

OOP's \Rightarrow object oriented programming

Q.1 Class के "Instances" को किस नाम से जाना जाता है?

- (A) member (B) member class (C) object (D) variable

Q.2 ऐसी language जो class के object पर माध्यारित हैं।
वह क्या कहलाती है।

→ oop's

Q.3 ----- object का blueprint होता है।

- (A) function (B) class (C) variable (D) NONE

Q.4 निम्न में से कौनसा एक C++ का Text editor नहीं है।

- (A) Eclipse (B) CodeLite (C) scienca (D) bluefish

Q.5 निम्न में से कौनसी एक C++ का editor-oop's
Language नहीं है।

- (A) Scala (B) kotlin (C) java (D) Smalltalk

Q.6 C++ की code updation के रूप में किसी भी update किया
पुका है।

- (A) 2 (B) 6 (C) 15 (D) Infinite

Q.7 C++ program का कौनसा extension होता है।

- (A) .cc (B) .cpp (C) .hpp (D) all

Q.8 निम्न में से कौनसा एक C++ का informal Name नहीं है?

- (A) .cc (B) .cpp (C) .hpp (D)

- (A) C++0x (B) C++1y (C) C++1z (D) C++0z

Q.9 C++ 2.0 ઓન્ડ Released કિંદ્યા ગયા થા?

- (a) 1998 (b) 1999 (c) 2000 (d) None

Q. 10 "C++1Y" का एक Released किया गया था?

- (a) 2003 (b) 2011 (c) 2014 (d) 2017

१॥ निम्न में से कौनसा C++ का Compiler है।

- (a) microsoft visual c++

- (b) Dev C++

- (c) Net Beans IDE

~~(d)~~ All

Q.12 Stat 1 :- C++ compiler based OOP's ~~exist~~ ^{exist} \rightarrow True.

Stat 2 :- oop's का class का design करने का तरीका
(अक्षयारणा) होता है। → True

Q.13 match the following :-

Col I

Col II

- (A) Bluefish → (1) Feature of OOP's
 - (B) Ruby → (2) OOP's Language
 - (C) modularity → (3) C++ Compiler
 - (D) C++ IDE → (4) C++ 11

੧. ੧੪ COPS ਅਤੀ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਲਾ-ਧਰਨ ਅੰਗਿਏ

- (a) portability (b) Re-usability (c) modularity ~~(d) all~~

9.15 stat 1 → C++ can case sensitive op's ~~diff~~ → [f]

Statement 2 → object के अंतर्गत मुख्यालयीय होने वाले object के अनुसार यहाँ किया जाता है। \rightarrow [True]

Stat 3 → SQL's are the high Level language. → True

Date
10/01/23

OOP's Concept using C++ object oriented programming

C++

Object :- (1) Function / method / procedure
(2) variable List → data type

(3) format specifier (access specifier) " %d "

↳ **OOP's**

void main () → function

{

int a=5, b=10;

Print (" %d ", a+b);

