Inheritance:

1. Employee:

```
using namespace std;

#include<iostream>

#include<string.h>

struct Employee                          //Base class

{

        int emp_id;

        char name[20];

        double salary;

        Employee()

        {

                cout<<"\n\nEmp default constructor called\n";

                this->emp_id=0;

                strcpy(this->name,"not_given");

                this->salary=0;

        }

        Employee(int i,const char* n,double s)

        {

                cout<<"\n\nEmp parameterised called\n";

                this->emp_id=i;

                strcpy(this->name,n);

                this->salary=s;

        }

        void setId(int i)  //setters(mutators)
```

```
{
        this->emp_id=i;
}
void setName(const char* n)     //setters(mutators)
{
        strcpy(this->name,n);
}
void setSalary(double s) //setters(mutators)
{
        this->salary=s;
}
int getId()         //getters(accessors)
{
        return this->emp_id;
}
char* getName()          //getters(accessors)
{
        return this->name;
}
double getSalary()         //getters(accessors)
{
        return this->salary;
}
void display()
{
		{

			cout<<"\nemployees detail: \nid: "<<this->emp_id<<"\tname: "<<this->name<<"\tsalary: "<<this->salary<<"\n";

		}

};

struct SalesMan:public Employee		//step 1

{

	int target;

	double intensive;

	SalesMan():Employee()  //step 2(a)

	{

		cout<<"\n\nSM default constructor called\n";

		this->target=0;

		this->intensive=0;

	}

	SalesMan(int i,const char* n,double s,int t,int in):Employee(i,n,s)			//step 2(b)

	{

		cout<<"\n\nSM parameterised constructor called\n";

		this->target=t;

		this->intensive=in;

	}

	void setTarget(int t)			//extra setters(mutator) required for Sales manager

	{

		this->target=t;

	}

	void setIntense(double in)			//extra setters(mutator) required for Sales manager

	{
```

```
                this->intensive=in;
        }
        int getTarget()             //extra getters(accessor) required for sales manager
        {
                return this->target;
        }
        double getIntense()                 //extra getters(accessor) required for sales manager
        {
                return this->intensive;
        }
        void display()
        {
                Employee::display();      //step 3
                cout<<"\ntarget: "<<this->target<<"\tintensive: "<<this->intensive;
        }
};
struct Admin:public Employee    //step 1
{
        double allowance;
        Admin():Employee()      //step 2(a)
        {
                cout<<"\n\nAdmin default constructor called\n";
                this->allowance=0;
        }
        Admin(int i,const char* n,double s,double a):Employee(i,n,s)      //step 2(b)
```

```
	{
		cout<<"\n\nAdmin parameterised constructor called\n";
		this->allowance=a;
	}
	void setAllow(double a)		//extra setters(mutator) required for Admin
	{
		this->allowance=a;
	}
	double getAllow()		//extra getters(accessor) required for Admin
	{
		return this->allowance;
	}
	void display()	//step 3
	{
		Employee::display();
		cout<<"\tallowance: "<<this->allowance<<"\n";
	}
};
struct HrManager:public Employee	//step 1
{
	double commission;
	HrManager():Employee()		//step 2(a)
	{
		cout<<"\n\nHR default constructor called\n";
		this->commission=0;
	}
	HrManager(int i,const char* n,double s,double c):Employee(i,n,s)		//step 2(b)
	{
		cout<<"\n\nHR parameterised constructor called\n";
		this->commission=c;
	}
	void setComm(double c)			//extra setters(mutator) required for HR manager
	{
		this->commission=c;
	}
	double getComm()	//extra getters(accessor) required for HR manager
	{
		return this->commission;
	}
	void display()
	{
		Employee::display();	//step 3
		cout<<"\tCommission: "<<this->commission<<"\n";
	}
};
int main()
{
	int id,target;
	char name[20];
	double salary,intensive,allowance,commission;
```

```
Employee e1;

Employee e2(42,"pragati",60000);

SalesMan m1;

cout<<"enter sale managers id:\n";

cin>>id;

cout<<"\nenter the name of sales manager:\n";

cin>>name;

cout<<"\nenter salary of sales manager:\n";

cin>>salary;

cout<<"\nenter target of sales manager:\n";

cin>>target;

cout<<"\nenter intensive for target completion:\n";

cin>>intensive;

m1.setId(id);

m1.setName(name);

m1.setSalary(salary);

m1.setTarget(target);

m1.setIntense(intensive);

m1.display();

SalesMan m3;

m3.display();

SalesMan m2(22,"pragati",50000,45,4500);

m2.display();

Admin a1;

a1.display();
```

```
cout<<"enter admin id:\n";

cin>>id;

cout<<"\nenter name of the admin:\n";

cin>>name;

cout<<"\nenter salary of admin:\n";

cin>>salary;

cout<<"\nallowance for admin:\n";

cin>>allowance;

a1.setId(id);

a1.setName(name);

a1.setSalary(salary);

a1.setAllow(allowance);

cout<<"\nafter setting values\n";

a1.display();

Admin a2(101,"pragati",50000,4500);

a2.display();

HrManager h1;

h1.display();

cout<<"\nenter hr managers id:\n";

cin>>id;

cout<<"\nenter name of hr manager:\n";

cin>>name;

cout<<"\nenter salary of hr manager:\n";

cin>>salary;

cout<<"\nenter commission for hr manager:\n";
```

```
        cin>>commission;
        h1.setId(id);
        h1.setName(name);
        h1.setSalary(salary);
        h1.setComm(commission);
        h1.display();
        HrManager h2(202,"pragati",50000,5000);
        h2.display();
        return 0;
}
```

