SDF Tutorial - 10

Q1.)

Procedural languages :-

⮚ In procedural programming, program is divided into small parts called functions.

⮚ Procedural programming does not have any proper way for hiding data so it is less secure.

⮚ Procedural programming is based on unreal world.

⮚ In procedural programming, function is more important than data.

⮚ In procedural programming, overloading is not possible.

⮚ Examples:- C, FORTRAN, Pascal, Basic etc.

OBJECT ORIENTED LANGUAGE :-

⮚ In object oriented programming, program is divided into small parts called objects.

⮚ Object oriented programming provides data hiding so it is more secure.

⮚ Object oriented programming is based on real world.

⮚ In object oriented programming, data is more important than function.

⮚ Overloading is possible in object oriented programming.

⮚ Examples:- C++, Java, Python, C# etc.

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q2.)

# Advantage :-

The C programming language isn’t going anywhere anytime soon; However, the tools available to developers to start using C++ and beginning the transition are available. C++ offers developers an opportunity to start using object oriented methodologies while at the same time, if necessary, continue to use legacy C code. There are many reasons to start using C++, and plenty to continue using C, but don’t be surprised in the next few years when more and more example code starts showing up in C++.

# Characteristics :-

⮚ Encapsulation :– Encapsulation is capturing data and keeping it safely and securely from outside interfaces.

⮚ Inheritance:- This is the process by which a class can be derived from a base class with all features of base class and some of its own. This increases code reusability.

⮚ Polymorphism:- This is the ability to exist in various forms. For example an operator can be overloaded so as to add two integer numbers and two floats.

⮚ Abstraction:- The ability to represent data at a very conceptual level without any details.

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q3.)

There are 6 types of inheritence:-

⮚ Single Inheritance

⮚ Multilevel Inheritance

⮚ Multiple Inheritance

⮚ Heirarchical Inheritance

⮚ Hybrid Inheritance

⮚ Multipath Inheritance

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q4.)

## INLINE FUNCTION :-

Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call. This substitution is performed by the C++ compiler at compile time.

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q5.)

PURE VITAL FUNCTION :- A pure virtual function or pure virtual method is a virtual function that is required to be implemented by a derived class if the derived class is not abstract. Classes containing pure virtual methods are termed "abstract" and they cannot be instantiated directly.

## ABSTRACT CLASS:-

Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation. Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods.

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q6.)

## FRIEND CLASS:-

⮚ It is a class used with a friend keyword to access the private members of another class.

⮚ A friend class can be used when a class is created on the top of another class.

## FRIEND FUNCTION:-

⮚ It is a function used with a friend keyword to grant a non-member function access to the private members of a class.

⮚ A friend function can be used in some situation of operator overloading.

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q7.)

OPTION:- B - void virtual f()=0 { }

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q8.)

STATEMENT:- A - / B - ( ) C - :: D - %

## ANSWER:- “ :: “

REASON:- The sizeof operator returns the size of the object or datatype as an operand. This is evaluated by the compiler. It cannot be evaluated during runtime. So we cannot overload it.

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q9.)

#include <iostream>

using namespace std; class human

{ public: string prof; int age;

human(): prof("UNEMPLOYED"), age(16) { }

void display()

{

cout<<"\nMY PROFESSION IS : "<<prof<<endl; cout<<"\nMERI UMAR H : " <<age<<endl;

walk(); talk();

}

void walk() { cout<<"\nMN CHL SKTA HU."<<endl; } void talk() { cout<<"\nMN BOL SKTA HU."<<endl; }

};

class kabadi : public human

{ public:

void teach() { cout<<"\nMN KABADI MN PIROO PLAYER HU."<<endl; }

};

class pubg : public human

{ public:

void play() { cout<<"\nMN PUBG KA PIROO PILAYER BHI HU."<<endl; }

};

int main()

{

kabadi k;

k.prof = "KABADI PLAYER";

k.age = 19;

k.display();

k.teach(); cout<<"\n\n\n"; pubg p;

p.prof = "PUBG PLAYER";

p.age = 19;

p.display();

p.play();

return 0; }

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q10.)

