IOE Pulchowk All Subject Notes

C++ Object Oriented 32 OK Programming IOE Nepal

notepulchowkcampus.blogsport.com

Project-75

pulchowkcampus.blogspot.com

Inline function:

Eg:-

```
#include < iostream.h >
#include < conio.h >
inline int sum(int a, int b)
{
int s;
s = a + b;
return(s);
}
void main()
{
int a1, b1, s1;
cout << "Enter the value of a1 and b1" << endl;
cin >> a1 >> b1;
s1 = sum(a1, b1); //function call
cout << "The sum is" << s1 << endl;
getch();
}
```

Inline function: It is the function that is expanded inline when it is called with corresponding code.
The syntax for making inline is
Inline return-type function-name (no arguments/arguments)
Note: - inline function should always have return type

Advantage:

1. It reduces the exception time of a program with function having fewer lines of codes.

Disadvantage:

1. The execution time of inline function is more than normal function if the function has the larger number of codes or more lines of code.
2. It does not work for the recursive course.

Default Arguments:

C++ allows programmers to call a function without specifying all its arguments. In such cases function assigned a default value to the parameter which does not have matching arguments in function call. These values in function definition are called **default arguments**.

Note: while assigning a default value, we should place the default from right to left.
eg:-
float simple (int P, int t, float r=0.15) (default argumernt or legal)
float simple(int p, int t=2,float r) (illegal)

http://notepulchowkcampus.blogspot.com/

float simple(int p, int t=2,float r=0.15) (Illegal)

{IOE Pulchowk All Subjects Note (notepulchowkcampus.blogspot.com)}
subscribe and support to us

Inline and default arguments:

```
#include < iostream.h >
#include < conio.h >
using namespace std;
inline float call_sim(int p,int t = 2,float r = 0.15)
{
return (p * t * r/100);
}
void main()
{
int p1;
cout << Ënter the value of P1" << endl;
cin >> p1;
float I1 = call_sim(p1);
cout << "The simple intrest is" << I1 << endl;
getch();
}
```

Function overloading:

Function overloading means using same function name to perform variety of task. But in function overloading we must consider two things. They are:

1. Either the number of arguments in function should be different (the number of input to function should vary).
2. If the numbers of arguments are same then the data type of the function arguments should be different.

number of parameter different.

Note: **return type has no role in function overloading**

Type I: Different no of parameters

```
#include<iostream.h>
#include<conio.h>
using namespace std;
int sum(int a, int b)
{
return (a + b);
}
```

http://notepulchowkcampus.blogspot.com/

```
int sum(int a1,int b1,int c1)
{
return(a1+b1+c1);
}
int sum(int d)
{
return (d+5);
}
void main()
{
cout<<"The sum of one variable"<<sum(10)<<endl;
cout<<"The sum of two variable"<<sum(10,20)<<endl;
cout<<"The sum of three vriable"<<sum(10,20,30)<<endl;
getch();
}
```

Q. writes a program to calculate the area of rectangle and triangle using concept of function overloading.

Type II: Different Data types

```
#include<iostream.h>
#include<conio.h>
using namespace std;
int area(int l,int b)
{
return (l*b);
}
float area(int b,float h)
{
return (0.5*b*h);
}
void main()
{
int l1,b1;
float  h;
cout<<"Enter the value of l1,b1 & h1 <<endl;
cin>>l1>>b1>>h1;
cout<<"The area of rectangle is  <<area(l1,b1)<<endl;
cout<<"The area of triangle is <<area(l1,h1);
getch();
}
```

Reference variable: It is an alternative name given to the original variable. any changes made to reference variable will change original variable and vice versa.
The syntax for making reference variable is:-
data-type & reference_variable = original variable
eg:- int &a(reference variable)=X(original variable)

```
#include<iostream.h>
#include<conio.h>
using namespace std;
void main()
{
```

http://nojepulchowkcampus.blogspot.com/

```
int x = 5;
int &a = x; // making refrance variable
a = a + 5;
cout << "The value of x is" << x << endl;
getch();
}
```
output =10

Q. Is the concept of alternative variable used for pass by reference? If yes show with an appropriate example.

Pass by reference: provision of reference variable in c++ allows us to pass parameter to the function by refe rence. When we pass argument by reference the formal argument in the function call (the input variable of function definition) becomes alternative variable (or reference variable) of the variable in function call.

```
#include < iostream.h >
#include < conio.h >
using namespace std;
void swap (int &a, &b)
{
int temp = a;
a = b;
b = temp;
}
void main()
{
int x1 = 5, y1 = 6;
cout << "before swapping x1 = " << x1 << endl << "y1 = " << y1 << endl;
swap(x1, y1); // function call
cout << "After swapping" << "x1 = " << x1 << endl << "y1" << y1 << endl;
getch();
}
```