2. Shape:

```
using namespace std;
#include<iostream>
struct Shape
{
        double area;
        Shape()
        {
                cout<<"\nShape default constructor called\n";
                this->area=0;
        }
        Shape(double a)
        {
                cout<<"\nShape parameterised constructor called\n";
```

```
		this->area=a;
	}
	void setArea(double a)
	{
		this->area=a;
	}
	double getArea()
	{
		return this->area;
	}
	void drawn()
	{
		cout<<"\nArea of shape is: "<<this->area<<"\n";
	}
};
struct Circle:public Shape
{
	double radius;
	Circle():Shape()
	{
		cout<<"\nCircle default constructor called\n";
		this->area=0;
	}
	Circle(double r,double a):Shape(a)
	{
```

```
                cout<<"\nCircle Parameterised constructor called\n";
                this->radius=r;
        }
        void setRadius(double r)
        {
                this->radius=r;
        }
        double getRadius()
        {
                return this->radius;
        }
        void drawn()
        {
                cout<<"\nRadius of circle is: "<<this->radius<<"\n";
                Shape::drawn();
        }
};
struct Rectangle:public Shape
{
        double length,breadth;
        Rectangle():Shape()
        {
                cout<<"\nRectangle default constructor called\n";
                this->length=0;
                this->breadth=0;
```

```
{


}
Rectangle(double l,double b,double a):Shape(a)
{
        cout<<"\nRectangle parameterized constructor called\n";
        this->length=l;
        this->breadth=b;
}
void setLength(double l)
{
        this->length;
}
void setBreadth(double b)
{
        this->breadth;
}
double getLength()
{
        return this->length;
}
double getBreadth()
{
        return this->breadth;
}
void drawn()
{
		{
			cout<<"\nLength of Rectangle is: "<<this->length<<"\nLength of Rectangle is: "<<this-
>breadth<<"\n";
			Shape::drawn();
		}
};
struct Triangle:public Shape
{
		double height,base;
		Triangle():Shape()
		{
			cout<<"\nTriangle default constructor called\n";
			this->base=0;
			this->height=0;
		}
		Triangle(double b,double h,double a):Shape(a)
		{
			cout<<"\nTriangle parameterised constructor called\n";
			this->base=b;
			this->height=h;
		}
		void setBase(double b)
		{
			this->base=b;
		}
		void setHeight(double h)
		{
```

```
		this->height=h;
	}
	double getBase()
	{
		return this->base;
	}
	double getHeight()
	{
		return this->height;
	}
	void drawn()
	{
		cout<<"\nBase of Triangle is: "<<this->base<<"\nHeight of Triangle is: "<<this-
>height<<"\n";
		Shape::drawn();
	}
};
int main()
{
	double radius,length,breadth,base,height;
	Circle c1;
	cout<<"\nEnter radius of circle: ";
	cin>>radius;
	c1.setRadius(radius);
	c1.setArea(2*3.14*radius);
	Circle c2(5,2*3.14*5);
```

```
            c2.drawn();

            Triangle t1;

            cout<<"\nEnter base and height of triangle: ";

            cin>>base>>height;

            t1.setBase(base);

            t1.setHeight(height);

            t1.setArea(0.5*base*height);

            Triangle t2(3,5,0.5*3*5);

            t2.drawn();

            Rectangle r1;

            cout<<"\nEnter length and breadth of rectangle: ";

            cin>>length>>breadth;

            r1.setLength(length);

            r1.setBreadth(breadth);

            r1.setArea(length*breadth);

            Rectangle r2(5,3,5*3);

            r2.drawn();

            return 0;

}
```

