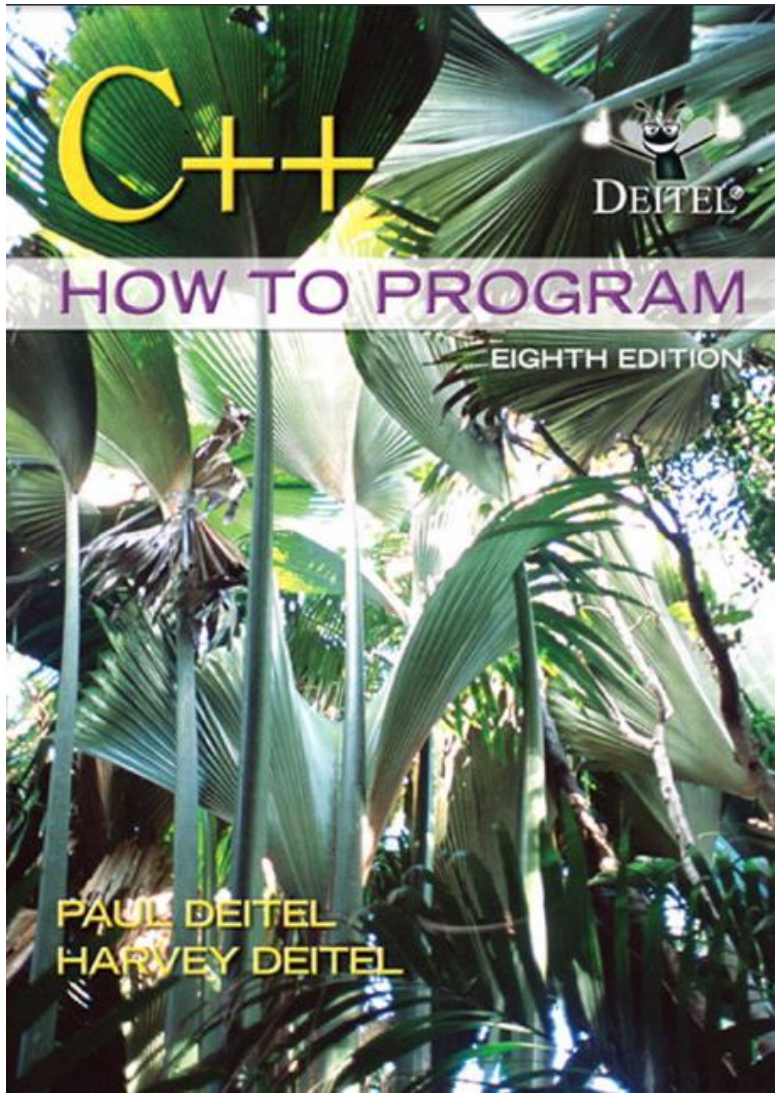
Programming Language

Vafaee Sharbaf

Points

Assignments	3
Midterm	3
Final exam	12
Lab	2

Reference


C++
DEITEL
HOW TO PROGRAM
EIGHTH EDITION
PAUL DEITEL
HARVEY DEITEL

Syllabus

Basic Concepts

C++

- Variables and Constant
- Conditional Statements
- Loops
- Arrays
- String
- Functions
- Files
- Object Oriented Programming (OOP)

Basic Concepts

Types of programming languages:

Low-level	assembly
Middle-level	C++
High-level	Python

Basic Concepts

Compiler

Integrated development environment (IDE)

