1. Differences between procedural , OOP , and functional programming?

procedural : is a sequence of procedures (functions) , the flow of procedures is linear.

OOP: depending on classes and objects , it encapsulates the data and methods that treat these data.

functional programming: it is an approach that depends on functions essentially in building the program, as it characterizes its first class citizens or object , this makes it more flexible because you can assign functions to variables , return function from another and pass function as an argument to parameters of another function.

1. OOP have Four principles , what are ?

1- Encapsulation

2- Inheritance

3- Abstraction

4- polymorphism

3- What is the first principle of Encapsulation in oop means? give me an example

it means hide and protect data of a class this does not prevent happening errors , it become less possibility .As it gives a developer a full control his data by access modifier (public ,private, protected).

Example:

class Bank\_Account

{

private:

double balance;

public:

void setBalance(double b)

{

if (b > 0)

{

balance = b;

}

else

{

cout << "Error, please enter another number" << endl;

}

};

Here , I make balance private in order to not allow other classes or other functions outside this class access it directly and that is to avoid errors happening and misuse of the data.

4. What abstraction means in OOP? give me an example

Abstraction refers to hide implementation, and this occurs within an abstract class or interface. Inside them abstract functions(that is without scoop or body).Their implementation inside child classes which has different behavior about abstract functions.

Example:

class Animal {

public:

virtual void sound() = 0;

void eat()

{ std::cout << "Eating..." << std::endl; }

};

class Dog : public Animal {

public:

void sound()

{ std::cout << "Woof!" << std::endl; }

};

Cat : public Animal {

public:

void sound()

{ std::cout << "Meow!" << std::endl; }

};

5. Differences between encapsulation and abstraction?

Encapsulation focuses on hiding and protecting data by restricting direct access to some members of a class as the external parts can not access these .

Abstraction hides detailed implementation of its methods and provides the necessary functions.

6. what is composition in OOP? give me an example

it is a relationship “Has a” , as something is a part of another for example human has a heart. Here the heart is the part of the human which can not live without it. In OOP it represents this type through a base class that has an object from the derived class. Therefore, the

Example

class Engine {

public: void start()

{ std::cout << "Engine started" << std::endl; }

};

class Wheel {

public: void rotate()

{ std::cout << "Wheel is rotating" << std::endl; }

}; class Car {

private:

Engine engine;

Wheel wheels[4];

public:

void drive() {

engine.start();

for (int i = 0; i < 4; ++i) {

wheels[i].rotate();

}

std::cout << "Car is driving" << std::endl; }

};

7. why composition is consider in many cases better than inheritance?

Because one important of its benefits is a low dependency, this allows you to change what you need in a class without modifying or even altering something in another class that is owing it.

8.Differences between abstract and concrete class?

A concrete class, all its functions have implementation ,therefore, you can instantiate of concrete class unlike an abstract class that contains a pure virtual function (without implementation) .Consequently, it can not instantiate of abstract class.

9- differences between abstract class and interface in c++?

Abstract class is more flexible as it consists of both abstract (pure virtual functions)

and non-abstract methods (with their implementation) whereas the interface contains only abstract methods. Furthermore, you can specify the access level of members in an abstract in contrast to an interface where all its members are public by default. Also, abstract class is one type of classes , but interface is not class. You can define attributes in abstract class .On other hand, it is rarely doing that in the interface.

10. Difference between private , public ,protected Inheritance , and what are the types of

inheritance?

In public inheritance, the derived class inherits the members of the base class as they remain the same access modifier in base (public become public , private remain as it is , protected remain protected).

Whereas private inheritance, a derived class inherits the public and protected members of a base class which are private inside it.

While protected Inheritance, the derived class inherits public members of the base class are protected for it , as it uses the inherited members freely inside it and only other derived classes can use these members, which inherit from a previous derived class too(multilevel inheritance).

types of inheritance:

1- single inheritance

2-multiple inheritance

3- multilevel inheritance

4- hierarchical inheritance

5- hybrid inheritance

11. what is Polymorphism ?

It is the ability of variable , function and object to have multiple forms.For example, one method in the base class can have different behaviors according to how this child class behaves.

