1. What are the different data types present in C++?

The 4 data types in C++ are given below:

Primitive Datatype(basic datatype). Example- char, short, int, float, long, double, bool, etc.

Derived datatype. Example- array, pointer, etc.

Enumeration. Example- enum

User-defined data types. Example- structure, class, etc

2. What is the difference between C and C++?

The main difference between C and C++ are provided in the table below:

C	C++
C is a procedure-oriented pro	C++ is an object-oriented programming language.
C does not support data hidin	Data is hidden by encapsulation to ensure that data structures and operators are used as intended.
C is a subset of C++	C++ is a superset of C.
Function and operator overloa supported in C	Function and operator overloading is supported in C++
Namespace features are not p	Namespace is used by C++, which avoids name collisions.
Functions can not be defined i	Functions can be defined inside structures.
calloc() and malloc() functions memory allocation and free() f memory deallocation.	new operator is used for memory allocation and deletes operator is used for memory deallocation.

3. What are class and object in C++?

A class is a user-defined data type that has data members and member functions. Data members are the data variables and member functions are the functions that are used to perform operations on these variables.

An object is an instance of a class. Since a class is a user-defined data type so an object can also be called a variable of that data type.

A class is defined as-

```
class A{
private:
 int data;
public:
 void fun(){

 }
};
```

4. What is the difference between struct and class?

In C++ a structure is the same as a class except for a few differences like security. The difference between struct and class are given below:

Structure	Class
Members of the structure are pub	Members of the class are private by default.
When deriving a struct from a cla specifiers for base class/struct are	When deriving a class, default access specifiers are private.

5. What is operator overloading?

Operator Overloading is a very essential element to perform the operations on userdefined data types. By operator overloading we can modify the default meaning to the operators like +, -, *, /, <=, etc.

For example -

The following code is for adding two complex number using operator overloading-

```
class complex{
private:
 float r, i;
public:
 complex(float r, float i){
  this->r=r;
  this->i=i;
 }
 complex(){}
 void displaydata(){
  cout<<"real part = "<<r<<endl;
  cout<<"imaginary part = "<<i<<endl;
 }
 complex operator+(complex c){
  return complex(r+c.r, i+c.i);
 }
};
int main(){
complex a(2,3);
complex b(3,4);
complex c=a+b;
c.displaydata();
return 0;
}
```

6. What is polymorphism in C++?

Polymorphism in simple means having many forms. Its behavior is different in different situations. And this occurs when we have multiple classes that are related to each other by inheritance.

For example, think of a base class called a car that has a method called car brand(). Derived classes of cars could be Mercedes, BMW, Audi - And they also have their own implementation of a cars

The two types of polymorphism in c++ are:

- Compile Time Polymorphism
- Runtime Polymorphism

7. Explain constructor in C++

The constructor is a member function that is executed automatically whenever an object is created. Constructors have the same name as the class of which they are members so that compiler knows that the member function is a constructor. And no return type is used for constructors.

Example:

```
class A{
 private:
  int val;
 public:
  A(int x){              //one argument constructo r
   val=x;
  }
  A(){                     //zero argument constructo r
  }
}
int main(){
 A a(3);

 return 0;
}
```

8. Tell me about virtual function

Virtual function is a member function in the base class that you redefine in a derived class. A virtual function is declared using the virtual keyword. When the function is made virtual, C++ determines which function is to be invoked at the runtime based on the type of the object pointed by the base class pointer.

9. Compare compile time polymorphism and Runtime polymorphism

The main difference between compile-time and runtime polymorphism is provided below:

Compile-time polymorphism	Run time polymorphism
In this method, we would com compile time which method w call is resolved by the compile	In this method, we come to know at run time which method will be called. The call is not resolved by the compiler.

It provides fast execution beca the compile time.	It provides slow execution compared to compile-time polymorphism because it is known at the run time.
It is achieved by function overl operator overloading.	It can be achieved by virtual functions and pointers.
Example -	Example -

```
int add(int a, int b){
return a+b;
```

```
class A{      public:
```

Compile-time polymorphism	Run time polymorphism

```
}
int add(int a, int b, int c){        return a+b+c;
}
int main(){
cout<<add(2,3)<<endl;
cout<<add(2,3,4)<<endl;

     return 0;
}
```

```
                                virtual void fun(){
                                     cout<<"base ";
          }
};
class B: public A{      public:
          void fun(){                    cout<<"derived ";
          }
}; int main(){       A *a=new B;
a->fun();

     return 0;
}
```

10. What do you know about friend class and friend function?

A friend class can access private, protected, and public members of other classes in which it is declared as friends.

