Using *typedef* keyword with C Structures

typedef is a keyword used in C language to assign alternative names to existing datatypes. It is mostly used with user defined datatypes, when names of the datatypes become slightly complicated to use in programs.

The basic syntax of a C structure is as,

```
struct [tag name] {

   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```

Structure definition using typedef

We have defined structures using the struct keyword and a tag name. Like,

```
struct Book

{
        int ref_no;
        char name[15];
        char author[50];

};
```

When creating variables of the defined type we used the struct keyword.

```
int main(){
        struct Book book1;
        book1.ref_no= 11562;
        return 0;
}
```

The same structure can be written using typedef as

```
typedef struct Book
{
        int ref_no;
        char name[15];
        char author[50];

}Book;
```

The **tag name** after the struct key word is optional here. You can write your structure by omitting it too, like follows.

```
#include <stdio.h>

typedef struct
{
        int ref_no;
        char name[15];
        char author[50];

}Book;

int main(){

        Book book1;
        book1.ref_no = 11562;

        return 0;
}
```

Using a **typedef** avoids having to write *struct* every time you declare a variable of that type.

The purpose of typedef is to give a name to a type specification. Here we use it with the structures and then we could use the structure names much like any of the built-in types of the language to declare variables.

Eg: *Instead of struct Book book1, we could use Book book1;*

So the **typedef** keyword reduces the length of the code and complexity of data types. It also helps in understanding the program.