**QUESTION BANK C++**

**LOW ACHIEVERS**

Q1. Explain Operators?

Ans: 1. Arithmetic operators :-

Those operators are operates only on numeric data types operands are known as arithmetic operators.

2. Unary - Result is the negation of operand’s value (Reverse the sign of operand’s value)

If a=5 then – a means -5.

If a = - 4 then – a means 4.

3. Unary + The result is the value of its

operand

If a=5 then +a means 5.

If a = - 4 then +a means -4.

4. + (Addition Operator) : Addition ( it adds two numbers) 4+20 results is 24.

If a = 5 then a + 5 results 10.

5. - ( Subtraction Operator) :Subtraction ( Subtract the second operand from first)

14 – 3 evaluates to 12.

If a = 2 , b = 3 then b – a evaluates 1.

6.\*(Multiplication Operator) :Multiplies the values of its operands 3\*4 evaluates to 12.

If a=2, b=3 then a\*b evaluates

to 6.

7. / (Division Operator) :Divides its first operand by the second 100/5 evaluates 20.

If a =10 , b = 5 then a/b evaluates 2

8. % (Modulus Operator) :It produce the remainder of dividing the first operand by second

19%6 evaluates to 1.

If a = 14 , b = 3 then a%b evaluates to 2.

Modulus operator requires that both operands be integer and second operand be non-zero.

9. Increment and Decrement Operators (++ , - -) :

The increment operator (++) adds 1 to its operand and decrement operator (--) subtract one from

its operand. In other word a = a + 1; is same as ++a; or a++; & a = a – 1 ; is same as --a; or a--;

Both the increment & decrement operators comes in two version :

(i)Prefix increment/decrement :- When an increment or decrement operator precedes its

operand, it is called prefix increment or decrement(or pre-increment / decrement). In prefix increment/decrement , C++ perform the increment or decrement operation before using the value of the operand. e.g.,

If sum = 10 and count =10 then Sum = sum +(++count); First count incremented and then evaluate sum = 21.

(ii)Postfix increment/decrement :- When an increment or decrement operator follows its operand, it is called postfix increment or decrement (or post-increment / decrement).

In postfix increment/decrement , C++ first uses the value of the operand in evaluating the expression before incrementing or decrementing the operand’s value. e.g.,

If sum = 10 and count =10 then

Sum = sum +(count++);

First evaluate sum = 20 , and then increment count to 11.

Q2. Explain Relational Operator?

Ans: These operators are used to compare two values. If comparison is true, the relational expression results into the value 1 and if the comparison is false its result will be 0.

The six relational operators are:

1. Operator Meaning
2. = = Equal to

(c ) != Not equal to

(d) < Less than

(e) <= Less than or equal to

(f) > Greater than

(g) >= Greater than or equal to

Q3. Explain Logical Operators :

In addition to the relational operator, C++ contains three logical operators. Relational operators often are used with logical operators to construct more complex decision making expressions.

Q4. Explain Assignment Operator:

Ans: C++ offers an assignment operator (=) to assign a value to an identifier. The assignment statement that make use of this operator are written in the form :

var = expression ;

where var generally represents a variable and expression may be a constant or a variable or an

expression.

Q5. Explain Shorthand operators.

Ans: C++ offers special shorthand operators that simplify the coding of a certain type of assignment statement .

e.g.,

a = a + 10 ; can be written as a+=10 ;

This shorthand works for all binary arithmetic operators. The general form of this shorthand is

Var = var operator expression ; is same as var operator = expression ;

Following are some examples of C++ shorthands:

x -=10 ; equivalent to x = x -10 ;

x\*=3 ; equivalent to x = x \* 3 ;

x/=2 ; equivalent to x = x/2 ;

x%=z equivalent to x = x % z ;

Q6. Explain Conditional operator ( ? : )

Ans: The conditional operator (? :) is a ternary operator i.e., it require three operands. The general form of conditional operator is:

expression1? expression2: expression3 ;

Where expression1 is a logical expression , which is either true or false. If expression1 evaluates to true i.e., 1, then the value of whole expression is the value of expression2, otherwise, the value of the whole expression is the value of expression3.

For example 14 min = a<b? a : b ;

Here if expression (a<b ) is true then the value of a will be assigned to min otherwise value of b

will be assigned to min.

Q7. Explain Comma operator ( , )

Ans: The comma operator (,) is used to separate two or more expressions that are included where only one expression is expected. When the set of express ions has to be evaluated for a value, only the rightmost expression is considered.

For example, the following code:

a = (b =3 , b +2 );

Would first assign the value 3 to b, and then assign b+2 to variable a. So, at the end, variable a

would contain the value 5 while variable b would contain value 3.

Q8. Explain sizeof() operator?

Ans: This operator returns the size of its operand in bytes. The operand may be an expression

or identifier or it may be a data type.

a= sizeof (char);

This will assign the value 1 to a because char is a one-byte long type.