Like friend class, friend function can also access private, protected, and public members. But, Friend functions are not member functions.

For example -

```
class A{
private:
  int data_a;
public:
A(int x){
data_a=x;
  }   friend int fun(A,
B);
} class
B{
private:
  int data_b;
public:
A(int x){
data_b=x;
  }
  friend int fun(A, B);
}
int fun(A a, B b){
 return a.data_a+b.data_b;
} int
main(){  A
a(10);
 B b(20);
 cout<<fun(a,b)<<endl;
 return 0;
}
```

Here we can access the private data of class A and class B.

11. What are the C++ access specifiers?

In C++ there are the following access specifiers:

Public: All data members and member functions are accessible outside the class.

Protected: All data members and member functions are accessible inside the class and to the derived class.

Private: All data members and member functions are not accessible outside the class.

12. Define inline function

If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time. One of the important advantages of

using an inline function is that it eliminates the function calling overhead of a traditional function.

13. What is a reference in C++?

A reference is like a pointer. It is another name of an already existing variable. Once a reference name is initialized with a variable, that variable can be accessed by the variable name or reference name both. For example-

```
int x=10; int &ref=x;
//reference variable
```

If we change the value of ref it will be reflected in x. Once a reference variable is initialized it cannot refer to any other variable. We can declare an array of pointers but an array of references is not possible.

14. What do you mean by abstraction in C++?

Abstraction is the process of showing the essential details to the user and hiding the details which we don't want to show to the user or hiding the details which are irrelevant to a particular user.

15. Is deconstructor overloading possible? If yes then explain and if no then why?

No destructor overloading is not possible. Destructors take no arguments, so there's only one way to destroy an object. That's the reason destructor overloading is not possible.

16. What do you mean by call by value and call by reference?

In call by value method, we pass a copy of the parameter is passed to the functions. For these copied values a new memory is assigned and changes made to these values do not reflect the variable in the main function.

In call by reference method, we pass the address of the variable and the address is used to access the actual argument used in the function call. So changes made in the parameter alter the passing argument.

17. What is an abstract class and when do you use it?

A class is called an abstract class whose objects can never be created. Such a class exists as a parent for the derived classes. We can make a class abstract by placing a pure virtual function in the class.

18. What are destructors in C++?

A constructor is automatically called when an object is first created. Similarly when an object is destroyed a function called destructor automatically gets called. A destructor has the same name as the constructor (which is the same as the class name) but is preceded by a tilde.

Example:

```
class A{  private:   int val;
public:
  A(int x){
val=x;
  }
  A(){
  }
  ~A(){                    //destructor
  }
}
int main(){
 A a(3);
 return 0;
}
```