↳ **format specifier**

}

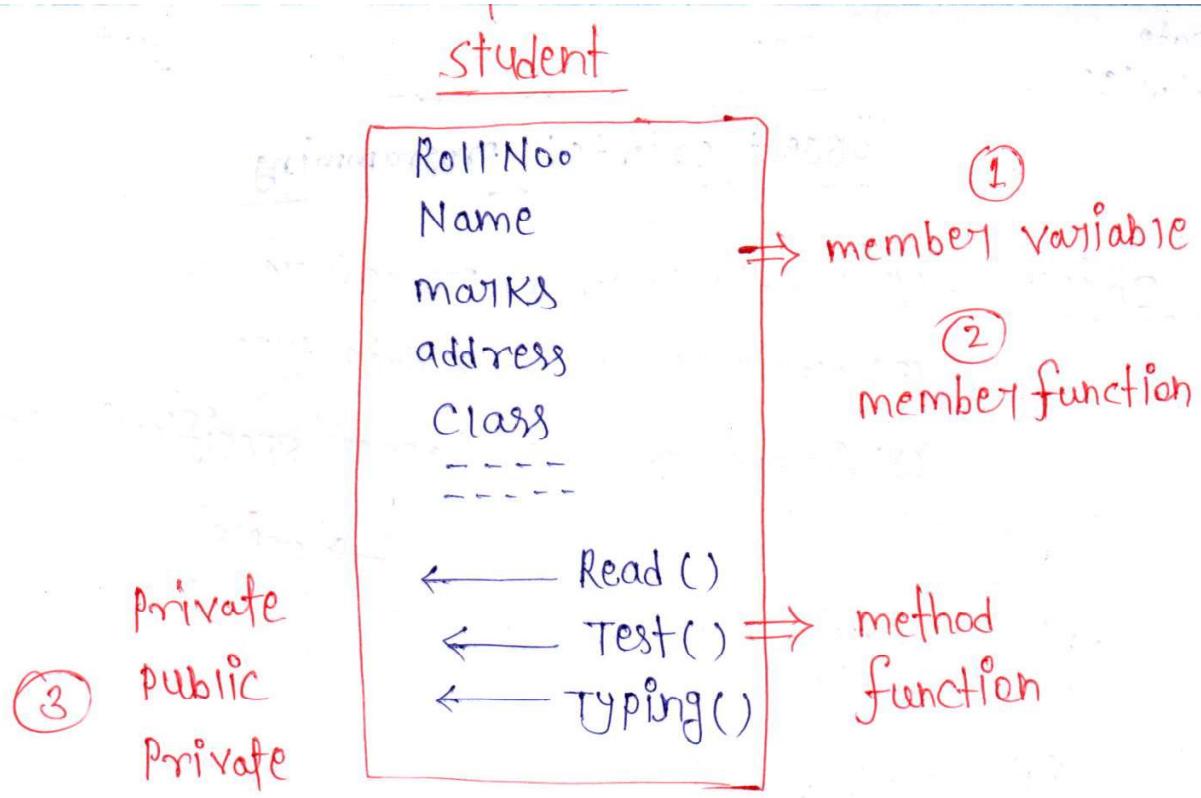


private
(3)
↳ **Access Specifier**

{ deposit ()
withdraw ()
enquiry () } **data type**

class A/C

variable (1) { Ac No. → int
Balance → float
Type → char } data type
↳ **member variable**



Class

1. member variable
2. member function
3. access specifier
4. data type

5. format specifier (%d, %c, %f)

6. Keywords

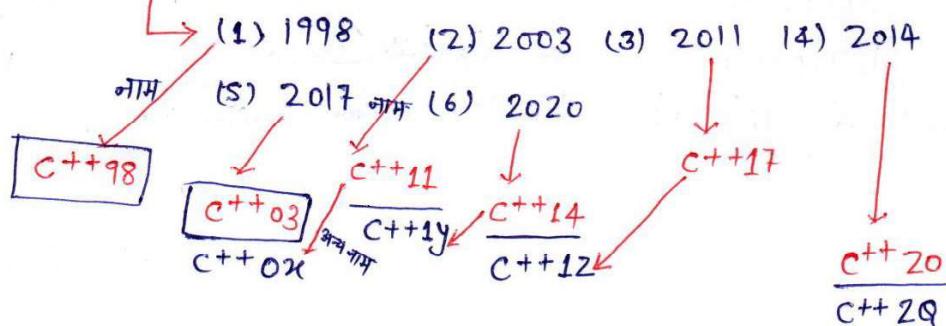
* Basic of C++ (in opp's) :-

यह एक उच्च स्तरीय शोरामिंग भाषा है, जिसे 1979-80 में बजारीने स्ट्राउप (Bjarne Stroustrup) ने बनाया था।

इसे भी AT&T Bell Laboratories में बनाया गया था।

American Telephone & Telegraph.