Q9. Explain Expressions?

Ans: An expression in C++ is any valid combination of operators, constants, and variables.

Pure Expressions:-

If an expression have all operand of same data types then it is called a pure expression.

Mixed Expressions :-

If an expression have operands of two or more different data types then it is called a mixed expression.

Arithmetic Expressions:-

Arithmetic expression can either be integer expressions or real expressions. Sometimes a mixed expression can also be formed which is a mixture of real and integer expressions.

Integer Expressions:- Integer expressions are formed by connecting all integer operands using

integer arithmetic operators.

Real Expressions:-

Real expressions are formed by connecting real operands by using real arithmetic operators.

Logical Expressions:-

The expressions which results evaluates either 0 (false) or 1 (true) are called logical expressions. The logical expressions use relational or Logical operators.

Q10. Explain Type Conversion:-

Ans: The process of converting one predefined data type into another is called type conversion.

C++ facilitates the type conversion in two forms:

(i)Implicit type conversion

:- An implicit type conversion is a conversion performed by the compiler without programmer’s intervention.

An implicit conversion is applied generally whenever different data types are intermixed in an expression. The C++ compiler converts all operands upto the data type of the largest data type’s operand, which is called type promotion.

(ii)Explicit type conversion

:- An explicit type conversion is user-defined that forces an expression to be of specific data type.

Q11. Explain Type Casting?

Ans: The explicit conversion of an operand to a specific type is called type casting.

Type Casting Operator - (type) :-Type casting operators allow you to convert a data item of a given type to another data type. To do so , the expression or identifier must be preceded by the name of the desired data type , enclosed in parentheses . i. e., (data type) expression Where data type is a valid C++ data type to which the conversion is to be done.

For example , to make sure that the expression (x+y/2) evaluates to type float , write it as:

(float) (x+y/2)

**AVERAGE ACHIEVERS**

Q12. Explain Precedence of Operators

Ans:- Operator precedence determines which operator will be performed first in a group of operators with different precedence. For instance

5 + 3 \* 2 is calculated as 5 + (3 \* 2), giving 11

Q13. Explain Statements?

Ans: Statements are the instructions given to the Computer to perform any kind of

action.

Q14. Explain Null Statement?

Ans: A null statement is useful in those case where syntax of the language requires the presence of a statement but logic of program does not give permission to do anything then we can use null statement. A null statement is nothing only a ;. A null (or empty statement have the following form:

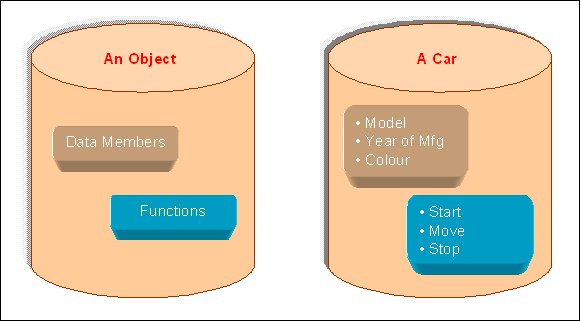
; // only a semicolon (;)

Q15. Explain Compound Statement ?

Ans: A compound statement is a group of statements enclosed in

**Q16. Explain Objects?**

Ans: Object is the basic unit of object-oriented programming. Objects are identified by its unique name. An object represents a particular instance of a class.



An Object is a collection of data members and associated member functions also known as methods.

**Q17. Explain Classes?**

**Ans:** Classes are data types based on which objects are created.

* Thus a Class represents a set of individual objects. Characteristics of an object are represented in a class as Properties. The actions that can be performed by objects become functions of the class and are referred to as Methods.
* Classes are the blueprints upon which objects are created. E.g when we design a map of the house, the architect first designs it. Once it is designed, the raw material is used to build the house. In this example, Design of the house is CLASS (blueprint) and the house built on the basis of design is an object.

No memory is allocated when a class is created. Memory is allocated only when an object is created, i.e., when an instance of a class is created.

**Q18. Explain Inheritance?**

Ans: Inheritance is the process of forming a new class from an existing class or base class. The base class is also known as parent class or super class. The new class that is formed is called derived class.

* Derived class is also known as a child class or sub class. Inheritance helps in reducing the overall code size of the program, which is an important concept in object-oriented programming. It is the process by which one class inherits the properties of another Class.

Q19. Explain **Data Abstraction?**

**Ans:**

Data Abstraction increases the power of programming language by creating user defined data types.

* Data Abstraction also represents the needed information in the program without presenting the details.

The concept of abstraction relates to the idea of hiding data that are not needed for presentation. The main idea behind data abstraction is to give a clear separation between properties of data type and the associated implementation details.

An **Abstract Data Type** is defined as a data type that is defined in terms of the operations that it supports and not in terms of its structure or implementation.

Q20. Explain the features of Object oriented Programming?

