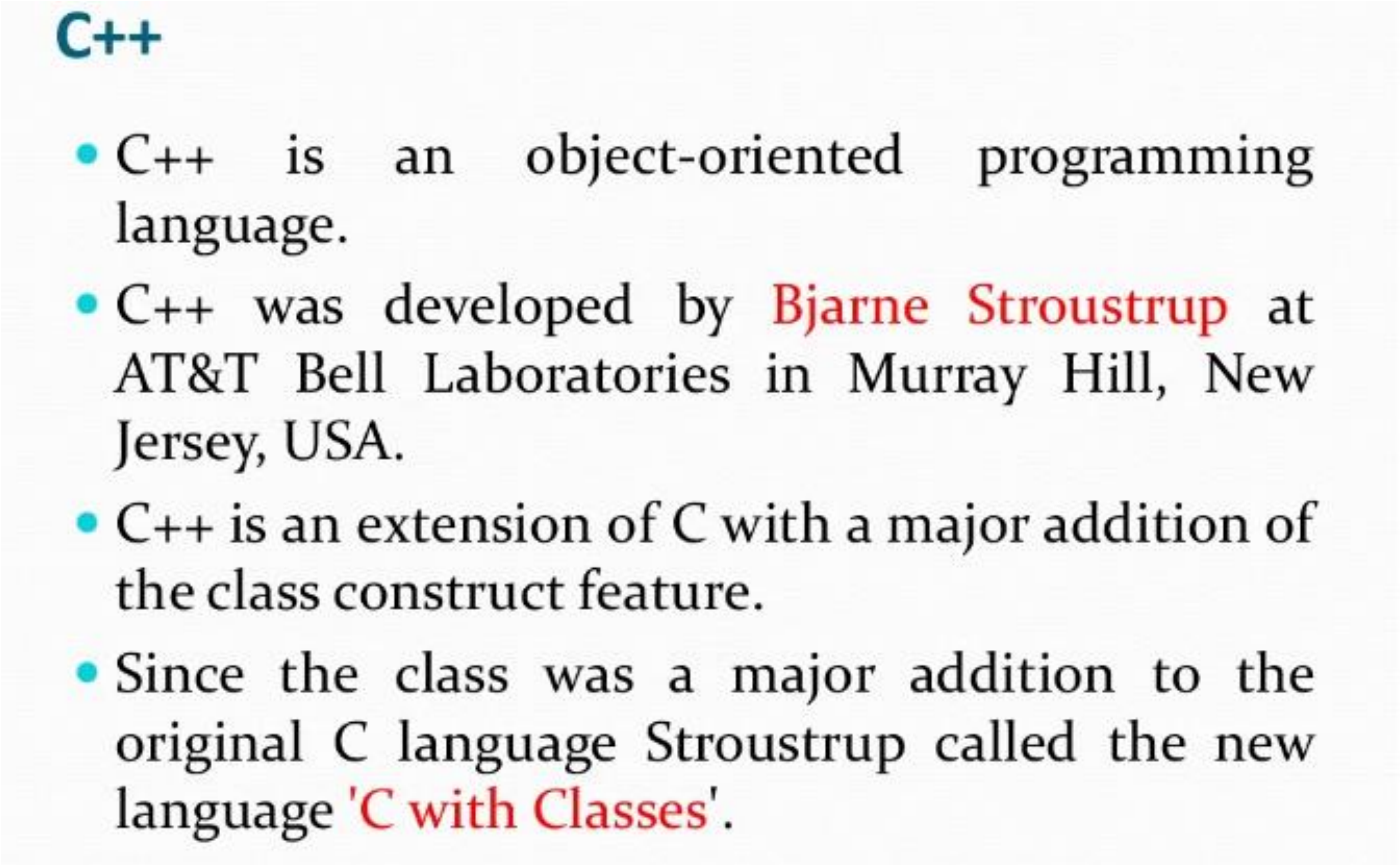
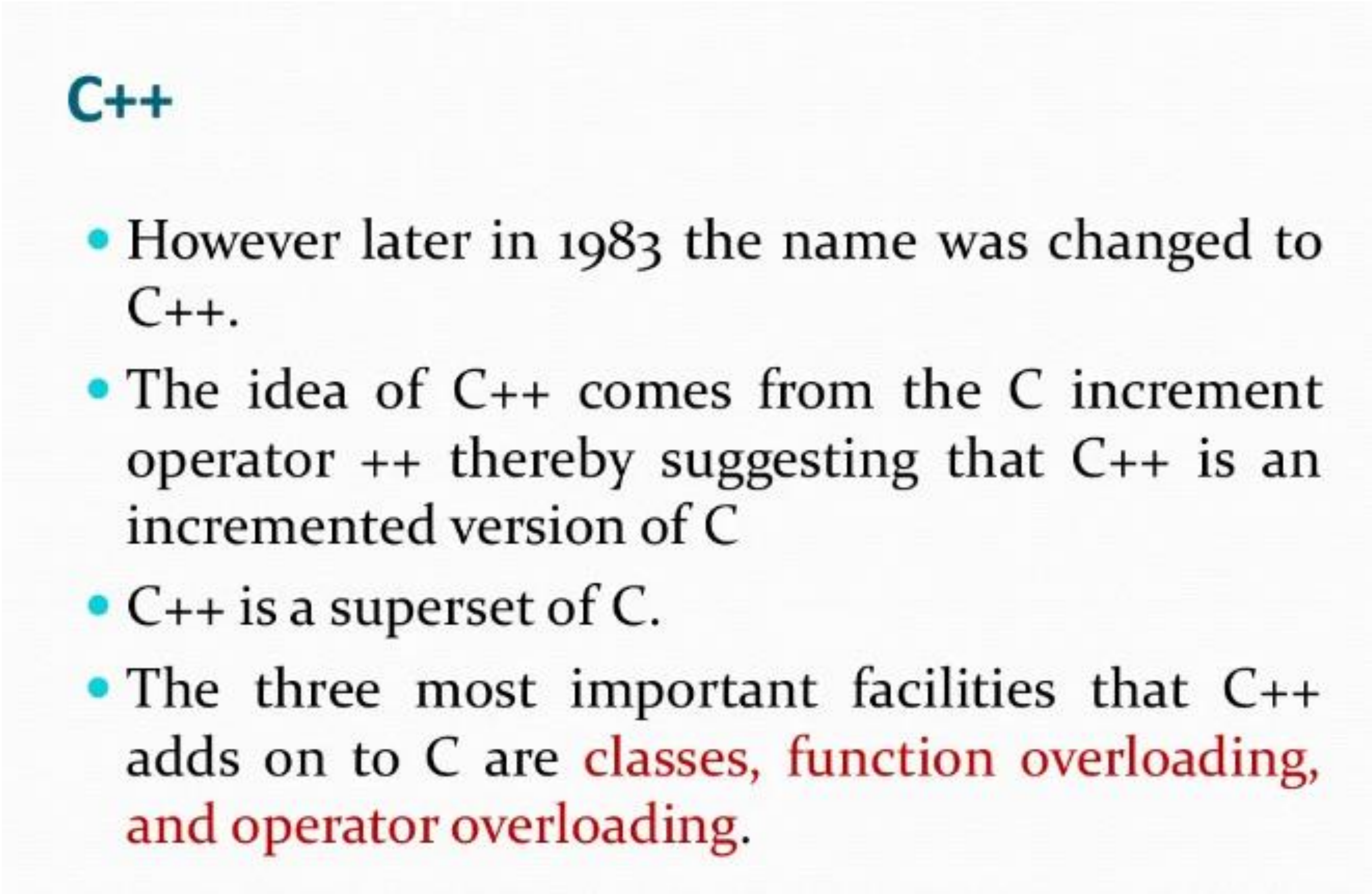
Lecture 2 – C++