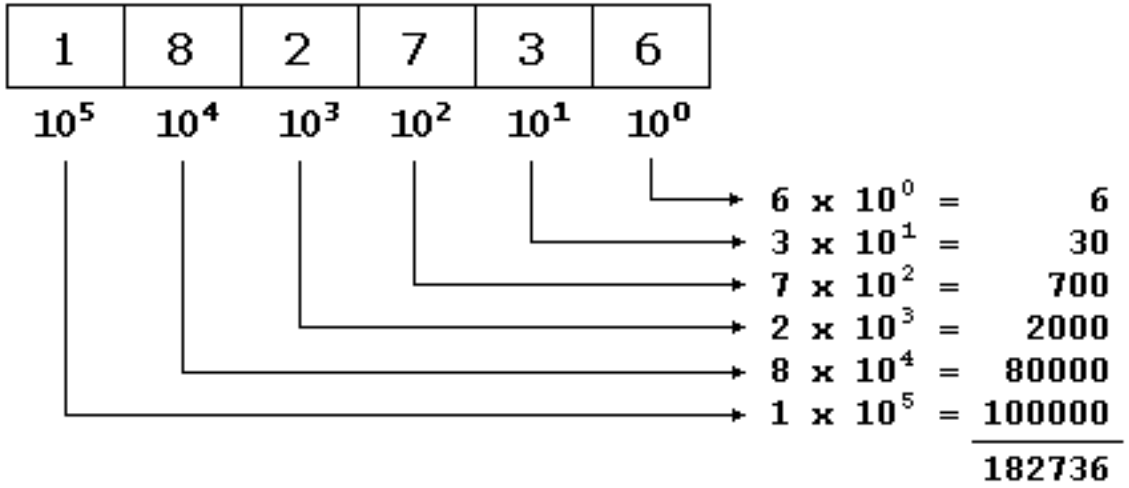
Editor

https://hpc-wiki.info/hpc/Compiler

Basic Concepts

Numeric System

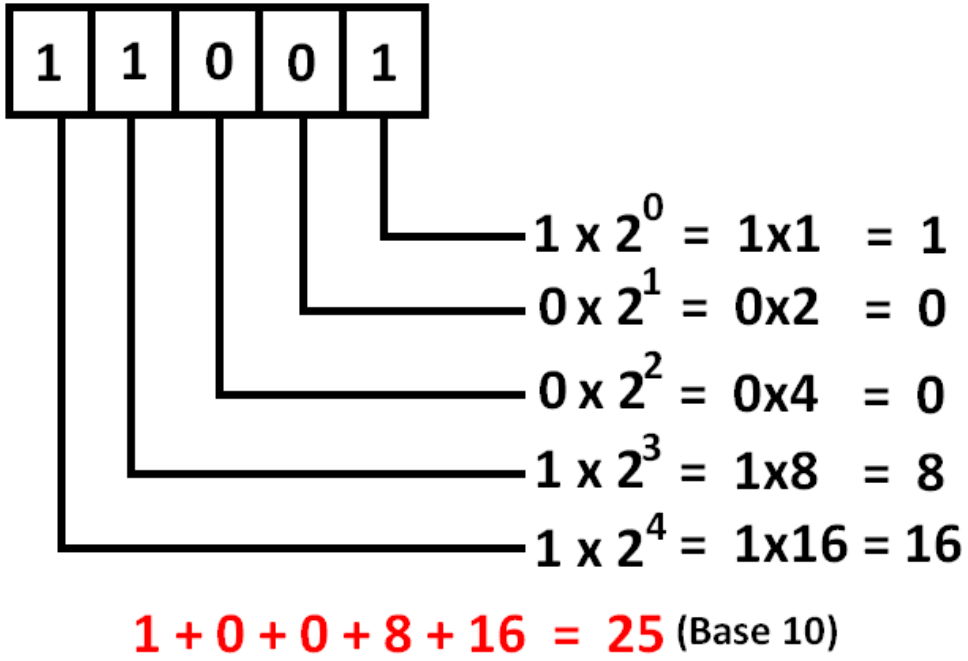
1. Decimal number system (Base- 10)