Q. Wap to calculate area and perimeter of rectangle in a function and display the result in main().

```
#include < iostream.h >
#include < conio.h >
using namespace std;
void rectangle(int l,int b,int &A1,int &P1)
{
A1 = l * b;
P1 = 2 * (l + b);
}
void main()
{
int l1, b1, A, P;
cout << "Enter the value of l1 and b1" << endl;
cin >> l1 >> b1;
rectangle (l1, b1, A,P);
cout << "Area of rectangle is " << A << endl << "Perimeter of rectangle is" << P << endl;
getch();
}
```

Return by reference: In return by reference in true case c++ assign a reference value . in return by the function is called as:

syntax: function call = reference value;

http://notepulcrowkcampus.blogspot.com/

eg:-
Max (a,b)=200; (reference value)
The syntax for function definition in return by reference is

$return - type \ \& \ function\_name(data\_type1 \ \& \ variable1, data \ type2 \ and \ variable \ 2 \ldots\ldots upto \ n)$

Example of return by reference:

```
#include < iostream.h >
#include < conio.h >
using namespace std;
int &Max(int &a, int &b)
{
if (a > b);
{
        return (a);
}
else
{
return (b);
}
}
void main()
{
int a = 10, b = 5;
Max(a, b) = 200; // function call
cout << "a = " << a << endl << "b = " << b << endl;
getch();
}
```

Output: a=10, b=5

Q. How memory is allocated dynamically for normal variable and array?

Ans: Memory is allocated dynamically for normal variable by using syntax:-
pointer-variable = new data type
eg:-
int *p;
p=new int;
alternatively we can combine to allocate dynamically
syntax: $data - type \ * pointer \ variable \ = \ new \ data \ type$

for array
syntax
$data \ type \ * pointer \ = \ new \ data \ type \ (size)$
eg:-
int *p = new int [20];

http://notepulchowkcampus.blogspot.com/

Class and object:

Class:

1. A class is a way to bind data and associated function together.
2. It allows data and function to hidden if necessary for external use.
3. Defining a class means creating a user defined data type that behaves as built in data.
4. Once a class has been declared we can create any number of objects belonging to that class.

Syntax:

```
class class_name
{
private:
data member;
member function;
public:
data member;
member function;
};//end of class
```

Generally data are placed in private section and functions are placed in public section.

5. Class contains both data and function. Data are called data member and function are called member function.
6. Data and function are defined in private section are not accessible out of the class.
7. Data and function placed in public section are only accessible from outside the class.
8. The word private is optional. The data in function of class by default private.

Object:

⇨ Objects are the basic entities of oop. The object may be person place or anything's that have some physical attribute. Objects are the variable of class in terms of programming.
syntax:
class_name objects_name(variable name)

Accessing the data member of class

1. (.) operator:
the data member or member function of a class is generally accessed by using (.) operator.
The syntax for accessing data member of classis
syntax: object.function_name() or object.data_member;

Q.wap to calculate SI using concept of class and object.

```
#include < iostream.h >
#include < conio.h >
using namespace std;
class simple
{
private:
float p,t,r,I;
public:
void input()
{
cout << "Enter the value of P,T,R" << endl;
cin >> p >> t >> r;
}
void process()
{
I = p * t * r;
}
void display()
{
cout << "The value of I is" << I << endl;
}
}; //end of class
void main()
```

http://notepulchowkcampus.blogspot.com/

```
{
simple s;
s.input();
s.process();
s.display();
getch();
}
```

Q. Wap to calculate area & perimeter of rectangle using the concept of class and object.

```
#include < iostream.h >
#include < conio.h >
using namespace std;
class rectangle
{
private:
int l,b,A,P;
public:
void input()
{
cout << "Enter the value of length and breadth" << endl;
cin >> l >> b;
}
void calculate_area_perimeter()
{
A = l * b;
P = 2 * (l + b);
}
void display()
{
cout << "Area = " << A << "perimeter" << P << endl;
}
};  //end of class
void main()
{
rectangle R;
R.input();
R.calculate_area_perimeter();
R.display();
getch();
}
```

