Subsections

Conditionals

This Chapter deals with the various methods that C can control the ***flow*** of logic in a program. Apart from slight syntactic variation they are similar to other languages.

As we have seen following logical operations exist in C:

==, !=, ||, &&.

One other operator is the unitary - it takes only one argument - ***not*** **!**.

These operators are used in conjunction with the following statements.

The `if` statement

The `if` statement has the same function as other languages. It has three basic forms:

```
if  (expression)
                        statement
```

...or:

```
if  (expression)
                        statement1
            else
                        statement2
```

...or:

```
if  (expression)
                        statement1
            else if (expression)
                        statement2
            else
                        statement3
```

For example:-

```
int x,y,w;

            main()
                        {

                        if (x>0)
                                    {
                                    z=w;
                                    ........
                                    }
                        else
                                    {
                                    z=y;
                                    ........
                                    }

                        }
```

The `?` operator

The `?` (***ternary condition***) operator is a more efficient form for expressing simple `if` statements. It has the following form:

```
expression1 ? expression2:  expression3
```

It simply states:

`if` *expression1* `then` *expression2* `else` *expression3*

For example to assign the maximum of `a` and `b` to `z`:

```
z = (a>b) ? a : b;
```

which is the same as:

```
if (a>b)
                        z = a;
            else
                        z=b;
```

The `switch` statement

The C `switch` is similar to Pascal's `case` statement and it allows multiple choice of a selection of items at one level of a conditional where it is a far neater way of writing multiple `if` statements:

```
switch (expression) {
                        case item1:
                                        statement1;
                                        break;
                        case item2:
                                        statement2;
                                        break;
                                        ⋮                       ⋮                       cas
                                        statementn;
                                        break;
                        default:
                                        statement;
                                        break;
                      }
```

In each case the value of $item_i$ must be a constant, variables are not allowed.

The `break` is needed if you want to terminate the `switch` after execution of one choice. Otherwise the next case would get evaluated. **Note:** This is unlike most other languages.

We can also have **null** statements by just including a ; or let the switch statement ***fall through*** by omitting any statements (see ***e.g.*** below).

The `default` case is optional and catches any other cases.

For example:-

```
        switch (letter)
                        {
                         case `A':
                         case `E':
                         case `I':
                         case `O':
                         case `U':
                                        numberofvowels++;
                                        break;

                         case ` ':
                                        numberofspaces++;
                                        break;

                         default:
                                        numberofconstants++;
                                        break;
                        }
```

In the above example if the value of `letter is `A', `E', `I', `O' or `U' then numberofvowels is incremented.`

`If the value of letter is ` ' then numberofspaces is incremented.`

`If none of these is true then the default condition is executed, that is numberofconstants is incremented.`

Exercises

Exercise 12304

Write a program to read two characters, and print their value when interpreted as a 2-digit hexadecimal number. Accept upper case letters for values from 10 to 15.

Exercise 12305

Read an integer value. Assume it is the number of a month of the year; print out the name of that month.

Exercise 12306

Given as input three integers representing a date as day, month, year, print out the number day, month and year for the following day's date.

Typical input: 28 2 1992 Typical output: Date following 28:02:1992 is 29:02:1992

Exercise 12307

Write a program which reads two integer values. If the first is less than the second, print the message up. If the second is less than the first, print the message down If the numbers are equal, print the message equal If there is an error reading the data, print a message containing the word Error and perform exit(0);

Dave Marshall
1/5/1999