Ans: FEATURES OF OBJECT ORIENTED PROGRAMMING:

•Inheritance:

Inheritance is the process of forming a new class from an existing class or base class. The

base class is also known as parent class or super class.

•Derived class is also known as a child class or sub class. Inheritance helps in

reusability of code , thus reducing the overall size of the program

•Data Abstraction

It refers to the act of representing essential features without including the background

details .Example : For driving , only accelerator, clutch and brake controls need to be

learnt rather than working of engine and other details.

Data Encapsulation:

•It means wrapping up data and associated functions into one single unit called class..

•A class groups its members into three sections :public, private and protected, where

private and protected members remain hidden from outside world and thereby helps in

implementing data hiding.

•Modularity :

The act of partitioning a complex program into simpler fragments called modules is

called as modularity.It reduces the complexity to some degree and It creates a number of well defined boundaries within the program .

•Polymorphism

means many and morphs mean form, so polymorphism means one name multiple

forms.

•It is the ability for a message or data to be processed in more than one form.

•C++ implements Polymorhism through Function Overloading .

Q21. Explain Access Specifiees in Class.

Ans: Access specifiers in Classes:

Access specifiers are used to identify access rights for the data and member functions of the class.

There are three main types of access specifiers in C++ programming language:

•private

•public

•protected

Member-Access Control

Type of AccessMeaning

PrivateClass members declared as privatecan be used only by memberfunctions and friends (classes or functions) of the class.

Protected

Class members declared as protected can be used by member functions and friends (classes or functions) of the class. Additionally, they can be used by classes derived from the class

.

Public

Class members declared as public can be used by any function

Importance of Access Specifiers :

Access control helps prevent you from using objects in ways they were not intended to be

used. Thus it helps in implementing data hiding and data abstraction.

**HIGH ACHIEVERS**

Q22. Explain Function?

Ans: Name given to group of statements that does some specific task and may return a

value. Function can be invoked(called)any no. of time and anywhere in the program.

Q23. Explain Function prototypes?

Ans: Function declaration that specifies the function name, return type and

parameter list of the function.

syntax: return\_type function\_name(type var1,type var2,....,type varn );

Q24. Explain Actual and Formal Parameters?

Ans: Variables associated with function name during function call statement.

Formal ParametersVariables which contains copy of actual parameters inside the function definition.

Q25.What are Local variables?

Ans:Declared inside the function only and its scope and lifetimeis function only and hence

accessible only inside function.

Q26. What are Global variables?

Ans:Declared outside the function and its scope and lifetime is whole program and hence

accessible to all function in the program from point declaration.

Example :

#include <iostream.h>

int a=20;

// global

void main()

{

int b=10;

// local

cout<<a<<b;

}

Q27. What is Passing by value?

Ans:In this method separate memory created for formal arguments and if

any changes done on formal variables , it will not affect the actual var

iables.So actual

variables are preserved in this case

Q28. What is Passing by address/reference?

Ans:Passing by address/reference:

In this method no separate memory created for formal variables i.e formal variables share the same location of actual variables and hence any change on formal variables aut

omatically reflected back to actual variables.

Example :

void sample( int a, int &b)

{

a=a+100;

b=b+200;

cout<<a<<b;

}

void main()

{

int a=50, b=40;

cout<<a<<b; // output 50 40

sample(a,b) // output 150 240

cout<<a<<b; // output 50 240

}

Q29. What is Function overloading?

Ans. Processing of two or more functions having same name but different list of parameters

Function overloading in C++

A function name having several definitions that are differentiable by the number or types

of their arguments is known as function overloading.

Example : A same function print() is being used to print different data types:

#include <iostream.h>

class printData

{

public:

void print(int i) {

cout << "Printing int: " << i << endl;

}

void print(double f) {

cout << "Printing float: " << f << endl;

}

void print(char\* c) {

cout << "Printing character: " << c << endl;

}

};

int main(void)

{

printData pd;// Call print to print integer

pd.print(5);// Call print to print float

pd.print(500.263);// Call print to print character

pd.print("Hello C++");

return 0;

}

When the above code is compiled and executed, it produces following result:

Printing int: 5

Printing float: 500.263

Printing character: Hello C++

Q30 What is Function recursion?

Ans:Function that call itself either directly or indirectly.

Q31. What is Structure?-

Ans:Collection of logically related different data types (Primitive and Derived) referenced

under one name.

e.g.

struct employee

{

int empno;

char name[30];

char design[20];

char department[20];

}

Declaration:

employee e;

Input /Output : cin>>e.empno; // members are accessed using dot(.) operator.

cout<<e.

empno;

Q32. What is a Nested structure?

Ans:A Structure definition within another structure.

A structure containing object of another structure.

e.g.

struct address

{

int houseno;char city[20];char area[20];long int pincode;};

struct employee

{

int empno;

char name[30];char design[20];char department[20];

address ad; //nested structure

}

Declaration:

employee e;