Basic Concepts

Numeric System

2. Binary number system (Base- 2)

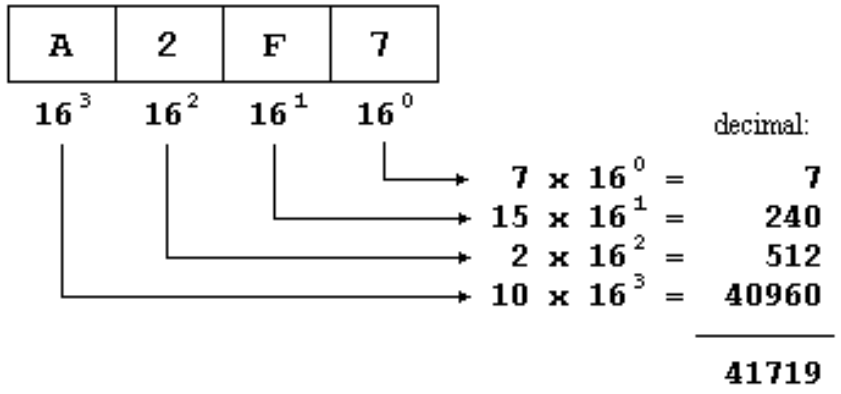


Basic Concepts

Numeric System

3. Hexadecimal number system (Base- 16)

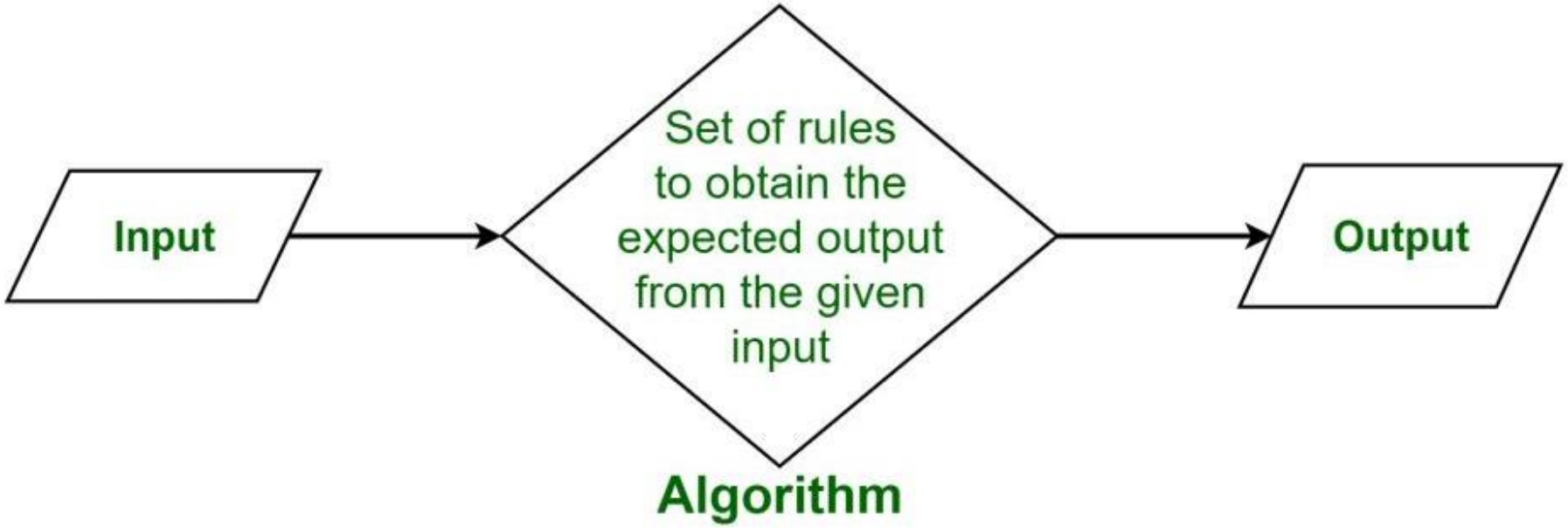
Decade	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadec.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F



Basic Concepts

What is an algorithm?

A step-by-step procedure that defines a set of instructions that must be carried out in a specific order to produce the desired result.



Programming By C++

Data types

Data Type	Meaning	Size (in Bytes)
`int`	Integer	2 or 4
`float`	Floating-point	4
`double`	Double Floating-point	8
`char`	Character	1
`bool`	Boolean	1
`void`	Empty	0

Variables

Syntax

```
type variableName = value;
```

Example

```
int myNum = 5;                // Integer (whole number without decimals)
double myFloatNum = 5.99;     // Floating point number (with decimals)
char myLetter = 'D';          // Character
string myText = "Hello";      // String (text)
bool myBoolean = true;        // Boolean (true or false)
```

Namespace

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int x;
    x = 0;
    double x; // Error here
    x = 0.0;
    system("pause");
}
```

```
#include <iostream>
#include <cstdlib>
using namespace std;
namespace first
{
    int val = 500;
}

int main()
{
    int val = 200;
    cout << first::val << '\n';
    system("pause");
}
```

First experience

```
#include <iostream>
using namespace std;

int main()
{
 cout<<"Hello, World.";
 cout<<std::endl<<"and its very easy";
 return 0;
}
```

The std namespace includes all classes, objects and functions of the C++ standard library.

Constant - Unchangeable and read-only

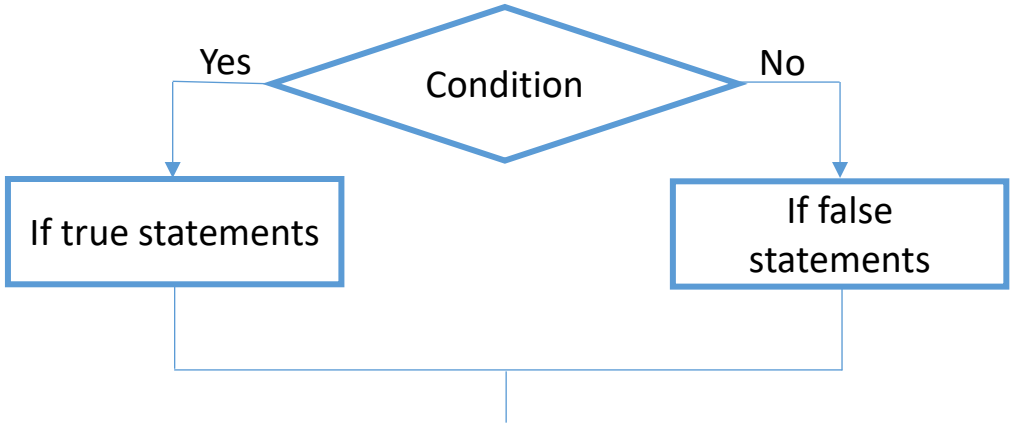
Syntax

