Object Oriented Programming

Using C++ Programming Language

RECAP

Information hiding

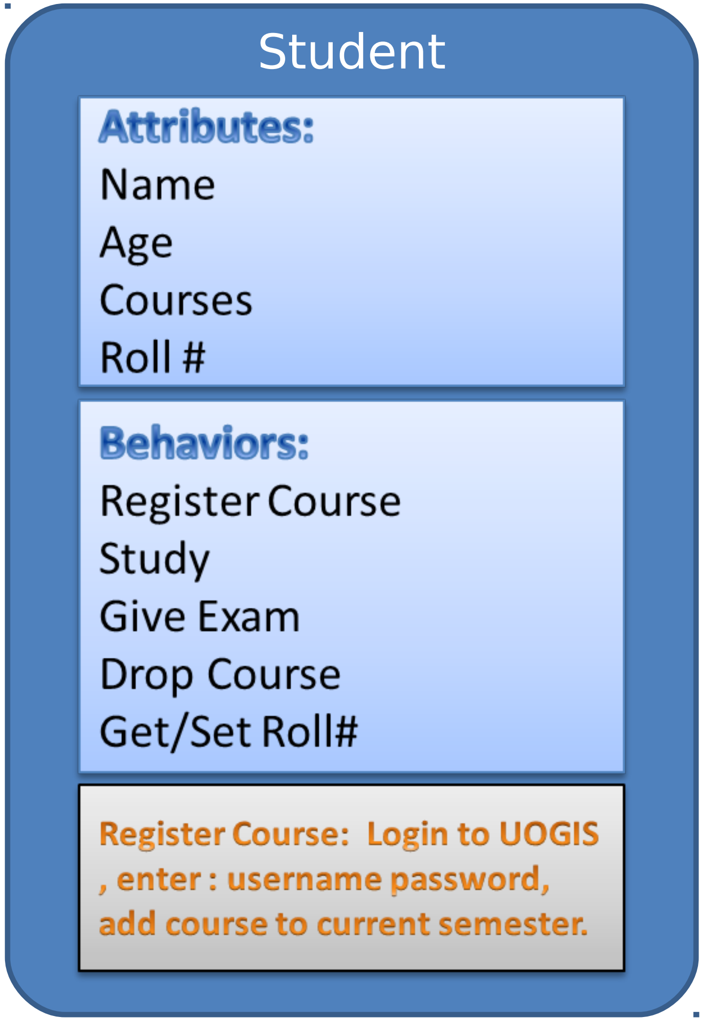
Encapsulation

Implementation

Interface

Messages

Abstraction



Student

Attributes:
Name
Age
Courses
Roll #

Behaviors:
Register Course
Study
Give Exam
Drop Course