इसे पहले बार update किया गया है।



⇒ Text editors & Compiler :-

- | | |
|-------------------|------------------------|
| 1. Sublime Text 3 | 8. Ultra edit |
| 2. Notepad | 9. Eclipse |
| 3. Atom | 10. Code lite |
| 4. Notepad ++ | 11. visual studio code |
| 5. blue fish | 12. Net Beans |
| 6. Brackets | 13. QT edit |
| 7. Komoda edit | 14. QT creator |

⇒ Compilers of C++ →

- | | | |
|-----------------------|------------------|----------------|
| (1) TurboC++ | (6) Netbeans IDE | (12) Code Lite |
| (2) visual studio C++ | (7) Code blocks | |
| (3) Dev C++ | (8) QT creator | |
| (4) vim | (9) c lang | |
| (5) Eclipse IDE | (10) C++ builder | |

IDE → Integrated development Environment

* features of oop's (characteristics) c++ :-

- 1. Re-usability
- 2. portability
- 3. Extensibility
- 4. High Level
- 5. Case Sensitive
- 6. Compiler based
- 7. fast & efficient
- (8) Existence of libraries
- 9. modularity
- 10. Easy to debug
- 11. uses of structure
- 12. Easy to understand
- 13. syntax Based
- 14. Recursion supported
- 15. Simple
- 16. Structured
- 17. procedural
- 18. memory management
- 19. compatibility with 'c' language
- 20. very popular

Constructor & destructor

Allocate free

* OOP's Supported languages :-

- 1. C++
- 2. Java
- 3. Python
- 4. Ruby
- 5. Smalltalk
- 6. C Sharp
- 7. Red
- 8. C#
- 9. PHP
- 10. VB.net
- 11. JavaScript
- 12. Jain Advance
- 13. object Pascal
- 14. Swift
- 15. Sech
- 16. Kotlin
- 17. Lisp
- 18. delphi
- 19. Eiffel
- 20. LUO
- 21. Self
- 22. Erlang
- 23. TCI
- 24. Groovy
- 25. Go lang

* Advance of oop's :-

1. Provide high security → Password
2. Easy to troubleshoot → step by step solution
3. Better Productivity
4. Code flexibility → Changeable
5. Easy to design → class & object
7. Provide modular structure →
8. Hiding (complexity) of the program
9. help to keep "DRY" → Re-useability
"DON'T Repeat yourself" concept
10. Easy to maintain
11. Easy to problem solving

Finish
next →

* Extension of C++ :-

- | | | |
|---------|---------|----------|
| 1. .CPP | 6. .hpp | 9. .hxx |
| 2. .C | 7. .h | 10. .h++ |
| 3. .CC | 8. .H | |
| 4. .CXX | | |
| 5. .H | | |

Q.1 निम्न में से कौनसा है C++ का (oop's) आधार स्तरीय नहीं है।

- (A) Inheritance (B) Abstraction (C) Polymorphism (D) Class & Object

Q.2 oop's concept को किसके द्वारा inverted किया गया है।

- (A) Alan Kay (B) Bjarne Stroustrup (C) Tony Hoare (D) Alan Turing

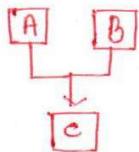
Q.3 निम्न में से कौनसा है Inheritance का Type नहीं है।

- (A) Hybrid (B) Hierarchical (C) multiLevel (D) multi hybrid

Q. 4 किसी भी parent class की feature को derived class में उपयोग में ले ना क्या अद्वितीय है।

- (A) Abstraction (B) hierarchical (C) multi level (D) multi hybrid
(E) inheritance (F) polymorphism (G) derivation

Q. 5 निम्नलिखित diagram पर क्या लिखित है।



- (A) hierarchical
(B) Hybrid
(C) multi level
(D) NONE

Q. 6 ऐसी class जिसे parent class से inherit किया जाता है, वह किस नाम से जानी जाती है।

- (A) child class (B) sub class (C) derived class (D) All

Q. 7 Re-usability को किसमें उपयोग किया जाता है।

- (A) Abstraction (B) encapsulation (C) Inheritance (D) All

Q. 8 OOP's concept कौनसे year में आया था।

- (A) 1954-55 (B) 1966-67 (C) 1974-75 (D) 1998-99

Q. 9 ----- निम्न में से कौनसे mechanism के द्वारा objects के class के अंतर्गत transfer किया जाता है।

- (A) paradigm (B) Encapsulation (C) Inheritance (D) None

Q. 10 match the following :-

- | | |
|----------------------|----------------------------|
| (A) Inheritance | (1) more than one form |
| (B) Polymorphism | (2) Collection in a single |
| (C) Encapsulation | (3) Transfer feature |
| (D) Data Abstraction | (4) Hide complexity |

