LCD Lab

RPN calculator - “Debugger” style

Print Stack

2016

State Machine

Transition 2

CMPE 013/L: "C" Programming

Gabriel Hugh Elkaim – Spring 2013
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
Enumerations
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 $label_0 = 0, label_1 = 1, label_n = n$

Example

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

Label Values:
SUN = 0, MON = 1, TUE = 2, WED = 3, THR = 4, FRI = 5, 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 $label_0 = const_0, label_1 = (const_0 + 1), ...$

Example

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

Label Values:
Rob = 0, Steve = 1, Paul = 7, Bill = 8, Gary = 9
Enumerations

How to Declare an Enumeration Type Variable

- Declared along with type:

  Syntax

  ```
  enum typeName {const-list} varname_1,...;
  ```

- Declared independently:

  Syntax

  ```
  enum typeName varName_1,...,varName_n;
  ```

Example

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

How to Declare an Enumeration Type with `typedef`

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

**Syntax**

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

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

**Example**

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

Enumerations

How to Use an Enumeration Type Variable

If enumeration and variable have already been defined:

**Syntax**

```
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;

day = WED;
day = 6;               //May only use values from 0 to 6
if (day == WED)
    { ... }
```
Lab Exercise 18

Enumerations

Exercise 18
Enumerations

• Open the project’s workspace:

On the class website

/Examples/Lab18.zip -> Load “Lab18.mcw”

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

If you already have a project open in MPLAB, close it by selecting **Close Workspace** from the **File** menu before opening a new one.
Exercise 18
Enumerations

• 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.

#define 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

Questions?