C 位域 C 输入 & 输出

# C typedef

C 语言提供了 **typedef** 关键字，您可以使用它来为类型取一个新的名字。下面的实例为单字节数字定义了一个术语 **BYTE**：

```
typedef unsigned char BYTE;
```

在这个类型定义之后，标识符 BYTE 可作为类型 **unsigned char** 的缩写，例如：

```
BYTE  b1, b2;
```

按照惯例，定义时会大写字母，以便提醒用户类型名称是一个象征性的缩写，但您也可以使用小写字母，如下：

```
typedef unsigned char byte;
```

您也可以使用 **typedef** 来为用户自定义的数据类型取一个新的名字。例如，您可以对结构体使用 typedef 来定义一个新的数据类型名字，然后使用这个新的数据类型来直接定义结构变量，如下：

**实例**

```
#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, "C 教程");
strcpy( book.author, "Runoob");
strcpy( book.subject, "编程语言");
book.book_id = 12345;
printf( "书标题 : %s\n", book.title);
printf( "书作者 : %s\n", book.author);
printf( "书类目 : %s\n", book.subject);
printf( "书 ID : %d\n", book.book_id);
return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
书标题 : C 教程
书作者 : Runoob
书类目 : 编程语言
书 ID : 12345
```

## typedef vs #define

**#define** 是 C 指令，用于为各种数据类型定义别名，与 **typedef** 类似，但是它们有以下几点不同：

- **typedef** 仅限于为类型定义符号名称，**#define** 不仅可以为类型定义别名，也能为数值定义别名，比如您可以定义 1 为 ONE。
- **typedef** 是由编译器执行解释的，**#define** 语句是由预编译器进行处理的。

下面是 #define 的最简单的用法：

**实例**

```
#include <stdio.h>
#define TRUE 1
#define FALSE 0
int main( )
{
printf( "TRUE 的值: %d\n", TRUE);
printf( "FALSE 的值: %d\n", FALSE);
return 0;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
TRUE 的值: 1
FALSE 的值: 0
```

C 位域 C 输入 & 输出