19CSE201 Advanced Programming



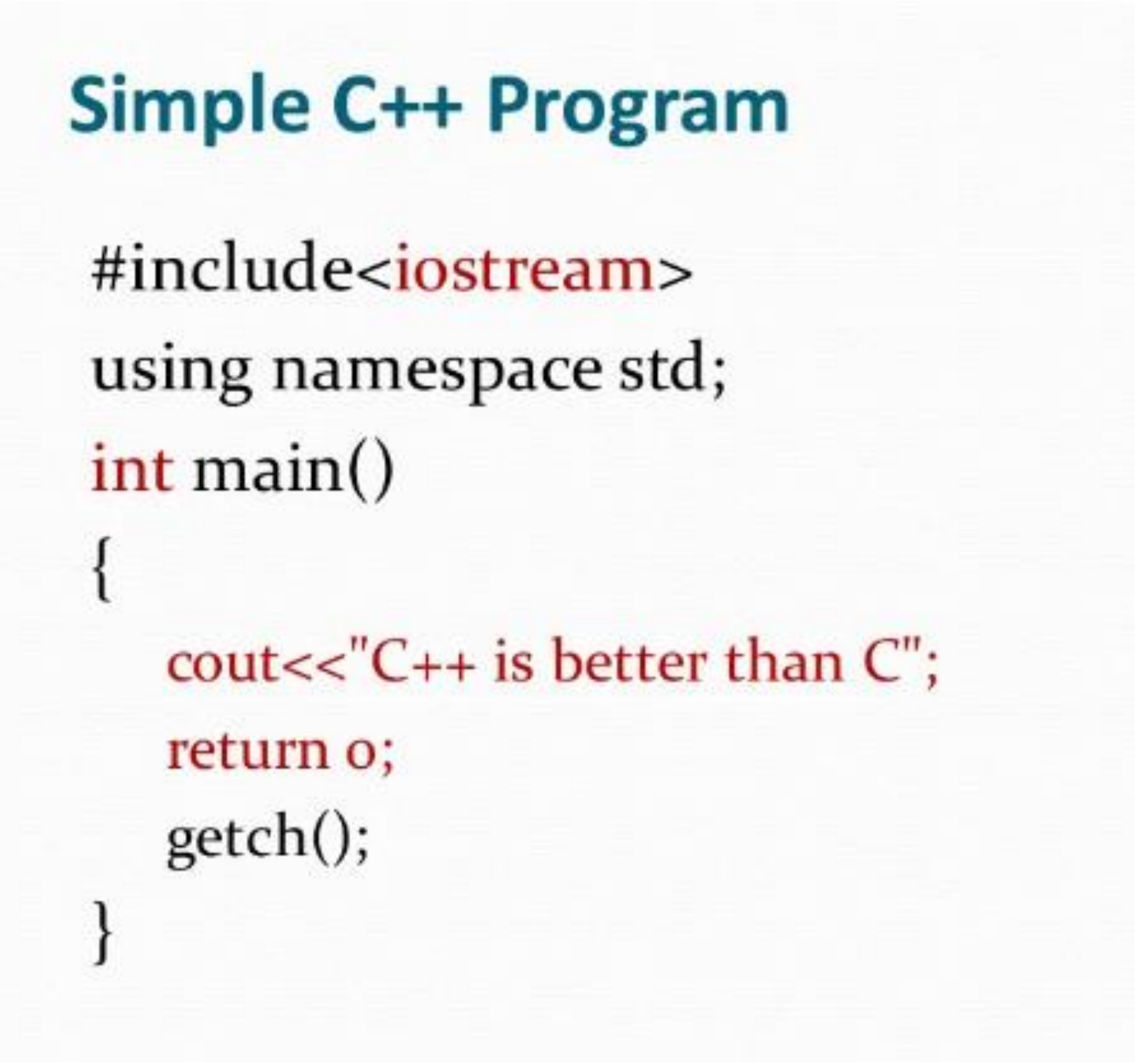
C++

- C++ is an object-oriented programming language.
- C++ was developed by Bjarne Stroustrup at AT&T Bell Laboratories in Murray Hill, New Jersey, USA.
- C++ is an extension of C with a major addition of the class construct feature.
- Since the class was a major addition to the original C language Stroustrup called the new language 'C with Classes'.



C++

- However later in 1983 the name was changed to C++.
- The idea of C++ comes from the C increment operator ++ thereby suggesting that C++ is an incremented version of C
- C++ is a superset of C.
- The three most important facilities that C++ adds on to C are classes, function overloading, and operator overloading.



Simple C++ Program

```
#include<iostream>
using namespace std;
int main()
{
    cout<<"C++ is better than C";
    return 0;
    getch();
}
```

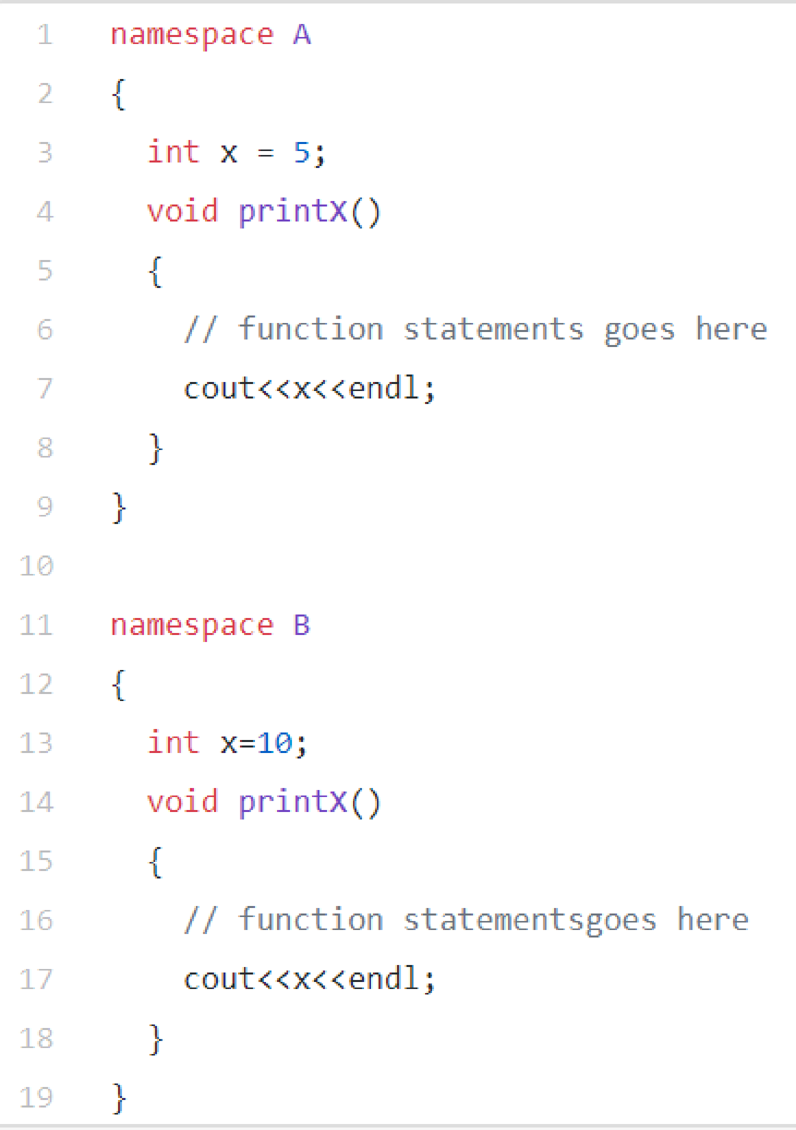
#include<iostream>

- iostream- input output stream
- tell the compiler that we need to include the contents of this file

using namespace std;

- New concept in c++
- A **namespace** is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) that are used in program.
- Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries
- In std, all C++ standard class libraries are defined.
- All built-in classes, functions, templates are declared within the namespace named std.
- these are defined in the std namespace in the iostream file.

- In programming, we cannot have variables, functions, etc. with the same name. So to avoid those conflicts we use namespaces.



```
namespace A
{
  int x = 5;
  void printX()
  {
    // function statements goes here
    cout<<x<<endl;
  }
}

namespace B
{
  int x=10;
  void printX()
  {
    // function statementsgoes here
    cout<<x<<endl;
  }
}
```

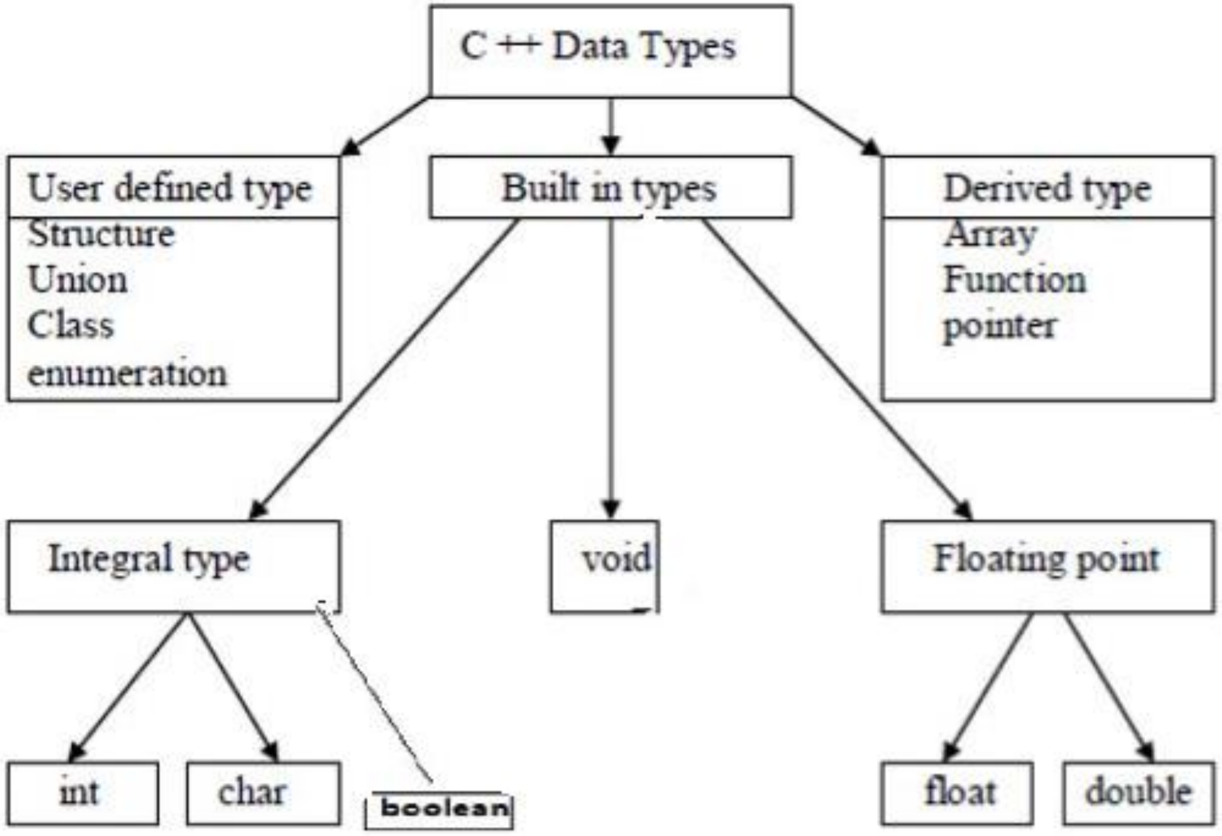
int main()

