# CONTENTS

**Chapters and Topics** **page**

# CHAPTER – 13

# STRUCTURES

## ACCESSING MEMBERS OF A STRUCTURE

Each member of a structure can be used just like a normal variable, but its name will be a bit longer. To return to the examples above, member name of structure st_rec will behave just like a normal array of char, however we refer to it by the name

st_rec.name

Here the dot is an operator which selects a member from a structure.

Where we have a pointer to a structure we could dereference the pointer and then use dot as a member selector. This method is a little clumsy to type. Since selecting a member from a structure pointer happens frequently, it has its own operator -> which acts as follows. Assume that st_ptr is a pointer to a structure of type student We would refer to the name member as

st_ptr -> name

## STRUCTURES AS FUNCTION ARGUMENTS

A structure can be passed as a function argument just like any other variable. This raises a few practical issues.

Where we wish to modify the value of members of the structure, we must pass a pointer to that structure. This is just like passing a pointer to an int type argument whose value we wish to change.

If we are only interested in one member of a structure, it is probably simpler to just pass that member. This will make for a simpler function, which is easier to re-use. Of course if we wish to change the value of that member, we should pass a pointer to it.

When a structure is passed as an argument, each member of the structure is copied. This can prove expensive where structures are large or functions are called frequently. Passing and working with pointers to large structures may be more efficient in such cases.

## FURTHER USE OF STRUCTURES

As we have seen, a structure is a good way of storing related data together. It is also a good way of representing certain types of information. Complex numbers in mathematics inhabit a two dimensional plane (stretching in real and imaginary directions). These could easily be represented here by

```
struct numbers{
        double real;
        double imag;
} complex;
```

doubles have been used for each field because their range is greater than floats and because the majority of mathematical library functions deal with doubles by default.

In a similar way, structures could be used to hold the locations of points in multi-dimensional space. Mathematicians and engineers might see a storage efficient implementation for sparse arrays here. Apart from holding data, structures can be used as members of other structures. Arrays of structures are possible, and are a good way of storing lists of data with regular fields, such as databases. Another possibility is a structure whose fields include pointers to its own type. These can be used to build chains (programmers call these linked lists), trees or other connected structures. These are rather daunting to the new programmer, so we won't deal with them here.