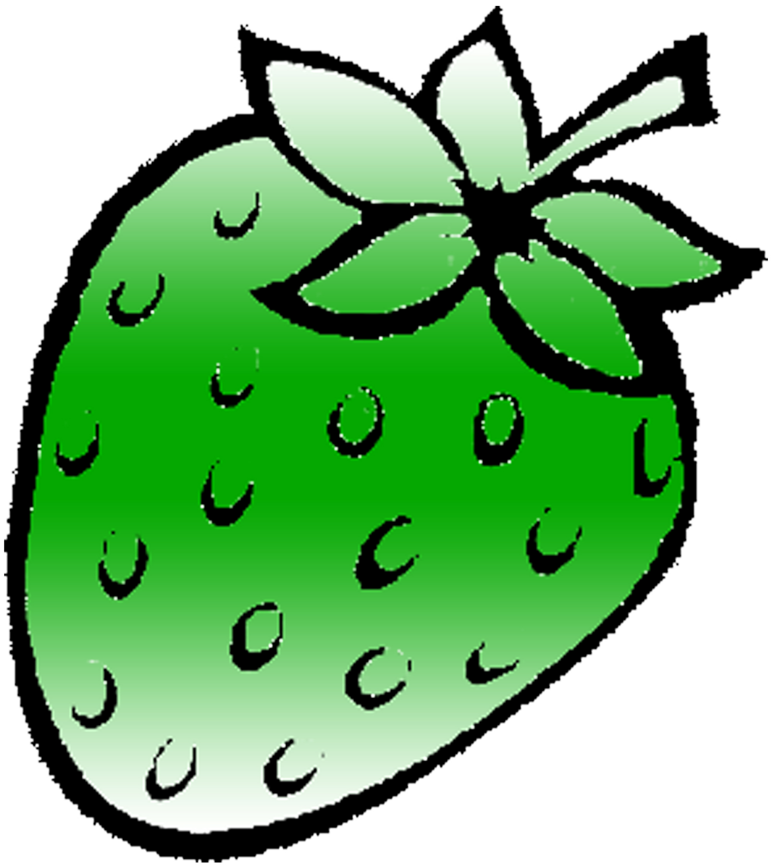
STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com

Experiment No. 2

PART A

(PART A: TO BE REFFERED BY STUDENTS)

A.1 Aim: To understand basic concepts of class as below:

1. Objects
2. Class
3. Data Members
4. Access Specifiers
5. Scope resolution operator
6. Member functions and Friend functions

P1: Modify P1 program of practical no 1, Write a class called *DM* with data members *meter* and *centimeter*. The class must include member functions *1. convert* for converting *meter* to *centimeters* and a *display* function to display the data of object.

P2: Modify P1 (practical 2); add a class *DB* to store distance in *feet* and *inches* (Data Members) and below member functions:

1. *Convert* – For converting feet into inches
2. *Display*- For displaying data of objects

P3: Modify P2 (practical 2); Write a program that can read values for the class objects and add one object of DM with another object of DB. Use friend function to carry out the addition operation.

A.2 Prerequisite:

Knowledge of using input/output statements in C++.

A.3 Outcome:

After successful completion of this experiment students will be able to

1. Know use of classes and objects.
2. Know how to use function for solving various problem.
3. Understand the use of access specifiers and scope resolution operator.

A.4 Theory:

Objects, Classes, Data members and Member functions:

Classes are an expanded concept of *data structures*: like data structures, they can contain data members, but they can also contain functions as members.

An *object* is an instantiation of a class. In terms of variables, a class would be the type, and an object would be the variable.

