**CPP programs**

Develop a C++ program illustrating Inline Functions

//inline function

#include<iostream>

using namespace std;

inline int show(int a, int b)

{

return (a>b?a:b);

}

int main()

{

int a,b;

cout<<"enter a and b value";

cin>>a>>b;

int x=show(a,b);

cout<<x;

}

1. **Develop a C++ program demonstrating a Bank Account with necessary data members and member functions.**

**Or**

**Develop a C++ program for illustrating Access Specifiers :public and private.**

#include<iostream>

using namespace std;

class bank

{

private:

string fname,lname;

int alc, pin;

public:

void get()

{

cout<<"enter first name:"<<endl;

cin>>fname;

cout<<"enter last name:"<<endl;

cin>>lname;

cout<<"enter alc number and pin no."<<endl;

cin>>alc>>pin;

}

void show()

{

cout<<"first name="<<fname<<endl<<"last name="<<lname<<endl<<"alc no="<<alc<<endl<<"pin no"<<pin<<endl;

}

};

int main()

{

bank b;

b.get();

b.show();

}

1. Develop a C++ program to illustrate this pointer.

#include<iostream>

using namespace std;

class pointers

{

private:

string fname, lname;

int pin, alc;

public:

void get()

{

cout<<"enter first name:"<<endl;

cin>>this->fname;

cout<<"enter last name:"<<endl;

cin>>this->lname;

cout<<"enter alc number and pin no."<<endl;

cin>>this->alc;

cin>>this->pin;

}

void show()

{

cout<<"first name="<<this->fname<<endl<<"last name="<<this->lname<<endl<<"alc no="<<this->alc<<endl<<"pin no="<<this->pin<<endl;

}

};

int main()

{

pointers p;

p.get();

p.show();

}

1. **Develop a C++ program illustrate function overloading.**

//function overloading

#include<iostream>

using namespace std;

class shape

{

public:

void area(int side)

{

cout<<"Area of square is"<<side\*side<<endl;

}

void area(float r)

{

cout<<"Area of circle is"<<3.14\*r\*r<<endl;

}

void area(int l, int b)

{

cout<<"Area of rectangle is"<<l\*b<<endl;

}

};

int main()

{

shape s;

s.area(5);

s.area(5.5f);

s.area(10, 20);

}

1. Develop a C++ program to illustrate the use of default arguments.

//default argument

#include<iostream>

using namespace std;

int interest(int p, int t, int r=5)

{

return (p\*t\*r)/100;

}

int main()

{

int p,t,r;

cout<<"enter p,t,r"<<endl;

cin>>p>>t>>r;

cout<<interest(p,t)<<endl;

cout<<interest(p,t,r)<<endl;

}

1. Develop a C++ program illustrating friend function.

//friend function

#include<iostream>

using namespace std;

class arithematic

{

private:

int a,b;

public:

void get()

{

cout<<"enter a and b"<<endl;

cin>>a>>b;

}

friend void arith(arithematic &s);

};

void arith(arithematic &s)

{

s.get();

cout<<"Addition ="<<s.a+s.b<<endl;

cout<<"subtraction ="<<s.a-s.b<<endl;

cout<<"multiplication ="<<s.a\*s.b<<endl;

}

int main()

{

arithematic s;

arith(s);

}

1. **Develop a C++ Program to illustrate the use of Constructors and Destructors.**

#include<iostream>

using namespace std;

class volume

{

private:

int l,b,h;

public:

volume()

{

cout<<"enter length, breadth and height"<<endl;

cin>>l>>b>>h;

}

void show()

{

cout<<"volume of a box="<<l\*b\*h<<endl;

}

~ volume()

{

cout<<"destructor is involved"<<endl;

}

};

int main()

{

volume v;

v.show();

}

1. **Develop a C++ program illustrating Constructor overloading.**

//construction overloading

#include<iostream>

using namespace std;

class Rectangle

{

private:

int length, breadth;

public:

Rectangle()

{

length=10;

breadth=20;

}

Rectangle(int x, int y)

{

length=x;

breadth=y;

}

void area()

{

cout<<"Area of rectangle:"<<length\*breadth<<endl;

}

};

int main()

{

Rectangle r1;

Rectangle r2 (20,20);

r1.area();

r2.area();

}

1. **Develop a C++ program illustrating Copy Constructor.**

//copy constructor

#include<iostream>

using namespace std;

class person

{

private:

string fname,lname;

public:

person(string f, string l)

{

fname=f;

lname=l;

}

person(person &p1)

{

fname=p1.fname;

lname=p1.lname;

}

void show()

{

cout<<"first name= "<<fname<<endl;

cout<<"Last name= "<<lname<<endl;

}

};

int main()

{

person p1 ("sanjit", "das");

p1.show();

person p2(p1);

p2.show() ;