- Execution of C++ pgm starts from main.
- Only one main in a pgm
- Always return type of main is int
- Member functions of a class is called inside the main

Tokens

- Tokens-Basic Units of Programming
    - Keywords
    - Identifiers
        - Variables
    - Constants
    - Operators

Data types



C ++ Data Types

User defined type
Structure
Union
Class
enumeration

Built in types

Derived type
Array
Function
pointer

Integral type

void

Floating point

int

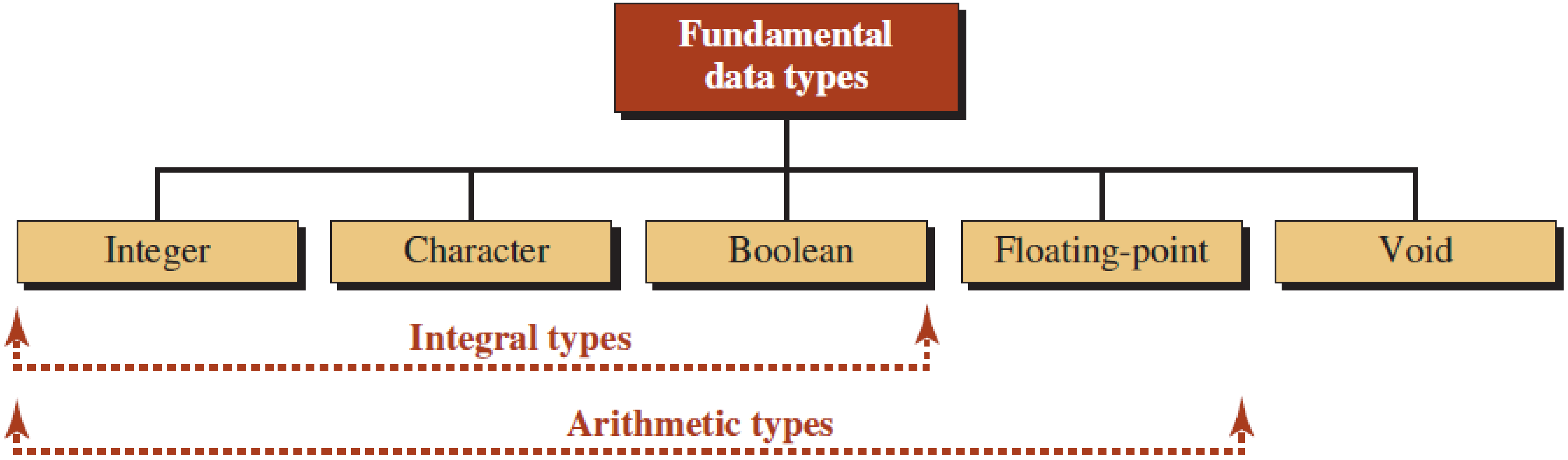
char

boolean

float

double

Built-in type



int

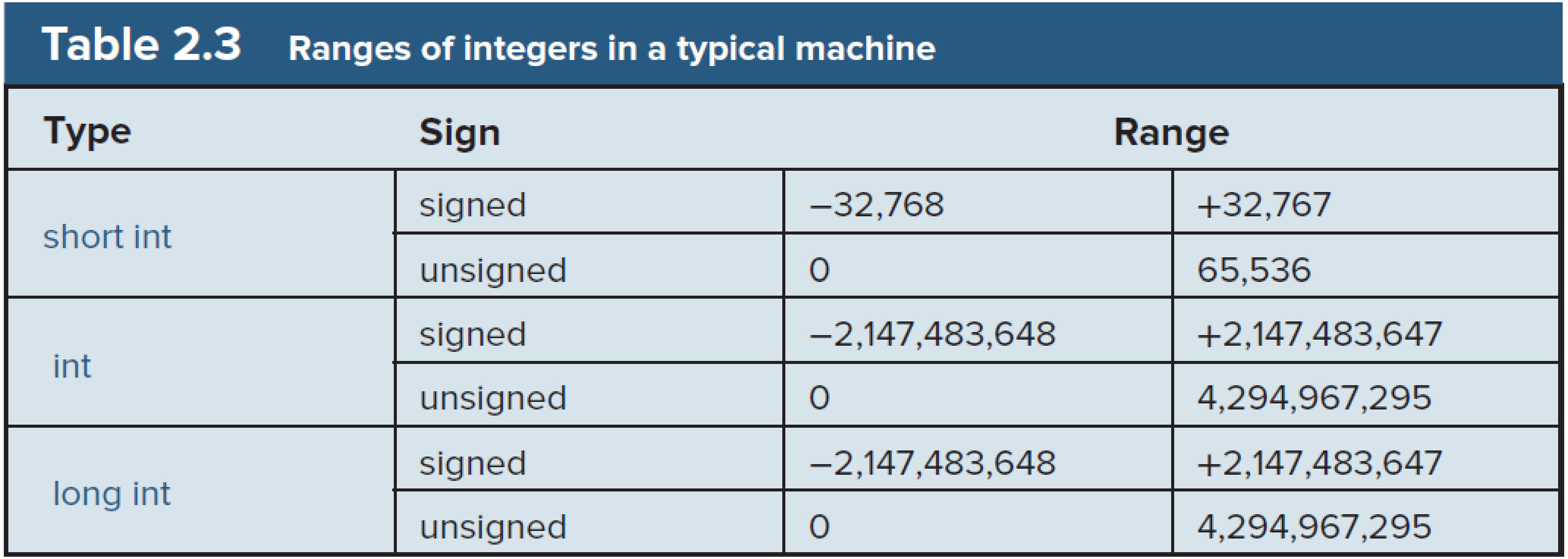
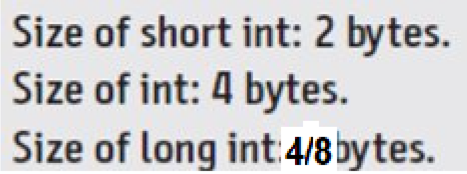


Table 2.3 Ranges of integers in a typical machine

Type	Sign	Range	
short int	signed	–32,768	+32,767
	unsigned	0	65,536
int	signed	–2,147,483,648	+2,147,483,647
	unsigned	0	4,294,967,295
long int	signed	–2,147,483,648	+2,147,483,647
	unsigned	0	4,294,967,295



```
Size of short int: 2 bytes.
Size of int: 4 bytes.
Size of long int:4/8bytes.
```

char

- char – 1 byte
- Character literals are always enclosed within single quotes.

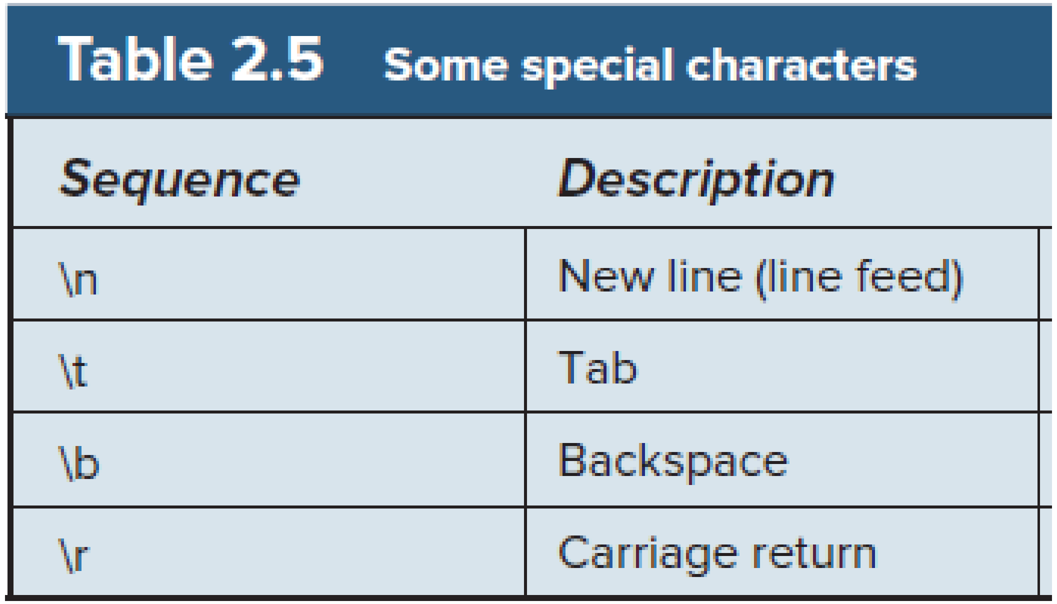
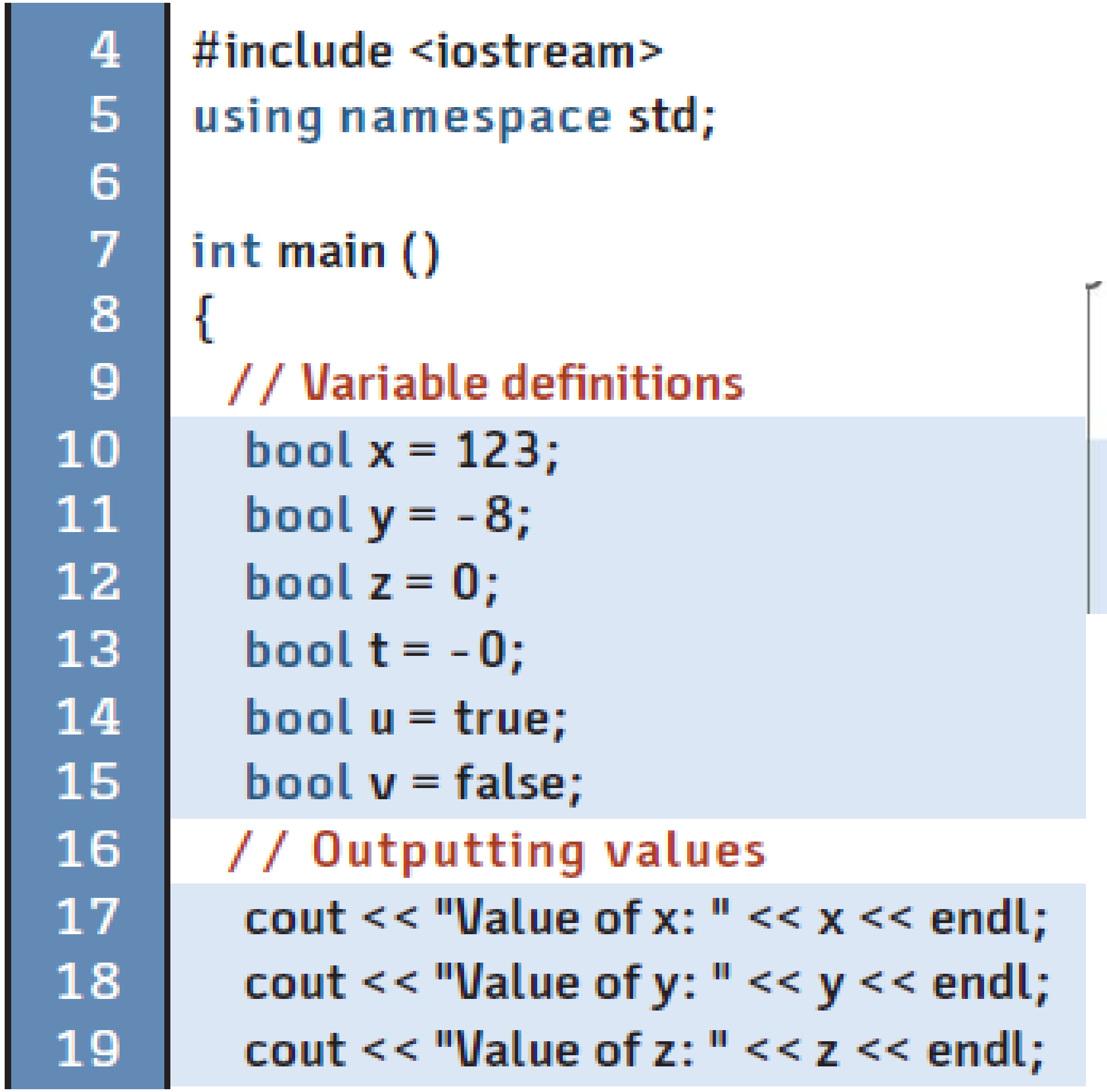