12. what are the two types of polymorphism ? give an example for each type

1. run-time polymorphism like overriding functions and virtual functions. In this type of polymorphism the compiler can not select which function to call it, so virtual functions are always dynamic. Briefly, the function call is resolved at run-time.
2. compiler-time such as operator overloading and function overloading , here the compiler can specify any function to call at compile-time before run- time.Therefore, the function call is resolved at compile-time.

13. polymorphyims allow us to pass childs to function that accept parent as paramater , explain.

ou can create function which accept parameter from parent class for instance pointer and pass it object of the derived class . When the base class have virtual function in order to can apply this feature in your code.

14.what is execption means? and why do we need exception handling?

An exception is a problem which occurs during executing the program,as the execution stops suddenly due to run-time errors that can not discover it within compile-time like divide by zero and error stack overflow. In addition, the exception is a base class for many derived classes.Exception class has a virtual method called What() that returns a string of exception description. We need exception handling for many purposes which are ,can resolve exceptions , allow a program to continue executing ,notify the user about the problem and terminate the program in a controlled manner , making the program robust and fault-tolerant.

15. explain the different types of exception exist.

1. A derived class logic- failure is related to logic errors of the program generally, there 4 classes inherit from logic-failure including the derived class out of range , it occurs like when array index is out of array size range , also the derived class invalid argument results from pass argument to inappropriate parameters of function.
2. A derived class runtime\_error is a base class to the derived class overflow\_error such as integer is overflow and underflow-error and range-error.
3. A derived class bad-alloc is another type of exception which occurs when you try to store a large size in filled memory or in not enough memory space .

16. explain try , catch in c++ , and explain a program with multiple catch blocks.

try is a block which contains the code of unexpected error and keyword “throw”.

Also, catch is a block which comes directly after the try block and inside it occurs exception handling via catch the exception is passed by keyword “throw” in the try block as parameter of the block catch . In multiple catch blocks each block catch receives a different type of exception as an argument , so each throw passes the exception to only one catch block according to a type of passed exception.

17.what is the stack trace that exist in the exception?

stack trace is a recording of the exception path as it registers all relevant occurring exception for instance , a line which happened inside it the exception ,a file of exception and all functions that contained an exception.

18. what is throw keyword do ? difference between throw and throw ex in c++?

1. throw keyword come after it any data type of exception , compiler when see throw keyword transfers to another part of program automatically which is the catch block at the same time throw keyword passes this exception to the catch block.Difference between throw and throw ex is preserve stack trace that occur in throw , it demonstrates all path of exception unlike throw ex that is not preserve stack trace. Therefore, it clarifies only the last error and not the path of exception from its beginning.

19. Create object in stack , and create another object in the heap and compare stack and heap .

class student {

string name;

int ID;

public:

student(string n = "Aya", int id = 113690) :name(n),ID(id)

{

}

void set\_name(string n) {

name = n;

}

string get\_name() {

return name;

}

void set\_ID(int n) {

ID = n;

}

int get\_ID() {

return ID;

}

};

int main() {

student\* t = new student();

t->set\_name("shahd");

cout << t->get\_name() <<"\t" << t->get\_ID() << endl;

student d("Sara", 139032);

cout << d.get\_name() <<"\t"<< d.get\_ID() << endl;

delete t;

}

The heap is manually both allocated and deallocated by “new” and “delete”, it has a large space , so you can store variables of unknown size or just its large size.Its error type is memory leak.

Stack is a limited and smaller , it stores local variables and function calls,it is managed automatically without control from the programmer by management of stack, it is possible easily occurring stack overflow due to much memory is used or infinity recursion.

20. SOLID principles are important . why we need them?

We need them to many things including organization , testing, reusability , lower coupling,coherence,easier to understand, avoid many errors and maintainability.

21. explain each principle in the SOLID principle

1. Single-responsibility principle (SRP)

A class should have one and only one reason to change , meaning that a class should have only one job.

1. Open-closed principle(OCP)

Objects or entities should be open for extension but closed for modification.

1. Liskov substitution (LSP)

Objects of base class should be replaceable with instances of subtype without altering the correctness of the program.

1. Interface segregation

A client should never be forced to implement an interface that it does not use or clients should not be forced to depend on a method they do not use.

1. Dependency inversion principle (DIP)

The high level module and low level-module depend on the abstraction.

Abstractions should not depend on details , details should depend on abstraction.