* OOP'S 8 CONCEPT:-

- oop's, program लिखने का एक ही सा आदर्श होता है। जिसके अन्तर्गत class and object को शामिल किया जाता है।
- oop's, एक ही programming model होता है। जिसके अन्तर्गत सभी program, object पर आधारित होते हैं।
- oop's concept की अध्यारणा को " Alan Kay " के द्वारा सन् 1966-67 में विवरित किया गया था।
- oop's concept निम्नलिखित 4 भाषार संम्बोध पर आधारित होता है।

1. INHERITANCE
2. ENCAPSULATION
3. POLYMORPHISM
4. ABSTRACTION

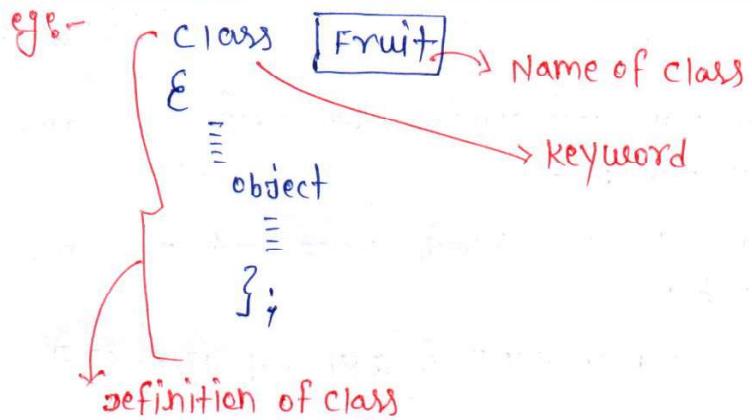
- oop's के अन्तर्गत class and object को बनाया जाता है। क्योंकि प्रत्येक program के साथ में संबंधित object जुड़े हुए होते हैं।

* Class object :-

- एक class प्रत्येक object के लिए एक template होती है।
- एक class सभी object के लिए blueprint ↑ pre-defined होती है।
- data type :-
 - pre-defined → int, float, char, double
 - user-defined - class
- एक class user-defined data type होती है जिसके माध्यम से प्रत्येक object को describe किया जाता है।
- एक class को माध्यम से object को behavior को describe किया जाता है।
- एक object class का छोटा होता है जिसे class को एक भाग " instance of the class " के नाम से जाना जाता है।

Class

- Blueprint
- Template
- user defined data type
- "class" keyword is used make a class
- syntax :-



object :-

- "Instance of class"
- variable (member variable)
- Function (method)/member Function
- access Specifier (public, private, protected)

*

```

class {
    private:
        int a,b,c,r;
    public:
        void multiply()
    {
        a=2, b=3;
        r=a*b; → ② output
        cout<<r;
    }
}

```

access specifier

```

void main()
{
    mult obj;
    obj multiply();
}

```

* class abc

{

private:

int x, y, z;

public:

void display()

{

x=5, y=15;

25

z = x+y - x+y; ~~x~~; ~~5+15-5+15~~ 5+5-5+15*5

~~20~~

15 + 75
= 190

cout << z;

}

void main()

{

class

abc a;

object

a. display();

}

* class & object:

class xyz

{

private:

int a, b, c;

public:

void int()

{

a=5, b=20;

}

```

void cal()
{
    n = a + b * a - 5;
}

```

$$s + 20 * s - s$$

$$100 + 8 - 8$$

$$\text{out} = \boxed{100}$$

```

void show()
{
    cout << n;
}

```

```

void main()
{
    xyz obj;
    obj.ini();
    obj.cal();
    obj.show();
}

```

* Class abc

```

private:
    int n;
public:
    int disp(int y)
    {
        n = y;
        n = n * n;
        , Return n
    }

