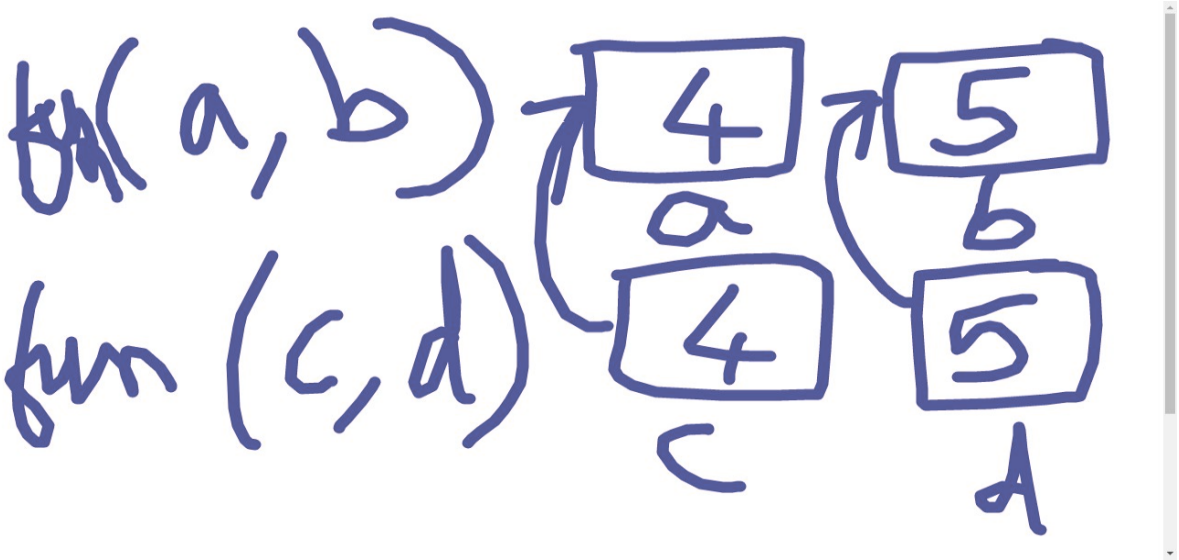
C++

- For competitive programming

Function calling methods - Call by Value and Call by Reference

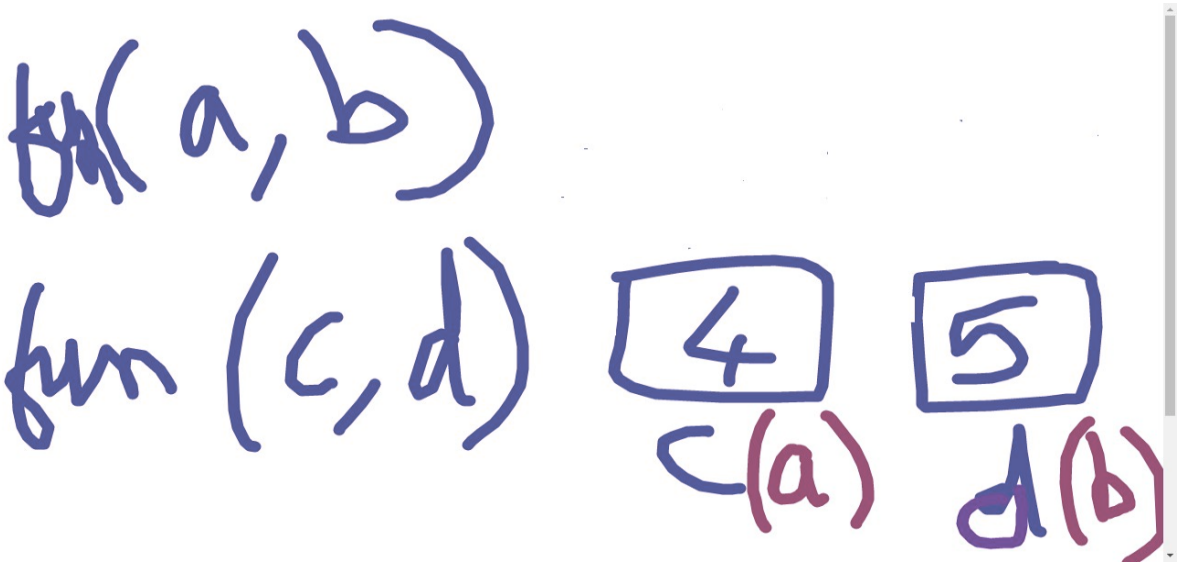
We have function fun(int a, int b) .
Let us say, in main(), we call it as:

int c=4,d=5;
fun(c,d)
By default, the function call uses a **call by value** mechanism. In call by value, function creates different copies of all the input paremeter. So, any change in value inside the function is not reflected outside that function.



