STEP ONE STUDY NOTES

This document contains some brief notes on topics covered in *Step One* of the course '*Advanced C Programming: Pointers*'. You will find a similar set of study notes for each step of the course.

You may want to refer to use these notes as a revision aid or to help clarify important points discussed in the course. I do *not* recommend that you read them *before* watching the course lectures. These notes do not attempt to explain pointers in detail. The course lectures do that. Instead, they are aimed at *summarising* key concepts described in the lectures. So you may find it useful to read the study notes *after* you have watched the lectures in each step of the course.

In addition, be sure to read the *FAQ/ReadMe* supplied with this course. That contains answers to some specific questions that have been posed by students. It also contains any errata and corrections.

Pointer Basics

Key Points

Pointers and Addresses

- A pointer is a variable whose value is an address in memory.
- An address is a location in memory (where some data may be stored).
- The value of a pointer is a number that indicates a specific memory location.
- 'Indirection' means accessing data stored at the address given by a pointer.
- 'Dereferencing' is another term used to describe indirection.

Pointers and Addresses

Pointer variables

A pointer variable is declared like this:

```
<type> * <variable name>;
```

So, for example, here `numPtr` is a pointer variable that may 'point to' (that is, it may store the address of) an integer:

```
int* numPtr;
```

Addresses

To obtain the address of some data stored in a variable, use the ampersand: `&`

```
int num;
int* numPtr;
num = 100;
numPtr = &num;
```

This code assigns the address of the `num` variable to the `numPtr` variable. It is important to understand that the address-of operator does *not* return the *value* of the `num` variable, that is 100. It returns the *address* in memory at which that value is stored.

INDIRECTION

In order to access the data at an address given by a pointer, you need to use the *indirection* or *dereferencing* operator which, once again, is the asterisk or 'star': `*`

Here I dereference `numPtr` in order to obtain the value stored at that address – here the integer 100 – and assign that value to the variable `num2`:

```
int num;
int* numPtr;
int num2;
num = 100;
numPtr = &num;
num2 = *numPtr;
```