Name : Palash Bajpai

Enroll no : DX2000356

Roll no : 2011220

Subject : Object-oriented programming c++

CPP program with comments and function's

```cpp
#include<iostream>
using namespace std;
void message(),line();

int main()
{
    cout<<"Main program start's"<<endl;
    line();
    message();
    line();
    cout<<"At the end of main()";
    return 0;
}

void line()
{
    cout<<"----------------------------"<<endl;
}

void message()
{
    cout<<"In function message()"<<endl;
}
```

CPP program with function

```cpp
#include<iostream>
using namespace std;
void pause();

int main()
{
    cout<<endl<<"Dear reader";
    cout<<endl<<" have a ";
    pause();
    return 0;
}

void pause()
{
    cout<<"Break";
}
```

Static in cpp

```cpp
// C++ program to demonstrate
// the use of static Static


// variables in a Function
#include <iostream>
#include <string>
using namespace std;

void demo()
{
    // static variable
    static int count = 0;
    cout << count << " ";

    // value is updated and
    // will be carried to next
    // function calls
    count++;
}

int main()
{
    for (int i=0; i<5; i++)
    {
        demo();
    }
    return 0;
}
```

```cpp
// C++ program to demonstrate static
```

Static in cpp

```cpp
// variables inside a class

#include<iostream>
using namespace std;

class GfG
{
public:
    static int i;

    GfG()
    {
        //i++; value of i can be
        //incremented by here
        // Do nothing
    };
};

int GfG::i = 1;

int main()
{
    GfG obj;
    // prints value of i
    cout << obj.i;
}
```

Function overloading cpp

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

void print(int i) {
  cout << " Here is int " << i << endl;
}
void print(double  f) {
  cout << " Here is float " << f << endl;
}
void print(char const *c) {
  cout << " Here is char* " << c << endl;
}

int main() {
  print(10);
  print(10.10);
  print("ten");
  return 0;
}
```

Function overloading cpp

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

class a
{
 public:
void print(int i) {
  cout << " Here is int " << i << endl;
}
void print(double  f) {
  cout << " Here is float " << f << endl;
}
void print(char const *c) {
  cout << " Here is char* " << c << endl;
}

};
int main()
{
  a ob;
  ob.print(10);
  ob.print(10.10);
  ob.print("ten");
  return 0;
}
```

Constructor overloading cpp

```cpp
// C++ program to illustrate
// Constructor overloading
#include <iostream>
using namespace std;
class construct
{
    public:float area;

    // Constructor with no parameters
    construct(int a)
    {
        area = a;
    }
    // Constructor with two parameters
    construct(int a, int b)
    {
        area = a * b;
    }
    void disp()
    {
        cout<< area<< endl;
    }
};
int main()
{
    // Constructor Overloading
    // with two different constructors
    // of class name
    construct o(10);
    construct o2( 10, 20);
```

```
    o.disp();
    o2.disp();
    return 1;
}
```

Inheritance

```
// C++ program to demonstrate implementation
// of Inheritance

#include <bits/stdc++.h>
using namespace std;

//Base class
class Parent
{
    public:
    int id_p;
};

// Sub class inheriting from Base Class(Parent)
class Child : public Parent
{
    public:
    int id_c;
};

//main function
int main()
{

        Child obj1;
        // An object of class child has all data
members
        // and member functions of class parent
        obj1.id_c = 7;
        obj1.id_p = 91;
```

```
        cout << "Child id is " << obj1.id_c <<
endl;
        cout << "Parent id is " << obj1.id_p <<
endl;
        return 0;
}
```

Constructor overloading cpp

```
//Program 2.
#include <iostream>
// Constructor Overloading
using namespace std;
class myclass
{
    int a, b, c;
    public:
    myclass()
    {cout<<"Good Afternoon BCA 3 B\n";}
    myclass(int x)
    {
     c=x;
     cout<<c;
    }

    myclass(int i, int j) {a=i; b=j;}
    void show()
    {
        cout << "a="<<a << " " <<"b="<< b;
    }
};

int main()
{
myclass ob;
myclass ob1(10);
myclass ob2(3,5);
ob2.show();
return 0;
}
```

1.Enumeration in c

```c
#include<stdio.h>
enum week{
Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,
Saturday
};
int main()
{
    week day;
    day=Sunday;
    printf("Today is day %d of this week :
",day+1);
    return 0;
}
```

1.Enumeration in cpp

```cpp
#include<iostream>
using namespace std;
enum week{
Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,
Saturday
};
int main()
{
    week day;
    day=Sunday;
    cout<<"Today is day "<<day+1<<" of this week
: ";
```