We use call by reference mechanism to return the new (changed) values of input parameters of function in main()

Consider same function, but in call by reference , it just creates an alias (or pet name) of the same variables. It don't create a new variable.



To use call by reference, use & sign in front of the parameters in function definition only.

```
#include <bits/stdc++.h>
using namespace std;

// Swap 2 variables without a third new variable.
```

```
void swapNums(int & a, int & b)
{
    cout<<"In function, before swapping : a="<<a<<"
b="<<b<<'\n';
    int t=b;
    b=a;
    a=t;
    cout<<"In function, after swapping : a="<<a<<"
b="<<b<<'\n';
}

int main()
{
    // = operator moves RHS value to LHS
    int a=3;
    int b=5;
    cout<<"In main(), before swapping : a="<<a<<"
b="<<b<<'\n';
    swapNums(a,b);
    cout<<"In main(), after swapping : a="<<a<<"
b="<<b<<'\n';
    return 0;
}
```

You use call by reference, when you want that input parameters of function should be changed outside that function (where it is called)
Which is faster ? Why ?
Call by reference, because it just gives a pet name to the original variables passed.

= operator (Assignment Operator)

= operator moves RHS value to LHS
For eg.
int a;
a = 2;
// After this line, value of a becomes 2

Recursive functions

Recursive functions are a function which calls itself in the function definition.

Eg. // Suppose, you are creating function for factorial calculation
// == operator in C++ is used for equality checking

```
int factorial(int n)
{
     if (n==0)
  {
```

```
    return 1;
  }
     return n * factorial(n-1);
}
int main()
{
cout<<factorial(4);
return 0;
}
```

Always a function stops when it encounters a return statement.

Eg. call factorial(4);
return 4 * factorial(3)
return 4 * 3 * factorial(2)
return 4 * 3 * 2 * factorial(1)
return 4 * 3 * 2 * 1 * factorial(0)
return 4 * 3 *2 * 1 * 1
…
This will stop at n = 0

So, for stopping (terminating) any recursive function, you always need a base condition. Like, in above function, if (n==1) return 5;
(If you remove that condition, it will give an infinitely running program)

Homework Questions on Recursion

1. Create a recursive function rec_power(int base, int p) to find the value of $base^p$ (base raised to power p)
2. Create a recursive function to find the sum of digits of a number
3. Create a recursive function to print the multiplication table of any number.

Basics of arrays

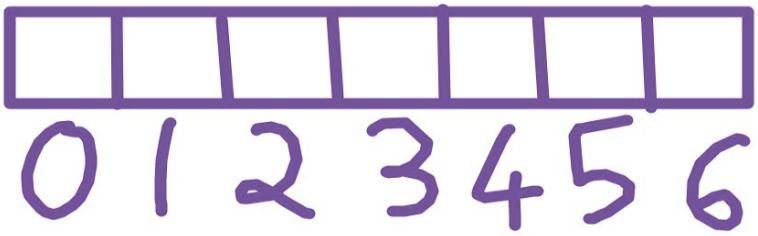
Array are used when you need to store a large number of variables of same data type.

Syntax for declaring array :
<data-type> <name-of-array> [<array-size>]

<array-size> should always be a **constant positive integer**. For example,
int marks [100];

const int size=100;
int marks[size];

Counting or indexing of elements of array starts with 0.



How to access a particular element of an array ?

<arry-name> [index or position of the element]

For eg. marks[0]; // gives first element
marks[1]; // gives second element
marks[2]; // gives third element
…… and so on
i.e. marks[i - 1] gives i^{th} element

Example 1 Take input of 2 students marks in an array of size 6

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
```

```
    int marks[6];
    cout<<"Enter marks of first student: ";
    cin>>marks[0];
    cout<<"Enter marks of second student: ";
    cin>>marks[1];
    cout<<"Marks are : "<<marks[0]<<"
"<<marks[1]<<'\n';
    return 0;
}
```

Example 2 Find size of an array

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int marks[6];
    cout<<" No. of element is array :
"<<sizeof(marks)/sizeof(int);
    return 0;
}
```

Relational Operators
(a few call it Conditional Operators)