```

```

void main()
{
    int n;
    abc obj;
}

```

$$n = obj.\text{disp}(10);$$

$\text{cout} << n;$

$$\boxed{100}$$

* class abc

{

 private :

 int a, b, x;

 public :

 Void main()

 {

 a = b, b = n, x = a

 -- return --

* class swap

{

 int a, b, c;

 public :

 Void disp()

 {

 a = 5, b = 20;

 c = a;

 a = b;

 b = c;

 Cout << a;

 Cout << b; } } { Cout << a << b;

 }

}

 Void main()

{

 swap obj;

* class swap

```
{  
    int a, b, c;
```

```
public:
```

```
void disp (int a, int b)
```

```
{
```

```
    c = a;
```

```
    a = b;
```

```
    b = c;
```

```
    cout << a << b
```

```
}
```

```
void main ()
```

```
{
```

```
    swapobj;
```

```
    obj.disp (5, 7);
```

```
}
```

* class path

```
{
```

```
private:
```

```
int a;
```

```
public:
```

```
void disp (int b)
```

```
{
```

```
    a = b;
```

```
a = a * b * 3 - 5;
```

```
cout << b << a;
```

```
}
```

```
6 → 103
```

```
void main ()
```

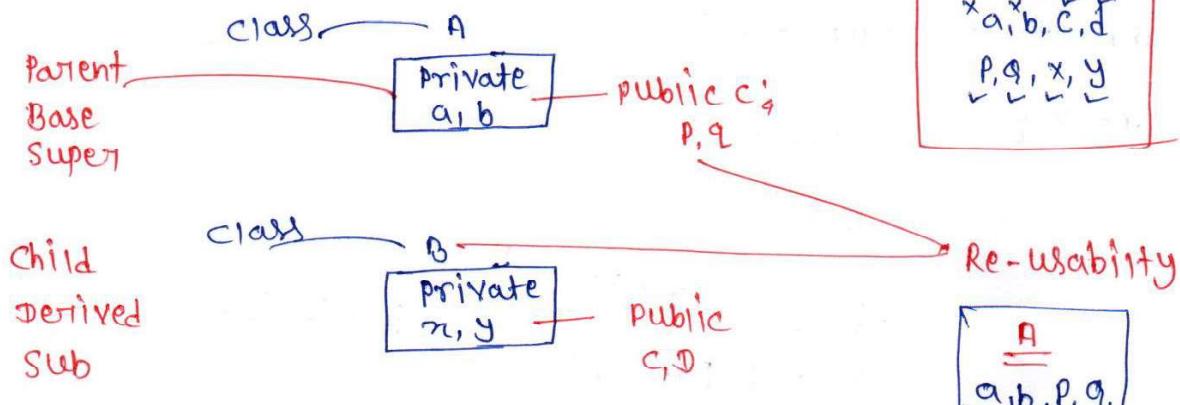
```
{
```

```
path p;
```

```
p.disp (6);
```

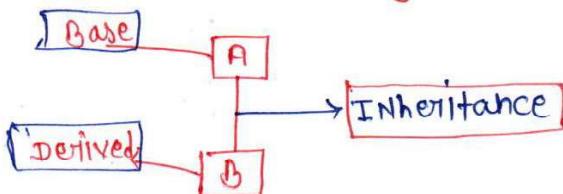
* PILLARS :-

1. INheritance :- इनरीटेन्स



- Inheritance, एक ही प्रोसेस होता है, जिसके अन्तर्गत **Base class** का **object** नों **Derived class** के द्वारा (**Inherit**) प्राप्त किया जाता है।
- Inheritance, एक ही तरीका होता है, जिसमें **Base class** के द्वारा **Derived class** के लिए **objects** को प्राप्त किया जाता है।

↳ इसका सबसे बड़ा फायदा 'Re-useability' (रुनिडपचोगिता) होता है।



* Base class / parent / super :- Base class, एक main class होती है, जैसा पर derived class के द्वारा, Base class का **object** नों inherit (प्राप्त) किया जाता है।