Q. Wap to calculate the distance between points using class and object.

```
#include < iostream.h >
#include < math.h >
#include < conio.h >
using namespace std;
class distance
{
private:
float x1,x2,y1,y2,d;
public:
void input()
{
```

http://notepulchowkcampus.blogspot.com/

```
cout << "Enter the value of x1, x2, y1, y2" << endl;
cin >> x1 >> x2 >> y1 >> y2;
}
void calculate_distance()
{
d = sqrt(pow(x2 − x1,2) + pow(y2 − y1,2));
}
void display()
{
cout << "Distance = " << d << endl;
}
};
void main()
{
distance D;
D.input();
D.calculate_distance();
D.display();
getch();
}
```

Q. Wap to calculate TSA,CSA & volume of cylinder using concept of class and object.

```
#include < iostream.h >
#include < conio.h >
using namespace std;
class cylinder
{
private:
float r, h, tsa, csa, vol;
public:
void input()
{
cout << "Enter the value of r, h" << endl;
cin >> r >> h;
}
void calculate_cylinder()
{
tsa = 2 * 3.1416 * r * (r + h);
csa = 2 * 3.1416 * r * h;
vol = 3.1416 * r * r * h;
}
void display()
{
cout << "TSA = " << tsa << "CSA = " << csa << "Volume = " << vol << endl;
}
};
void main()
{
distance D;
D.input();
D.calculate_cylinder();
D.display();
getch();
```

http://notepulchowkcampus.blogspot.com/

}

Passing object as function argument (input)

Q. Create a class time with data member hour, minute sec and day. use necessary member function and wap to add two type object (by passing object as function argument)

```
#include < iostream.h >
#include < conio.h >
using namespace std;
class time{
private:
 hr,min,sec, day;
public:
void input()
{
cout << "Enter hour,minute sec and day" << endl;
cin >> hr >> min >> sec >> day;
}
void add_time(time t1,time t2)
{
sec = t1. sec + t2. sec;
min = t1.min + t2. min + sec/60;
sec = sec%60;
hr = t1.hr + t2.hr + min/60;
min = min%60;
day = t1. day + t2. day + hr/24;
hr = hr%24;
}
void display()
{
cout << "Sec = " << sec << "Min = " << min << "Hour = " << hr << "Day = " << day << end;
}
};
void main()
{
time t1,t2,t3;
t1.input();
t2.input();
t3.add_time(t1,t2);
t3.display();
getch();
}
```

Q. Create a class called distance with data member feet and inch. use necessary member function if needed and wap to add two distance object.

```
#include < iostrean.h >
#include < conio.h >
using namespace std;
class distance
{
private:
float inch,feed;
public:
```

http://notepulchowkcampus.blogspot.com/

```
void input()
{
cout << "Enter inch and feet" << end;
cin >> inch >> feet;
}
void add_distance(distance d1,distance d2)
{
inch = d1.inch + d2.inch;
feet = d1.feet + d2.feet + inch/12;
inch = inch%12;
}
void display()
{
cout << "Inch = " << inch << "Feet = " << feet << endl;
}
};
void main()
{
distance D1, D2, D3;
D1. input();
D2. input();
D3. add_distance(D1, D2);
D3. display();
getch();
}
```

Q.create a class called complex with data member real and imaginary. Use necessary function and wap to add two complex number and display result.

```
#include < iostream.h >
#include < conio.h >
using namespace std;
class complex
{
private:
int real,img;
public:
void input()
{
cout << "Enter real and imaginary" << endl;
cin >> real >> img;
}
void add_complex(complex c1,complex c2)
{
real = c1.real + c2.real;
img = c1. img + c2. img;
}
void display()
{
cout << "real = " << real << "imaginary" << img << endl;
}
};
void main()
{
```

http://notepulchowkcampus.blogspot.com/

```
complex c1,c2,c3;
c1.input();
c2.input();
c3.add_complex(c1,c2);
c3.display();
getch();
}
```

Passing argument to the function and returning value(returning value in main function)

Q. Create a class called distance with data member feet and inch. Use necessary member function and wap to add two distance object and display the result in main().

```
#include<iosteam.h>
#include<conio.h>
using namespace std;
class distance{
private:
int feet,inch;
public:
void input()
{
cout<<"Enter the value of inch and feet"<<endl;
cin>>inch>>feet;
}
distance add_distance(distance d1,distance d2)
{
distance d3; //temporary object
d3.inch=d1.inch+d2.inch;
d3.feet =d1.feet+d2.feet+d3.inch/12;
d3.inch=d3.inch%12;
return (d3);
}
};
void main()
{
distance D1,D2,D3;
D1.input();
D2.input();
D3=D3.add_distance(D1,D2);
D3.display();
getch();
}
```