}

**10) Develop a C++ program to Overload Unary, and Binary Operators using member function.**

**Unary operator using member function**

//uniary operator uning member function

#include<iostream>

using namespace std;

class sample

{

private:

int a,b,c;

public:

void get()

{

a=10;

b=20;

c=-30;

}

void show()

{

cout<<"a= "<<a<<endl;

cout<<"b= "<<b<<endl;

cout<<"c= "<<c<<endl;

}

void operator -()

{

a=-a;

b=-b;

c=-c;

}

};

int main()

{

sample s;

s.get();

s.show();

-s;

cout<<"after uniary minus"<<endl;

s.show();

}

**Binary using member function**

//binary operator

#include<iostream>

using namespace std;

class complex

{

private:

int real, imaginary;

public:

void get()

{

cout<<"enter real number:"<<endl;

cin>>real;

cout<<"enter imaginary number:"<<endl;

cin>>imaginary;

}

void operator +(complex c2)

{

cout<<real+c2.real<<"+"<<imaginary+c2.imaginary<<"i";

}

};

int main()

{

complex c1, c2;

c1.get();

c2.get();

c1+c2;

}

# 11)Develop a C++ program to Overload Unary, and Binary Operators using friend function.

**Uniary using friend function**

//uniary operator uning friend function

#include<iostream>

using namespace std;

class sample

{

private:

int a,b,c;

public:

void get()

{

a=10;

b=20;

c=-30;

}

void show()

{

cout<<"a= "<<a<<endl;

cout<<"b= "<<b<<endl;

cout<<"c= "<<c<<endl;

}

friend void operator -(sample &s);

};

void operator -(sample &s)

{

s.a=-s.a;

s.b=-s.b;

s.c=-s.c;

}

int main()

{

sample s;

s.get();

s.show();

-s;

cout<<"after uniary minus"<<endl;

s.show();

}

**Binary using friend function**

//binary operator using friend function

#include<iostream>

using namespace std;

class complex

{

private:

int real, imaginary;

public:

void get()

{

cout<<"enter real number:"<<endl;

cin>>real;

cout<<"enter imaginary number:"<<endl;

cin>>imaginary;

}

friend void operator +(complex &c1, complex &c2);

};

void operator +(complex &c1, complex &c2)

{

cout<<c1.real+c2.real<<"+"<<c1.imaginary+c2.imaginary<<"i";

}

int main()

{

complex c1, c2;

c1.get();

c2.get();

c1+c2;

}

**12)Develop C++ Programs to incorporate various forms of Inheritance**

**Single inheritance**

//single inheritance

#include<iostream>

using namespace std;

class student

{

protected:

string name, gender;

int age;

public:

void gets()

{

cout<<"enter name, gender, age of student"<<endl;

cin>>name>>gender>>age;

}

void shows()

{

cout<<"Name= "<<name<<endl;

cout<<"gender= "<<gender<<endl;

cout<<"age= "<<age<<endl;

}

};

class mark: public student

{

private:

string reg;

int m1,m2,m3,m4,m5;

public:

void getm()

{

cout<<"enter reg no."<<endl;

cin>>reg;

cout<<"enter 5 subject marks"<<endl;

cin>>m1>>m2>>m3>>m4>>m5;

}

void show()

{

cout<<"reg= "<<reg<<endl;

cout<<"marks="<<m1<<" "<<m2<<" "<<m3<<" "<<m4<<" "<<m5<<endl;

}

};

int main()

{

mark m;

m.gets();

m.getm();

m.shows();

m.show();

}

**Multilevel inheritance**

//multilevel inheritance

#include<iostream>

using namespace std;

class student

{

protected:

string name, gender;

int age;

public:

void gets()

{

cout<<"enter name, gender, age of student"<<endl;

cin>>name>>gender>>age;

}

void shows()

{

cout<<"Name= "<<name<<endl;

cout<<"gender= "<<gender<<endl;

cout<<"age= "<<age<<endl;

}

};

class mark: public student

{

protected:

string reg;

int m1,m2,m3,m4,m5;

public:

void getm()

{

cout<<"enter reg no."<<endl;

cin>>reg;

cout<<"enter 5 subject marks"<<endl;

cin>>m1>>m2>>m3>>m4>>m5;

}

void show()

{

cout<<"reg= "<<reg<<endl;

cout<<"marks="<<m1<<" "<<m2<<" "<<m3<<" "<<m4<<" "<<m5<<endl;

}

};

class percent:public mark

{

private:

float p;

public:

void showpe()

{

p=(m1+m2+m3+m4+m5)/5;

cout<<"percentage="<<p<<"%";

}

};

int main()

{

percent p;

p.gets();

p.getm();

p.shows();

p.show();

p.showpe();

