# Introduction to C++

# Agenda

- Introduction
- Comments
- Variables
- Datatypes
- Control Statements
- Operators
- Strings
- Arrays
- Functions
- Pointers
- Classes

# Introduction

# Introduction

- It is an amplification of C.
- It is not platform independent but it is machine independent.
- It is a compiled language.
- It supports object-oriented programming language.
- It supports Pointers.
- Execution speed is faster than other programming languages.
- It is not platform independent but it is machine independent.
- It is Case insensitive.

# C++ Syntax

| | |
|---|---|
| **#include<iostream>** | Header file that contains functions for i/o operations |
| **#include <string>** | string header for using string class |
| **using namespace std;** | namespace |
| **Int main(){** | Execution of a program starts from here |
| **cout<<"Hello World";** | It will print Hello World on screen |
| **return 0;** | Execution of program is successful |
| **}** | |

# Application

- It is used in the development of operating system and web browser
- It is also used in gaming applications

# Comments

# Comments

- Description of information or details of source code in a program.
- It is ignored by compiler and will not execute.
- Comments can be **single line** or **Multiline:**

**Single Line Comments:** //

**Syntax:**

```
// code information
```

**Multi Line Comments:**

**Syntax:**

```
/*
  code information
*/
```

# Variables

# Variables

- It is used to store values.

**Syntax:**

```
//Initialization
Data_type  var_name= val;
```

**Example:**

```
#include <iostream>
using namespace std;
int main() {
int score= 82;
cout<<" Marks scored is "<<score;
```

# Variables Scope

Variables scope is of two types primarily:

**Local variable-** It is declared inside the function or method and cannot be accessed outside the function or method.

```
#include<iostream>
using namespace std;
void print(){
        int marks=92;          // local variable
        cout<<marks;
}
int main()
{
        cout<<"Marks obtained is ";
        print();

        return 0;
}
```

# Variables Scope

**Global variable-** It is generally declared outside the function and can be accessed throughout the program.

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

int glob = 10;                  // global variable

void print() {
   cout<<glob<<endl;
}

int main() {
   print();

   glob = 100;
   print();
}
```

# User Input/Output

# User Input/Output

**cin -** Standard Input stream

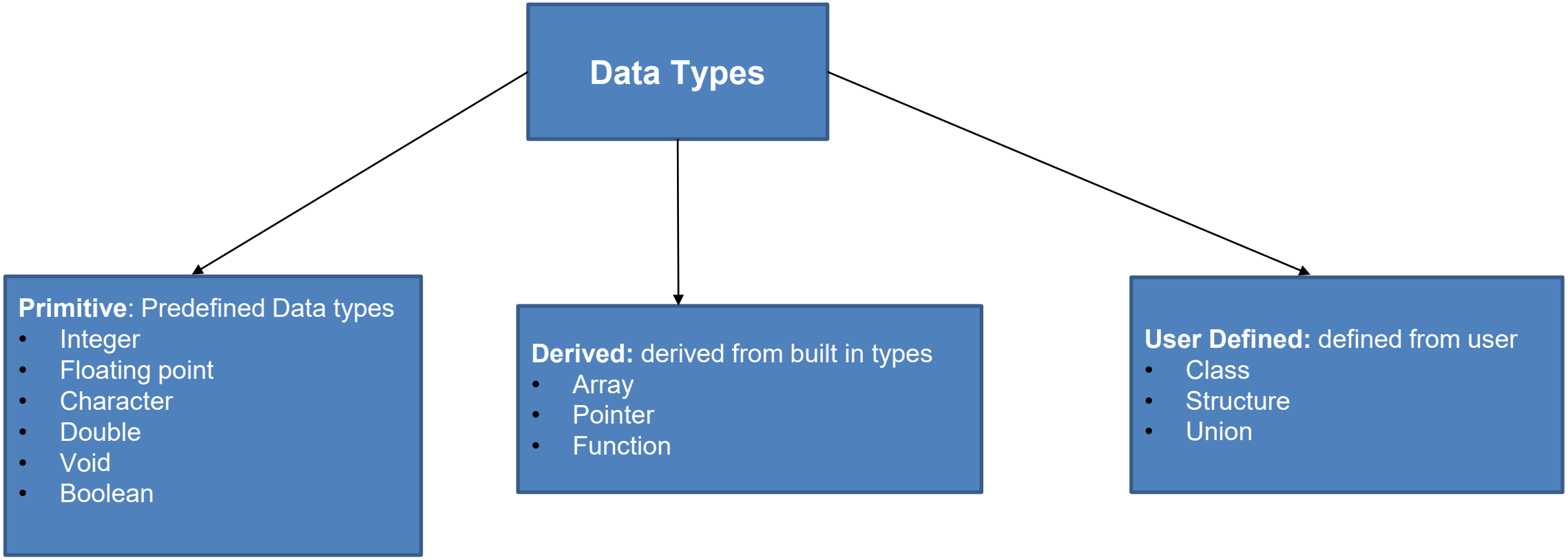
**cout -** Standard Output stream

**Demo:**

```
#include <iostream>
using namespace std;
int main(){
   int x,y;
   cout<<"Enter a number";
   cin>>x;
   cout<<"Enter a Number";
   cin>>y;
   cout<< " Addition of two numbers is "<<x+y<<endl;
   cout<< " Subtraction of two numbers is "<<x-y<<endl;
```

# Datatypes

# Datatypes



# Control Statements

# Control Statements

## Conditional statements:

### The if Statements-

Condition is true, Statements will be executed.
Condition is false, Statements will be skipped.

**Syntax:**

```
if (condition)
 {
 statement;
 }
```

# Control Statements

## Conditional statements:

**The if else Statements-** It is having two parts if and else.

Condition is True, if part will be executed.
Condition is False, else will be executed.

**Syntax:**