Get/Set Roll#

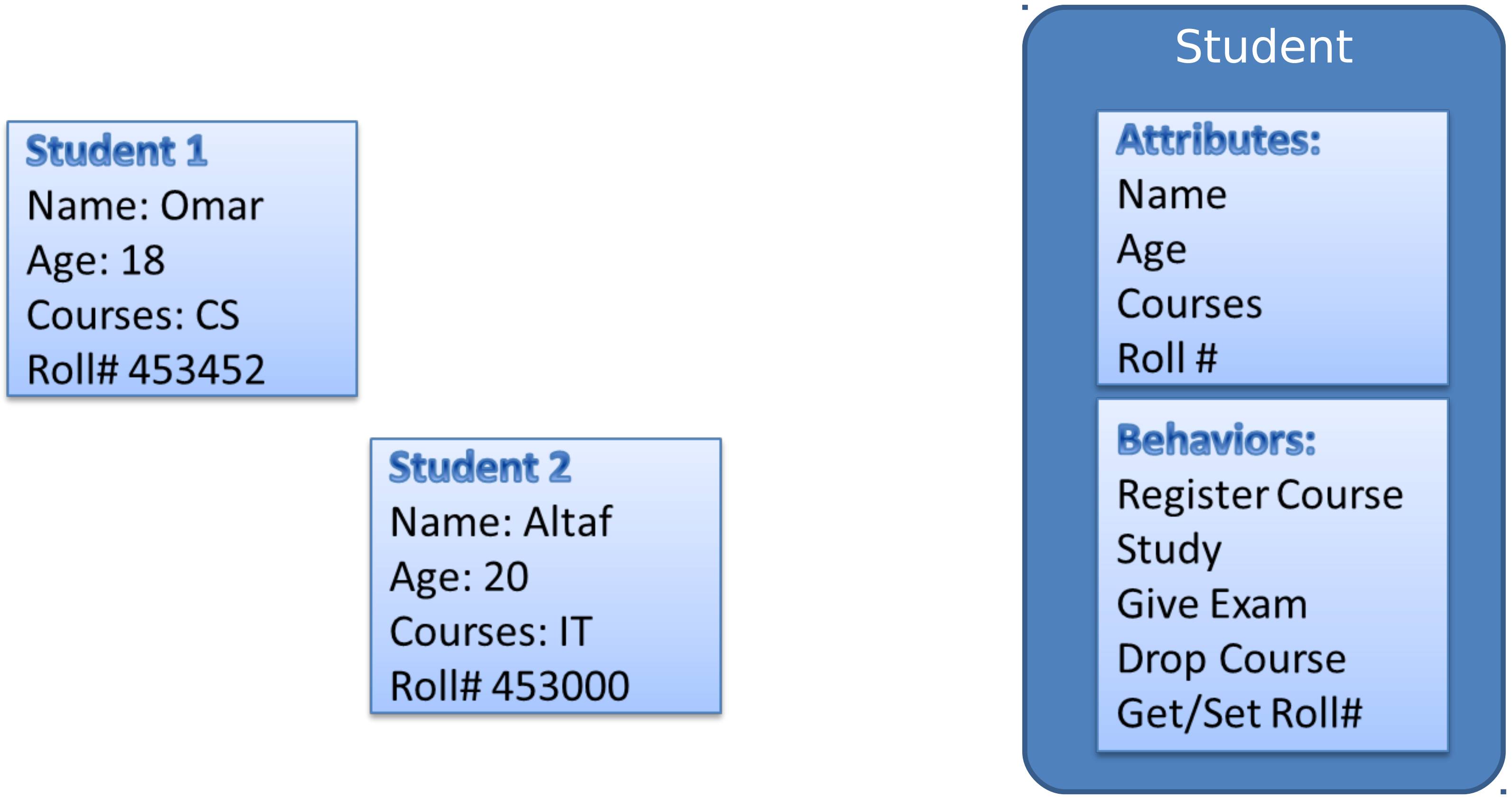
Register Course: Login to UOGIS , enter : username password, add course to current semester.

Lecture # 3

Classes, Abstract Data Types, Comparison with Structures

Classes

Objects having same data and behavior belong to same class.



