Enumerations

Definition

Enumerations are integer data types that you can create with a limited range of values. Each value is represented by a symbolic constant that may be used in conjunction with variables of the same enumerated type.

- Enumerations:
  - Are unique integer data types
  - May only contain a specified list of values
  - Values are specified as symbolic constants

How to Create an Enumeration Type

- Creates an ordered list of constants
- Each label’s value is one greater than the previous label

Syntax

```c
enum typeName {label0, label1, ..., labeln};
```

Where compiler sets \( \text{label}_0 = 0, \text{label}_1 = 1, \text{label}_n = n \)

Example

```c
enum weekday {SUN, MON, TUE, WED, THR, FRI, SAT};
```

Label Values:

\( \text{SUN} = 0, \text{MON} = 1, \text{TUE} = 2, \text{WED} = 3, \text{THR} = 4, \text{FRI} = 5, \text{SAT} = 6 \)

Enumerations

How to Create an Enumeration Type

- Any label may be assigned a specific value
- The following labels will increment from that value

Syntax

```c
enum typeName {label0 = const0, ..., labeln};
```

Where compiler sets \( \text{label}_0 = \text{const}_0, \text{label}_1 = (\text{const}_0 + 1), ..., \text{label}_n \)

Example

```c
enum people {Rob, Steve, Paul = 7, Bill, Gary};
```

Label Values:

\( \text{Rob} = 0, \text{Steve} = 1, \text{Paul} = 7, \text{Bill} = 8, \text{Gary} = 9 \)
Enumerations

How to Declare an Enumeration Type Variable

- Declared along with type:

```
enum typeName {const-list} varname1,...;
```

- Declared independently:

```
enum typeName varName1,...,varNameN;
```

Example

```
enum weekday {SUN, MON, TUE, WED, THR, FRI, SAT}; today;
enum weekday day; // day is a variable of type weekday
```

Enumerations

How to Use an Enumeration Type Variable

- Variables may be declared as type `typeName` without needing the `enum` keyword

```
typedef enum {const-list} typeName;
```

- The enumeration may now be used as an ordinary data type (compatible with `int`)

```
typedef enum {SUN, MON, TUE, WED, THR, FRI, SAT} weekday;
weekday day; // Variable of type weekday
```

Enumerations

How to Declare a ‘Tagless’ Enumeration Variable

- No type name specified:

```
enum {const-list} varName1,...,varNameN;
```

- Only variables specified as part of the `enum` declaration may be of that type
- No type name is available to declare additional variables of the `enum` type later in code

Example

```
enum {SUN, MON, TUE, WED, THR, FRI, SAT} today;
weekday day;
// Variable of type weekday
```

Enumerations

How to Declare an Enumeration Type with `typedef`

```
typedef enum {const-list} typeName;
```

```
typedef enum {SUN, MON, TUE, WED, THR, FRI, SAT} weekday;
weekday day;
// Variable of type weekday
```

Variables may be declared as type `typeName` without needing the `enum` keyword

```
varName = labelN;
```

- The labels may be used as any other symbolic constant
- Variables defined as `enumeration types` must be used in conjunction with the type's labels or equivalent integer

Example

```
enum weekday {SUN, MON, TUE, WED, THR, FRI, SAT};
enum weekday day;
    if (day == WED)
        // May only use values from 0 to 6
```

- Variables defined as `enumeration types` must be used in conjunction with the type's labels or equivalent integer
Lab Exercise 18
Enumerations

Exercise 18
Enumerations

Exercise 18
Enumerations

Exercise 18
Enumerations

Exercise 18
Enumerations

• Open the project’s workspace:

1. Open MPLAB® and select Open Workspace... from the File menu. Open the file listed above.

2. If you already have a project open in MPLAB, close it by selecting Close Workspace from the File menu before opening a new one.

• Compile and run the code:

2. Click on the Build All button.
3. If no errors are reported, click on the Run button.
4. Click on the Halt button.

typedef enum{BANDSTOP, LOWPASS, HIGHPASS, BANDPASS} filterTypes;

filterTypes filter;

/*============================================================================
 FUNCTION:     main()  
============================================================================*/
int main(void)
{
    filter = BANDPASS;
    switch (filter)
    {
    case BANDSTOP: BandStopFilter(); break;
    case LOWPASS: LowPassFilter(); break;
    case HIGHPASS: HighPassFilter(); break;
    case BANDPASS: BandPassFilter(); break;
    }
    while(1);
}
Exercise 18

Conclusions

- Enumerations provide a means of associating a list of constants with one or more variables.
- Make code easier to read and maintain.
- Variables declared as `enum` are essentially still `int` types.

```c
if (day < 1) {
    day = 1;
    if (day > 7) {
        day = 1;
    }
}
```

Questions?

```c
static int i = 0;
static int bln[4] = {0, 0, 0, 0};

bln[i++] = BUTTON_UPSIDE;

"Real" clausal output
```