```
if (condition)
 {
 statement1;
 }
else{
statement2;
}
```

# Control Statements

## Conditional statements:

**The else if Statements-** used to set out a new condition when first condition is False.

**Syntax:**

```
if (con 1)
 {
 statement1;          // statement will execute if con1 is true
 }
 else if(con 2){
 statement2;         // statement will execute if con1 is false and con2 is true
 }
else{
statement3;        // execute if both the above condition are false
}
```

# Control Statements

## Conditional statements:

**Switch case:**

multiway branch statement
alter of if-else

**Syntax:**

```
switch (x)
 {
    case 1: // execution of code if x = 1;
      break;
    case 2: // execution of code if x = 2;
      break;

   default: // execution of code when x value does not match in above any cases
 }
```

# Control Statements

**Loops-** Repetition of a sequence of instruction to reach a certain condition

**For loop-** It will repeat the same statement till the condition is achieved.

**Syntax:**

```
for(i=0;i<n;i++){
 // statement
 }
```

**While loop-** It is having one control condition and will execute till the condition is True.

**Syntax:**

```
while(condition is True){
  //information
 }
```

# Operators

# Operators

| Types | Operators |
| --- | --- |
| Arithmetic Operator | **+, - , * , / , %** |
| Relational Operator | **< , > , <= , >= , ==, !=** |
| Logical Operator | AND(**&&**), OR(**||**), NOT(**!**) |
| Bitwise Operator | **^ , & , | , << , >>** |
| Assignment Operator | **= , += , -=, *=** |
| | |

# Strings

# Strings

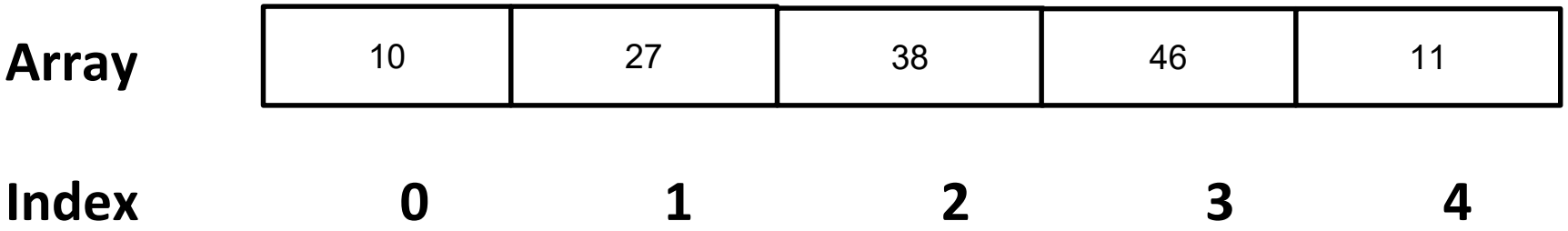
- It is used to store sequence of characters.
- Concatenation in string is done by attaching two strings.
- String functions are used by importing <string> library
  **length()-** gives size or length of string
  **substr(x,y)-** gives size of sub-string in a string
- **Demo**

```
#include <iostream>
using namespace std;
int main(){
string g="gaurav";
cout<<g;
}
```

# Arrays

# Arrays

- Collection of same data type elements stored in adjacent memory locations.
- Each element in an array has a distinct index.
- Any elements in an array can be accessed through indexing.



# Initialization of an Array

## Array Initialization:

### 1) Static

```
int x[6]={3,5,9,2,8,7}

char c[6]= {‘g’,’a’,’u’,’r’,’a’,’v’}
```

### 2) Dynamic

```
int x[6];
for(int i=0;i<6;i++)
cin>>x[i]
```

# Arrays

Arrays are of two types:

**Single Dimensional Array:**

**Syntax:**

data_type  name_of_an_array [size_of_an_array]

int a[5]; // Single dimensional array

**Multi Dimensional Array:**

**Syntax:**

data_type name_of_an_array[s1][s2][s3]..[sN]

int a[10][5]; // 2-D array

# Functions

# Functions

Block of code runs on invoking or calling.

Instead of writing similar code again and again functions can be invoked to execute for different outputs

**Syntax:**

```
returntype  fxn_name(arg_type arg_name,.....)  {
 Code
   }
```

# Function Overloading

- It is a process in which two or more functions have same name but different parameters.
- To perform function overloading return type, type of parameters or number of parameter must be different.

**Demo:**

```
int sub(int a,int b){
return a-b;
 }
double sub(double a,double b){
return a-b;
}
int main(){
cout<<sub(10,7)<<endl;
cout<<sub(5.25,3.66);
}
```

# Pointers

# Pointers

It is a variable which is used to store address of another variable.

**Syntax:**

```
int x= 5;

int *ptr;  //ptr is a pointer

ptr= &x;   // Storing address of x in ptr

cout<<ptr ; //  address of pointer
```

# Classes

# Classes

## Class

- It is a blueprint for an object.
- C++ supports an object-oriented programming language.
- It is user defined data type.

## Object

- It is a real world entity.
- It is an instance of a class.

# Classes

## Constructor

- It is a special type of method which is used to initialize an object.
- Different types of Constructor are-

  **Default Constructor-** no parameter or argument is passed.

  **Parameterized Constructor-** used to pass arguments.

# Classes

## Inheritance

- It is a feature of object-oriented programming.
- It is a process in which a class acquires properties and behaviour from different class.

**Super class-** Class which is inherited.

**Sub Class-** Class which inherits from different class

# Hands on

# Summary

- C++ basic concepts
- Variables and Datatypes
- Control Statements
- Object oriented Programming

# Thank You