Enumerations

Suppose I have 3 colors of paint…

- RED
- GREEN
- BLUE

How do I represent these colors in C?

- I could put the color name in a string…

```
char can1[6];

strcpy(can1,”green”);
if (0==strcmp(can1,”red”) printf(“First can is red paint”);
```

- Takes lots of space
- Not very clear what is going on

How do I represent these colors in C?

- I could use the first letter… R/G/B

```
char can1;

can1='G';
if (can1=='R') printf ("First can is red paint");
```

- Takes less space
- Not very clear what is going on – what if I have Grey and Black?

How do I represent these colors in C?

- I could use a number->color mapping… 0=red, 1=green, 2=blue

```
char can1;
can1=1;
if (can1==0) printf (“First can is red paint”);
```

- Takes less space
- Not very clear what is going on – was green 2 or 3?

C has “Enumerations”

- Special C construct to make code clear
- An enumeration is any finite list of items

```
enum colors {
        red,
        green,
        blue
};
```

C Enumeration Definition

- Defines a new data type
- Defines constant values of that type

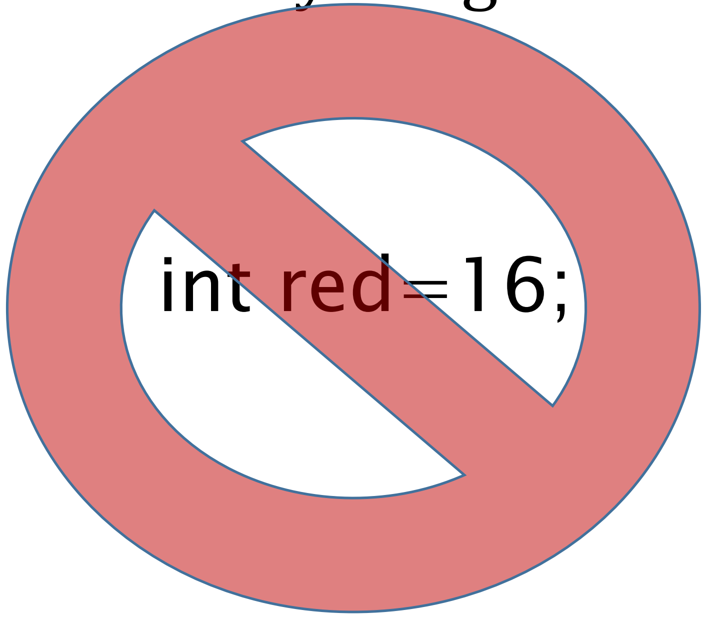
```
enum colors {
    red,
    green,
    blue
};
```

Define a new type called enum colors

Define 3 constants of type enum colors with anonymous values

Enumeration Constants

- Enumeration constants are variable names
- Cannot use these names for anything else



```
int red=16;
```

C Enumeration Declaration

- Create a variable with an enumerated type

Type specification
enum colors must be defined above

```
enum colors can1;
```

Variable name

- Variable can have any value as long as it's a color enumeration constant.

```
can1=green;
```

Putting it all together

```
enum colors {red, green, blue} can1;

can1=green;

if (can1==red) printf("First can is red paint");
```

Why enums?

- Makes reading the code crystal clear
- Compiler can check to make sure everything is correct

```
can1=orange; // compiler error – orange not a valid color
```

- Space efficient
- Easy to extend

Problem: Using enums in messages

```
printf("The value of can1 is %d",can1);
```

```
The value of can1 is 5
```

- Problem… enum values don't really make sense to people!

Solution: Provide a function to translate

```
char * colorName(enum colors inc) {
	switch(inc) {
		case red : return “red”;
		case green: return “green”;
		case blue: return “blue”;
	}
	return “unknown”;
}
```

Printing with a translator function

```
printf("First can has %s paint\n",colorName(can1));
```

```
First can has green paint
```

- Note… update translator function when enum changes!

Enums with Specified Values

```
enum colors {
        red = 14,
        green = 12,
        blue = 15
};
```

- red, green, and blue are still constants, but now they have specific values
- default values are 0,1,2 …

Why specific values?

```
enum escapes={
        BELL=‘\a’,
        BACKSPACE=‘\b’,
        TAB=‘\t’,
        NEWLINE=‘\n’,
        RETURN=‘\r’,
        VTAB=‘\v’
};
```

Resources

- Programming in C, Chapter 13 (Enumerated Data Types)
- Wikipedia Enumerated Type https://en.wikipedia.org/wiki/Enumerated_type
- Enumeration Tutorial http://www.programiz.com/c-programming/c-enumeration
- Example Code http://www.cs.binghamton.edu/~tbarten1/CS211_Fall_2015/examples/xmp_enum/