Table 2.5 Some special characters

Sequence	*Description*
\n	New line (line feed)
\t	Tab
\b	Backspace
\r	Carriage return

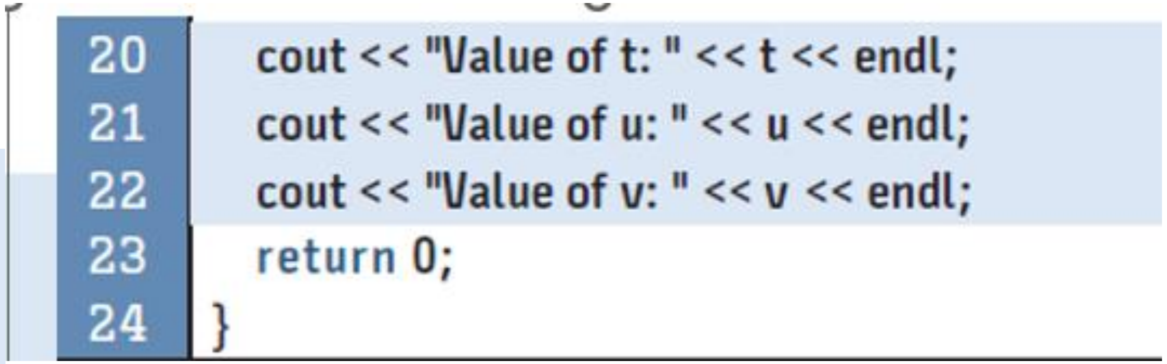
boolean

- The keyword used is bool
- **The size of a bool data type is 1 byte.**
- used to represent the result of comparing two values.
- if we compare the integers 23 and 24 for equality, we get **false**
- However, if we compare them for inequality, we get **true**.

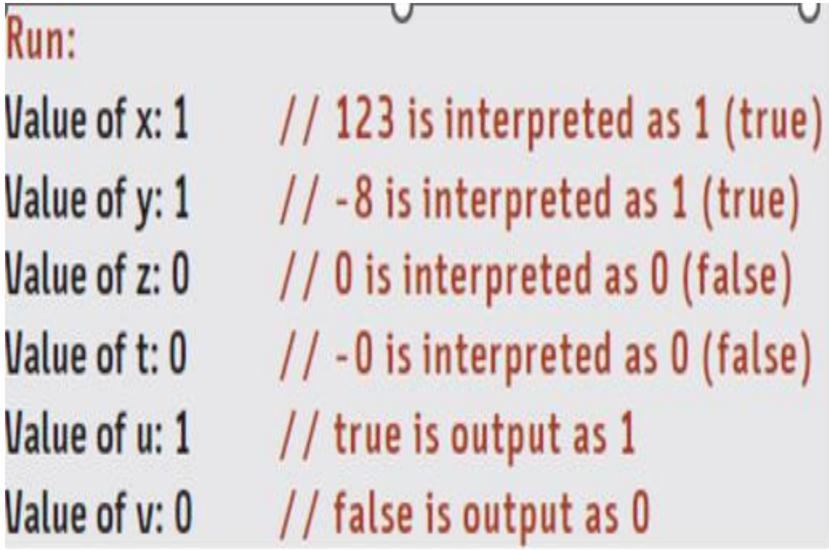


```
#include <iostream>
using namespace std;

int main ()
{
  // Variable definitions
    bool x = 123;
    bool y = -8;
    bool z = 0;
    bool t = -0;
    bool u = true;
    bool v = false;
  // Outputting values
    cout << "Value of x: " << x << endl;
    cout << "Value of y: " << y << endl;
    cout << "Value of z: " << z << endl;
```



```
    cout << "Value of t: " << t << endl;
    cout << "Value of u: " << u << endl;
    cout << "Value of v: " << v << endl;
    return 0;
}
```



```
Run:
Value of x: 1     // 123 is interpreted as 1 (true)
Value of y: 1     // -8 is interpreted as 1 (true)
Value of z: 0     // 0 is interpreted as 0 (false)
Value of t: 0     // -0 is interpreted as 0 (false)
Value of u: 1     // true is output as 1
Value of v: 0     // false is output as 0
```

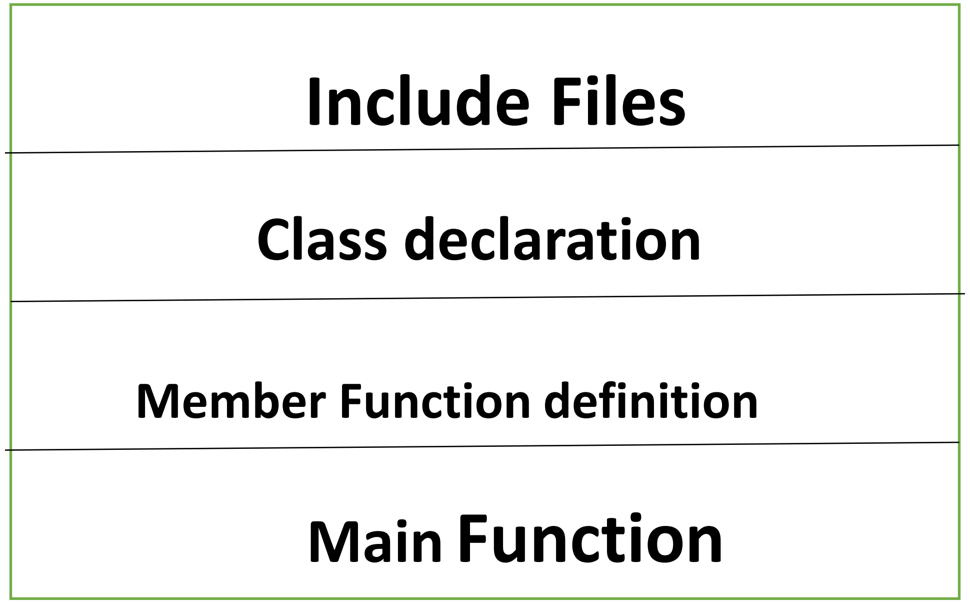
Floating point type

- float
- double
- long double
- **All floating-point numbers are signed.**

void

- A special type that has no value.
- We can use **void** to show that a function returns no value.

