Creating a constant type

Once an enumerated sequence is declared it can be considered to be like a new data type in its own right, with properties of its specified constant names and their associated values.

Variables of this **enum** data type can be declared in the same way that variables are declared of other data types – using this syntax:

data-type variable-name **;**

The example on the previous page created an enumerated sequence named **SNOOKER**, which can be regarded as a data type of **enum SNOOKER**. So a variable named "pair" of that data type can be created with the declaration **enum SNOOKER pair ;** and can store values of the enumeration set defined by that type.

To explicitly assign an integer value to a variable of an enumerated data type the C standard recommends that a cast be used to convert the **int** data type to the **enum** data type, like this:

```
pair = ( enum SNOOKER ) 7 ;
```

In practice this is not needed though as enumerated values are always integers so are equivalent to the **int** data type.

An **enum** declaration can optionally also create a variable by specifying a variable name after the final brace. For example, the declaration **enum BOOLEAN { FALSE , TRUE } flag ;** defines an **enum** data type and creates a variable named "flag" of that type.

Custom data types can be defined using the **typedef** keyword and this syntax:

typedef ***definition type-name*** **;**

Declaration of custom data types can help make the program code more concise. For example, where a program uses a number of **unsigned short int** variables it would be useful to first create a custom data type with those modifiers, using this declaration:

```
typedef unsigned short int USINT ;
```

Each **unsigned short int** variable declaration can then simply use the custom data type name **USINT** in place of **unsigned short int**.

Hot tip

Although not essential using the recommended cast to explicitly assign values to an enumerated type variable serves as a reminder of its type.

constype.c

1. Begin a new program with a preprocessor instruction to include the standard input/output library functions

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

2. Add a main function that declares and initializes an enumerated set of constants starting at 1

```
int main()
{
  enum SNOOKER
  { RED=1,YELLOW, GREEN, BROWN, BLUE, PINK, BLACK } ;
}
```

3. Next in the main function block, declare and initialize a variable of the defined enum type, then display its value

```
enum SNOOKER pair = RED + BLACK ;
printf( "Pair value: %d \n" , pair ) ;
```

4. Now add a statement to create a custom data type

```
typedef unsigned short int USINT ;
```

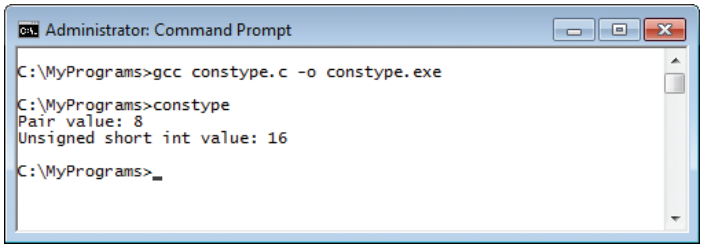
5. Then declare and initialize a variable of the custom data type and display its value

```
USINT num = 16 ;
printf( "Unsigned short int value: %d \n" , num ) ;
```

6. At the end of the main function block return a zero integer value as required by the function declaration

```
return 0 ;
```

7. Save the program file then compile and execute the program to see the value assigned to the enumerated type variable and to the custom data type



Don't forget

Custom data types must be defined in the program before variables of that type can be created.