2.Enumeration in c

```c
#include<stdio.h>
enum calc{
            a=100,b=20000,c=4000,d=70000000000
}c1;
int main()
{
   printf("Size of enum variable is :
%d",sizeof(c1));
   return 0;
}
```

2.Enumeration in cpp

```cpp
#include<iostream>
using namespace std;
enum calc{
            a=100,b=20000,c=4000,d=7000
}c1;
int main()
{
   cout<<"Size of enum variable is
:"<<sizeof(c1);
   return 0;
}
```

3.Enumeration in cpp

```
#include<stdio.h>
int main()
{
    enum gender { M,F};
    gender a=M;

    switch(a)
    {
        case M:
            printf("Gender is male");
        break;
        case F:
            printf("Gender is female");
        break;
        default:printf("Value can be male or
female");
    }
    return 0;
}
```

3.Enumeration in cpp

```
#include<iostream>
using namespace std;
int main()
{
    enum gender { M,F};
    gender a=M;

    switch(a)
    {
        case M:
            cout<<"Gender is male";
        break;
        case F:
            cout<<"Gender is female";
        break;
        default:cout<<"Value can be male or
female";
    }
    return 0;
}
```

4.Enumeration in c

```c
#include<stdio.h>
enum year{
jan,feb,march,april,may,june,july,august,sept,oc
t,nov,dec,};
int main()
{
    int i;
  for(i=0;i<dec;i++)
    {
        printf("%d\n",i);
    }
    return 0;
}
```

4.Enumeration in cpp

```cpp
using namespace std;
enum
year{jan,feb,march,april,may,june,july,august,se
pt,oct,nov,d};
int main()
{
    int i;
    for(i=0;i<=d;i++)
    {
       cout<<i<<endl;
    }
    return 0;
}
```

Structure (using dot operator) in c for single record

```c
#include<stdio.h>
struct emp
{
    int id;
    float salary;
    char name[50];
};
int main()
{
    struct emp e;

    printf("Enter Employee name : ");
    scanf("%s",e.name);
    printf("Enter employee id : ");
    scanf("%d",&e.id);
    printf("Enter employee salary : ");
    scanf("%f",&e.salary);

    system("cls");
    printf("Entered record:\n");
    printf("Employee id : %d \n",e.id);
    printf("Employee name : %s \n",e.name);
    printf("Employee salary :%f \n ",e.salary);
    return 0;
}
```

Structure (using dot operator) in c for multiple records

```c
#include<stdio.h>
```

```c
struct emp
{
  int emp_id;
  char name[50];
  float salary;
};
int main()
{
    struct emp e[100];
    int  i,n;

    printf("Enter number of record's you want to enter :
");
    scanf("%d",&n);

    for(i=0;i<n;i++)
    {
        printf("\n\nEnter record no %d\n\n",i+1);
        printf("Enter employee name : ");
        scanf("%s",e[i].name);
        printf("Enter employee id : ");
        scanf("%d",&e[i].emp_id);
        printf("Enter employee salary : ");
        scanf("%f",&e[i].salary);
    }
    system("cls");

    printf("\nEntered record's:\n");
    for(i=0;i<n;i++)
    {
        printf("\n\nEntered record no %d\n",i+1);
        printf("Employee name : %s",e[i].name);
        printf("\nEmployee id : %d",e[i].emp_id);
        printf("\nEmployee salary : %.2f",e[i].salary);
        printf("\n");
    }
    return 0;
}
```

Structure (using dot operator) in Cpp for multiple records

```c
#include<iostream>
```

```
using namespace std;

struct emp
{
    int emp_id;
    char name[50];
    float salary;
};
int main()
{
    emp e;
    int n,i;
    cout<<"Enter number of record's you want to enter : ";
    cin>>n;

    for(i=0;i<n;i++)
    {
      cout<<"Enter record no "<<i;
      cout<<"Enter name : ";
      cin>>e[i].name;
      cout<<"Enter employee id : ";
      cin>>e[i].emp_id;
      cout<<"Enter employee salary : ";
      cin>>e[i].salary;
    }
    system("cls");

    cout<<"Entered record :\n";
    for(i=0;i<n;i++)
    {
        cout<<"\nRecord no "<<i;
        cout<<"\nEmployee name : "<<e[i].name;
        cout<<"Employee id : "<<e[i].emp_id;
        cout<<"Employee salary : "<<e[i].salary;
        cout<<endl;
    }
    return 0;
}
```

Structure (using arrow operator) in c for single record

```
#include<stdio.h>
```

```c
#include<stdlib.h>

struct emp
{
  int emp_id;
  char name[50];
  float salary;
};

int main()
{

     struct  emp *e=(struct emp*)malloc(sizeof(struct
emp));
     printf("Enter employee id : ");
     scanf("%d",&e->emp_id);
     printf("Enter employee name : ");
     scanf("%s",e->name);
     printf("Enter employee salary : ");
      scanf("%f",&e->salary);

    system("cls");

    printf("Name: %s",e->name);
    printf("\nId:%d",e->emp_id);
    printf("\nSalary: %f",e->salary);
    return 0;
}
```