Basic Structure of C++ program



Include Files

Class declaration

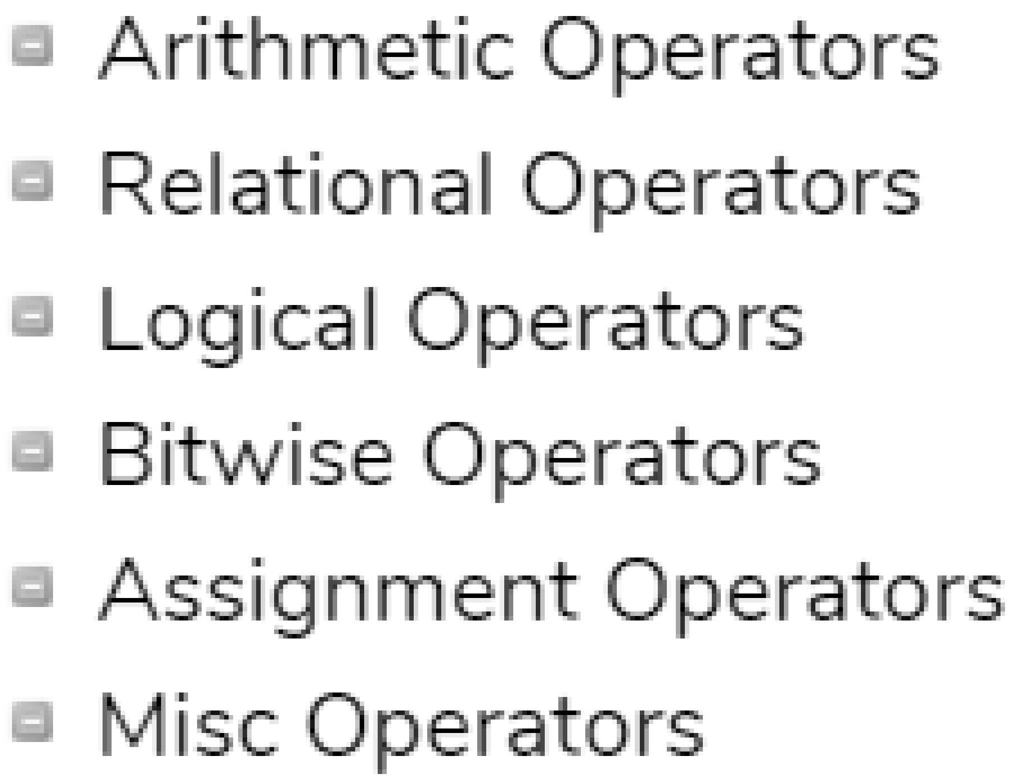
Member Function definition

Main Function

Basic C++ program without class

```
Include header files
int main()
{
-------
}
```

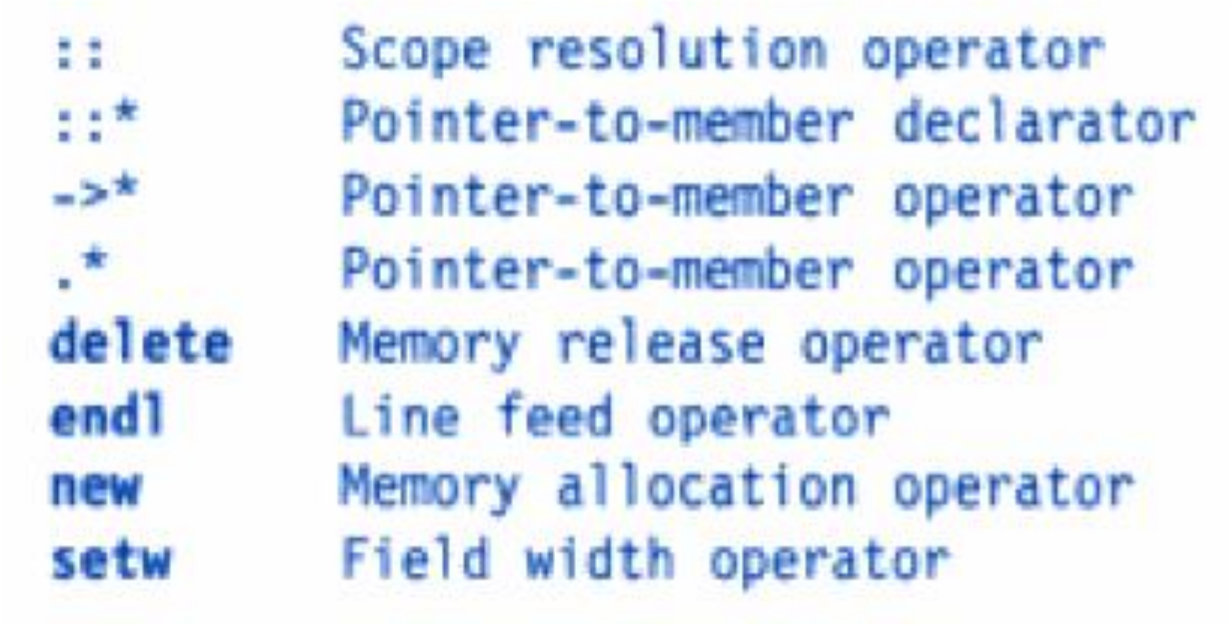
Operators in C++



- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Misc Operators

Operators in C++

- Insertion operator <<
- Extraction operator >>



::	Scope resolution operator
::*	Pointer-to-member declarator
->*	Pointer-to-member operator
.*	Pointer-to-member operator
delete	Memory release operator
endl	Line feed operator
new	Memory allocation operator
setw	Field width operator

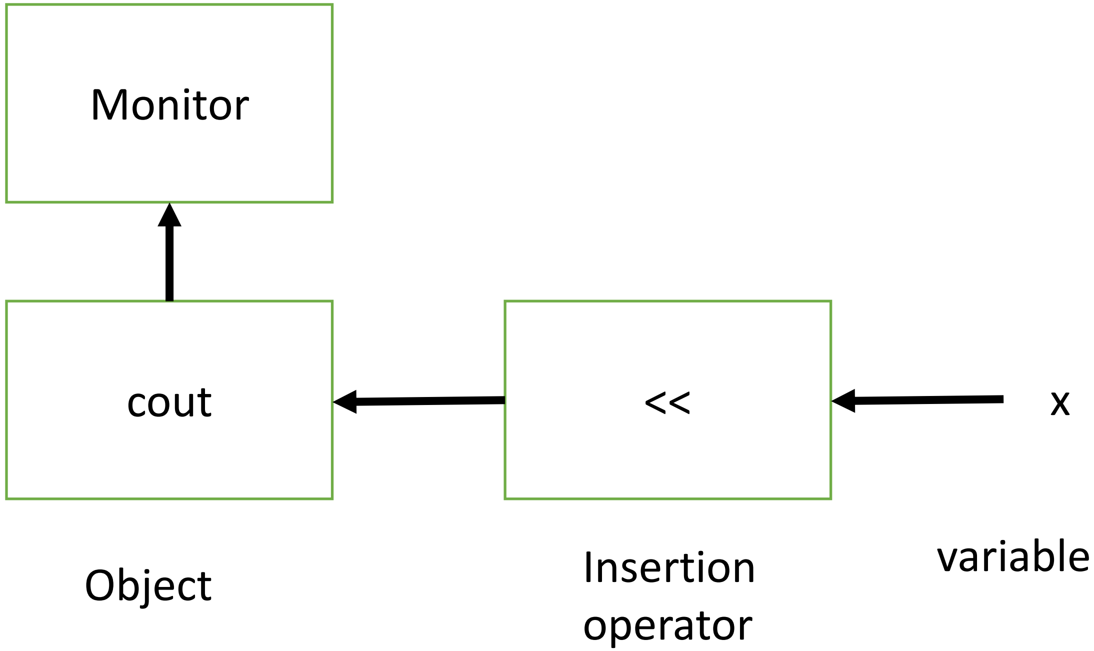
output operator

- Output operator is <<
- Also called **insertion or put to** operator

  eg: cout<<“helloworld”;

  cout<<“sum is”<<x;

  cout<<x<<“\n”<<y;
- cout is **object** of ostream class
- **cout<<x** - operator << inserts the value of variable x to the object cout
- cout will display the contents on the screen

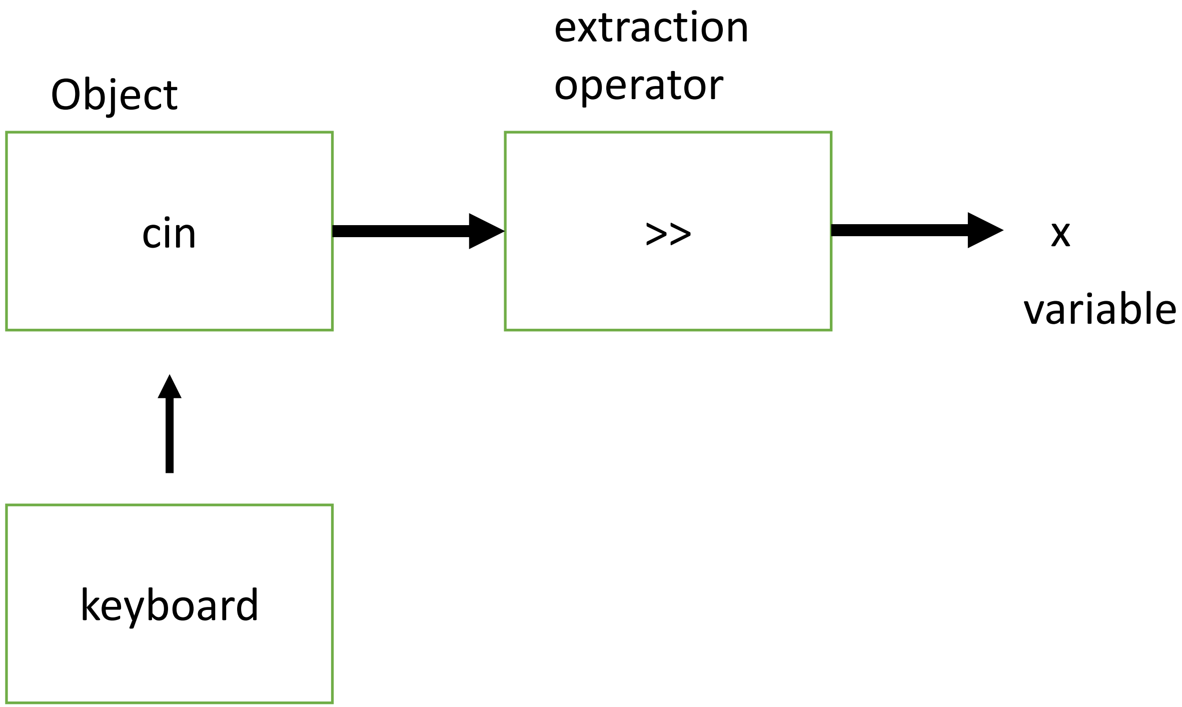

Monitor
cout
<<
x
Object
Insertion operator
variable

Input Operator

- Input operator is >>
- Also called **extraction or get from** operator

  eg: cin>>x;

  cin>>a>>b>>c;
- cin is **object** of istream class
- **cin>>x** - The object cin gets value from keyboard and operator >> assign the value to the variable x.