```
Const type ConstantName = value;
```

Example

```
const int myNum = 15;  // myNum will always be 15
myNum = 10;  // error: assignment of read-only variable 'myNum'
```

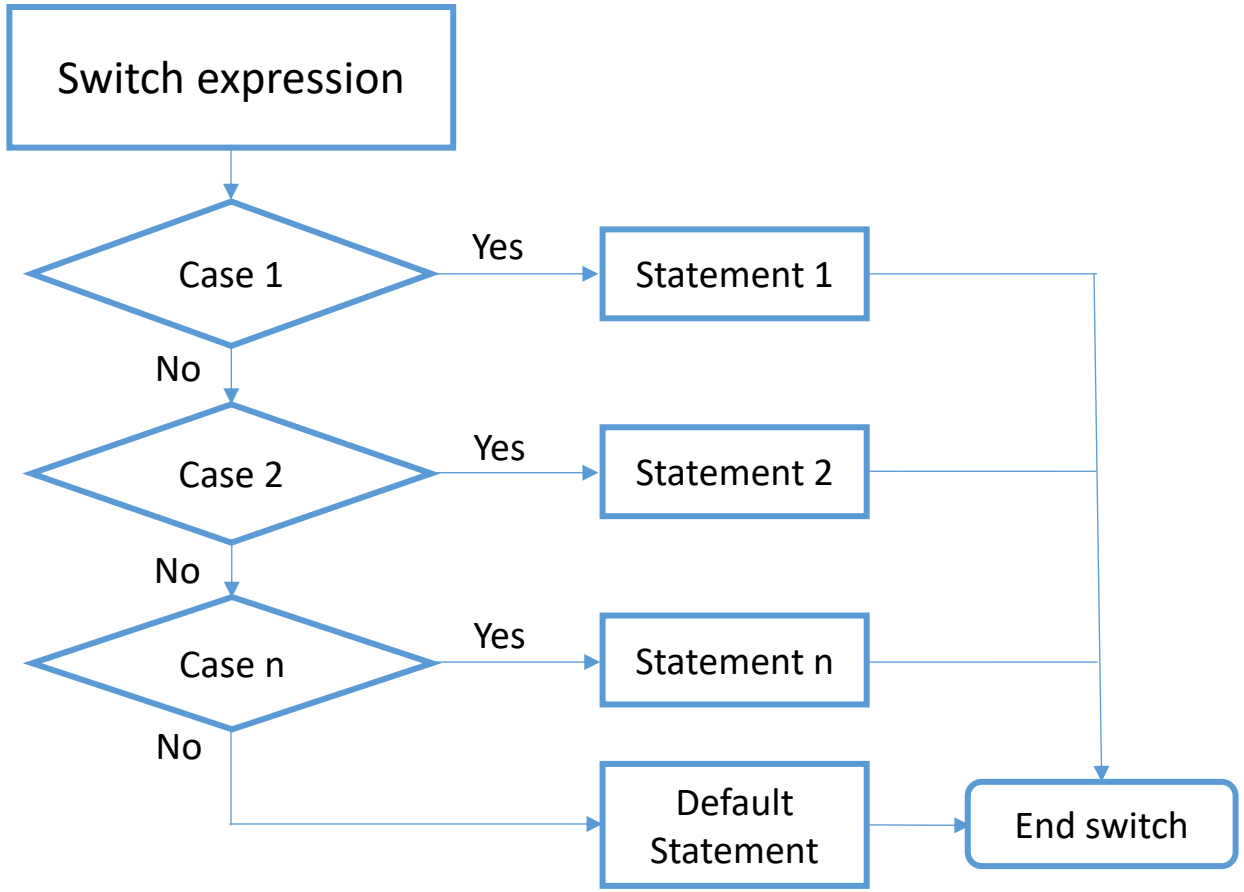
If-else