19. What are the static members and static member functions?

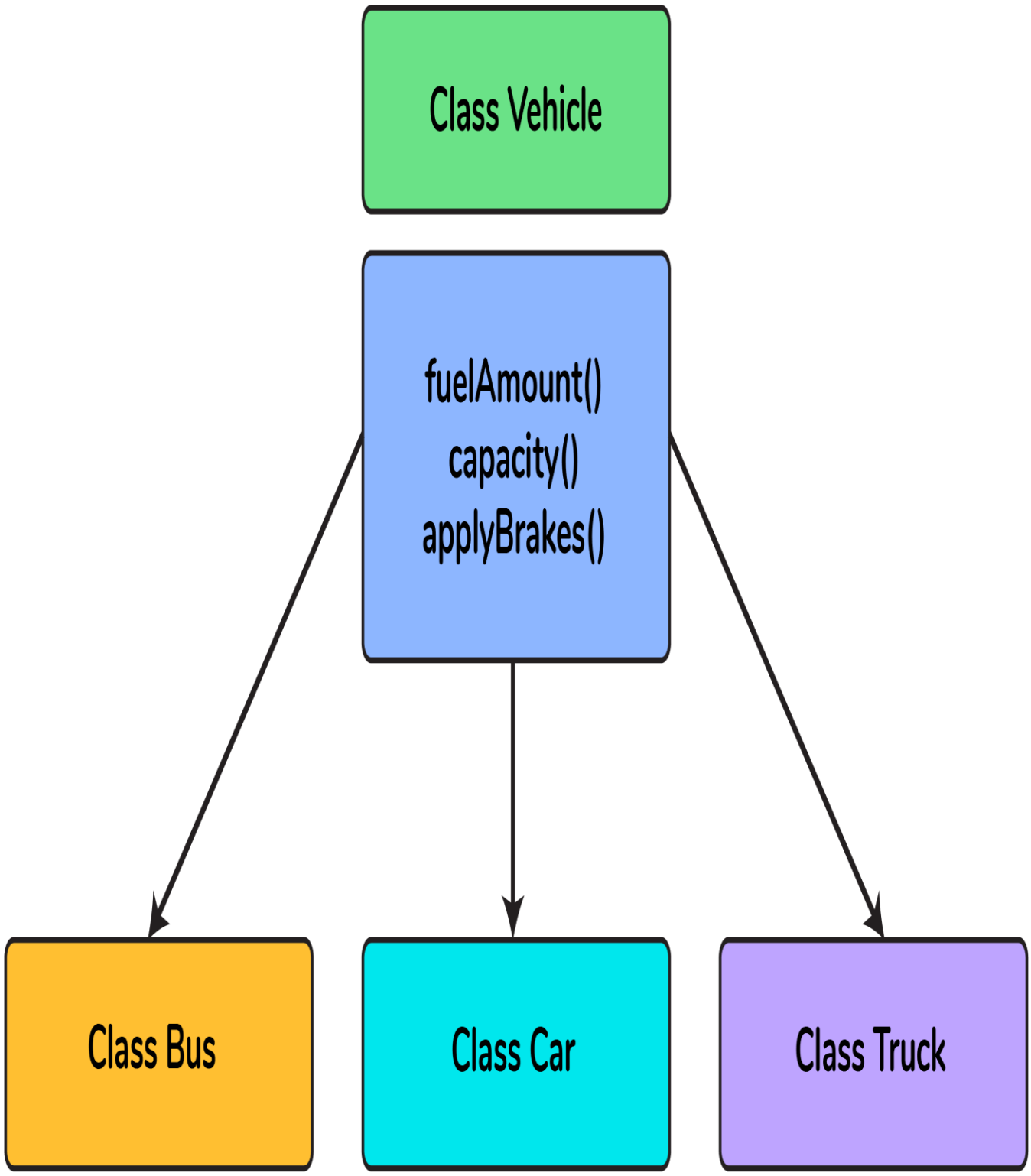
When a variable in a class is declared static, space for it is allocated for the lifetime of the program. No matter how many objects of that class have been created, there is only one copy of the static member. So same static member can be accessed by all the objects of that class.

A static member function can be called even if no objects of the class exist and the static function are accessed using only the class name and the scope resolution operator ::

20. Explain inheritance

Inheritance is the process of creating new classes, called derived classes, from existing classes. These existing classes are called base classes. The derived classes inherit all the capabilities of the base class but can add new features and refinements of their own.

Example-


Class Vehicle
fuelAmount()
capacity()
applyBrakes()
Class Bus
Class Car
Class Truck

Inheritance in C++

Class Bus, Class Car, and Class Truck inherit the properties of Class Vehicle.

The most important thing about inheritance is that it permits code reusability.

C++ Interview Questions For Experienced

21. What is a copy constructor?

A copy constructor is a member function that initializes an object using another object of the same class.

Example-

```
class A{
int x,y;
A(int x, int y){
 this->x=x;
 this->y=y;
}

};
int main(){
A a1(2,3);
A a2=a1;      //default copy constructor is called
return 0;
}
```

We can define our copy constructor. If we don't define a copy constructor then the default copy constructor is called.

22. What is the difference between shallow copy and deep copy?

The difference between shallow copy and a deep copy is given below:

Shallow Copy	Deep Copy
Shallow copy stores the referen to the original memory address	Deep copy makes a new and separate copy of an entire object with its unique memory address.
Shallow copy is faster.	Deep copy is comparatively slower.

