FIT ICT Software Development

Lecture 5

Setting Constant Values

Lecturer : Charles Tyner

FIT ICT Software Development

Declaring Program Constants

Constant values never change

Declared in same way as variables but using ***const*** keyword:

```
const int MILLION = 1000000
```

Declaration must initialise constant object with value it will contain

Can be used in array declaration if all element values are not to be changed by program

Constant names always in uppercase

FIT ICT Software Development

Declaring Program Constants

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

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

2. Add main function that declares a constant value approximating mathematical value of ***Pi***:

```
int main() {
        constnt float PI = 3.141593;
}
```

FIT ICT Software Development

Declaring Program Constants

3. Insert statements declaring four variables:

```
float diameter;
float radius;
float circ;
float area;
```

4. Insert statements prompting user input to be assigned to a variable:

```
printf("Enter diameter of circle in millimeters: ");
scanf("%d", &diameter);
```

FIT ICT Software Development

Declaring Program Constants

5. Add statements to calculate values for other variables using the constant value and user input:

```
circ = PI * diameter;
radius = diameter/2;
area = PI * (radius * radius);
```

6. Insert statements to output calculated values formatted to two decimal places:

```
printf("\n\tCircumference is %.2f mm", circ);
printf("\n\tThe area is %.2f sqmm\n", area);
```

FIT ICT Software Development

Declaring Program Constants

7. At end of function block return a zero integer

```
return 0;
```

8. Save, compile, execute, study

REMEMBER: the **&** Addressof operator must precede variable name in **scanf()** to assign input to the variable

FIT ICT Software Development

Enumerating Constant Values

Keyword *enum* allows creation of sequence of integer constants

Optionally, declaration can include name for the sequence after keyword

Constant names follow as comma separated list within braces

By default, first constant has value of zero: each following constant has value one greater than the one it follows

enum{MON,TUE,WED,THU,FRI}

WED has an integer value of 2

FIT ICT Software Development

Enumerating Constant Values

Constants can be assigned any individual integer value in the declaration, but next in list will always increment prededing value by one unless it, too, is assigned a value

To begin sequence at one instead of zero, assign one to first constant:

enum{MON=1,TUE,WED,THU,FRI}

WED has an integer value of 3

Enumeration sets may contain duplicate constant values – value zero could be assigned to constants NIL and NONE

FIT ICT Software Development

Enumerating Constant Values

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

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

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

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

FIT ICT Software Development

Enumerating Constant Values

3. In main block, declare an integer variable to store sum of some constant values:

```
int total;
```

4. Insert code to display values of some enumerated constants:

```
printf("\nI potted a red worth %d\n", RED);
printf("Then a black worth %d\n", BLACK);
printf("Followed by another red worth %d\n", RED);
printf("And finally a blue worth%d\n", BLUE);
```

FIT ICT Software Development

Enumerating Constant Values

5. Add statement to calculate sum total of constant values displayed by previous step:

```
total = RED + BLACK + RED + BLUE;
```

6. Add statement to output calculated total:

```
printf("\nAltogether I scored %d\n", total);
```

7. At end of function block return a zero integer

```
return 0;
```

8. Save, compile, execute, study

FIT ICT Software Development

Creating A Constant Type

A declared enumerated sequence can be considered as a new data type

It has properties of specified constant names and associated values

Variables of this *enum* data type can be declared in exactly the same way as variables of other data types

data-type variable-name;

For example, having created data type **enum SNOOKER**, variable named **pair** can be created:

enum SNOOKER pair;

FIT ICT Software Development

Creating A Constant Type

To explicitly assign integer value to variable of enumerated data type, C standard recommends casting to convert **int** data type to **enum** data type:

pair = (enum SNOOKER) 7;

In practice, not necessary, as enumerated values are always integers so equivalent to **int** data type

Variable can also be created by including its name after the final brace at declaration time:

enum BOOLEAN{FALSE, TRUE} flag

FIT ICT Software Development

Creating A Constant Type

Custom data types can be defined using **typedef** keyword

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

Example: where a program uses a number of **unsigned short int** variables, a lot of typing can be saved by creating custom data type with those modifiers:

typedef unsigned short int USINT;

Now, each **unsigned short int** variable can be declared simply using custom data type **USINT**

FIT ICT Software Development

Creating A Constant Type

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

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

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

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

FIT ICT Software Development

Creating A Constant Type

3. In main block, declare and initialise variable of defined enum type and display its value:

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

4. Add statement to create custom data type:

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

5. Declare and initialise variable of custom data type and display its value:

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

FIT ICT Software Development

Creating A Constant Type

6. At end of function block return a zero integer

```
return 0;
```

7. Save, compile, execute, study