extraction
operator
Object
cin
>>
x
variable
keyboard

Pgm to print helloworld

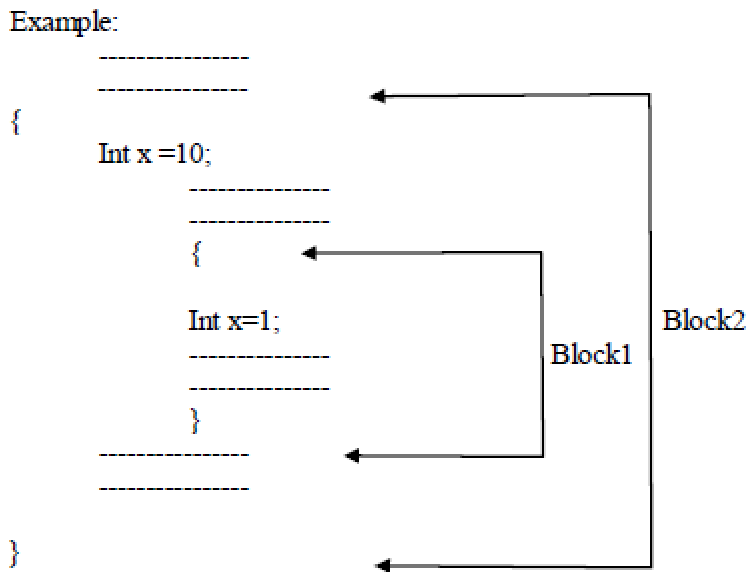
```
#include<iostream>
using namespace std;
int main()
{
cout<<“helloworld”;
}
```

Pgm to find sum of two numbers

```
#include<iostream>
using namespace std;
int main()
{
int x=2, y=4, z;
z=x+y;
cout<<“sum is”<<z;
}
```

Scope resolution operator ::

- In C, global version of a variable can't be accessed from with in the inner block.
- C++ resolves this problem by introducing a new operator ::



```
Example:
        ----------------
        ----------------
{
        Int x =10;
                ---------------
                ---------------
                {
                Int x=1;
                ---------------
                ---------------
                }
        ----------------
        ----------------
}
```

Can we access x with value 10 from inner block?

In C

```
int x=10;

main()
{         {int x=2;
          printf("%d",x);
          }
Printf(x)
}
```

In C++

```
int x=10;
main()
{         {int x=2;
          cout<<x;
          cout<<::x;
          }
}
```

```
int m=10;
main()
{
    int m=20;
    {
        int k=m;
        int m=30;
        cout<<”we are in inner block”;
        cout<<"k="<<k<<endl;
        cout<<"m="<<m<<endl;
        cout<<":: m="<<::m<<endl;
    }
    cout<<”\n we are in outer block \n”;
    cout<<"m="<<m<<endl;
    cout<<":: m="<<:: m<<endl;
}
```

Output

```
we are in inner block
k=20
m=30
::m=10          //always retrieve global value
we are in outer block
m=20
::m=10
```

Always :: will refer the global value.
If there is no global value, it gives error.

Collectively (to resolve name conflicts)

- So Scope resolution operator is **Accessing a global variable when there is a local variable with same name**.
- Scope resolution operator also used to access the members of another class/ Referring to a class inside another class

  Eg: std::cout<<x;
- Used to define member function outside the class
- Accessing a class’s static variables
- In case of multiple Inheritance

Reference variable - output?

```
int total =100;
int &sum =total;
cout<<total<<endl;
cout<<sum<<endl;
total= total+10;
cout<<total<<endl;
cout<<sum<<endl;
sum=0;
cout<<total<<endl;
cout<<sum;
```

Output

```
100
100
110
110
0
0
```

Control structures

- Decision making
  - Simple if, if else, else if ladder, nested if else
  - switch case
- Loops
  - for, while, do while

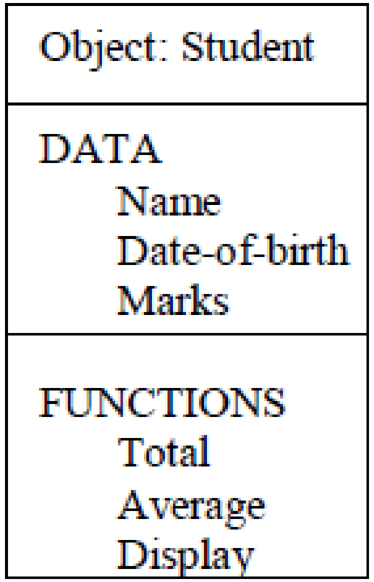
```
for(i=0;i<10;i++)
{if(i % 5==1)
   continue;
cout<<i;}
```

Object Oriented concepts

1. Objects
2. Classes
3. Data abstraction and encapsulation
4. Inheritance
5. Polymorphism
6. Dynamic binding
7. Message passing

Objects

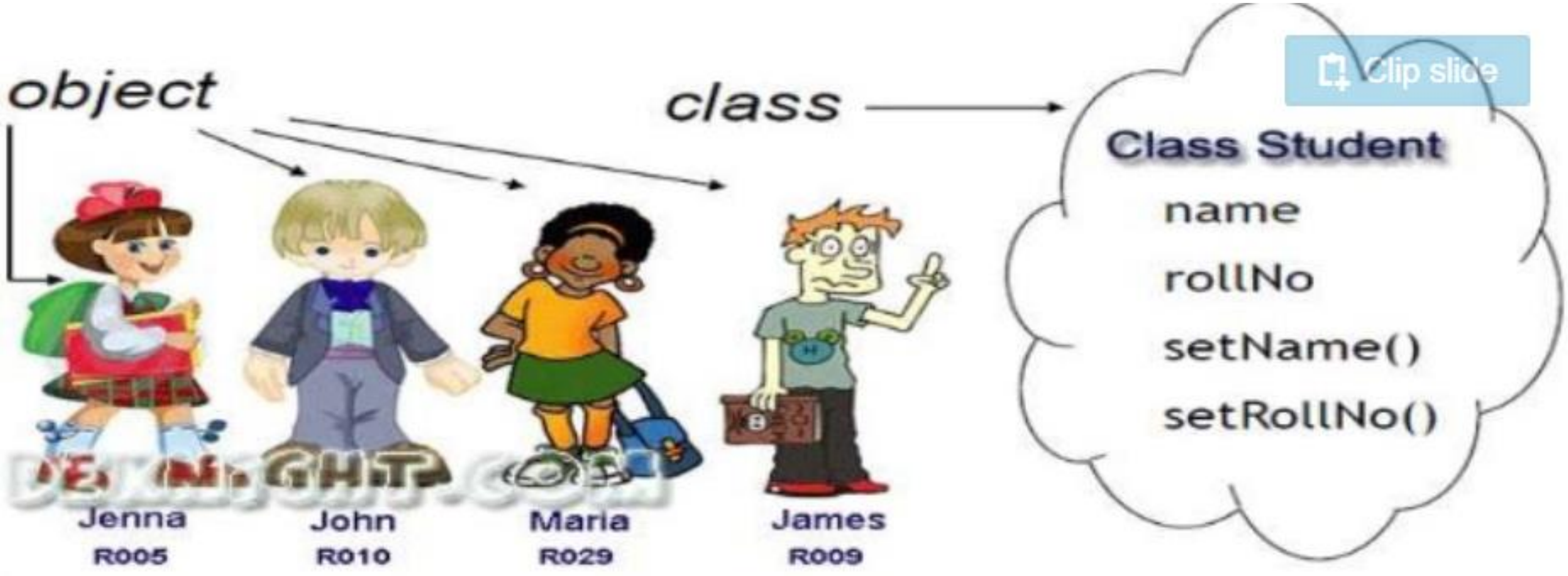
- Objects are the basic run-time entities in an object-oriented system.
- Objects will have some attributes or characteristics and that makes it different from others.
- They may represent a person, a place, a bank account, a table of data or any item that the program must handle.
- The fundamental idea behind object oriented approach is to combine both data and function into a single unit and these units are called objects.