Shallow copy reflects changes nDeep copy doesn't reflect changes made to the new/copied object in the origin new/copied object in the original object

23. What is the difference between virtual functions and pure virtual functions?

A virtual function is a member function in the base class that you redefine in a derived class. It is declared using the virtual keyword. **Example-**

```
class base{ public:
 virtual void fun(){

 }
};
```

A pure virtual function is a function that has no implementation and is declared by assigning 0. It has no body.

Example-

```
class base{
public:
 virtual void fun()=0;
};
```

Here, = sign has got nothing to do with the assignment, and value 0 is not assigned to anything. It is used to simply tell the compiler that a function will be pure and it will not have anybody.

24. If class D is derived from a base class B. When creating an object of type D in what order would the constructors of these classes get called?

The derived class has two parts, a base part, and a derived part. When C++ constructs derived objects, it does so in phases. First, the most-base class(at the top of the inheritance tree) is constructed. Then each child class is constructed in order until the most-child class is constructed last.

So the first Constructor of class B will be called and then the constructor of class D will be called.

During the destruction exactly reverse order is followed. That is destructor starts at the most-derived class and works its way down to base class.
So the first destructor of class D will be called and then the destructor of class B will be called.

25. Can we call a virtual function from a constructor?

Yes, we can call a virtual function from a constructor. But the behavior is a little different in this case. When a virtual function is called, the virtual call is resolved at runtime. It is always the member function of the current class that gets called. That is the virtual machine doesn't work within the constructor.

For example-

```
class base{
 private:
  int value;
 public:
  base(int x){
   value=x;
  }
  virtual void fun(){

  }
}

class derived{
 private:
  int a;
 public:
  derived(int x, int v):base(x){
   base *b;
   b=this;
   b->fun();       //calls derived::fun()
  }
  void fun(){
   cout<<"fun inside derived class"<<endl;
  }
}
```

26. What are void pointers?

A void pointer is a pointer which is having no datatype associated with it. It can hold addresses of any type. For example-

```
void *ptr;  char *str; p=str;                 // no error
str=p;                 // error because of type mismatch
```

We can assign a pointer of any type to a void pointer but the reverse is not true unless you typecast it as

```
str=(char*) ptr;
```

27. What is this pointer in C++?

The member functions of every object have a pointer named this, which points to the object itself. The value of this is set to the address of the object for which it is called. It can be used to access the data in the object it points to.

Example

```
class A{
 private:
  int value;
 public:
  void setvalue(int x){
   this->value=x;
  }
};

int main(){
 A a;
 a.setvalue(5);
 return 0;
}
```

28. How do you allocate and deallocate memory in C++?

The new operator is used for memory allocation and deletes operator is used for memory deallocation in C++.

For example-

```
int value=new int;              //allocates memory for storing 1 integer
delete value;                           // deallocates memory taken by value


int *arr=new int[10];           //allocates memory for storing 10 int
delete []arr;
```

Fibonacci Series in C++

Fibonacci Series in C++: In case of fibonacci series, next number is the sum of previous two numbers for example 0, 1, 1, 2, 3, 5, 8, 13, 21 etc. The first two numbers of fibonacci series are 0 and 1.

There are two ways to write the fibonacci series program:

- Fibonacci Series without recursion
- Fibonacci Series using recursion

Fibonaccci Series in C++ without Recursion

Let's see the fibonacci series program in C++ without recursion.

```
1.  #include <iostream>
2.  using namespace std;
3.  int main() {
4.  int n1=0,n2=1,n3,i,number;
5.  cout<<"Enter the number of elements: ";
6.  cin>>number;
7.  cout<<n1<<" "<<n2<<" "; //printing 0 and 1
8.  for(i=2;i<number;++i) //loop starts from 2 because 0 and 1 are already printed
9.  {
10. n3=n1+n2;
11. cout<<n3<<" ";
12. n1=n2;
13. n2=n3;
14. }
15. return 0;
16. }
```

Output:

```
Enter the number of elements: 10
0 1 1 2 3 5 8 13 21 34
```

Fibonnaci series using recursion in C++

Let's see the fibonacci series program in C++ using recursion.

```
1.  #include<iostream>
2.  using namespace std;
3.  void printFibonacci(int n){
4.  static int n1=0, n2=1, n3;
5.  if(n>0){
6.  n3 = n1 + n2;
7.  n1 = n2;
8.  n2 = n3;
9.  cout<<n3<<" ";
10. printFibonacci(n-1);
11. }
12. }
13. int main(){
14. int n;
15. cout<<"Enter the number of elements: ";
16. cin>>n;
17. cout<<"Fibonacci Series: ";
18. cout<<"0 "<<"1 ";
19. printFibonacci(n-2);  //n-2 because 2 numbers are already printed
20. return 0;
21. }
```