```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2
is true
} else {
  // block of code to be executed if the condition1 is false and condition2
is false
}
```

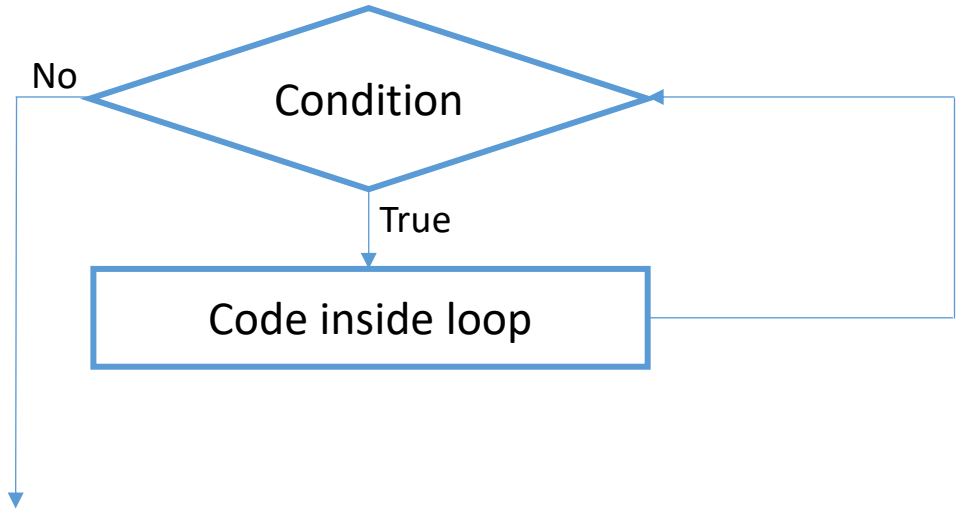
Switch-case

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```



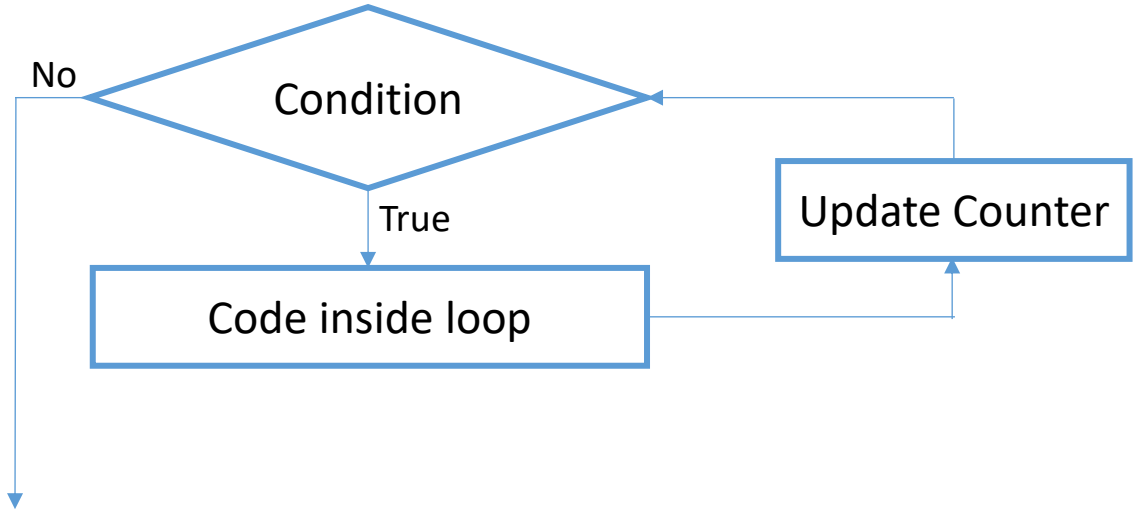
While Loop

```
while (condition) {
  // code block to be executed
}
```



For Loop

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```



Time complexity

```
for(int i=1 ; i<=n ; i++)
    for(int j=1 ; j<=n ; j++)
        for(int k=1 ; k<=n ; k++)
            cout<<"*";
```

$\Theta(n^3)$

```
for(int i=1 ; i<=n ; i++)
    for(int j=1 ; j<=i ; j++)
        cout<<"*";
```

$\Theta(n^2)$

```
for(int i=1 ; i<=n ; i++)
    for(int j=1 ; j<=i ; j++)
        for(int k=1 ; k<=j ; k++)
            cout<<"*";
```

$\Theta(n^3)$

Time complexity

```
for(int i=1 ; i<=n ; i++)
    for(int j=1 ; j<=n ; j+=i)
        cout<<"*";
```

$\Theta(nlogn)$

```
for(int i=1 ; i<=n ; i++)
    for(int j=1 ; j<=n ; j++)
    {
        cout<<"*";
        n--;
    }
```

$\Theta(n)$

```
for(int i=1 ; i<=n ; i++)
{
    for(int j=1 ; j<=n ; j++)
        cout<<"*";
        n--;
}
```

$\Theta(n^2)$

Time complexity

```
for(int i=1 ; i<=n ; i*=2)
    cout<<"*";
```

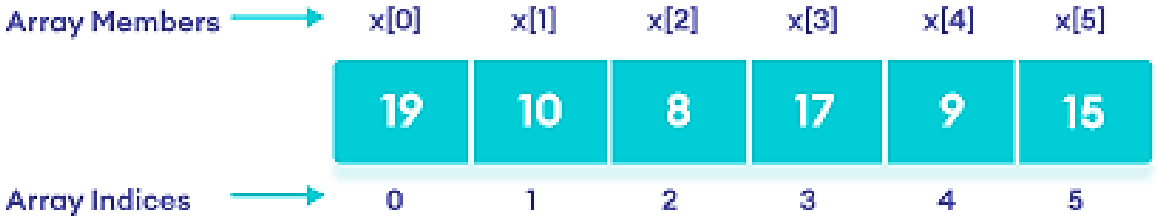
$\Theta\ (logn)$

```
for(int i=n ; i>=1 ; i/=2)
    cout<<"*";
```

$\Theta(logn)$

Array (Vector)