Classes are defined as below:

```
class class_name {

  access_specifier_1:

    member1;

  access_specifier_2:

    member2;

  ...

} object_names;
```

Where class_name is a valid identifier for the class, object_names is an optional list of names for objects of this class. The body of the declaration can contain *members*, which can either be data or function declarations, and optionally *access specifiers*

For example:

```
class Rectangle {
    int width, height;
  public:
    void set_values (int,int);
    int area (void);
} rect;
```

Declares a class (i.e., a type) called Rectangle and an object (i.e., a variable) of this class, called rect. This class contains four members: two data members of type int (member width and member height) with *private access* (because private is the default access level) and two member functions with *public access*: the functions set_values and area, of which for now we have only included their declaration, but not their definition.

Access Specifiers:

An *access specifier* is one of the following three keywords: private, public or protected. These specifiers modify the access rights for the members that follow them:

- private members of a class are accessible only from within other members of the same class (or from their *"friends"*).
- protected members are accessible from other members of the same class (or from their *"friends"*), but also from members of their derived classes.
- Finally, public members are accessible from anywhere where the object is visible.

Scope resolution operator:

Scope resolution operator (::) is used to define a function outside a class. A function can be defined within or outside class. When a function is declared within class, it can be defined outside class using scope resolution operator.

Syntax:

```
class Square{
  int x;
public:
 void set_value(int b)
{ x=b; }
  void find_square();
}
void Square: find_square()
{
x=x*x;
}
void main()
{
Square s;
s.set_value(5);
s.find_square();
}
```

Friend Function:

In principle, private and protected members of a class cannot be accessed from outside the same class in which they are declared. However, this rule does not apply to*"friends"*.

Friends are functions or classes declared with the friend keyword.
A non-member function can access the private and protected members of a class if it is declared a *friend* of that class. That is done by including a declaration of this external function within the class, and preceding it with the keyword friend:

A C++ friend functions are special functions which can access the private members of a class. They are considered to be a loophole in the Object Oriented Programming concepts, but logical use of them can make them useful in certain cases. For instance: when it is not possible to implement some function, without making private members accessible in them.

Friend functions have the following properties:

- 1) Friend of the class can be member of some other class.
- 2) Friend of one class can be friend of another class or all the classes in one program, such a friend is known as GLOBAL FRIEND.
- 3) Friend can access the private or protected members of the class in which they are declared to be friend, but they can use the members for a specific object.
- 4) Friends are non-members hence do not get “this” pointer.
- 5) Friends, can be friend of more than one class, hence they can be used for message passing between the classes.
- 6) Friend can be declared anywhere (in public, protected or private section) in the class.

Syntax:

```
class square
{
int x;
public:
set_value(int b)
{x=b; }
friend void find_square(int)
}
void find_square(square a)
{
  a.x=a.x*a.x
}
void main
```

```
{
square s;
s.set_value(5);
find_square(s);
}
```

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

Roll No. N008	Name: Akshay Banda
Program: MBA TECh CS	Division: C
Semester: 2	Batch : C1
Date of Experiment: 14/1/15	Date of Submission: 14/1/15
Grade :	

B.1 Software Code written by student:

(Paste your C++ code completed during the 2 hours of practical in the lab here)

1.
```
#include<iostream>
using namespace std;

class DM
{
private:
   double m,cm;
public:
   void input()
   {
      cout<<"Enter distance in meter : ";
      cin>>m;
   }
   void convert()
   {
      cm=m*100;
   }
   void display()
   {
      cout<<"Distance is "<<cm<<" cm";
   }
};

int main()
{
   DM d;
   d.input();
   d.convert();
   d.display();
   return 0;
}
```

2.
```
#include<iostream>
using namespace std;

class DB
{
private:
    double f,i;
public:
    void input()
    {
        cout<<"Enter distance in feet : ";
        cin>>f;
        cout<<"Enter distance in inches : ";
        cin>>i;
    }
    void convert()
    {
        i=i+(f*12);
    }
    void display()
    {
        cout<<"Distance in inches is "<<i<<"inches";
    }
};

int main()
{
    DB d;
    d.input();
    d.convert();
    d.display();
    return 0;
}
```

