Enum type

enumeration type (enum)
A user defined data type whose domain is an ordered set of literal values expressed as identifiers.

Examples:
(1) enum Days {SUN, MON, TUE, WED, THU, FRI, SAT};

notes: the identifiers are ordered : SUN < MON < TUE … < SAT
the default values for the identifiers are: SUN=0, MON=1, …SAT=6, (but the values can be changed if necessary)

(2) enum Vowel {'A', 'E', 'I', 'O', 'U'}; // wrong!! Why?

(3) enum Animals {CAT, DOG, BIRD, HORSE, SHEEP, TIGER, LION};
Animals firstAnimal, secondAnimal, thirdAnimal;

```
// assignment statements
firstAnimal = CAT;
secondAnimal = DOG;
thirdAnimal = firstAnimal;
firstAnimal = 0;   //wrong!
secondAnimal = 30;   // wrong!

// increment
firstAnimal = static_cast<Animals>(firstAnimal + 1);
```

enum used in switch statement:

```
switch  (firstAnimal)
{
case    CAT:    …
                break;
case    DOG:    …
                break;
case    BIRD:   …
                break;
case    HORSE: …
                break;
case    SHEEP: …
                break;
case     LION:  …
                break;
case     TIGER: …
                break;
}
```

enum used in array subscripts

(1)
```
Animals     oneAnimal;
float       weights[7];

for (oneAnimal = CAT;  oneAnimal <=TIGER; oneAnimal=static_cast<Animals>(oneAnimal+1))
      cout   << "The average weight for this animal is " << weights[oneAnimal] << endl;
```

(2)
```
const int   NUM_COLORS=5;
```

```
const int   NUM_MAKERS=5;

enum Color {RED, ORANGE, GREY, WHITE, BLACK};
enum Maker {TOYOTA, HONDA, BMW, JAGUAR, NISSAN};

float   crashRating[NUM_MAKERS][NUM_COLORS];

crashRating[TOYOTA][GREY] = 0.87;
…
crashRating[HONDA][BLACK] = 0.18;
```

typedef : define new data type names (give another name to existing, or newly created, data type)

Examples :

(1)
```
typedef   float     balance;
balance  saving, checking;
```

(2)
```
struct    employee
{
        int     id;
        char  name[ARRAY_SIZE];
        char  gender;
        int     numDependents;
        float  payRate;
};
typedef  struct  employee    EmployeeType;
EmployeeType    teachers[500];
```

Equivalent form:
```
typedef   struct    employee
{
        int     id;
        char  name[ARRAY_SIZE];
        char  gender;
        int     numDependents;
        float  payRate;
}   EmployeeType;
EmployeeType   chairman;
```

(3)
```
typedef   float  ClassScores[20] ;
ClassScores     test1, test2;
```