The C programming language provides a keyword called **typedef**, which you can use to give a type a new name. Following is an example to define a term **BYTE** for one-byte numbers −

```
typedef unsigned char BYTE;
```

After this type definition, the identifier BYTE can be used as an abbreviation for the type **unsigned char, for example.**.

```
BYTE  b1, b2;
```

By convention, uppercase letters are used for these definitions to remind the user that the type name is really a symbolic abbreviation, but you can use lowercase, as follows −

```
typedef unsigned char byte;
```

You can use **typedef** to give a name to your user defined data types as well. For example, you can use typedef with structure to define a new data type and then use that data type to define structure variables directly as follows −

```
#include <stdio.h>
#include <string.h>

typedef struct Books {
   char title[50];
   char author[50];
   char subject[100];
   int book_id;
} Book;

int main( ) {

   Book book;

   strcpy( book.title, "The C Programming Language");
   strcpy( book.author, "Brian Kernighan and Dennis Ritchie");
   strcpy( book.subject, "Learn how to program in C");
   book.book_id = 0131103628;

   printf( "Book title : %s\n", book.title);
   printf( "Book author : %s\n", book.author);
   printf( "Book subject : %s\n", book.subject);
   printf( "Book book_id : %d\n", book.book_id);

   return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
Book  title : The C Programming Language
Book  author : Brian Kernighan and Dennis Ritchie
Book  subject : Learn how to program in C
Book  book_id : 131103628
```

typedef vs #define

#define is a C-directive which is also used to define the aliases for various data types similar to **typedef** but with the following differences −

- **typedef** is limited to giving symbolic names to types only where as **#define** can be used to define alias for values as well, q., you can define 1 as ONE etc.
- **typedef** interpretation is performed by the compiler whereas **#define** statements are processed by the pre-processor.

The following example shows how to use #define in a program −

```
#include <stdio.h>

#define TRUE  1
#define FALSE 0

int main( ) {
   printf( "Value of TRUE : %d\n", TRUE);
   printf( "Value of FALSE : %d\n", FALSE);

   return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
Value of TRUE : 1
Value of FALSE : 0
```

<END>