3. Payement Method:

```
using namespace std;

#include<iostream>

#include<string.h>

struct PaymentMethod
```

```
{
	char sender[20],receiver[20];
	double amount;
	PaymentMethod()
	{
		cout<<"\nPayment default construct called\n";
		strcpy(this->sender,"not_given");
		strcpy(this->receiver,"not_given");
		this->amount=0;
	}
	PaymentMethod(char* s,char* r,double a)
	{
		cout<<"\nPayment parameterised construct called\n";
		strcpy(this->sender,s);
		strcpy(this->receiver,r);
		this->amount=a;
	}
	void setSender(char* s)
	{
		strcpy(this->sender,s);
	}
	void setReceiver(char* r)
	{
		strcpy(this->receiver,r);
	}
	void setAmount(double a)
	{
		this->amount=a;
	}
	char* getSender()
	{
		return this->sender;
	}
	char* getReceiver()
	{
		return this->receiver;
	}
	double getAmount()
	{
		return this->amount;
	}
	void display()
	{
		cout<<"\nPayment Details:\nSender name: "<<this->sender<<"\t Receiver name: 
"<<this->receiver<<"\t Amount: "<<this->amount<<"\n";
	}
};
struct DebitCard:public PaymentMethod
{
	int accNo,cvv;
	DebitCard():PaymentMethod()
```

```
{
        cout<<"\nDebit card default constructor called\n";
        this->accNo=0;
        this->amount=0;
}
DebitCard(char* s,char* r,double a,int acc,int cvv):PaymentMethod(s,r,a)
{
        cout<<"\nDebit card default constructor called\n";
        this->accNo=acc;
        this->cvv=cvv;
}
void setAccNo(int acc)
{
        this->accNo=acc;
}
void setCvv(int cvv)
{
        this->cvv=cvv;
}
int getAccNo()
{
        return this->accNo;
}
int getCvv()
{
```

```
                return this->cvv;
        }
        void display()
        {
                PaymentMethod::display();
                cout<<"\nAccount no: "<<this->accNo<<"\t cvv: "<<this->cvv<<"\n";
        }
};
struct UPI:public PaymentMethod
{
        int upiId;
        UPI():PaymentMethod()
        {
                cout<<"\nUpi default constructor called\n";
                this->upiId=0;
        }
        UPI(char* s,char* r,double a,int ui):PaymentMethod(s,r,a)
        {
                cout<<"\nUpi default constructor called\n";
                this->upiId=ui;
        }
        void setUpiId(int ui)
        {
                this->upiId=ui;
        }
```

```
	int getUpiId()
	{
		return this->upiId;
	}
	void display()
	{
		PaymentMethod::display();
		cout<<"\nUPI id: "<<this->upiId<<"\n";
	}
};
int main()
{
	char sender[20],receiver[20];
	int accNo,cvv,upiId;
	double amount;
	DebitCard d1;
	cout<<"\nEnter sender name: ";
	cin>>sender;
	cout<<"\nEnter receiver name: ";
	cin>>receiver;
	cout<<"\nEnter amount to pay: ";
	cin>>amount;
	cout<<"\nEnter account no.: ";
	cin>>accNo;
	cout<<"\nEnter cvv: ";
```

```
cin>>cvv;

d1.setSender(sender);

d1.setReceiver(receiver);

d1.setAmount(amount);

d1.setAccNo(accNo);

d1.setCvv(cvv);

d1.display();

DebitCard d2("Prakruti","Pragati",50000,2330409,234);

d2.display();

UPI u1;

cout<<"\nEnter sender name: ";

cin>>sender;

cout<<"\nEnter receiver name: ";

cin>>receiver;

cout<<"\nEnter amount to pay: ";

cin>>amount;

cout<<"\nEnter upi id: ";

cin>>upiId;

u1.setSender(sender);

u1.setReceiver(receiver);

u1.setAmount(amount);

u1.setUpiId(upiId);

u1.display();

UPI u2("Prakruti","Pragati",50000,2330904);

u2.display();
```

```
        return 0;
}
```