Student 1
Name: Omar
Age: 18
Courses: CS
Roll# 453452

Student 2
Name: Altaf
Age: 20
Courses: IT
Roll# 453000

Student

Attributes:
Name
Age
Courses
Roll #

Behaviors:
Register Course
Study
Give Exam
Drop Course
Get/Set Roll#

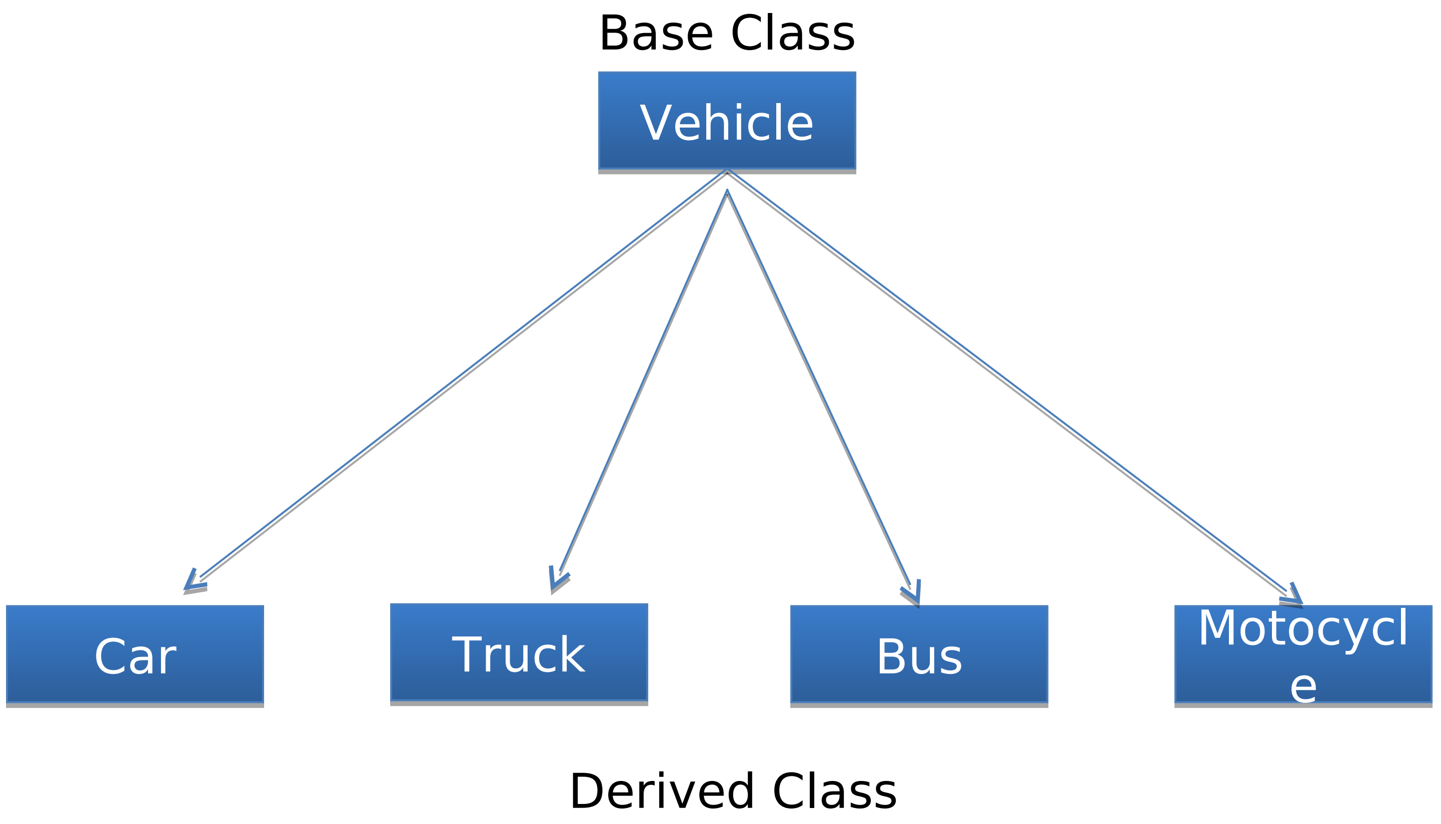
Example – Class

- Circle is an closed oval shape
- Square is a closed rectangular shape
- Line is open shape with start and ending point

- Each one is a shape
- We say these objects are *instances* of the shape class

Inheritance

- Idea of classes leads to idea of inheritance.
- Classes can be divided into subclasses



Reusability

- Inheritance extends reusability
- Subclasses can add more functionality to the base class with out modifying it.

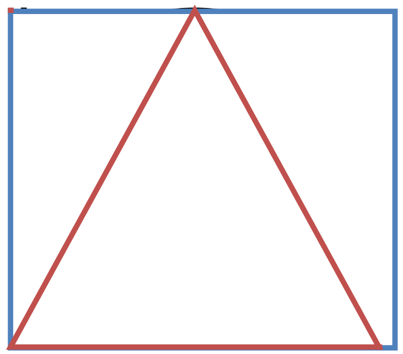
Example:

A Class TextFileReader can be extended to an XMLFileReader subclass.

Polymorphism

- 1+3 = 4
- D1 + D2 = D3
  - {D1, D2 and D3 are Distance: Int Feet Float Inches }

- Shape.Draw()



Overloading

- If we use Operators to achieve Polymorphism it is called operator Overloading

  + , -, =, >

Similarly if we use functions from Base Class for exhibiting polymorphic behavior in subclasses it will be called function overloading.

In this section we will learn how to define custom data types and their usage

STRUCTURES AND ENUMERATIONS

User defined Data Types

- Primitive data types
  - Int, float, char, double, bool etc.
- Composite Data Types
  - Date, Pencil Box, Words in a Sentence.