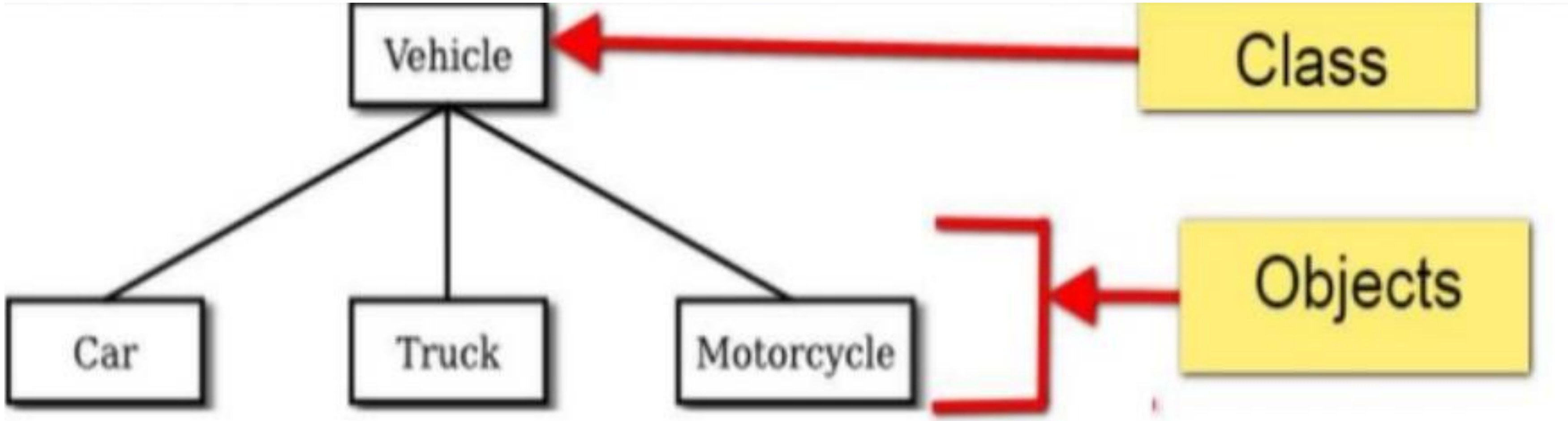


Object: Student

DATA
Name
Date-of-birth
Marks

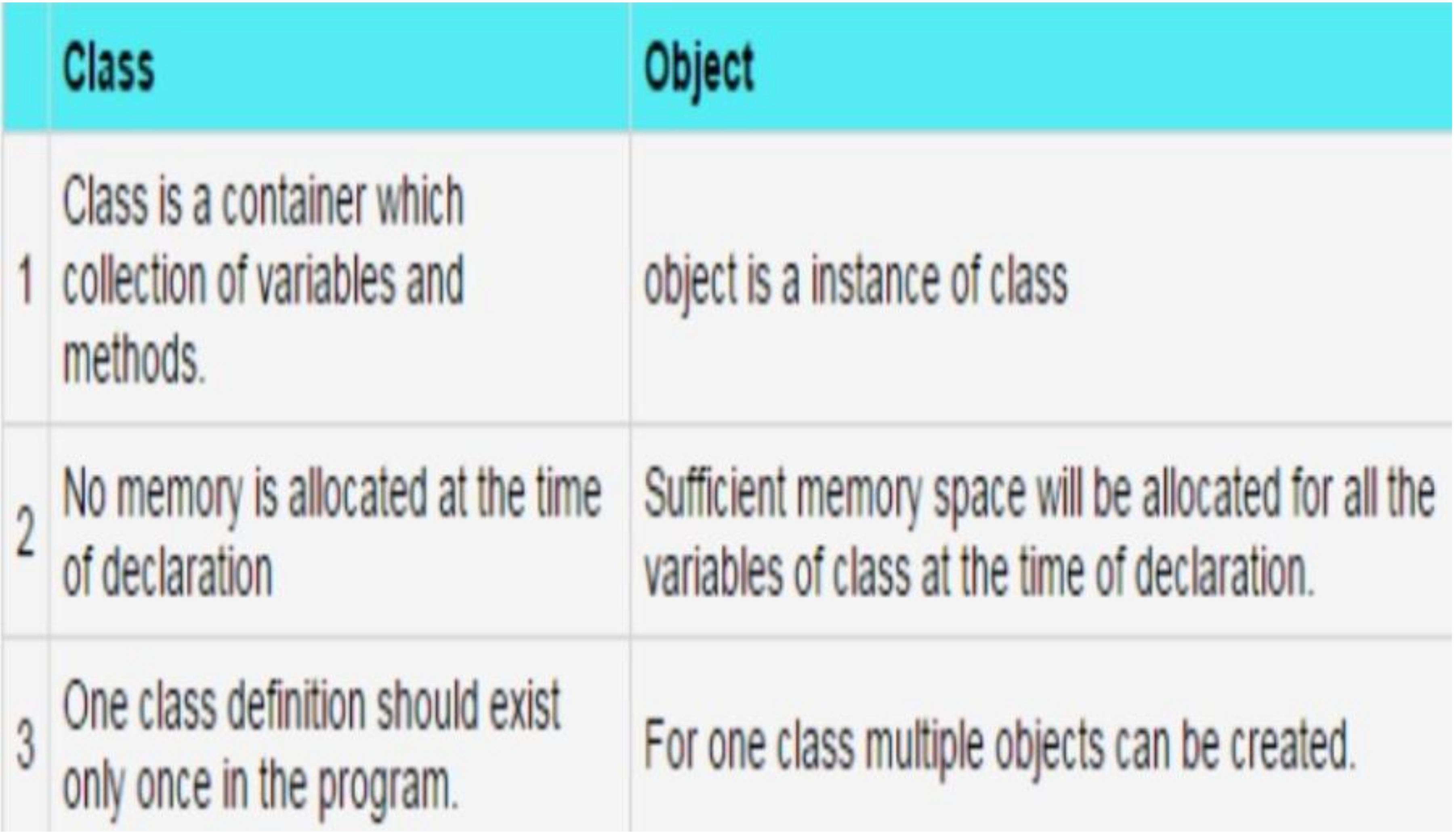
FUNCTIONS
Total
Average
Display


object
class
Clip slide
Class Student
name
rollNo
setName()
setRollNo()
Jenna
R005
John
R010
Maria
R029
James
R009


Vehicle
Class
Car
Truck
Motorcycle
Objects

Class

- A group of objects that share common properties for data part and some program part are collectively called as class.
- In C ++ a class is a new data type that contains member variables and member functions that operate on the variables.



	Class	Object
1	Class is a container which collection of variables and methods.	object is a instance of class
2	No memory is allocated at the time of declaration	Sufficient memory space will be allocated for all the variables of class at the time of declaration.
3	One class definition should exist only once in the program.	For one class multiple objects can be created.

How to declare a class

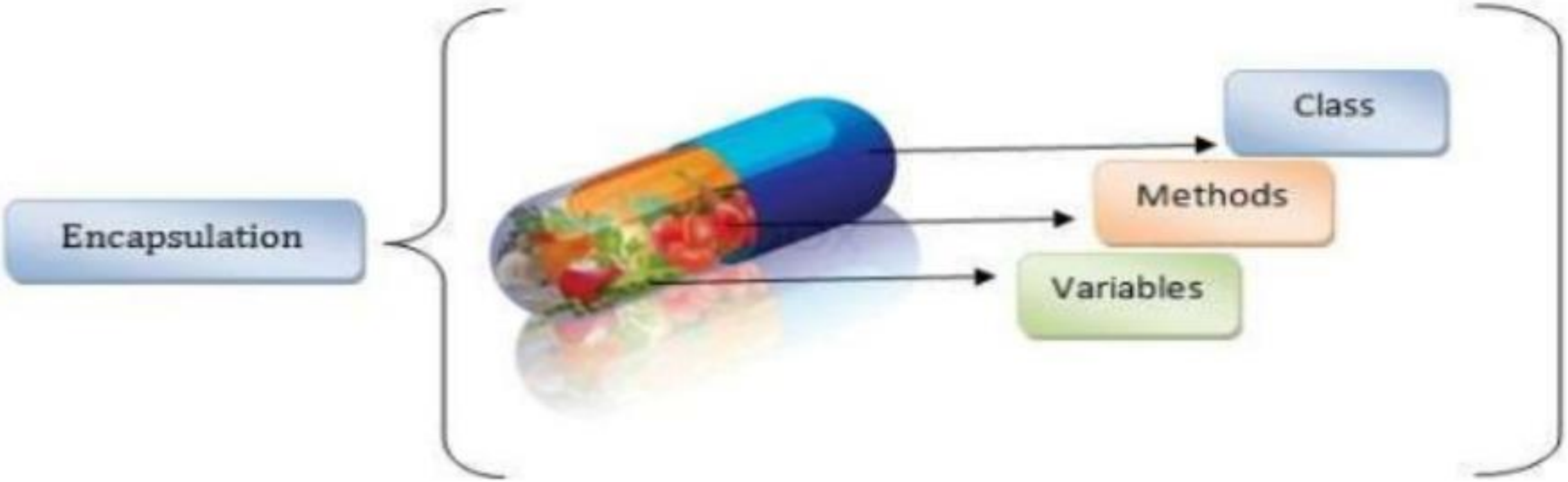
```
class classname
{
variable declaration;
function declaration;
};
main()
{
classname object;
}
```

or

```
class classname
{
variable declaration;
function declaration;
}object;
main()
{
-------
}
```