#include <iostream> using namespace std; class user

{

protected:

char name[30] ,gender; int empId; public:

void userdata(void)

{

cout<<"ENTER YOUR NAME : "; cin.getline(name,30); cout<<"ENTER YOUE EMPLOYEE ID : "; cin >>empId; cout<<"ENTER YOUR GENDER(M/F) : "; cin>>gender;

}

};

class deptInfo

{

protected:

char deptName[30]; char assignedWork[30]; int time2complete; public: void getDeptInfo(void)

{

cout<<"ENTER YOUR DEPATMENT NAME : ";

cin.ignore(1); cin.getline(deptName,30); cout<<"EMTER YOUR ASSINGED WORK : ";

fflush(stdin); cin.getline(assignedWork,30); cout<<"ENTER THE DEADLINE : "; cin >>time2complete;

}

};

class employee:private user, private deptInfo

{ public:

void getEmployeeInfo(void)

{

cout<<"ENTER EMPLOYESS DATA: " << endl;

userdata();

cout<<"\nENTER EMPLOYE WORK DATA: " << endl;

getDeptInfo();

}

void printEmployeeInfo(void)

{

cout<<"EMPLOYES INFO : " << endl; cout<<"NAME: " << name << endl; cout<<"EMPLOYEE ID : " << empId << endl; cout<<"GENDER: " << gender << endl << endl; cout<<"\n\n\n\n"; cout<<"WORK INFO:" << endl; cout<<"DEPARTMENT NAME : " << deptName << endl; cout<<"ASSINGED WORK : " << assignedWork << endl; cout<<"DEADLINE : " << time2complete<< endl;

}

};

int main()

{ employee emp;

emp.getEmployeeInfo(); emp.printEmployeeInfo(); return 0; }

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q11.)

#include<iostream>

using namespace std; int main()

{

int time = 0; int hour = 0; int min = 0; int sec = 0; cout<<"ENTER TIME IN SECONDS : "; cin>>time; hour = time/3600; time = time%3600; min = time/60; time = time%60; sec = time;

cout<<"\nTIME IN HH:MM:SS FORMAT IS : "<<hour<<" Hrs., "<<min<<" Min., and "<<sec<<" Sec!\n";

return 0; }

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q12.)

#include <iostream> using namespace std; class C{ private:

static int count; public:

C() { count++; } static int Count() { return count; }

};

int C::count; main() {

C obj1, obj2, bj3; int c1=0; c1 = C::Count();

cout << "NUMBER OF OBJECTS : " << c1; }

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q13.)

#include<stdio.h> #include<iostream> using namespace std; class NUM

{ private: int n; public:

void getNum(int x) { n=x; } void dispNum(void) { cout << "\nNUMBER IS : " << n; }

NUM operator +(NUM &obj)

{

NUM x;

x.n=this->n + obj.n; return (x); }

};

int main()

{

NUM num1,num2,sum;int x,y;

cout<<"\nENTER FIRST NO. :"; cin>>x; num1.getNum(x); cout<<"\nENTER SECOND NO. :"; cin>>y; num2.getNum(y); sum=num1+num2; sum.dispNum(); cout << endl; return 0; }

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q14.)

#include<iostream> using namespace std; int main()

{ int x,s=0; cout <<"ENTER A NO. : ";

cin >> x; while(x!=0)

{

s=s+ (x%10); x=x/10;

}

cout << "SUM OF DIGITS OF IS : "<<x<<s; return 0;

}

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q15.)

#include<iostream> using namespace std; float cyl(int r,int h) { return(3.14\*r\*r\*h); } float sph(float r1) { return((4\*3.14\*r1\*r1\*r1)/3); } int cube(int a) { return(a\*a\*a); }

int main()

{ int r,h,a; float r1;

cout<<"\nENTER RADIUS AND HEIGHT OF CYLINDER : "; cin>>r>>h; cout<<"\nENTER SIDE OF THE CUBE :"; cin>>a; cout<<"\nENTER RADIUS OF THE SPHERE : "; cin>>r1; cout<<"\nVOLUME OF THE CYLINDER IS : "<<cyl(r,h); cout<<"\nVOLUME OF THE CUBE IS : "<<cube(a); cout<<"\nVOLUME OF THE SPHERE IS : "<<sph(r1); return 0; }

---------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------

Q16.)

#include<iostream> using namespace std; class NUM

{ private: int n; public:

void getNum(int x) { n=x; } void dispNum(void) { cout << "\nNUMBER IS : " << n; }

NUM operator +(NUM &obj)

{

NUM x;

x.n=this->n + obj.n; return (x); }

};

int main()

{

NUM num1,num2,sum;int x,y; cout<<"\nENTER FIRST NO. :"; cin>>x; num1.getNum(x); cout<<"\nENTER SECOND NO. :"; cin>>y; num2.getNum(y); sum=num1+num2; sum.dispNum(); cout << endl;

return 0;

}