Lesson 4 Decisions

Computer languages must be able to perform different sets of actions depending on different outcomes to events. The C programming language has three main decision making structures, the **if** statement, the **ifelse** statement and the **switch** statement.

A fourth structure, the **conditional operator** may also be used.

The if statement

Load and display the code below for an example of our first conditional branching statement.

```
#include<stdio.h>
int main(void)
 {
 char ch;
 scanf("%c",&ch);
  if( ch=='z')
   printf("\n YOU HAVE INPUT   HAVEINPUT A z\n");
 return(0);
 }
```

This simple program informs the user that they have typed in a 'z' if and when a 'z' has been indeed typed in. If another character is typed, then the program prints nothing. Consider the **if** statement. It starts with the keyword **if** followed by an expression in parentheses. When the expression is evaluated and found to be true, the single statement following the **if** is executed, when its found to be false, the statement is skipped.

The single statement can be replaced by a compound statement composed of several statements bounded by braces. The expression "ch == z" is simply asking if the value of **ch** (which was typed in) is equal to the character z. You can see that the **if** statement is in fact similar to the **while** statement. The major difference is that in the **while** statement, if the condition is true, all the statements will be executed in the loop repeatedly until the condition is false.

Using **if** however, the statements in the parentheses will only be executed once.

Nested if statements

It is possible to nest **if** statements, that is, one **if** statement can be part of the body of another **if** statement. Compile and run the following code. Is the output what you expected ?

```
#include<stdio h>

int main(void)
{
 char ch1,ch2,ch3;
 scanf("%c%c%c",&ch1,&ch2,&ch3);
 if( ch1=='D')
   if(ch2=='A')
     if(ch3=='N')
 printf("\n YOU HAVE INPUT - DAN\n");
 return(0);
 }
```

If-else statements

The **if-else** statement is similar to the **if** statement with the addition of a new keyword, **else**.

This says that if the expression in the parentheses evaluates as true, the first expression is executed, otherwise the expression following the **else** is executed. Thus, one of the two expressions will always be executed.

Note that in the **if** construct example the single expression was either executed or skipped. Compile and run the program below to see if it does what you expect.

```
int main()
{
      int data;
      for(data = 0 ; data < 10 ; data = data + 1)
        {
         if(data == 2)
         printf("Data is now equal to %d\n", data);
         if(data < 5)
         printf("Data is now %d, which is less than 5\n", data);
         else
         printf("Data is now %d, which is greater than 4\n", data);
        }

     return(0);
}
```

Nested if-else statements

It is possible to nest an **if-else** construct within either the body of an **if** statement or the body of an **else** statement. The only possible confusion is perhaps, which **if** gets the **else** ?

We can answer this question with the aid of some code. Type in the following code and compile it. Before you run the code, see can you predict what the output will be.

```
 int main(void)
{
    int money;
    printf("HOW MUCH DO YOU EARN A WEEK(€0-500):\n");
    scanf("%d",&money);
    if( money < 350 )
     if( money > 250)
        printf("\nYour earnings are close to the average weekly wage\n");
        else
        printf("\nYour earnings are different to the average weekly wage\n");
     return(0)
}
```

Now suppose that €210 in typed in. What will the output be? When you run the program you will find that the second print statement is executed.

This is because the **else** statement is associated with the last **if** that dose not have an **else** of it own.

How can we make sure that the **else** goes with the first **if** statement?

The switch statement

Type in and compile the code below for an example of the **switch** construct. It begins with the keyword **switch** followed by a variable in parentheses which is the switching variable, in this case **i**.

As many “cases” as required are then enclosed within a pair of braces. The reserved word **case** is used to begin each case, followed by the value of the variable for that case, followed by a colon, and finally the statements to be executed.

```
int main()
{
 int i;
for (i=0;i<=2;i++)
    {
       switch (i)
        {
         case 0: printf("i=zero\n");   break;
         case 1: printf("i=one\n");    break;
         case 2: printf("i=two\n");     break;
         default:printf(" i is not 0,1,2 \n");   break;
         }
       }
  return(0);
}
```

In the example above, if the variable **i** contains the value 0 during this pass of the **switch** statement, the **printf()** will output "i=zero". Note that the **break** statement will cause the code to jump out of the **switch** construct.

If no **break** statement is used following a **case,** then control will fall through to the next **case**. Thus without the **break**, the program will execute not only the statements for a specific case, but also all the statements for the following cases as well. The key point here is remember the **break** !

It should be clear that any of the above constructs can be nested within each other or placed in succession, depending on the needs of the particular programming task at hand.

Logical Operators

Logical Operators provide a useful way to clarify complicated **if-else** constructs as well as other constructs.

There are three logical or Boolean operators in C. These are

‘||’ (logical OR)

‘&&’ (logical AND)

‘!’ (logical NOT)

The logical OR operator means that if any one of a number of expressions is true then the entire expression is true.

The logical AND operator means that all the expressions in a larger expression must be true for the larger expression to be true.

The logical NOT operator is used for reversing the logical value of an expression which it operates on. For example !(x<=50) means "NOT x less than or equal to 50". This is the same as *(x>50)*.

Note that the NOT operator is often used with an **if** construct to reverse the logical value of a single variable, eg if(!value) can be used instead of if(value==0). The second if is more readable than the first.

Operator Precedence

The table below gives the order of precedence of the operators which we have used so far.

Operator	**Type**
- !	unary:logical NOT ,minus
* / %	arithmetic:multiplicative
+ -	arithmetic:additive
<> <= >=	relational: inequality
== !=	relational: quality
&& ||	logical AND, OR
= += -= *= /= %=	assignment

Programming Exercises

1. Using the if-else construct write a program that reads in an exam mark (0-100) and prints out what grade the mark falls into. Use the following: A(80-100), B(60-79),C(40-59) and D(<40).
2. Develop a program of your choice that uses the logical operators ||, ! and &&.
3. What will the following code assign to the variable abs. (This is the conditional operator) abs =

   (num < 0) ? -num :num;

   Rewrite the above using if-else statements.
4. Write a program that will ask a car driver how fast they were driving on the motorway and then printout the response a Garda might give using the following speed ranges: >85, >70, >55, < 40.
5. The solution of a quadratic equation (inputs are a,b and c)

```
        a x^2 + b x + c = 0
        can be found from the following formulae
x1=(-b+sqrt(b*b - 4*a*c) ) / (2*a)          x2=(-b-
sqrt(b*b - 4*a*c) ) / (2*a)
```

   You are asked to write a general program to solve such an equation. The program should be able to handle the three distinct cases,

   (i) `b*b = 4*a*c`
   (ii) `b*b > 4*a*c`
   `(iii)b*b < 4*a*c`

6. Using a switch structure,write a program to act as a calculator which reads in one of four operations(+,-,*,/) along with two numbers that it operates on.
7. Write a program that counts the number of white spaces or blanks in a sentence which is terminated by a period (full stop).
8. Write a program using a do while loop which finds the length of a word. Use the C library function isspace() in ctype.h to check for white spaces.