```
if (w > 14)
    answer = 28;
else
    answer = 16;
```

Conditionals Example 1

Both types of conditionals are used equivalently here

```
#include <stdio.h>

int main(void)
{
    int x = 3, y = 10;

    printf("x is %d, y is %d\n\n", x, y);

    if (x > y)
        printf("x is larger\n\n");
    else
        printf("y is larger\n\n");

    /*  equivalent structure  */
    printf("Let's try with the conditional expression:\n");
    (x > y) ? printf("x is larger\n") : printf("y is larger\n") ;
}
```

Conditionals Example 1 cont.

The output is

```
x is 3, y is 10

y is larger

Let's try with the conditional expression:
y is larger
```

Conditionals Example 2

Here we have a conditional expression inside a function call:

```
#include <stdio.h>

int main(void)
{
    int x = 5;

    printf("x is %s\n", (x < 100) ? "small" : "large");
}
```

produces

```
x is small
```

Operators – Logical

C supports the following logical operators:

!	not
&&	and
||	or