1. == Checks the equality of LHS and RHS

Eg.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    cout<<(2==2)<<'\n';
    cout<<(3==2)<<'\n';
    return 0;
}
```

Output

1

0

(In C++, **true** is equivalent to **1** and **false** is equivalent to **0**)

2. != Checks non-equality

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
```

```
    cout<<(2!=2)<<'\n';
    cout<<(3!=2)<<'\n';
    return 0;
}
```

Output:
0
1

3. **< operator (Less than)**
4. **> operator (Greater than)**
5. **<= operator (Less than or equal)**
6. **>= operator (Greater than or equal)**

For eg.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    cout<<(2<=3)<<'\n';
    cout<<(2>3)<<'\n';
    return 0;
}
```

Try it yourself: You can compare strings, char, etc. with == operator, != operator etc.

Caution : Don’t compare floating point (or double or long double data types) with == or != . Because it may not give you an accurate value.

If statements

Syntax of if statement :

```
if ( condition )
{
// statements
// ...
// ...
}
```

For eg.

```
int a=2;
if (a < 5)
{
    cout<<"Less than 5";
}
```

```
int a=2;
if (a > 5)
{
    cout<<"Greater  than 5";
}
```

If else statement

Syntax:

```
if ( condition )
{
// statements
// ...
// ...
}
 else
{
// statements
// ...
// ...
}
```

Example:

```
#include <bits/stdc++.h>
using namespace std;
```

```
int main()
{
    int a=8;
    if (a < 5)
    {
    cout<<"Less than 5";
    }
    else {
    cout<<"Greater  than 5";
    }
    return 0;
}
```

Logical Operators

&& - AND operator in C++
|| - OR operator in C++
! - NOT operator in C++

Example 1 -

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
```

```
    int percentile;
    int boardPer;

    cout<<" Enter Percentile :";
    cin>>percentile;

    cout<<" Enter Board percentage :";
    cin>>boardPer;

    if ( (percentile > 85) || (boardPer > 75)
)
    {
        cout<<"You are qualified";
    }
    else
    {
        cout<<"Not qualified, sorry";
    }

    return 0;
}
```

Example 2 -

```
#include <bits/stdc++.h>
```

```
using namespace std;

int main()
{
    int percentile;
    int boardPer;

    cout<<" Enter Percentile :";
    cin>>percentile;

    cout<<" Enter Board percentage :";
    cin>>boardPer;

    if ( (percentile > 85) && (boardPer > 75)
)
    {
        cout<<"You are qualified";
    }
    else
    {
        cout<<"Not qualified, sorry";
    }

    return 0;
}
```

Example 3

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int percentile;
    int boardPer;

    cout<<" Enter Percentile :";
    cin>>percentile;

    cout<<" Enter Board percentage :";
    cin>>boardPer;

    if ( !((percentile > 85) && (boardPer >
75)) )
    {
        cout<<"You are qualified";
    }
    else
    {
        cout<<"Not qualified, sorry";
    }

    return 0;
```

```
}
```

Many if ..else can be combined also.

Example 4

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n;
    if (n==0)
    {
        cout<<"Zero";
    }
    else if(n==1)
    {
        cout<<"One";
    }
    else if(n==2)
    {
        cout<<"Two";
    }
    else if(n==3)
    {
        cout<<"Three";
    }
```

```
    return 0;
}
```

Loops

- You want to perform same piece of code, many number of times. Then loops are your saviour!
- In C++, there are 3 types of loops:
    1. While loop
    2. For loop
    3. Do while

While Loop

Syntax:

```
while (condition for running loop)
{
  // statements
}
```

Example: Program to print hello 10 times

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;

int main()
{
    int cnt=0;

    while(cnt<10)
    {
    cout<<"Hello ";
    cnt=cnt+1;
    }

    return 0;
}
```

2. Print hello Infinite times

Note that 1 in condition means true and 0 means false.

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int cnt=0;
```

```
    while(1)
    {
    cout<<"Hello ";
    cnt=cnt+1;
    }

    return 0;
}
```

Break statement
(I wana jump out of loop)

Example

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int cnt=0;

    while(1)
    {
    cout<<"Hello ";
    cnt=cnt+1;

    if (cnt == 10)
```

```
    {
        break; // break will stop your loop
    }

    }

    return 0;
}
```

Program for printing counting from 1 to 8

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int i = 1;
    while (i<=8)
    {
        cout<<i<< '\n';
        i++;
    }

    return 0;
}
```

For Loop

```
for (  <variable-initialisation> ; <condtion> ;
<update-variable>  )
{
  // statements
}
```

<variable-intilisation> means to give a starting value to any variable . It occurs only 1 time.