}

**Multiple**

//multiple inheritance

#include<iostream>

using namespace std;

class person

{

protected:

string name, gender;

int age;

public:

void getp()

{

cout<<"enter name, gender, age of student"<<endl;

cin>>name>>gender>>age;

}

void showp()

{

cout<<"Name= "<<name<<endl;

cout<<"gender= "<<gender<<endl;

cout<<"age= "<<age<<endl;

}

};

class student

{

protected:

string reg, course;

public:

void gets()

{

cout<<"enter reg no. and course "<<reg<<course<<endl;

cin>>reg>>course;

}

void shows()

{

cout<<"reg no.= "<<reg<<endl;

cout<<"course= "<<course<<endl;

}

};

class mark: public person, public student

{

private:

int m1,m2,m3,m4,m5;

public:

void getm()

{

cout<<"enter 5 subject marks"<<endl;

cin>>m1>>m2>>m3>>m4>>m5;

}

void showm()

{

cout<<"reg= "<<reg<<endl;

cout<<"marks="<<m1<<" "<<m2<<" "<<m3<<" "<<m4<<" "<<m5<<endl;

}

};

int main()

{

mark p;

p.getp();

p.gets();

p.getm();

p.showp();

p.shows();

p.showm();

}

**Hirerarchy inheritance**

//hirerachy inheritance

#include<iostream>

using namespace std;

class person

{

protected:

string name, gender;

int age;

public:

void getp()

{

cout<<"enter name, gender, age of student"<<endl;

cin>>name>>gender>>age;

}

void showp()

{

cout<<"Name= "<<name<<endl;

cout<<"gender= "<<gender<<endl;

cout<<"age= "<<age<<endl;

}

};

class student:public person

{

private:

string clz, reg, course;

public:

void gets()

{

cout<<"enter clz, reg no. and course "<<clz<<reg<<course<<endl;

cin>>clz>>reg>>course;

}

void shows()

{

cout<<"college= "<<clz<<endl;

cout<<"reg no.= "<<reg<<endl;

cout<<"course= "<<course<<endl;

}

};

class employee: public person

{

private:

string empid, depart;

public:

void gete()

{

cout<<"enter empid and department"<<endl;

cin>>empid>>depart;

}

void showe()

{

cout<<"employee id = "<<empid<<endl;

cout<<"department = "<<depart<<endl;

}

};

int main()

{

student s;

cout<<"student detail"<<endl;

s.getp();

s.gets();

s.showp();

s.shows();

cout<<"employee detail"<<endl;

employee E;

E.getp();

E.gete();

E.showp();

E.showe();

}

**Hybrid inheritance**

//hybrid inheritance

//hirerachy inheritance

#include<iostream>

using namespace std;

class person

{

protected:

string name, gender;

int age;

public:

void getp()

{

cout<<"enter name, gender, age of student"<<endl;

cin>>name>>gender>>age;

}

void showp()

{

cout<<"Name= "<<name<<endl;

cout<<"gender= "<<gender<<endl;

cout<<"age= "<<age<<endl;

}

};

class employee:public person

{

private:

string empid, depart;

public:

void gete()

{

getp();

cout<<"enter empid and department"<<endl;

cin>>empid>>depart;

}

void showe()

{

showp();

cout<<"employee id = "<<empid<<endl;

cout<<"department = "<<depart<<endl;

}

};

class student

{

protected:

string clz, reg, course;

public:

void gets()

{

cout<<"enter clz, reg no. and course "<<clz<<reg<<course<<endl;

cin>>clz>>reg>>course;

}

void shows()

{

cout<<"college= "<<clz<<endl;

cout<<"reg no.= "<<reg<<endl;

cout<<"course= "<<course<<endl;

}

};

class marks:public employee, public student

{

private:

int m1,m2,m3,m4,m5;

public:

void getm()

{

gete();

gets();

cout<<"enter 5 subject marks"<<endl;

cin>>m1>>m2>>m3>>m4>>m5;

}

void showm()

{

showe();

shows();

cout<<"reg= "<<reg<<endl;

cout<<"marks="<<m1<<" "<<m2<<" "<<m3<<" "<<m4<<" "<<m5<<endl;

}

};

int main()

{

marks m;

m.getm();

m.showm();

}

# 13)Develop a C++ program in C++ to illustrate the order of execution of constructors and destructors in inheritance.

**Code:**

//order of execution of constructors and destructors in inheritance.

#include<iostream>

using namespace std;

class A

{

public:

A()

{

cout<<"class a constructor is involved"<<endl;

}

~A()

{

cout<<"class a destructor is involved"<<endl;

}

};

class B:public A

{

public:

B()

{

cout<<"class B costructor is involved"<<endl;

}

~B()

{

cout<<"class B destructor is involved"<<endl;

}

};

int main()

{

B b;

}