```
int main()
{
    int a[5];
    for(int i = 0; i<5)
        cin>>a[i];
}
```



Bubble Sort

	5	**8**	**1**	**3**	**2**
1)	5	8	1	3	2
	5	8	1	3	2
	5	1	8	3	2
	5	1	3	8	2
	5	1	3	2	8

2)	5	1	3	2	8
	1	5	3	2	8
	1	3	5	2	8
	1	3	2	5	8

Bubble Sort

	5	**8**	**1**	**3**	**2**
3)	1	3	2	5	8
	1	3	2	5	8
	1	2	3	5	8

4)	1	2	3	5	8
	1	2	3	5	8

Bubble Sort

```
for( i=0; i<4 ;i++)
{
    for( j=0; j<(4-i); j++)
    {
        if( a[j] > a[j+1])
        {
            t=a[j];
            a[j]=a[j+1];
            a[j+1]=t;
        }
    }
}
```

$O(n^2)$ $\theta(n^2)$ $\Omega(n^2)$

```
bool sw=1;
for( i=0; i<4 && sw ;i++)
{
    sw=0;
    for( j=0; j<(4-i); j++)
    {
        if( a[j] > a[j+1])
        {
            t=a[j];
            a[j]=a[j+1];
            a[j+1]=t;
            sw=1;
        }
    }
}
```

$O(n^2)$ $\theta(n^2)$ $\Omega(n)$

Search

Ordered Search - O(n)

```
int main()
{
    int a[5], i, num;

    for( i=0; i<5; i++)
        cin>> a[i];

    cin>>num;

    for( i=0; i<5 ;i++)
    {
        if( a[i] == num)
        {
            cout<<"Found";
            break;
        }
    }
    if(i==5)
    {
        cout<<"Not Found";
    }
```

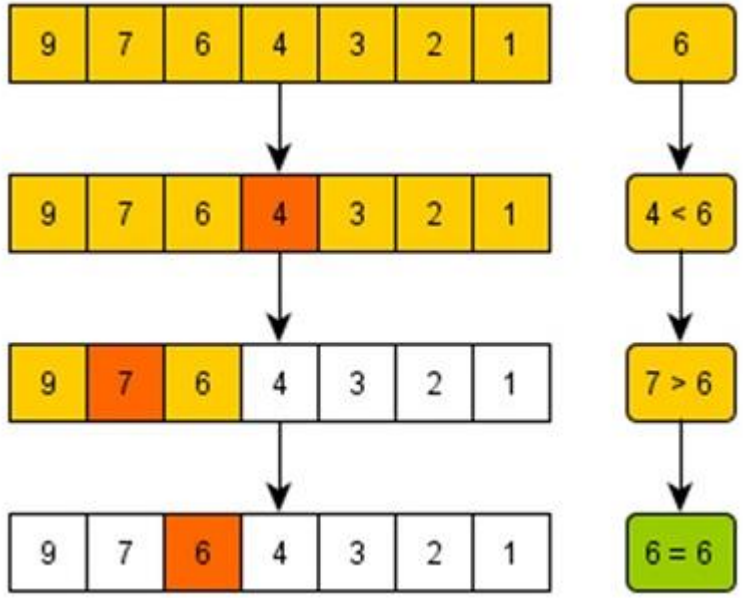
Binary Search – O(logn)

```
int min, low = 0, high = 4, mid;
while(low <= high)
{
    mid = (low+high)/2;
    if(num < arr[mid])
        high = mid-1;
    else if(num > arr[mid])
        low = mid +1;
    else
    {
        cout<<"Found";
        break;
    }
}
if(low > high)
    cout<<"Not Found";
```

Search

Binary Search – O(logn)

