Tag-names/Enumeration

Enumeration constants - use

Example: use for readability

```c
enum color { red, yellow, green, blue };  // readable
color screenbkgrnd
if (screenbkgrnd == green) { action };  // readable
if (screenbkgrnd == 2) { action };     // a mystery
```

Initializing variables - enum or define

```c
#define idSize    7;
#define nameSize 30;
enum {
    nameSize = 30};
class Student {
private:
    char id[idsize+1];
    char name[nameSize+1];};
```

Enumeration list examples:
```c
enum colors(white, black, orange, pink);
// default values - white = 0, black = 1, etc.
enum colors(white = 1, black, orange = 5, pink);
// values - white = 1, black = 2, orange = 5, pink = 6
```

Tag-names are not associated just with enumeration constants

Syntax examples:
```c
enum tag-name { enum list };
class tag-name { member list };
```

FString - a real, useful example
Composites - initialization sequence

Example:
```cpp
class Point;
class Rectangle {
public:
    Rectangle(int uplfX = 0, int uplfY = 0, int lwrtX = 0, int lwrtY = 0);
private:
    Point uplft, lwrgt; // constructed first
};
```

Important - order which constructor parameters are used
⇒ all members (including classes) are constructed before the class is constructed
⇒ the class is destroyed before all members are destroyed

Composites - initialization syntax

Alternative Initialization - Examples:
```cpp
class Point {
public:
    Point(int x, int y); // constructor
private:
    int xVal, yVal;
};
```

// initialization code - one way
Point::Point(int x, int y) { xVal = x; yVal = y;
    • • • // rest of initialization }

// initialization code - alternative
Point::Point(int x, int y) : xVal(x), yVal(y)
    (• • • // rest of initialization );
```

Static members in a class

Example:
```cpp
class Point {
public:
    Point() { numPoints++; }; // default constructor
    ~Point() { numPoints--; }; // destructor
private:
    static short numPoints;
    float xVal, yVal;
};
```

Recall: "Static" has many different meanings in C++
Strings - as in C

- Basic structure - char buffer[ML]; // an array of characters
- No operations, no overflow check, etc.
- String - char array term. with a NULL char - "\0"
- String utility in library <string.h>
- String: copy; concat.; compare; search; tokens; etc.

Example:
```c
char name[] = "My name is Jane"; // string created
char newName[30]; // just a buffer of chars
char* stgPtr; // pointer to character
newName = name; // error - no copy. newName loc fixed
strcpy(newName, name); // ok
stgPtr = name; // okay assigning a pointer
stgPtr = &newName[1]; // okay now points to 2nd char
```

Strings in C++ - FString a useful example

- Number of string class utilities - adopt one for your use
- FString simple, easy to use, does things you want
- Constructs from C or FString strings
- Assign and Append from C or FString strings
- Ops - compare, less than, equal, greater than - add !=
- Future - add operator functions, +, =, ==, >, <, etc.
- Return C string pointer (const) - use for C string ops.
- I/O operators >>, <<
- Fixed length - 255 (protected) - future extend with pointers

Links:
- FString casestudy
- header code
- source code
- test program

page 3