Q. Create a class called time with data member hr,min,sec. Use necessary member function and wap to two type of object and display the result in main().

```
#include<iostream.h>
#include<conio.h>
using namespace std;
class time{
private:
 hr,min,sec;
public:
```

http://notepulchowkcampus.blogspot.com/

```
void input()
{
cout << "Enter hour,minute & sec " << endl;
cin >> hr >> min >> sec;
}
time add_time(time t1,time t2)
{
time t3;
t3.sec = t1.sec + t2.sec;
t3.min = t1.min + t2.min + t3.sec/60;
t3.sec = t3.sec%60;
t3.hr = t1.hr + t2. hr + t3. min/60;
t3.min = t3.min%60;
return(t3);
}
void display()
{
cout << "Sec = " << sec << "Min = " << min << "Hour = " << hr << end;
}
};
void main()
{
time t1,t2,t3;
t1.input();
t2.input();
t3 = t3.add_time(t1,t2);
t3.display();
getch();
}
```

Ambiguity in multiple inheritance

when two or more than two base class has same function name and the object of derived that is derived publically from both these base class calls the function then a state of confusion occurs because both these class has same function name. This situation is called ambiguity in multiple inheritances.

Resolving ambiguity

Ambiguity can be resolved by specifying the base class during function call .
syntax : object.class_name::function_name();
Example:

```
class Base1
{
protected:
int a;
public:
void input()
{
cout << "Enter the value of a" << endl;
cin >> a;
}
void display()
{
cout << "The value of a is " << a << endl;
}
```

http://notepulrowkcampus.blogspot.com/

```
};
class Base2
{
protected:
int b;
public:
void input()
{
cout << Enter the value of b" << endl;
cin >> b;
}
void display()
{
cout << "The value of b is " << b << endl;
}
};
class derived:public Base1,public Base2;
{
private:
int s;
public:
void call_display()
{
s = a + b;
cout << "sum = " << s << endl;
}
};
void mian()
{
derived d;
d.input();
d.display();
d.Base1::input();
d.Base2::input();
d.Base1:: display ();
d.Base2:: display ();
d. call_display();
getch();
}
```

ambiguity occurs

ambiguity solved

Function Overriding

when a base class and derived class has same function name and the object of derived class that is derived publically calls that function then it only executes the function of derived by ignoring function of base class. This is known as function overriding.

Resolving function overriding

Function overriding can be resolved by placing the following syntax inside the function that is in derived class.
syntax: base_class_name::function_name();

```
class Base
{
```

http://notepulchowkcampus.blogspot.com/

```
protected:
int a;
public:
void input()
{
cout << "Enter the value of a" << end;
cin >> a;
}
void display()
{
cout << "The value of a is" << a << endl;
}
};
class derived:public Base
{
private:
int b,s;
public:
void input()
{
Base::input();//resolving function overriding
cout << "Enter the value of b" << endl;
cin >> b;
}
void sum()
{
s = a + b;
}
void display()
{
Base::display()//resolving ambiguity
cout << "Sum is = " << s << endl;
}
};
void main()
{
derived d;
d.input();
d.process();
d.display();
getch();
}
```

Data conversion (3-5 marks sure)

There are basically four types of data conversion. They are

1. Basic to basic data conversion
2. Basic to user defined (Basic to class type)
3. user defined to basic type of conversion(class to basic type)
4. user defined to user defined type(class to class type)

http://notepulchowkcampus.blogspot.com/

Basic to class type conversion

Q. How can you convert basic type of data to class type of data?

To convert a basic type of data to a class type of data we need a one argument constructor which is also called as constructor function.

syntax:

```
class_name(one argument )
{
//..conversion routine
}
```

Q. wap to convert a basic type of data min to class type of data sec.

```
#include < iostream.h >
#include < conio.h >
using namespace std;
class time
{
private:
int s;
public:
time()
{
s = 0;
}
time (int m)  // one argument constructor function
{
s = m * 60;
}
void display()
{
cout << "The value of second is = " << s;
}};
void main()
{
time t;  //class type of  data
int m;  // basic type of data
cout << "Enter the value of minute" << endl;
cin >> m;
t = m; //conversion routine call
t. display();
getch();
}
```

