Chapter 3 - Input

In the previous chapter the value of a variable was determined when the program was compiled, by setting x=10, for example, when we wrote the code. But what if you wanted to enter the value of x when the program was running?

We can use a keyword similar to cout, called "cin" to do this. "cin" stands for "Channel IN".

For example, this program "prompts" the user to enter a value and then it will print the value you entered.

```
1  #include<iostream.h>
2
3  int main()
4  {
5  int x;
6
7  cout<< "Enter a number" <<endl;
8
9  cin>>x;
10
11 cout<< "Your number is " << x <<endl;
12
13 return 0;
14 }
```

Example 3a

Line 9, will wait indefinitely for the user to input a value and press enter. Notice the direction of the arrows for a cin statement, they are pointing inwards, as data is now flowing into the program. For a cout statement the arrows point in the opposite direction.

Another example:

```
1  #include<iostream.h>
2
3  int main()
4  {
5  int x,y,z;
6
7  cout<< "Enter two numbers" <<endl;
8
9  cin>> x >> y;
10 z=(x+y);
11
12 cout<< "The sum is " << z <<endl;
13
14 return 0;
15 }
```

Example 3b

What if you wanted to write a program that asks for your name? Well, we can't use a data type such as int or float to store it - as your name is made up of letters, not numbers!

So let's introduce a new data type called "char". A char data type can store a character i.e. a single character enclosed in single quotation marks, e.g. 'a' or 'b' etc.

We could declare and initialise a char variable with the following lines:

```
char var1
var1 = 'a'

cout<<"var1 has the value "<<var1<<endl;
```

The result of this would be: var1 has the value a

That's all very nice, but most people's names are usually longer than one letter! So we need to find a way of storing multiple chars! We can do this by using an array of chars.

Here is how we would declare a char array to store a person's name:

```
char name[10];
```

You will probably guess that this line will let you store a name up to 10 letters in length. Well, you're *almost* correct. In fact it will only hold up to 9 letters. This is because char arrays store info as follows:

```
'd '  'a'  'v '  'i'  'd'  '\0'
```

Here, the name David takes up five spaces. The character at the end ('\0') indicates to the compiler where the end of the string is and so we must allow for this when we declare char arrays.

i.e. char word[20] will hold a word up to 19 characters in length plus '\0'.
char name[15] will hold a word up to 14 characters in length plus '\0'.

Now let's do a program example. Compile and run the following code.

```
1  #include<iostream.h>
2
3  int main()
4  {
5  char MyName[10];
6
7  cout<<"Enter a name"<<endl;
8
9  cin>>MyName;
10
11 cout<<"Hello "<< MyName <<endl;
12
13 return 0;
14 }
```

Example 3c

Now try typing in your first and second name. What happens when you run the program? The computer only prints out your first name! This is because it stops when it sees a blank space (or so-called whitespace). This minor problem can be resolved by using the function cin.getline() as follows:

```
1  #include <iostream.h>
2
3  int main()
4  {
5  char MyName[20];
6
7  cout << "Please enter your name" << endl;
8  cin.getline(MyName,20);
9
10 cout << "your name is " << MyName << endl;
11
12 return 0;
13 }
```

Notice that the function cin.getline() takes 2 arguments in order to use it. The first is the name of the variable (in our case it's called MyName) and the second is the maximum length of the variable (here it is 19 taking into account the '\0' character).