LECTURE-17

Type Conversions

In a mixed expression constants and variables are of different data types. The assignment operations causes automatic type conversion between the operand as per certain rules.

The type of data to the right of an assignment operator is automatically converted to the data type of variable on the left.

Consider the following example: int x; float y = 20.123;
x=y ;

This converts float variable y to an integer before its value assigned to x. The type conversion is automatic as far as data types involved are built in types. We can also use the assignment operator in case of objects to copy values of all data members of right hand object to the object on left hand. The objects in this case are of same data type. But of objects are of different data types we must apply conversion rules for assignment.

There are three types of situations that arise where data conversion are between incompatible types.

1. Conversion from built in type to class type.
2. Conversion from class type to built in type.
3. Conversion from one class type to another.

Basic to Class Type

A constructor was used to build a matrix object from an int type array. Similarly, we used another constructor to build a string type object from a char* type variable. In these examples constructors performed a defacto type conversion from the argument's type to the constructor's class type

Consider the following constructor:

```
string :: string (char*a)
{
length = strlen (a);  name=new char[len+1];  strcpy (name,a);
}
```

This constructor builds a string type object from a char* type variable a. The variables length and name are data members of the class string. Once you define the constructor in the class string, it can be used for conversion from char* type to string type.

Example

```
string si   , s2;
```

```
char* namel =  “Good Morning”;  char* name2 =  “ STUDENTS” ;
s1 = string(namel);  s2 = name2;
```

The program statement

```
si = string (namel);
```

first converts name 1 from char* type to string type and then assigns the string type values to the object s1. The statement

```
s2  = name2;
```

performs the same job by invoking the constructor implicitly.
Consider the following example class time

```
{ int hours;  int minutes; public:  time (int t) // constructor
{ hours = t  /  60;   //t  is  inputted  in minutes minutes  =   t   %  60;
}
        };
```

In the following conversion statements :

```
time Tl;                    //object  Tl  created
int period =  160;
Tl  =  period;  //int   to  class   type
```

The object Tl is created. The variable period of data type integer is converted into class type time by invoking the constructor. After this conversion, the data member hours ofTl will have value 2 arid minutes will have a value of 40 denoting 2 hours and 40 minutes.

Note that the constructors used for the type conversion take a single argument whose type is to be converted.

In both the examples, the left-hand operand of = operator is always a class object. Hence, we can also accomplish this conversion using an overloaded = operator.