"Such data type in C++ perspective known as structures"

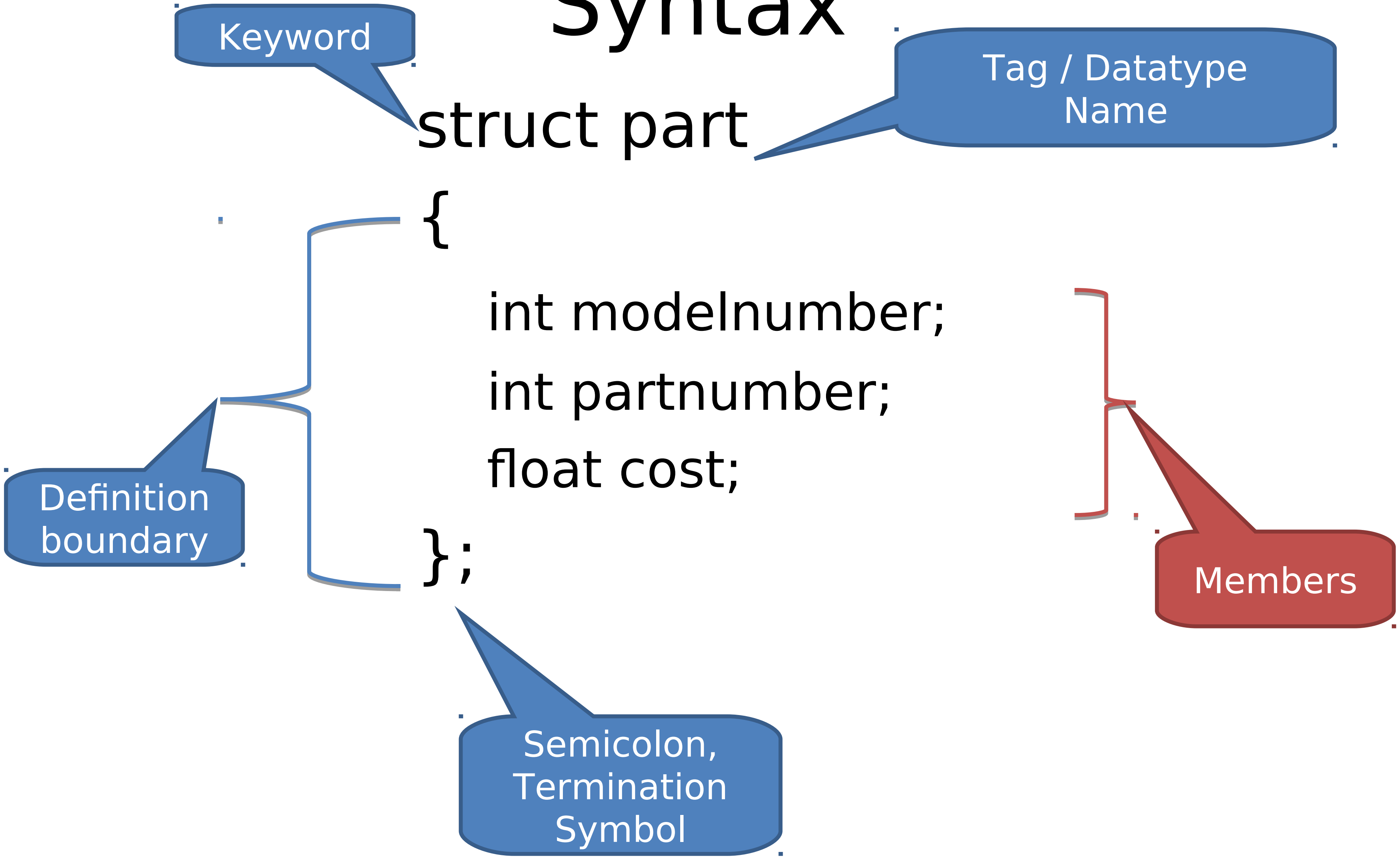
Structures

- A collection of variables
- Variables may be Same in Type , Differ in Type but store different information
- Data item are known as members of structure.
- Almost identical to Class
- Usually used to store only Data Information.
- Classes stores both data and functionality

```
// parts.cpp
// uses parts inventory to demonstrate structures
#include <iostream>
using namespace std;
////////////////////////////////////////////////////////////////
struct part                     //declare a structure
   {
   int modelnumber;             //ID number of widget
   int partnumber;              //ID number of widget part
   float cost;                  //cost of part
   };
////////////////////////////////////////////////////////////////
int main()
   {
   part part1;                  //define a structure variable

   part1.modelnumber = 6244;    //give values to structure members
   part1.partnumber = 373;
   part1.cost = 217.55F;
                                //display structure members
   cout << "Model "    << part1.modelnumber;
   cout << ", part "   << part1.partnumber;
   cout << ", costs $" << part1.cost << endl;
   return 0;
   }
```

Defining a Structure / Syntax



```
struct part
{
    int modelnumber;
    int partnumber;
    float cost;
};
```

Defining a structure variable

- Similarly as we define int or

A float variable

- Type varName;

  e.g. Part partA;

```
struct part
{
        int modelnumber;
        int partnumber;
        float cost;
};
```

Memory
…
10101010
10101010
10101010
10101010
11010101
11010101
11010101
11010101
00011011
00011011
00011011
00011011
…

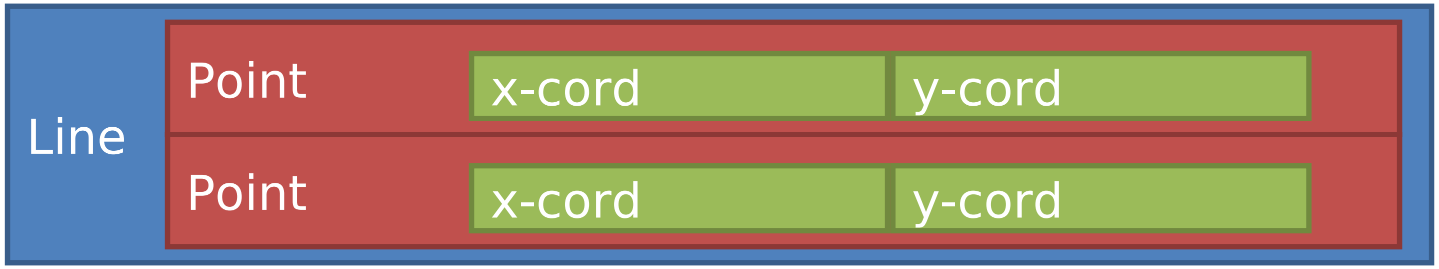
Complex Structures

- Structures within structure