* Derived / child / sub class :-

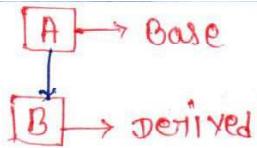
→ यह हीसे class होती है, जिसे Base class में से inherit (प्राप्त) किया जाता है।

* TYPES OF INHERITANCE (त्रिकार)

- S → single inheritance (सिंगल)
- multiple inheritance (मल्टीपल)
- multi Level .. (मल्टी लेवल)
- Hierarchy .. (हीरार्की)
- Hybrid .. (हायबिड)

(i) single inheritance:-

जिसमें one base class और one derived class हो, उसे single inheritance कहलाता है।



Syntax :-

Class A

{

 |||

 };

Class B

{

 |||

 };

 → colon (कोलन)

 → inherits (इन्हें)

: public A

* class test

{

 private :

 int x, y;

 public :

 Void cal (int a)

{

 x = a;

 y = x * 9 + x * 5;

Void main()

{

 test p;

 p.cal(9);

}

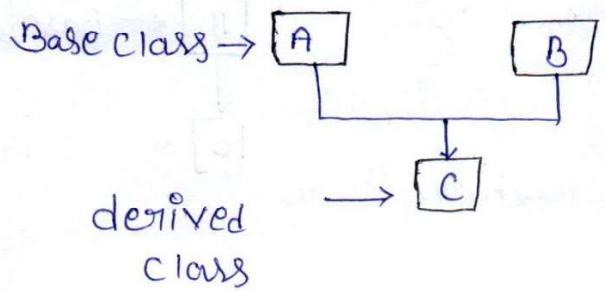
$$81 + 45 \Rightarrow 125$$

cout << y;

}

3.

2. multiple inheritance:-



→ जब multiple Base classes को सिक्के single derived class करा inherit किया जाता है तो वह multiple inheritance कहलाती है।

Syntax:-

Class A
{
 ;
 ;
 ;
}

Class B
{
 ;
 ;
 ;
}

Class C : public A, public B

{
 ;
 ;
 ;
}

Q. "Class : public A", is a syntax of ---- inheritance.

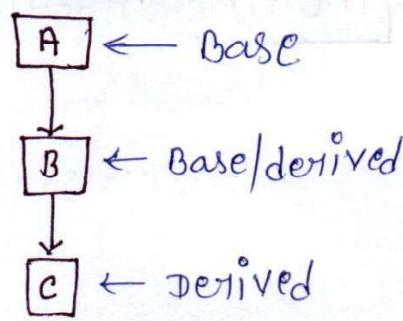
→ Single inheritance

Q. "Class : public X; public Y" ?

→ multiple inheritance

3. multi level inheritance :-

→ जब derived class को किसी अन्य
किसी derived class को दूसरे inherit
किया जाता है तो वह multilevel inheritance
नामलाता है।



Syntax:-

Class A

{

 ≡

 { ;

Class B : public A

{

 ≡

 { ;

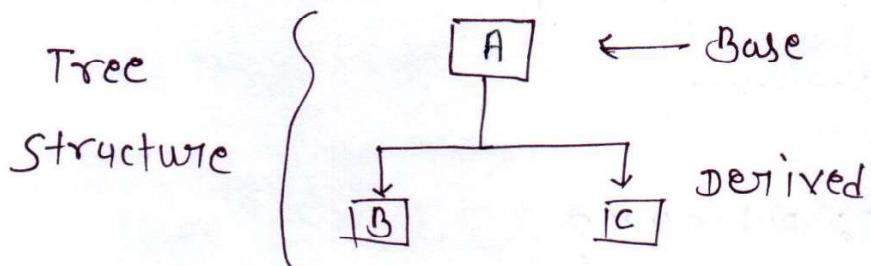
Class C : public A

{

 ≡

 { ;

4. Hierarchical inheritance :-



→ जब किसी भी Base class को multiple derived classes के
inherit किया जाता है तो वह Hierarchical Inheritance
नामलाता है।

Syntax :-

Class A

{
≡
};

Class B : public A

{
≡
};

