Topic 2

Type Definitions and Complex Data Types

Variable Prefixes

&

The prefix of an & with a variable returns the address of the variable in memory. For example an integer i would have the address returned by &i.

*****

The prefix of a * with a variable returns the variable pointed to by the variable prefixed. Also the prefix * can be used to declare a pointer to a data type. For example an integer i exists and a pointer p also exists, then

int i, *p ; // i is declared as an integer, p is declared as a pointer to integer data types.

The pointer variable p is assigned the address of i.

p= &i ;

Then the value of i can be gotten using *p

Pointers and typedef structures:

See next page for more introductory explanation of typedef structures. A typedef structure has sub variables. For example a variable George and a pointer to a person aMan:

```
typedef struct {int age; char *name} person;
person George, *aMan;

aMan= &George;  // aMan now points to George.
```

The . can be used with George because the George variable is a structure.
George.age // references the structure “George” and gets the value of “age” within the structure.

To get the same value using pointer variable “aMan” you need to use the -> symbols.
aMan->age // Evaluates the address contained in “aMan” to find the structure “George”, and then gets the value of “age” within the structure.

ENUM

ENUM allows you to define a list of aliases which represent integer numbers. For example if you find yourself coding something like:

```
#define MON 1
#define TUE 2
#define WED 3
```

You could use **enum** as below.

```
enum week { Mon=1, Tue, Wed, Thu, Fri Sat, Sun} days;
```

or

```
enum escapes { BELL   = '\a', BACKSPACE = '\b', HTAB = '\t',
                        RETURN = '\r', NEWLINE   = '\n', VTAB = '\v' };
```

or

```
enum boolean { FALSE = 0, TRUE };
```

TYPEDEF

TYPEDEF Every variable has a data type. **typedef** is used to define new data type names to make a program more readable to the programmer.

For example:

```
                              |
    main()                    |   main()
    {                         |   {
        int money;            |       typedef int Pounds;
        money = 2;            |       Pounds money = 2
    }                         |   }
```

These examples are EXACTLY the same to the compiler. But the right hand example tells the programmer the type of money he is dealing with.

A common use for typedef is to define a boolean data type as below.
Note: Recent C++ compilers have introduced a boolean datatype.

```
    typedef enum {FALSE=0, TRUE} Boolean

    main ()
    {
        Boolean flag = TRUE;
    }
```

And as a final example, how about creating a string datatype?

```
typedef char *String;

main()
{
    String Text = "Thunderbird";

    printf("%s\n", Text);
}
```

The main use for typedef seems to be defining structures. For example:

```
    typedef struct {int age; char *name} person;
    person people;
```

Take care to note that **person** is now a type specifier and NOT a variable name.

As a final note, you can create several data types in one hit.

```
    typedef int Pounds, Shillings, Pennies, Dollars, Cents;
```