```
struct  point
{
int x-cord;
int y-cord;
};
```

- Accessing members variables
  - Line LineX;
  - Cout << LineX.B.x-cord;

```
struct  Line
{
Point A;
Point B;
};
```



Initialization of structure

- Defination only
  - Line LineAB; {not recommended}
- Empty or Zero valued
  - Line LineAB = {0}; {recommended}
- Initialization with values
  - Line LineAB = {{5,2},{6,4}};
- Assigning value direct to member variable
  - LineAB.PointA.y-cord = 5;

Structures and Classes

- Structures are usually used to hold data only
- classes hold both data and functions

- In C++ structures can in fact hold both data and functions

- Major difference between class and a structure is that in a class members are private by default while in a structure they are public by default.

Enumerations

- *Enumeration* different approach to defining your own data type.
- We can write a perfectly fine program without enumeration but enumeration provides a lot of ease in programming by simplify and clarifying programming code.

- *Enumerated types works when you know in advance a finite list of values*.

Enumerations Example

```
#include <iostream>
using namespace std;
//specify enum type
enum days_of_week { Sun, Mon, Tue, Wed, Thu, Fri, Sat }
int main()
{
days_of_week day1, day2;   //define variables   of type days_of_week
day1 = Mon;                  //give values to
day2 = Thu;                  //variables
int diff = day2 - day1;    //can do integer arithmetic
cout << “Days between = “ << diff << endl;
if(day1 < day2)              //can do comparisons
cout << “day1 comes before day2\n”;
return 0;
}
```

Clarity of code with enums

```
int day1,
switch (day1)
{
      case 1:
      ....
      break;

      case 2:
      ....
      break;

      case 3:
      ....
      break;
......
}
```