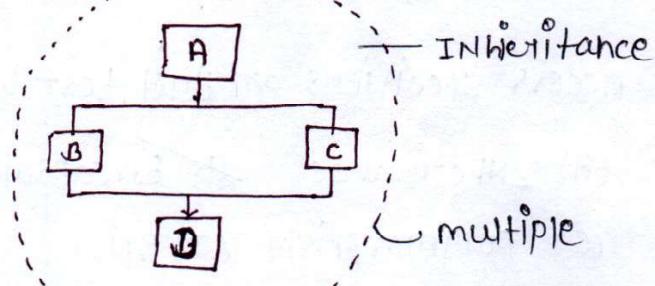
Class C : public A

{
≡
};

Class D : public A

{
≡
};

5. Hybrid Inheritance :-



→ जब किसी भी ही से अधिक inheritance को आपस में group कर दिया जाता है तो वह Hybrid inheritance कहलाती है।

Syntax :-

Class A

{
≡
};

Class B : public A

{
≡
};

Class C : public A

{
≡
};

Class D : public B, public C

{
≡
};

* ENCAPSULATION :-

Q.1 निम्न में से कौनसा encapsulation ओं प्रकार नहीं है।

- (a) class encapsulation
- (b) method encapsulation
- (c) variable encapsulation

~~(d)~~ All

Q.2 निम्न में से encapsulation के संदर्भ में कौनसा व्याप्ति सत्य है।

Stat 1 : → यह data ओं उन्हें और आधिक security provide करता है।

Stat 2 : → इसके माध्यम से Re-useability feature ओं provide करता है।

Stat 3 : → अलग सभी objects को एक unit के अन्तर्गत बंध (bind) कर के रखता है। → 

Q.3 Access specifiers का सबसे best use किसके अन्तर्गत होता है।

- (A) Inheritance
- ~~(B)~~ Encapsulation
- (C) polymorphism
- (D) All

Q.4 ----- member ओं derived class को किसी उपयोग में लिया जा सकता है।

- (a) private
- ~~(b)~~ protected
- (c) Normal
- (d) None of these

Q.5 निम्न में से कौनसे member ओं program में ऑफ अक्सेस किया जा सकता है।

- (a) private
- (b) protected
- (c) normal
- ~~(d)~~ None of these

Q.6 Class के private member ओं derived class को किसी भी विद्युत संचय से class किया जा सकता है।

- (a) friend keyword
- (b) derived keyword
- (c) grant keyword
- (d) accessibility key word

Q.7 math of the following : -

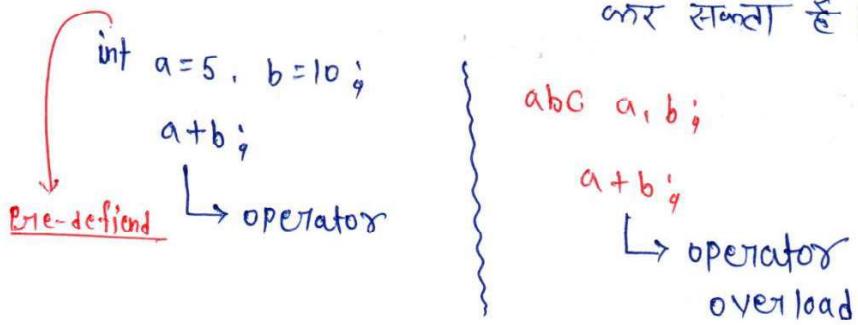
- (A) friend function.
 - (B) private class
 - (C) Derived class
 - (D) protected
- (1) access specifier
 - (2) Encapsulation
 - (3) access private function
 - (4) Inheritance

2. Encapsulation :-

Date 21/01/23

(3) operator overloading :-

अचार्प पहले बना हुआ operator user define datatype के साथ में भी काम कर सकता है।



⇒ The following operators can not be overloaded.