Structure (using arrow operator) in cpp for single record

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

```
struct emp
{
  int emp_id;
  char name[50];
  float salary;
};
int main()
{
    emp *e=(emp*)malloc(sizeof(emp));
    cout<<"Enter employee id : ";
    cin>>e->emp_id;
    cout<<"Enter employee name: ";
    cin>>e->name;
    cout<<"Enter employee salary : ";
    cin>>e->salary;

    system("cls");

    cout<<"Employee name : "<<e->name;
    cout<<endl<<"Employee id : "<<e->emp_id<<endl;
    cout<<"Employee salary : "<<e->salary;
    return 0;
}
```


Structure (using arrow operator) in cpp for multiple record's

Structure (using dot operator) in cpp

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

struct a
{
    public: int a,b,c;
};
int main()
{
    a obj;
    obj.a=2;
    obj.b=3;
    obj.c=4;

    cout<<obj.a<<endl<<obj.b<<endl<<obj.c;
    return 0;
}
```

Structure (using arrow operator) in cpp

```cpp
#include<iostream>
using namespace std;
struct a
{
    public : int a=2,b=3,c=3;
};
int main()
{
    a obj,*ptr;
    ptr=&obj;

    ptr->a=5;
    ptr->b=9;
    ptr->c=10;
    // new values assigned in struct a object's

    cout<<ptr->a<<endl;
    cout<<ptr->b<<endl;
    cout<<ptr->c<<endl;
    return 0;
}
```

Class accesing member using dot operator

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

class a
{
    public:
        int a,b,c;
};
int main()
{
    a obj;
    obj.a=1;
    obj.b=2;
    obj.c=3;

    cout<<obj.a<<endl;
    cout<<obj.b<<endl;
    cout<<obj.c<<endl;
    return 0;
}
```

Class accesing member using arrow operator

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

class a{
          int a,b,c;
};

int main()
{
     a obj,*ptr;

     ptr=&obj;

     cout<<ptr->a<<endl;
     cout<<ptr->b<<endl;
     cout<<ptr->c<<endl;

     return 0;
}
```

Class printing integer table in cpp through class

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

class multiply
{
    public :
    int n,i;
    multiply();
    void table();
};

multiply::multiply()
{
  cout<<"Enter a number : ";
  cin>>n;
}

void multiply::table()
{
    for(int i=0;i<=10;i++)
    {
        cout<<i<<"*"<<n<<"="<<i*n<<endl;
    }
}
int main()
{
    multiply m;
    m.table();
```

```
    return 0;
}
```

Class printing both integer and float table in cpp through class

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

class a{
    public : int n,i;
    a();
    void inttable();
};

a :: a()
{
    cout<<"Enter a integer value : \n";
    cin>>n;
}

void a :: inttable()
{
    for(int i=1;i<=10;i++)
    {
        cout<<i<<"*"<<n<<"="<<i*n<<endl;
    }
    cout<<endl;
}

class b{
```

```
      public : float n,i;
    b();
    void floattable();

};

b::b()
{
    cout<<"Enter a integer value : \n";
    cin>>n;
}
void b :: floattable()
{
    for(int i=1;i<=10;i++)
    {
        cout<<i<<"*"<<n<<"="<<i*n<<endl;
    }
}
int main()
{
    a i;
    i.inttable();
    b f;
    f.floattable();
    return 0;
}
```

Parameterized constructor in cpp

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

class add
{
    public : int x,y;
             float a,b;
    add(int x,int y)
    {
        cout<<" Addition of "<<x<<" and " << y
<<" is = "<<x+y<<endl;
    }

    void addfloat(float a,float b)
    {
        cout<<" Addition of "<<a<<" and " <<b<<"
is = "<<a+b<<endl;
    }

};

int main()
{
    add obj(1,2);

    obj.addfloat(1.5,1.5);
    return 0;
}
```

Parameterized constructor in cpp

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

class add
{
    public : int x,y;
             float a,b;
             add(int x,int y);
             void addfloat(float a,float b);
};
    add::add(int x,int y)
    {
        cout<<" Addition of "<<x<<" and " << y
<<" is = "<<x+y<<endl;
    }

    void add::addfloat(float a,float b)
    {
        cout<<" Addition of "<<a<<" and " <<b<<"
is = "<<a+b<<endl;
    }

int main()
{
    add obj(1,2);

    obj.addfloat(1.5,1.5);
    return 0;
}
```

```
#include<iostream>
```