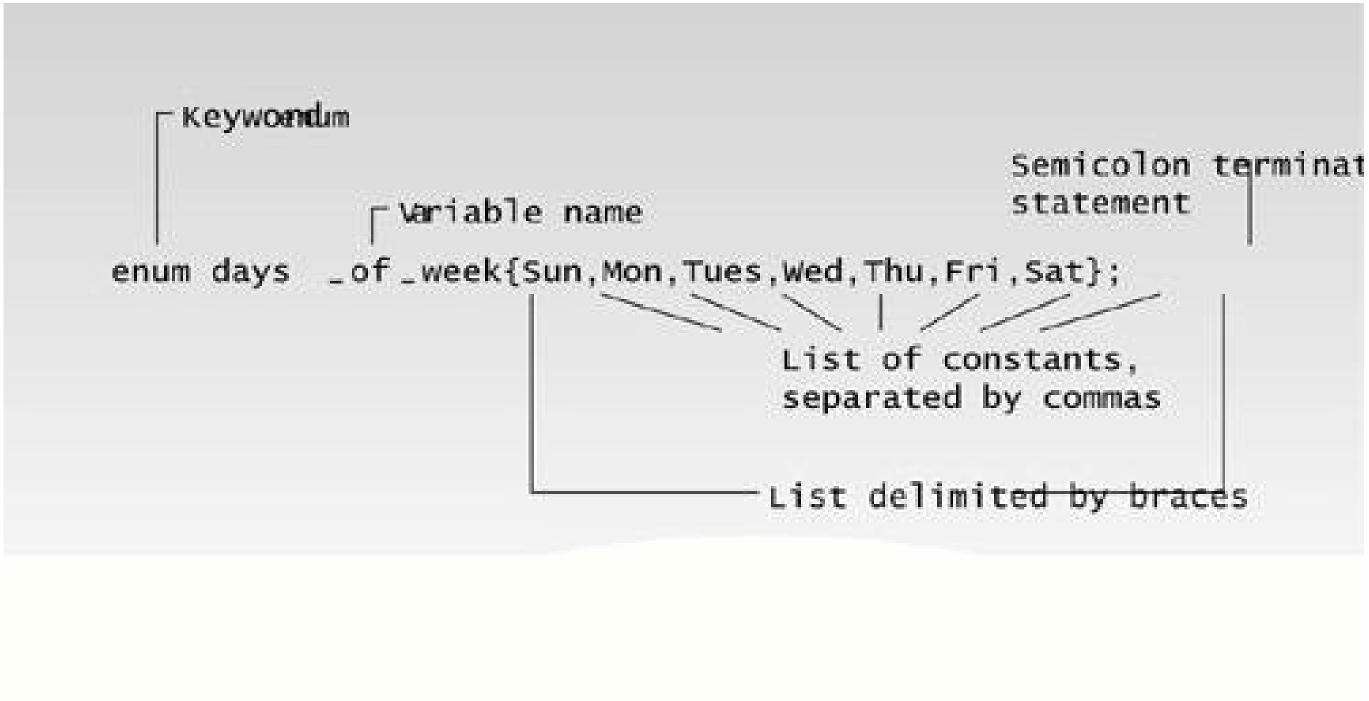
```
days_of_week day1,
switch (day1)
{
      case SAT:
      ....
      break;

      case SUN:
      ....
      break;

      case MON:
      ....
      break;
......
}
```

Syntax of declaring Enums

- **Enum** declaration defines the set of all names that will be permissible values of the type.
- Permissible values are called enumerators.


Keyword enum
Variable name
Semicolon terminat
statement
enum days _of _week{Sun,Mon,Tues,Wed,Thu,Fri,Sat};
List of constants, separated by commas
List delimited by braces

Enumeration

- Enumerations internally treated as integers.
- By default in enum declaration, the first enumerator's integer value is 0, the second value 1, and so on.
- We can also change the starting index or number of enums. e.g.

```
enum Suit { clubs=1, diamonds, hearts, spades };
```

Another Example

```
enum direction { north, south, east, west };
direction dir1 = south;
cout << dir1;
```

C++ I/O treats variables of enum types as integers, so the output would be 1

Q & A