1. scope Resolution operator
2. Ternary operator (`? :`)
3. member selector `"(.)"`
4. size of operator

* we can overload the following operators:-

निम्न लिखित operator का overload किया जा सकता है।

(i) unary operator

$\begin{matrix} + \\ - \\ \sim \end{matrix}$

(ii) binary operator

$+, -, *, /$

* 1. Scope Resolution operator (`::`)

```
class abc
{
    public:
        void cal()
    {
        = x
    }
    :: void cal()
}
```

→ इस operator की सहायता से function की definition का class में प्राप्त लिख सकते हैं।

→

* Ternary operator :-

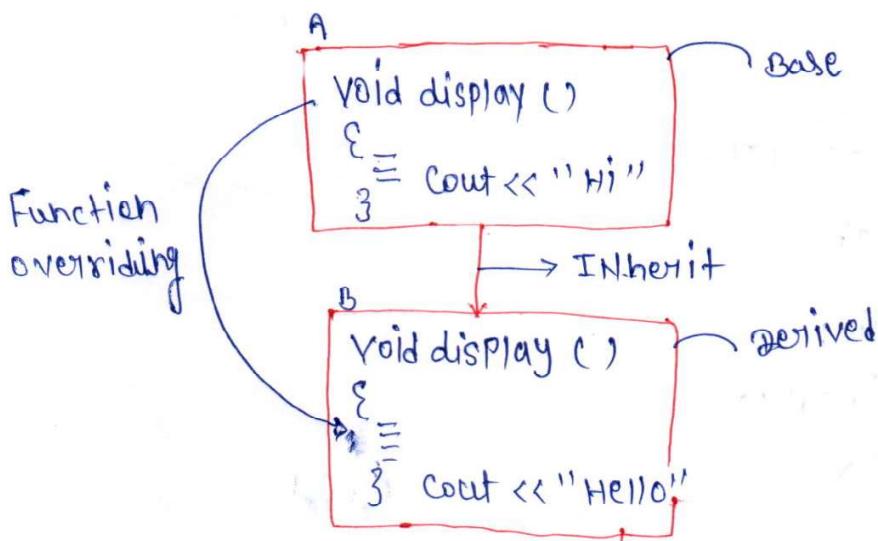
```
int a=5, b=50;  
a = (a < b) ? 30 : 32;  
cout << a;
```

* Member selector :-

```
Class abc  
{  
    public:  
        void disp()  
    }  
    cout << "RSSB";  
}  
void main()  
{  
    abc obj;  
    obj.Lisp();  
}
```

member selector

* Run TIME polymorphism :-



→ जब कोई नाम के Function को definition base class
और derived class दोनों classes में लिखी हुई होती है तो
program को run करते समय compiler के द्वारा होने वाले
function को determine नहीं किया जाता है। जिसे ही RUN
time polymorphism के नाम से जाना जाता है।

Example:-

Function Overriding-

→ जब कोई नाम के दो function को over
class के माध्यम से उपयोग में लिया जाता है तो वह
Function-overriding कहलाता है।

अर्थात् इसके अंतर्गत **base class** के **Function** को **derived class**
के माध्यम से उपयोग में लिया जाता है।

149

Date
23/1/22

→ enumeration :- (एन्यूमरेशन)

अह एक प्रैसा process होता है। इसके अन्तर्गत numeric value की
वर define data type को assigned किया जाता है।

→ इसके अंतर्गत value को 0 से assigned किया जाता है।

→ ऐ value से आधिक होती रहती है।

e.g. enum test { ⁰mango, ¹orange, ²apple};

Cout << orang << mango;
 ↓ ↓

O/P → 1, 0

e.g. enum coaching { pathsalai=1, abiggan=3, missian }⁴

Cout << "Top in IA" pathsalai << "Number"
O/P: Top in IA 1 Number

e.g. disp { ⁰A, ¹B, ²C = ⁷7, ⁸8, E = ¹²12, F, ¹³13, ¹⁴14 };

Cout << F-B << " << B+D << ", << A*B
O/P ⇒ 6 , 22 , 0

* \Rightarrow Infin Inline function \rightarrow इसके अन्तर्गत function call overhead को कम किया जाता है।

जबकि जबकि एक फ़ंक्शन का कोई overhead नहीं होता है।

जबकि inline function का overhead कम होता है।