4. Bicycle:

```
using namespace std;
#include<iostream>
#include<string.h>
struct Cycle
{
        int no_of_wheel,no_of_bottle_holder;
        char cname[20],brake_type[20];
        Cycle()
        {
                cout<<"\nCycle default constructor called\n";
                strcpy(this->cname,"not given");
                this->no_of_wheel=0;
                this->no_of_bottle_holder=0;
                strcpy(this->brake_type,"not given");
        }
        Cycle(char* cn,int w,int bh,char* b)
        {
                cout<<"\nCycle default constructor called\n";
                strcpy(this->cname,cn);
                this->no_of_wheel=w;
                this->no_of_bottle_holder=bh;
```

```
		strcpy(this->brake_type,b);
	}
	void setCompany(char* cn)
	{
		strcpy(this->cname,cn);
	}
	void setWheels(int w)
	{
		this->no_of_wheel=w;
	}
	void setBottleHolder(int bh)
	{
		this->no_of_bottle_holder=bh;
	}
	void setBrake(char* bt)
	{
		strcpy(this->brake_type,bt);
	}
	char* getCompany()
	{
		return this->cname;
	}
	int getWheel()
	{
		return this->no_of_wheel;
	}
	int getBottleHolder()
	{
		return this->no_of_bottle_holder;
	}
	char* getBrake()
	{
		return this->brake_type;
	}
	void display()
	{
		cout<<"\nCycles details:\nCompany name: "<<this->cname<<"\nNo of wheels: "<<this->no_of_wheel<<"\nNo of bottle holder: "<<this->no_of_bottle_holder<<"\nBrake type: "<<this->brake_type<<"\n";
	}
};
struct GearCycle:public Cycle
{
	int no_of_gears;
	GearCycle():Cycle()
	{
		cout<<"\nGear cycle default constructor called\n";
		this->no_of_gears=0;
	}
	GearCycle(char* cn,int w,int bh,char* bt,int g):Cycle(cn, w, bh, bt)
	{
		cout<<"\nGear cycle parameterised constructor called\n";
		this->no_of_gears=g;
	}
	void setGear(int g)
	{
		this->no_of_gears=g;
	}
	int getGear()
	{
		return this->no_of_gears;
	}
	void display()
	{
		Cycle::display();
		cout<<"No of gears: "<<this->no_of_gears<<"\n";
	}
};
struct ElectricCycle:public Cycle
{
	double battery_power;
	ElectricCycle():Cycle()
	{
		cout<<"\nElectric cycle default constructor called\n";
		this->battery_power=0;
	}
```

```
        ElectricCycle(char* cn,int w,int bh,char* bt,double b):Cycle(cn, w, bh, bt)
        {
                cout<<"\nElectric cycle default constructor called\n";
                this->battery_power=b;
        }
        void setBattery(double b)
        {
                this->battery_power=b;
        }
        double getBattery()
        {
                return this->battery_power;
        }
        void display()
        {
                Cycle::display();
                cout<<"Battery power: "<<this->battery_power<<"\n";
        }
};
int main()
{
        GearCycle g1;
        g1.display();
        GearCycle g2("keysto",2,1,"disc_brake",14);
        g2.display();
```

```
	ElectricCycle e1;

	e1.display();

	ElectricCycle e2("keysto",2,2,"drum_brake",5000);

	e2.display();

	return 0;

}
```