Output:

```
Enter the number of elements: 15
 Fibonacci Series: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Prime Number Program in C++

Prime number is a number that is greater than 1 and divided by 1 or itself. In other words, prime numbers can't be divided by other numbers than itself or 1. For example 2, 3, 5, 7, 11, 13, 17, 19, 23.... are the prime numbers.

Let's see the prime number program in C++. In this C++ program, we will take an input from the user and check whether the number is prime or not.

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.  int n, i, m=0, flag=0;
6.  cout << "Enter the Number to check Prime: ";
7.  cin >> n;
8.  m=n/2;
9.  for(i = 2; i <= m; i++)
10. {
11. if(n % i == 0)
12. {
13. cout<<"Number is not Prime."<<endl;
14. flag=1;
15. break;
16. }
17. }
18. if (flag==0)
19. cout << "Number is Prime."<<endl;
20. return 0;
21. }
```

Output:

```
Enter the Number to check Prime: 17
```

```
 Number is Prime.
Enter the Number to check Prime: 57   Number is not Prime.
```

Palindrome program in C++

A **palindrome number** is a number that is same after reverse. For example 121, 34543, 343, 131, 48984 are the palindrome numbers.

Palindrome number algorithm

- Get the number from user
- Hold the number in temporary variable
- Reverse the number
- Compare the temporary number with reversed number
- If both numbers are same, print palindrome number
- Else print not palindrome number

Let's see the palindrome program in C++. In this program, we will get an input from the user and check whether number is palindrome or not.

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.  int n,r,sum=0,temp;
6.  cout<<"Enter the Number=";
7.  cin>>n;
8.  temp=n;
9.  while(n>0)
10. {
11. r=n%10;
12. sum=(sum*10)+r;
13. n=n/10;
14. }
```

```
15. if(temp==sum)
16. cout<<"Number is Palindrome.";
17. else
18. cout<<"Number is not Palindrome.";
19. return 0;
20. }
```

Output:

```
Enter the Number=121
Number is Palindrome.
Enter the number=113   Number
is not Palindrome.
```

Factorial program in C++

Factorial Program in C++: Factorial of n is the product of all positive descending integers. Factorial of n is denoted by n!. For example:

1. 4! = 4*3*2*1 = 24
2. 6! = 6*5*4*3*2*1 = 720

Here, 4! is pronounced as "4 factorial", it is also called "4 bang" or "4 shriek".

The factorial is normally used in Combinations and Permutations (mathematics).

There are many ways to write the factorial program in C++ language. Let's see the 2 ways to write the factorial program.

- Factorial Program using loop
- Factorial Program using recursion

Factorial Program using Loop

Let's see the factorial Program in C++ using loop.

```
#include <iostream>
using namespace std;
int main()
{
int i,fact=1,number;
cout<<"Enter any Number: ";
cin>>number;
for(i=1;i<=number;i++){
fact=fact*i;
}
cout<<"Factorial of " <<number<<" is: "<<fact<<endl;
return 0;
}
```

Output:

```
Enter any Number: 5
Factorial of 5 is: 120
```

Factorial Program using Recursion

Let's see the factorial program in C++ using recursion.

```
#include<iostream>
using namespace std;
int main()
{
int factorial(int);
int fact,value;
cout<<"Enter any number: ";
cin>>value;
fact=factorial(value);
cout<<"Factorial of a number is: "<<fact<<endl;
```

```
11. return 0;
12. }
13. int factorial(int n)
14. {
15. if(n<0)
16. return(-1); /*Wrong value*/
17. if(n==0)
18. return(1);  /*Terminating condition*/
19. else
20. {
21. return(n*factorial(n-1));
22. }
23. }
```

Output:

```
Enter any number: 6
Factorial of a number is: 720
```

Armstrong Number in C++

Before going to write the C++ program to check whether the number is Armstrong or not, let's understand what is Armstrong number.

Armstrong number is a number that is equal to the sum of cubes of its digits. For example 0, 1, 153, 370, 371 and 407 are the Armstrong numbers.

Let's try to understand why **371** is an Armstrong number.

1. 371 = (3*3*3)+(7*7*7)+(1*1*1)
2. where:
3. (3*3*3)=27
4. (7*7*7)=343
5. (1*1*1)=1
6. So:
7. 27+343+1=371

Let's see the C++ program to check Armstrong Number.

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int n,r,sum=0,temp;
6. cout<<"Enter the Number=  ";
7. cin>>n;
8. temp=n;
9. while(n>0)
10. {
11. r=n%10;
12. sum=sum+(r*r*r);
13. n=n/10;
14. }
15. if(temp==sum)
16. cout<<"Armstrong Number."<<endl;
17. else
18. cout<<"Not Armstrong Number."<<endl;
19. return 0;
20. }
```

Output:

```
Enter the Number= 371
Armstrong Number.
Enter the Number= 342    Not Armstrong Number.
```

Sum of digits program in C++

We can write the sum of digits program in C++ language by the help of loop and mathematical operation only.

Sum of digits algorithm

To get sum of each digit by C++ program, use the following algorithm:

- **Step 1:** Get number by user
- **Step 2:** Get the modulus/remainder of the number
- **Step 3:** sum the remainder of the number
- **Step 4:** Divide the number by 10
- **Step 5:** Repeat the step 2 while number is greater than 0.

Let's see the sum of digits program in C++.

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int n,sum=0,m;
6. cout<<"Enter a number: ";
7. cin>>n;
8. while(n>0)
9. {
10. m=n%10;
11. sum=sum+m;
12. n=n/10;
13. }
14. cout<<"Sum is= "<<sum<<endl;
15. return 0;
16. }
```

Output:

```
Enter a number: 23
 Sum is= 5
Enter a number: 624
Sum is= 12
```

C++ Program to reverse number

We can reverse a number in C++ using loop and arithmetic operators. In this program, we are getting number as input from the user and reversing that number.

Let's see a simple C++ example to reverse a given number.

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.  int n, reverse=0, rem;
6.  cout<<"Enter a number: ";
7.  cin>>n;
8.  while(n!=0)
9.  {
10. rem=n%10;
11. reverse=reverse*10+rem;
12. n/=10;
13. }
14. cout<<"Reversed Number: "<<reverse<<endl;
15. return 0;
16. }
```

Output:

```
Enter a number: 234
Reversed Number: 432
```

C++ Program to swap two numbers without third variable

We can swap two numbers without using third variable. There are two common ways to swap two numbers without using third variable:

1. By + and -
2. By * and /

Program 1: Using * and /

Let's see a simple C++ example to swap two numbers without using third variable.

```
#include <iostream>
using namespace std;
int main()
{
int a=5, b=10;
cout<<"Before swap a= "<<a<<" b= "<<b<<endl;
a=a*b; //a=50 (5*10)
b=a/b; //b=5 (50/10)
a=a/b; //a=10 (50/5)
cout<<"After swap a= "<<a<<" b= "<<b<<endl;
return 0;
}
```

Output:

```
Before swap a= 5 b= 10      After
swap a= 10 b= 5
```

Program 2: Using + and -

Let's see another example to swap two numbers using + and -.

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

```
5. int a=5, b=10;
6. cout<<"Before swap a= "<<a<<" b= "<<b<<endl;
7. a=a+b; //a=15 (5+10)
8. b=a-b; //b=5 (15-10)
9. a=a-b; //a=10 (15-5)
10. cout<<"After swap a= "<<a<<" b= "<<b<<endl;
11. return 0;
12. }
```

Output:

```
Before swap a= 5 b= 10
 After swap a= 10 b= 5
```

Decimal to Binary Conversion Algorithm

Step 1: Divide the number by 2 through % (modulus operator) and store the remainder in array

Step 2: Divide the number by 2 through / (division operator) **Step**

3: Repeat the step 2 until the number is greater than zero Let's

see the C++ example to convert decimal to binary.