Q. Wap to convert a basic type of data meter to class type of data kilometre.

```
#include < iostream.h >
#include < conio.h >
using namespace std;
class convert
{
private:
int km;
public:
convert ()  //default constructor
```

http://notepulchowkcampus.blogspot.com/

```
{
km = 0;
}
convert (int m)  // one argument
{
km = m/1000;
}
void display()
{
cout << "The value of Kilometer is = " << km;
}};
void main()
{
convert d;  //class type of data
int m;  // basic type of data
cout << "Enter the value of meter" << endl;
cin >> m;
d = m; //conversion routine call
d.display();
getch();
}
```

Class to basic type of data conversion

To convert a class type of data to basic type of data we need an operator function

syntax for operator function:

```
operator return_type or data_type()
{
// conversion routine
}
```

Q. Wap to convert a class_type of data minute(m) to basic type of data second(s);

```
#include < iostream.h >
#include < conio.h >
using namespace std;

class time
{
private:
int m;
public:
void input()
{
cout << "Enter the value of minute" << endl;
cin >> m;
}
operator int()  //operator function
{
int s = m * 60;
return s;
}
};
void main()
{
```

http://notepulchowkcampus.blogspot.com/

```
time t;
int s;  //basic type of data
t.input();
s = t;// conversion routine call
cout << "second = " << s;
getch();
}
```

Q. Wap to convert a class_type of data meter(m) to basic type of data kilometre(km);

```
#include < iostream.h >
#include < conio.h >
using namespace std;

class convert
{
private:
int m;
public:
void input()
{
cout << "Enter the value of meter" << endl;
cin >> m;
}
operator int()   //operator function
{
int s = m * 60;
return s;
}
};
void main()
{
convert d;
int km;  //basic type of data
d.input();
km = d;// conversion routine call
cout << "Kilometer = " << km;
getch();
}

class to class conversion
class one
{
private:
: : : : :
public:
: : : : :
};
class two
{
private:
: : : :
public:
```

http://notepulchowkcampus.blogspot.com/

```
: : : :
};
void main()
{
one O;
two t;
O = t; // conversion class two to class one
: : :
}
```

Here class two type is converted to one type class. Class two is called source class and class one is called destination class. We can convert a class type of data to another class type of data by either using constructor function or by operator function.

- ⇨ if we use conversion routine in source class then we use operator function.
- ⇨ if we use conversion routine in destination class then we use one argument constructor function.

Conversion routine in source class

syntax:

```
class source;
class destination
{
private:
data members;
public:
// we use constructor to catch value returned from source class and display it.
};
class source
{
private:
data member;
public:
// input
operator function()
{
//conversion
}};
void main()
{
source s;
destination d;
d = s;   // conversion call
}
```

Q. Wap to convert ^{0}c to f – scale where degree and Fahrenheit and are two different classes.

Q. Wap to convert a class_type of data meter (m) to basic type of data kilometre (km);

```
#include < iostream.h >
#include < conio.h >
using namespace std;
 class degree;
class Fahrenheit
{
```

http://notepulchowkcampus.blogspot.com/

```
private:
float F;
public:
Fahrenheit()
{
F = 0.0;
}
Fahrenheit(float F1)
{
F = F1;
}
void display()
{
cout << " Fahrenheit scale is " << F << endl;
}};
class degree
{
private:
float deg;
public:
void input()
{
cout << "Enter the temperature  in degree calcius " <<  endl;
cin >> deg;
}
operator Fahrenheit() //operator function
{
float F1 = (9/5) * deg + 32;
return (Fahrenheit(F1));
}};
void main()
{
Fahrenheit F;
degree d;
d.input();
F = d;  //conversion routine call
F.display ();
getch();
}
```

Array of objects

If we have to scan or display more than two objects then we use array of objects. The syntax for declaring;-

Syntax:

```
class_name object[number of object];
```

eg:-

```
student s[10]  //array of object
```

Q. Wap to scan and display the details of 5 students.

```
#include < iostream.h >
#include < conio.h >
using namespace std;
class student
{
```

http://notepulchowkcampus.blogspot.com/

```
private:
char name[20];
int id;
public:
void input()
{
cout << "Enter name and ID of student" << endl;
cin >> name >> id;
}
void display()
{
cout << "Name:<< name << "ID:" << id << endl;
}
};
void main()
{
student s[5];   //array of object
int i;
cout << "Enter the detail of 5 student" << endl;
for (i = 0;i < 5;i++)
{
s[i].input();
}
cout << "Details of 5 students are" << endl;
for(i = 0;i < 5; i++)
{
s[i].display();
}
getch();
}
```