3.

```
#include<iostream>
using namespace std;

class DB;

class DM
{
private:
    double m,cm;
public:
    void input()
    {
        cout<<"Enter distance in meter : ";
        cin>>m;
    }
    void convert()
    {
        cm=m*100;
    }
    void display()
    {
        cout<<"Distance is "<<cm<<" cm";
    }
    friend void sum(DM a, DB b);
};

class DB
{
private:
    double f,i,c;
public:
    void input()
    {
        cout<<"Enter distance in feet : ";
        cin>>f;
        cout<<"Enter distance in inches : ";
        cin>>i;
    }
    void convert()
    {
        i=i+(f*12);
        c=i*2.54;
    }
    void display()
    {
        cout<<"Distance in inches is "<<i<<"inches";
    }
    friend void sum(DM a, DB b);

};
```

```
    void sum(DM a, DB b)
    {
       cout<<"Result is "<<(a.cm+b.c)<<" cm";
    }

int main()
{
    DM d;
    d.input();
    d.convert();
    DB b;
    b.input();
    b.convert();
    sum(d,b);
    return 0;
}
```

B.2 Input and Output:

(Paste your program input and output in following format. If there is error then paste the specific error in the output part. In case of error with due permission of the faculty extension can be given to submit the error free code with output in due course of time. Students will be graded accordingly.)

1.
```
Enter distance in meter : 7
Distance is 700 cm
```

2.
```
Enter distance in feet : 7
Enter distance in inches : 7
Distance in inches is 91inches
```

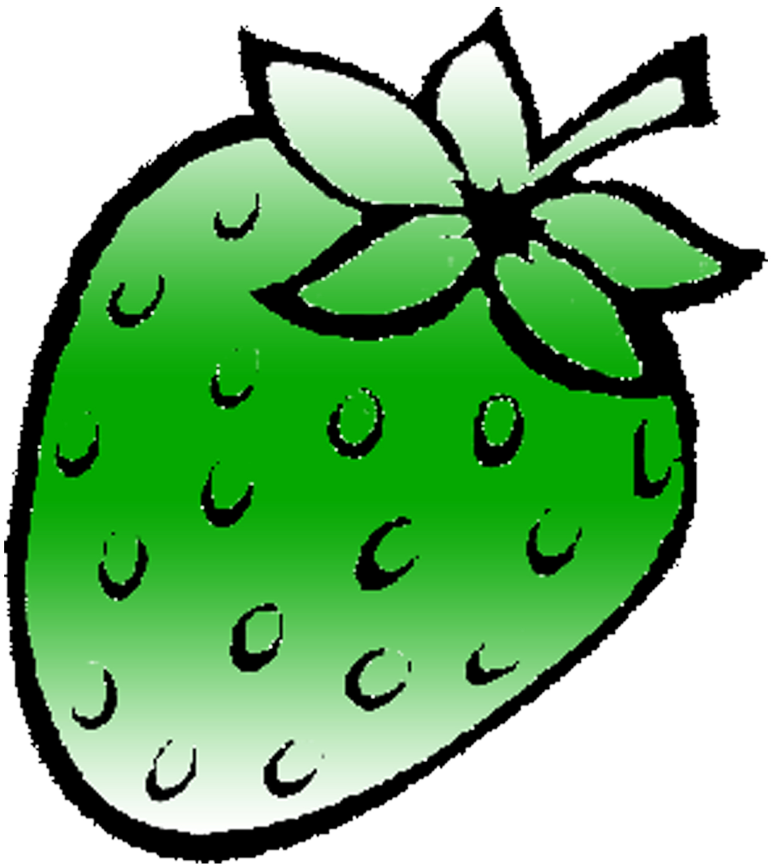
3.
```
Enter distance in meter : 7
Enter distance in feet : 7
Enter distance in inches : 7
Result is 931.14 cm
```

B.3 Conclusion:

(Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.1)

I learned to work with classes and objects and friend function;

STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com