For eg.

```
for (int i=1 ; i<=8; i=i+1 )
{
    cout<<i<<'\n';
}
```

++ operator increases value of variable by 1

For eg.

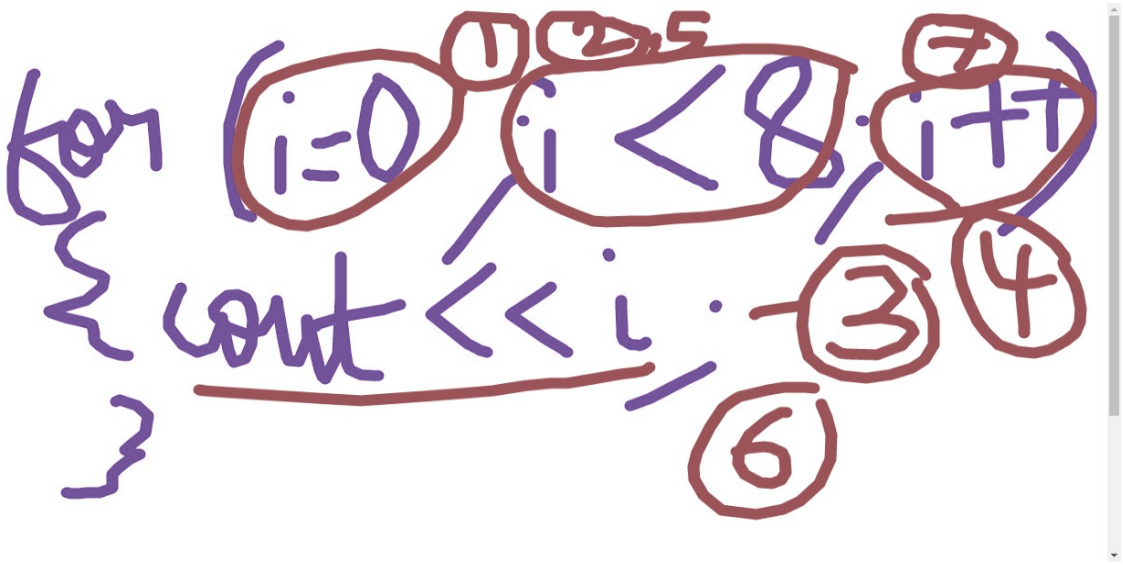
```
int a=4;
a++;
cout<<a<<'\n';
```

-- decrease value of variable by 1

For eg.

```
int a=4;
a--;
cout<<a<<'\n';
```

Order of execution of statement in a for loop



Example:

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    for (int i=1 ; i<=8; i++ )
    {
```

```
        cout<<i<<'\n';
    }
    return 0;
}
```

Do while loop

Syntax-

```
do
{
 // statements
} while (condition) ;
```

It always executes at least one time

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int i=1;
    do
    {
        cout<<i<<'\n';
        i++;
    } while(i<=4);
```

```
    return 0;
}
```

auto for loop for strings

Q. Print all characters of a string in different line

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    string str="Hello world";
    for(auto c: str)
    {
        cout<<c<<'\n';
    }
    return 0;
}
```

Practice Questions (Do try these)

1. Take a number as input from user and print whether the number is odd or even.

[**Hint** : Use % operator to find remainder. In C++, a%b gives remainder when a is divided by b]

2. Take marks of a student as input. And print his grade as per the following distribution

Marks	Grade
0-40	F
41-60	C
61-80	B
81-100	A

3. Write a program to take input of a number n and print the multiplication table of n using a for loop.
4. Write a program to take input of a number and print sum of its digits using a while loop.
5. Write a program to find the sum of the series: (using while loop)
1+11+111+....111 upto n.
(Take n as input from user)
For eg. if n=4, the series is : 1+11+111+1111.
6. Take marks of 10 students as an input in an array of size 10 from user and print the average marks and highest marks using for loop.
7. Write a program to take a number n as input and print the following star triangle pattern with n lines: (using for loop)

Eg. For n=4, the pattern would be :

*

**

***

****

8. Write a program to take a number n as input and print the following numerical triangle pattern with n lines: (using for loop)

Eg. For n=4, the pattern would be :

1

1 2

1 2 3

1 2 3 4

9. https://www.hackerrank.com/challenges/staircase/problem

10. https://www.hackerrank.com/challenges/kangaroo/problem