5. Water Source:

```
using namespace std;

#include<iostream>

#include<string.h>

struct WaterSource

{

	char name[20];

	double waterSalinity;

	char waterQuality[20];

	WaterSource()

	{

		cout<<"\nWater source default constructor called\n";

		strcpy(this->name,"not given");

		this->waterSalinity=-1;

		strcpy(this->waterQuality,"not_given");

	}

	WaterSource(char* n,double s,char* q)

	{
}

        cout<<"\nWater source parameterised constructor called\n";

        strcpy(this->name,n);

        this->waterSalinity=s;

        strcpy(this->waterQuality,q);

}

void setName(char* n)

{

        strcpy(this->name,n);

}

void setWaterQuality(char* q)

{

        strcpy(this->waterQuality,q);

}

void setWaterSalinity(double s)

{

        this->waterSalinity=s;

}

char* getName()

{

        return this->name;

}

char* getWaterQuality()

{

        return this->waterQuality;

}
```

```
	}

	double getWaterSalinity()
	{
		return this->waterSalinity;
	}
	void display()
	{
		cout<<"\nName: "<<this->name<<"\t  Quality: "<<this->waterQuality<<"\t  Salinity: 
"<<this->waterSalinity<<"\n";
	}
};
struct Dam:public WaterSource
{
	int noDoors;
	double capacity;
	Dam():WaterSource()
	{
		cout<<"\nDam default constructor called\n";
		this->noDoors=0;
		this->capacity=0;
	}
	Dam(char* n,double s,char* q,int d,double c):WaterSource(n,s,q)
	{
		cout<<"\nDam parameterised constructor called\n";
		this->noDoors=d;
		this->capacity=c;
	}
```

```
        void setDoors(int d)

        {

                this->noDoors=d;

        }

        void setCapacity(double c)

        {

                this->capacity=c;

        }

        int getDoors()

        {

                return this->noDoors;

        }

        double getCapacity()

        {

                return this->capacity;

        }

        void display()

        {

                cout<<"\nWater source: Dam\nDetails:\n";

                WaterSource::display();

                cout<<"\nNo of doors: "<<this->noDoors<<"\tCapacity: "<<this->capacity<<" tmc\n";

        }

};

struct River:public WaterSource

{
```

```
double depth,width;

char origin[20];

River():WaterSource()

{

        cout<<"\nRiver default constructor called\n";

        strcpy(this->origin,"not given");

        this->depth=0;

        this->width=0;

}

River(char* n,double s,char* q,char* sn,double d,double w):WaterSource(n,s,q)

{

        cout<<"\nRiver default constructor called\n";

        strcpy(this->origin,sn);

        this->depth=d;

        this->width=w;

}

void setOrigin(char* sn)

{

        strcpy(this->origin,sn);

}

void setDepth(double d)

{

        this->depth=d;

}

void setWidth(double w)
```

```
	{
		this->width=w;
	}
	char* getOrigin()
	{
		return this->origin;
	}
	double getDepth()
	{
		return this->depth;
	}
	double getWidth()
	{
		return this->width;
	}
	void display()
	{
		cout<<"\nWater source: River\nDetails:\n";
		WaterSource::display();
		cout<<"\nOrigin of river: "<<this->origin<<"\t  Depth: "<<this->depth<<"\t Width: 
"<<this->width<<"\n";
	}
};
int main()
{
	Dam d1;
```

```
        Dam d2("koyana_dam",20,"Medium",36,98.77);

        d2.display();

        River r1;

        River r2("koyana",32,"good","krishna_river",130,81);

        r2.display();

        return 0;

}


//  Short notes:
```

## # Software Developement Life Cycle (SDLC):

The growth of an info. system, passes through the various stages & this stages put together are reffered to as the software development life cycle (SDLC)

└ It is the process used by software industry to design, developed & test high quality software.

└ It aims to produce a high quality software that meet customers expectation, reaches completion within time & cost estimates.

— Activities of SDLC :

i) Prelimary investigation

ii) ~~System Design~~ Determination of system requirement

iii) system Design

iv) System development

v) system testing

vi) System implementation

vii) System maintanance

1) Preliminary investigation:

└ Any person related with the system, initiate the request, when the request is made, preliminary investigation start. It has 3 parts:

1) request clarification

2) feasibility study

i) techniqual feasibility

ii) Economical feasibility

iii) Operational feasibility.

3) request approval

ii) Determination of system requirements:

- Business analyst & project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective.
- Before creating a product, core understanding or knowledge of the product is very necessary.

iii) Designing :

- This step/phase is the product of inputs from customer & requirement gathering
- This phase is about to bring down all the knowledge of requirements, analysis & design of software project.

iv) Development :

- The implementation of design begins concerning writing code
- Developers have to follow coding guidelines described by their management & programming tools like compiler, interpreters, debuggers etc. are used to develop & implement the code.

v) Testing:

- After the code is generated, it is tested against the requirements to make sure that the product are solving the needs addressed & gathered during the requirements stage.
- During this stage, unit testing, integration testing, system testing, acceptance testing is done.

vi) System implementation :

- Once system has been clearify fully developed & tested, It is ready for implementation/ deployment.
- Then based on the assessment, software may be released as it is or with suggested enhancement in the object segment

vii) Maintenance :

- Once when the client starts using the developed systems, then the real issues come up & requirements to be solved from time to time.
- This procedure where the care is taken for the developed product is known as ~~co~~ maintenance.

# Reference variable :

- It is defined as an alias for another variable.
- In short, it is like giving a different name to pre-existing variable
- Once reference is initialized to the variable, we can use either the reference name or the variable to refer to that variable

eg.
```
~~in~~ void myfun (int & );
int main()
{ int a=10;
  myfun (a)
  cout <<" a is: " <<a ;
}
void myfun (int& b)
{  b = 30;
}
```