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int a[10], n, i;
6. cout<<"Enter the number to convert: ";
7. cin>>n;
8. for(i=0; n>0; i++)
9. {
```

```
10. a[i]=n%2;
11. n= n/2;
12. }
13. cout<<"Binary of the given number= ";
14. for(i=i-1 ;i>=0 ;i--)
15. {
16. cout<<a[i];
17. }
18. }
```

Output:

```
Enter the number to convert: 9
Binary of the given number= 1001
```

C++ Program to Convert Number in Characters

In C++ language, we can easily convert number in characters by the help of loop and switch case. In this program, we are taking input from the user and iterating this number until it is 0. While iteration, we are dividing it by 10 and the remainder is passed in switch case to get the word for the number.

Let's see the C++ program to convert number in characters.

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
```

```
5.  long int n,sum=0,r;
6.  cout<<"Enter the Number= ";
7.  cin>>n;
8.  while(n>0)
9.  {
10. r=n%10;
11. sum=sum*10+r;
12. n=n/10;
13. }
14. n=sum;
15. while(n>0)
16. {
17. r=n%10;
18. switch(r)
19. {
20. case 1:
21. cout<<"one ";
22. break;
23. case 2:
24. cout<<"two ";
25. break;
26. case 3:
27. cout<<"three ";
28. break;
29. case 4:
30. cout<<"four ";
31. break;
32. case 5:
33. cout<<"five ";
34. break;
```

```
35. case 6:
36. cout<<"six ";
37. break;
38. case 7:
39. cout<<"seven ";
40. break;
41. case 8:
42. cout<<"eight ";
43. break;
44. case 9:
45. cout<<"nine ";
46. break;
47. case 0:
48. cout<<"zero ";
49. break;
50. default:
51. cout<<"tttt ";
52. break;
53. }
54. n=n/10;
55. }
56. }
```

Output:

```
Enter the Number= 74254
seven four two five four
```

C++ Program to Print Alphabet Triangle

There are different triangles that can be printed. Triangles can be generated by alphabets or numbers. In this C++ program, we are going to print alphabet triangles.

Let's see the C++ example to print alphabet triangle.

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.  char ch='A';
6.  int i, j, k, m;
7.  for(i=1;i<=5;i++)
8.  {
9.  for(j=5;j>=i;j--)
10. cout<<" ";
11. for(k=1;k<=i;k++)
12. cout<<ch++;
13. ch--;
14. for(m=1;m<i;m++)
15. cout<<--ch;
16. cout<<"\n";
17. ch='A';
18. }
19. return 0;
20. }
```

Output:

```
    A
   ABA
  ABCBA
 ABCDCBA   ABCDEDCBA
```

C++ Program to print Number Triangle

Like alphabet triangle, we can write the C++ program to print the number triangle. The number triangle can be printed in different ways.

Let's see the C++ example to print number triangle.

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.  int i,j,k,l,n;
6.  cout<<"Enter the Range=";
7.  cin>>n;
8.  for(i=1;i<=n;i++)
9.  {
10. for(j=1;j<=n-i;j++)
11. {
12. cout<<" ";
13. }
14. for(k=1;k<=i;k++)
15. {
16. cout<<k;
17. }
18. for(l=i-1;l>=1;l--)
19. {
20. cout<<l;
21. }
22. cout<<"\n";
23. }
24. return 0;
25. }
```

Output:

```
Enter the Range=5
    1
```

```
    121
   12321
  1234321
 123454321
Enter the Range=6
      1
     121
    2321
   1234321
  123454321  12345654321
```

C++ Program to generate Fibonacci Triangle

In this program, we are getting input from the user for the limit for fibonacci triangle, and printing the fibonacci series for the given number of times (limit).

Let's see the C++ example to generate fibonacci triangle.

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.  int a=0,b=1,i,c,n,j;
6.  cout<<"Enter the limit: ";
7.  cin>>n;
8.  for(i=1; i<=n; i++)
9.  {
10. a=0;
11. b=1;
12. cout<<b<<"\t";
13. for(j=1; j<i; j++)
14. {
15. c=a+b;
16. cout<<c<<"\t";
17. a=b;
18. b=c;
```

```
19.         }
20.         cout<<"\n";
21.         }
22.         return 0;
23.         }
```

Output:

```
Enter the limit: 10
  1
  1       1
  1       1       2
  1       1       2       3
  1       1       2       3       5
  1       1       2       3       5       8
  1       1       2       3       5       8       13
  1       1       2       3       5       8       13      21
  1       1       2       3       5       8       13      21      34      1
1       2       3       5       8       13      21      34      55
```