```
int min, low = 0, high = 4, mid;
while(low <= high)
{
    mid = (low+high)/2;
    if(num < arr[mid])
        high = mid-1;
    else if(num > arr[mid])
        low = mid +1;
    else
    {
        cout<<"Found";
        break;
    }
}
if(low > high)
    cout<<"Not Found";
```



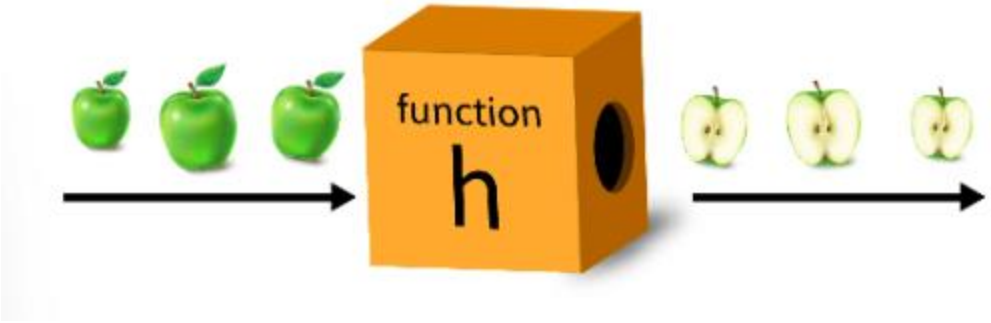
2D Array

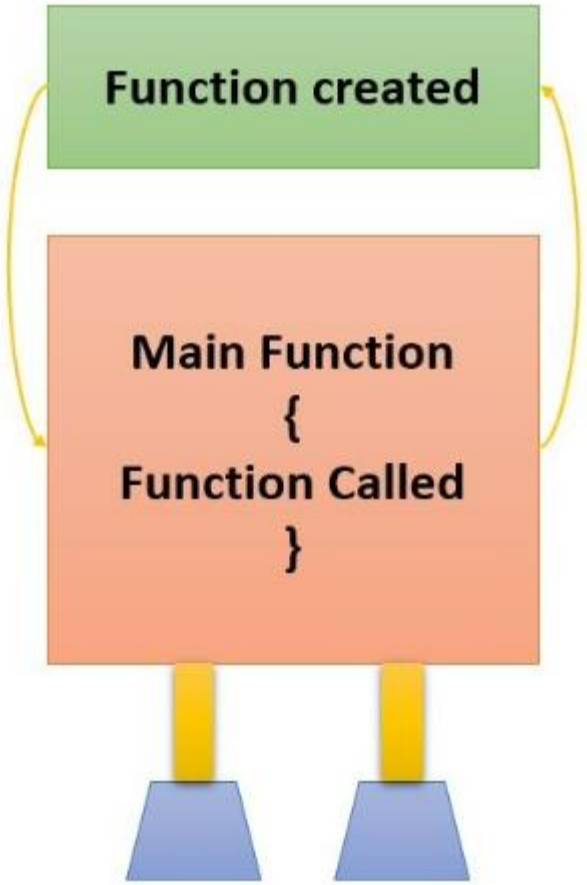
```
int main()
{
    int a[3][4];
    for(int i=0; i<3; i++)
        for(int j=0; j<4; j++)
            cin>>a[i][j];

    return 0;
}
```

	Column 1	Column 2	Column 3	Column 4
Row 1	X[0][0]	X[0][1]	X[0][2]	X[0][3]
Row 2	X[1][0]	X[1][1]	X[1][2]	X[1][3]
Row 3	X[2][0]	X[2][1]	X[2][2]	X[2][3]

Function


function
h


Function created
Main Function
{
Function Called
}

Function

Method 1

```
// Create a function
void myFunction() {
  cout << "I just got executed!";
}

int main() {
  myFunction(); // call the function
  return 0;
}
```

Method 2

```
// Function declaration
void myFunction();

// The main method
int main() {
  myFunction();  // call the function
  return 0;
}

// Function definition
void myFunction() {
  cout << "I just got executed!";
}
```

Function

Method 1

```
using namespace std;

int add(int x, int y)
{
    int sum;
    sum = x + y;
    return sum;
}

int main()
{
    int a,b;
    cin>>a>>b;
    cout<<add(a,b);
    return 0;
}
```

Method 2

```
int add(int,int);
int main()
{
    int a,b;
    cin>>a>>b;
    cout<<add(a,b);
    return 0;
}

int add(int x, int y)
{
    int sum;
    sum = x + y;
    return sum;
}
```

Function

Array

```
void myFunction(int myNumbers[5]) {
  for (int i = 0; i < 5; i++) {
    cout << myNumbers[i] << "\n";
  }
}

int main() {
  int myNumbers[5] = {10, 20, 30, 40, 50};
  myFunction(myNumbers);
  return 0;
}
```

Function

Call by Value

```
#include<iostream>
using namespace std;
void func(int x)
{
 x += 10;
}
int main()
{
 int x=50;
 cout<<"Before func x is: "<<x<<endl;
 func(x);  // passing value to function
 cout<<"Before func x is: "<<x<<endl;
 return 0;
}
```

Call by Reference

