Question & Answers

1) Define the following?

a) Class b) Object c) Inheritance

Class :- Collection of objects is called as class. It is a logical entity

Object :- Any entity that has state and behaviour is known as an object.
For example :- Chair, pen, table etc
It can be physical and logical

Inheritance :- When one object acquires all the properties and behaviors of parent object is known as Inheritance.
It provides code reusability.
It is used to achieve runtime polymorphism

2) Briefly Explain Operators in CPP?

An operator is a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise etc. There are following types of operations in C language.

* Arithmetic Operators
* Relational Operators
* Logical Operators
* Bitwise Operators
* Assignment Operators
* Unitary Operators.
* Ternary or Conditional Operator.
* Misc Operator.

Binary Operators are :-

Operator	Type
+, -, *, /, %	arithmetic operators
<, <=, >, >=, ==, !=	Relational operators
&&, ||, !	Logical operators
&, |, <<, >>, ~, ^	Bitwise operators.
=, +=, -=, *=, %=, /=	assignment operators

Unary Operators & Ternary Operator

Operator	Type
++, --	Unary Operator
? :	Ternary or conditional operators

The precedence of operator specifies that which operator will be evaluated first & next. The associativity specifies the operators direction to be evaluated. It may be left to right or right to left.

④

1 int data = 5 + 10 * 10;

The "data" variable will contain 105 because * is evaluated before + (addition operator). The precedence and associativity of C++ is given below

Category	Operator	Associativity
Postfix	( ) [] → , ++ --	left to right
Unary	+ - ! ~ ++ -- (type)*	right to left
Multiplicative	* / %	left to right
Additive	+ , -	right to left
Shift	<< >>	left to right
Relational	< <= > >=	left to right
Equality	== != / td >	right to left
Bitwise XOR	^	left to right
Bitwise AND	&	left to right
Bitwise OR	1	right to left
Logical AND	&&	left to right

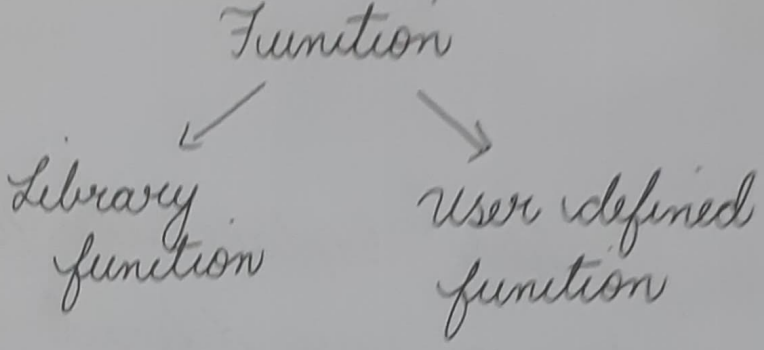
logical AND	&&	left to right
logical OR	||	left to right
conditional	?:	Right to left
Assignment	=+=-=*=/=%=>>=<<=&=^=|=	Right to left
Comma	,	left to right

3) Define function? Explain the procedure for creating a user defined function?

The function in C++ language is also known as procedure or subroutine in other programming language. To perform any task, we can create function. A function can be called many times. It provides modularity and code reusability.

Library Function are the functions which are declared in the C++ header files such as. cal(x), cos(x), exp(x) etc

User defined funtions are the funtion which are created by the c++ program so that he/she can use it many times. It reduces complexity of a big program and optimises the code.



Declaration of funtion

The syntax of creating funtion in c++ language is given below:

```
return-type funtion_name(data_type parameter
{
  //code to be executed.
}
```

User defined funtion:

1) Funtion declaration

2) Function calling
3) Function defination

4 Types of function declaration

1) no return type no parameters
Syntax : void fun_name();
ex : void add(); // function declaration statement

2) no return type with parameters
Syntax = void fun_name (parameters);
ex: void add (int a, int b);

3) Return type, with no parameter
Syntax : data_type fun_name();
ex : int add();
float add();

4) Return type, with parameters
Syntax - data_type fun_name();
ex : int odd (int a, int b);
ex : float sum (float x, float y, float z)
ex : float addition (int a, float b)

Function Calling

Call by value
call by reference

Syntax: data_type var_name = fun_name(); // int x = add();

data_type var name = fun_name(arguments); // int x = odd(a,b);

fun_name(); // odd();
fun_name(arguments); // odd(x,y);

Function defination.

1) Function header
2) Function body

Function header:-
void add().
int add(int a, int b).

function body
void odd() // function header

{ ...
...
...
}.

// function body } function definition

4) Explain about Dynamic Memory allocation?

In C language, we can dynamically allocate memory using malloc() & calloc() function where pointer is used.
- Memory space required can be specified at the time execution.
- C supports allocating & freeing memory dynamically using lib library routines

Malloc:
allocates requested number of bytes & return a pointer to the first byte of the allocated space

Calloc: allocates space for an array of elements, initializes them then returns a pointer to the memory.

- free: free previously allocated space.
- realloc: Modifies the size of previously allocated space.
- A block of memory can be allocated using the function malloc

general format

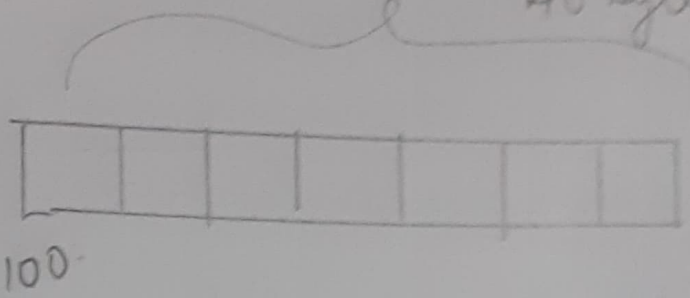
Ptr = (type *) malloc (byte-size);

Examples

P = (int *) malloc (10 * sizeof (int));

- A memory space equivalent to "10 times the size of an int" byte is reserved.
- The address of the first byte of the allocated memory is assigned to the pointer P of type int.

P=100



char* cptr

cptr = (char*) malloc (20);

Allocates 20 bytes of space for the pointer cptr of type char.

sptr : (struct student*) malloc (10* sizeof (struct student))

malloc always allocates a block of contiguous bytes the allocation can fail if sufficient contiguous memory space is not available.

If it fails, malloc returns null

—— x ——