HIGH-LEVEL PROGRAMMING I

Enumerations

by Prasanna Ghali

Confusing use of integers in code

- Quite often, you might want to assign integer codes to different items in your program, e.g.,

```
int month; // Jan = 1, Feb = 2,...
month = 5; // May
```

- Someone reading your program that does not know your integer code will be confused, e.g.,

```
int team;   // Ferrari = 1,
            // McLaren = 2,...
team = 6;   // What does this mean?
```

- C *enumeration types* do this in a better way

Declaring enumerations

In enumeration declaration, identifiers or *enumerators* given for each possible value that enumeration type can contain

②

Enumeration specifies set of *integer* values of type `int`
Enumeration declarations have similar syntax to structure, only difference is use of `enum` keyword

①

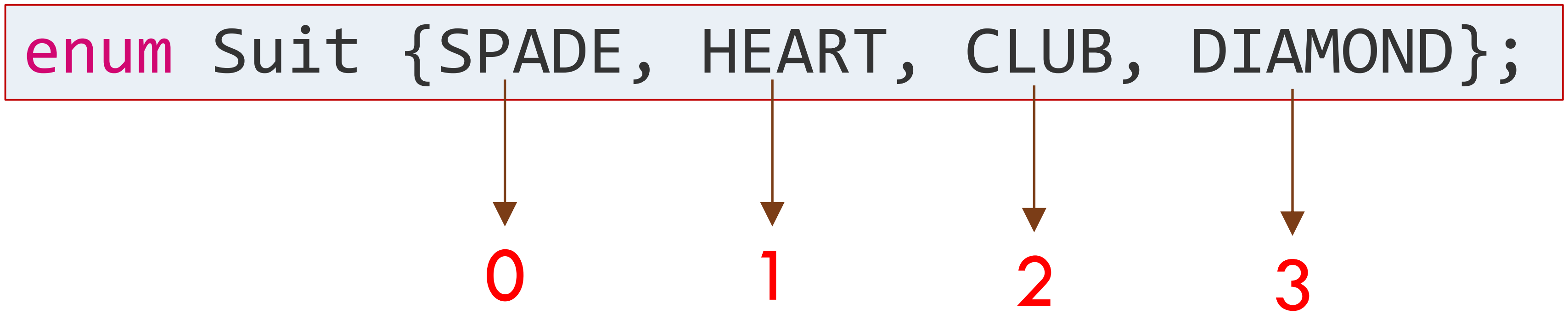
```
enum Team {
   FERRARI, MCLAREN,
   BMW, WILLIAMS,
   RENAULT, TOYOTA
};

enum Team my_team = BMW;
```

`my_team` is variable of type `enum Team` and is initialized with value `BMW`

③

Enumerators and values (1/3)

- By default, enumerators are assigned values 0, 1, 2, … in order



```
enum Suit {SPADE, HEART, CLUB, DIAMOND};
```

Enumerators and values (2/3)

- Enumerators can be explicitly specified values:

```
enum Suit {
  SPADE = 4, HEART = 3,
  CLUB = 2, DIAMOND = 1
};
```

Enumerators and values (3/3)

- Unspecified values are assigned value of previous member plus one

If 1st enumerator value is unspecified, by default, it has value of zero

```
enum Suit {
  SPADE, HEART = 8,
  CLUB = 2, DIAMOND
};
```

DIAMOND has value 3 - value of previous member SPADE plus one

Enumerations: Use cases (1/3)

- Since enumerators are `int`s, you can use `enum` variable anywhere an `int` is legal:

```
enum Suit {SPADE, HEART, CLUB, DIAMOND};
enum Suit s = CLUB;

int i = DIAMOND; // i is 3
s = SPADE;       // s is 0 (SPADE)
s++;             // s is 1 (HEART)
i += s;          // i is 4
```

Enumerations: Use cases (2/3)

```
enum Fish { trout, bass, carp, salmon };

enum Fish myfish = bass;
if (myfish == trout)
  grill_fish(myfish);
else
  bake_fish(myfish);
```

Enumerations: Use cases (3/3)

Enumerators are compile-time constants and therefore can be used to define array sizes.
This is preferable to preprocessor macros!!!

```
// not preferred
#define ARRAYSIZE 10
int arr[ARRAYSIZE];
```

```
// preferred!!!
enum {ARRAYSIZE = 15};
int arr[ARRAYSIZE];
```

Unnamed enumerations

Unnamed enum is used when all we need is set of integer constants, rather than a type for defining integer variables

②

enum declaration need not have enumeration tag

①

```
enum {trout = 2,  bass   = 5,
      carp  = 10, salmon = 15};

int myfish = carp;
if (myfish == trout)
  grill_fish(myfish);
else
  bake_fish(myfish);
```

Summary

- Enumeration can be used to give identifiers to integer codes
- Enumeration type variable is `int` and can be used in similar ways
- Type qualifier `const` and `enum` type can satisfy all symbolic constant operations for which `#define` might be used