```
#include<iostream>
using namespace std;
void func(int *x)
{
 *x += 10;
}
int main()
{
 int x=50;
 cout<<"Before func x is: "<<x<<endl;
 func(&x);  // passing value to function
 cout<<"Before func x is: "<<x<<endl;
 return 0;
}
```

Recursive Function

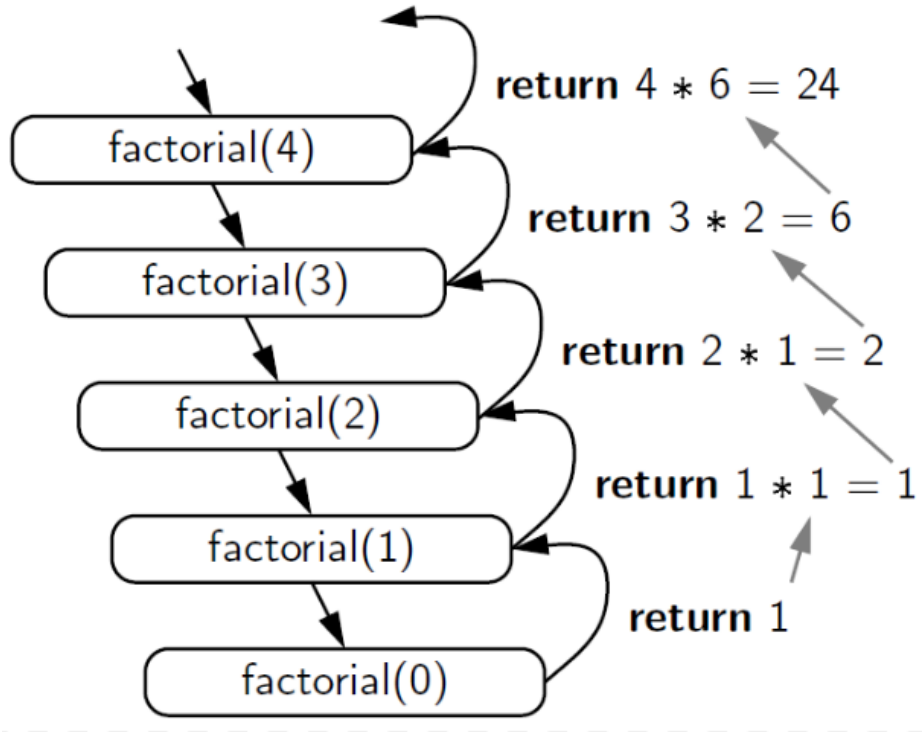
```
void recurse()
{
    ... .. ...
    recurse();
    ... .. ...
}

int main()
{
    ... .. ...
    recurse();
    ... .. ...
}
```

```
int fact(int x)
{
    if(x!=0)
        return x*fact(x-1);
    return 1;
}
int main() {
    int a = 5;
    cout<<fact(a);
  return 0;
}
```

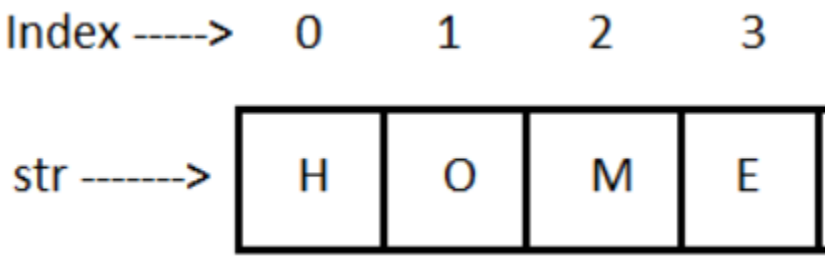
Recursive Function

```
int fact(int x)
{
    if(x!=0)
        return x*fact(x-1);
    return 1;
}
int main() {
    int a = 5;
    cout<<fact(a);
  return 0;
}
```



String

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str;
    cout << "Type your full name: ";
    getline(cin, str);
    cout << "Your name is: " << str;
}
```



File

Types

- Text
- Binary

Different classes in <fstream>

- ifstream (reading a file)
- ofstream (writing in a file)
- fstream (readding and writing in a file)

When a file is opened, an object is created and a **stream** is associated with it.

File: Open

Create an object

```
ifstream p1;
ofstream p2;
fstream p3;
```

Open file

```
P1.open("test.txt");
if(!p1)
  cout<<"cannot open file";
```

Close file

```
P1.close();
```

File: Read

```
#include <fstream>
using namespace std;
int main()
{
    //input file stream
    ifstream fin;
    fin.open("./test.txt");

    //ifstream fin("./test.txt");

    if(!fin)
        cout<<"File Not Open!";

    string str;
    fin>>str;
    cout<<str;

    fin.close();

    return 0;
}
```

File: write

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    //Output file stream
    ofstream fout("./test.txt",ios::out);
    if(!fout)
        cout<<"File Not Open!";

    fout<<"Hello"<<endl;
    return 0;
}
```

Object-oriented programming (OOP)

```
class Car {
  public:                 // Access specifier
  string Color;  // Attribute (string variable)
  void steer() { // Method defined inside the class
      cout << “rotate";
    }
};
```

```
int main() {
  Car myObj;  // Create an object of Car

  // Access attributes and set values
   myObj.Color = ”red”;

  // Print attribute values
  myObj.steer();
  return 0;
}
```

OOP - Access Specifiers

Public - members are accessible from outside the class

Private - members cannot be accessed (or viewed) from outside the class

```
class Car
{
    public:
        int x;
    private:
        int y;

};

int main()
{
    Car obj;
    obj.x = 10;
  //obj.y = 20; [Error] 'int Car::y' is private
    return 0;
}
```

OOP - Constructors

```
#include <iostream>
using namespace std;
class Car
{
    public:
        string color;
        Car(string s)
        {
            color = s;
        }

};

int main()
{
    Car obj("green");
    cout<<obj.color<<endl;
    return 0;
}
```