Constructor

1. Default constructor:

```
class test
{
private:
int x;
public:
test()   // default constructor
{
x = 10;  //default initallization
}
};
void main()
{
test t;// default constructor called
getch();
}
```

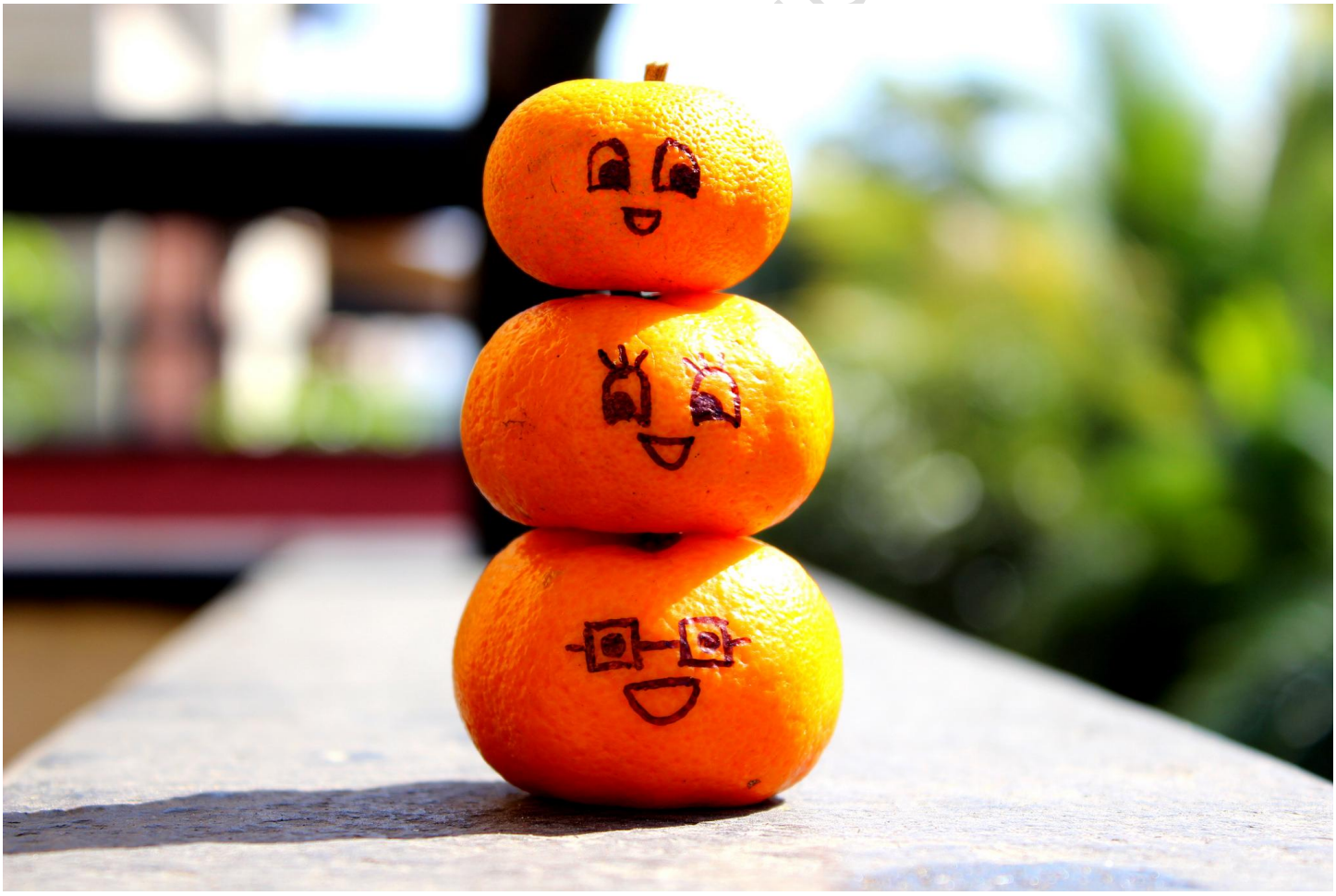
2. Parameterized constructor

```
class test
{
```

http://notepulchowkcampus.blogspot.com/

```
private:
int x;
public:
test( int a)   // default constructor
{
x = a;  //default initallization
}
void display()
{
cout << "the value of x = " << x;
}
};
void main()
{
test t(10);// parameterized constructor called
t. display();
getch();
}
```

Remains note will be updated soon
Subscribe to http://notepulchowkcampus.blogspot.com/



notepulchowkcampus.blogspot.com/