

# Structure and Nesting

Objectives:

This section brings together the various looping mechanisms available to the C programmer with the program control constructs we met in the last section.

We also demonstrates a neat trick with random numbers.

It is one of the great discoveries of programming that you can write any program using just simple while loops and if statements. You don't need any other control statements at all. Of course it might be nice to include some other types of control statement to make life easy - for example, you don't need the for loop, but it is good to have! So as long as you understand the if and the while loop in one form or another you can write any program you want to.

If you think that a loop and an if statement are not much to build programs then you are missing an important point. It's not just the statements you have, but the way you can put them together. You can include an if statement within a loop, loops within loops are also OK, as are loops in ifs, and ifs in ifs and so on. This putting one control statement inside another is called nesting and it is really what allows you to make a program as complicated as you like.

Think of a number:

Now let's have a go at writing the following program: 'It thinks of a number in the range 0 to 99 and then asks the user to guess it'. This sounds complicated, especially the 'thinks of a number' part, but all you need to know is that the statement:

```
r = rand()
```

will store a random number in the integer variable r. The standard library function rand() randomly picks a number within the range 0 to 32767, but this might vary from machine to machine. Look upon rand() as being a large dice.

Our problem is to select a number between 0 and 99 and not between 0 and 32767. How can we get our random number to within our range? The rand() function will produce numbers such as:

```
2567
134
20678
15789
32001
15987
etc...
```

If you look at the last two digits of all of these numbers they would form our random set! To select just these numbers we can use an arithmetic calculation of the following form:

```
r = rand() % 100
```

That is, to get the number into the right range you simply take the remainder on dividing by 100, ie a value in the range 0 to 99. You should remember this neat programming trick, you'll be surprised how often it is required.

Our solution to the problem is as follows:

```
#include <stdio.h>

main()
{
    int target;
    int guess;
    int again;
```

```

printf("\n Do you want to guess a number 1 =Yes, 0=No ");
scanf("%d",&again);

while (again)
{
    target = rand() % 100;
    guess = target + 1;

    while(target!=guess)
    {
        printf("\n What is your guess ? ");
        scanf("%d",&guess);

        if (target>guess) printf("Too low");
        else printf("Too high");
    }

    printf("\n Well done you got it! \n");
    printf("\nDo you want to guess a number 1=Yes, 0=No");
    scanf("%d",&again);
}
}

```

This looks like a very long and complicated program, but it isn't. Essentially it used two loops and an if/else which in English could be summarised as:

```

while(again) {
    think of a number
    while (user hasn't guessed it)
    {
        get users guess.
        if (target < guess) tell the user the guess is low
        else          tell the user the guess is high
    }
}

```

The integer variable again is used to indicate that the user wants to carry on playing. If it is 0 then the loop stops so 0 = No, and 1, or any other non-zero value, = Yes.

If you try this program out you will discover that it has a slight flaw - not so much a bug, more a feature. If the user guesses the correct value the program still tells the user that the guess is too high and then congratulates them that they have the correct value. Such problems with how loops end are common and you have to pay attention to details such as this. There are a number of possible solutions, but the most straight forward is to change the inner loop so that the first guess is asked for before the loop begins. This shifts the test for the loop to stop to before the test for a high or low guess:

```

#include <stdio.h>

main()
{
    int target;
    int guess;
    int again;

    printf("\n Do you want to guess a number 1 =Yes, 0=No ");
    scanf("%d",&again);

    while (again)
    {
        target = rand() % 100;

        printf("\n What is your guess ? ");
        scanf("%d",&guess);

        while(target!=guess)
        {
            if (target>guess) printf("Too low");

```

```
    else printf("Too high");
    printf("\n What is your guess ? ");
    scanf("%d",&guess);
}

printf("\n Well done you got it! \n");
printf("\n Do you want to guess a number 1=Yes, 0=No");
scanf("%d",&again);
}
}
```

If you want to be sure that you understand what is going on here, ask yourself why the line:

```
guess = target + 1;
```

was necessary in the first version of the program and not in the second?