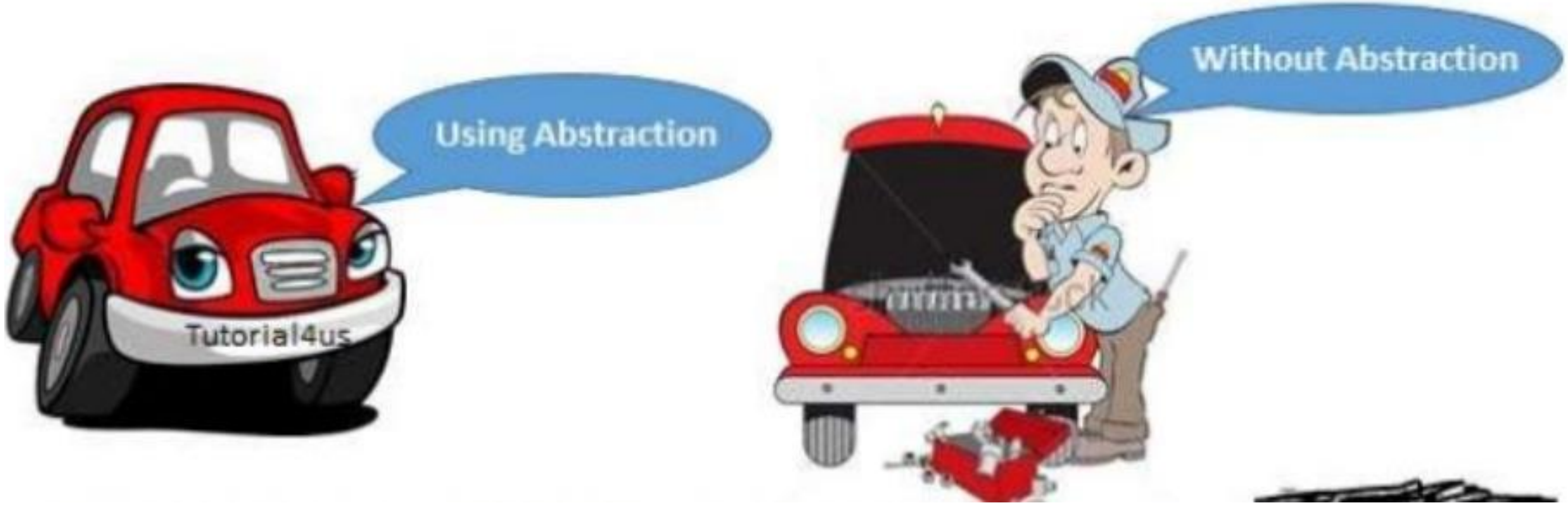
Data Encapsulation

- The wrapping up of data and function into a single unit (called class) is known as encapsulation.
- The data is not accessible to the outside world and only those functions which are wrapped in the class can access it.
- The feature encapsulation is supported using class.


Encapsulation
Class
Methods
Variables

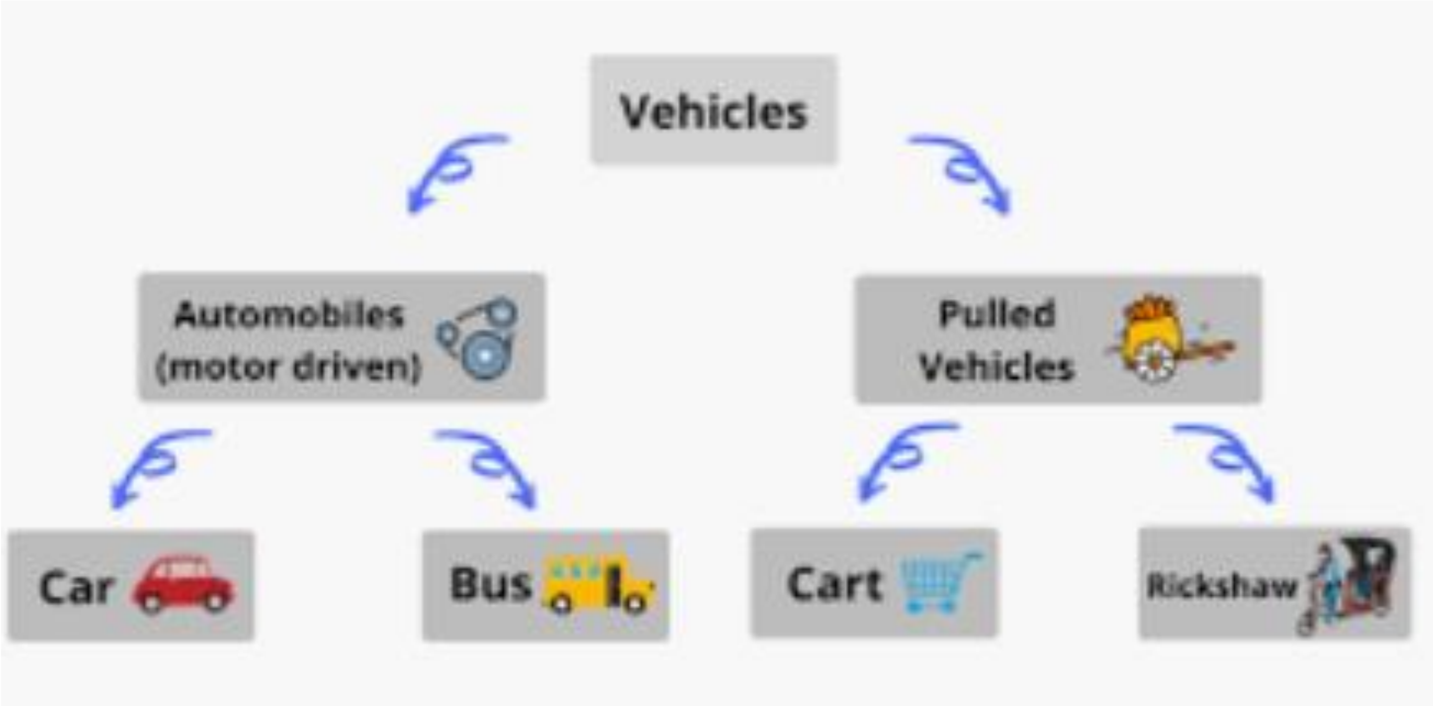
Abstraction

- Abstraction refers to the act of representing essential features without including the back ground details or explanations.
- Since class use the concept of data abstraction, it is called as Abstract Data Type(ADT).



Inheritance

- Inheritance is the process by which objects of one class acquire the properties of another class.
- inheritance provides the idea of reusability.
- This mean that we can add additional features to an existing class with out modifying it.



Polymorphism

- Polymorphism means the ability to take more than one form.
- An operation may exhibit different instance.
- The behaviour depends upon the type of data used in the operation.

In Shopping malls behave like Customer

In Bus behave like Passenger

In School behave like Student

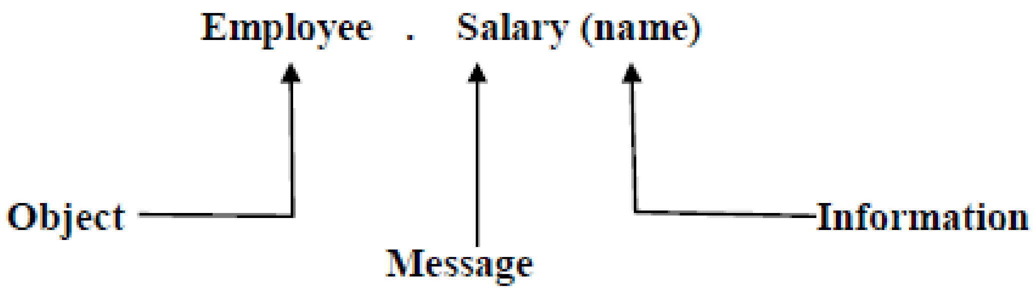
At Home behave like Son

Dynamic Binding

- Binding refers to the linking of a procedure call to the code to the executed in response to the call.
- Dynamic binding means the code associated with a given procedure is unknown until the time of execution.
- Runtime only it will get the code

Message Passing

- An object oriented program consists of a set of objects that communicate with each other.
- A message for an object is a request for execution of a procedure and therefore will invoke a function (procedure) in the receiving object that generates the desired result.
- Message passing involves specifying the name of the object, the name of the function (message) and information to be sent.



Employee . Salary (name)

Object

Message

Information

- The variables declared inside the class are known as data members and functions are known as member functions
- Process of creating an object is called instantiation.
- Object is an instance of a class.

Structure of C++ program with classes

include header files

class declaration

member function definition

main pgm

Eg:

Regno, name, Sem, M1, M2, M3

Display

Average

```
#include<iostream>
using namespace std;
class classname
{
variable declaration;
function declaration;
};
function definition()
{
}
int main()
{
classname obj;
obj.fun();
}
```

- Is the class definition compulsory in C++ pgm?

Class and objects

class is a new data type that contains data members and member functions that operates on the data members.

Generally a class specification has two parts:-

- **Class declaration**
- **Class function definition**

Syntax:-

```
class class-name
{
private:
      variable declarations;
      function declaration ;
public:
      variable declarations;
      function declaration;
};
```

Access specifiers/Visibility Labels

- private
- public
- protected

private

- Class members declared as private can be accessed only within the class. ie other class members can not access it.
- Cant access from main also.
- By default, class members are private
- Data hiding is possible using private declaration.

public

- Can be accessed from outside class
- Can access from main
- So usually we will declare the data members as private and member functions as public
- So that we can access public member functions from main and through that public member functions we can access the private data members.

protected

- Will discuss during inheritance

```
class item
{
      int member;
      float cost;
public:
      void getdata (int a ,float b);
      void putdata (void);
};
```

Creating objects

```
class item
{
-----------
-----------
-----------
}x ,y ,z;
```

would create the objects x ,y ,z of type item.

```
class item
{
-----------
-----------
-----------
};
main()
{
item x, y, z;
}
```

would create the objects x ,y ,z of type item.

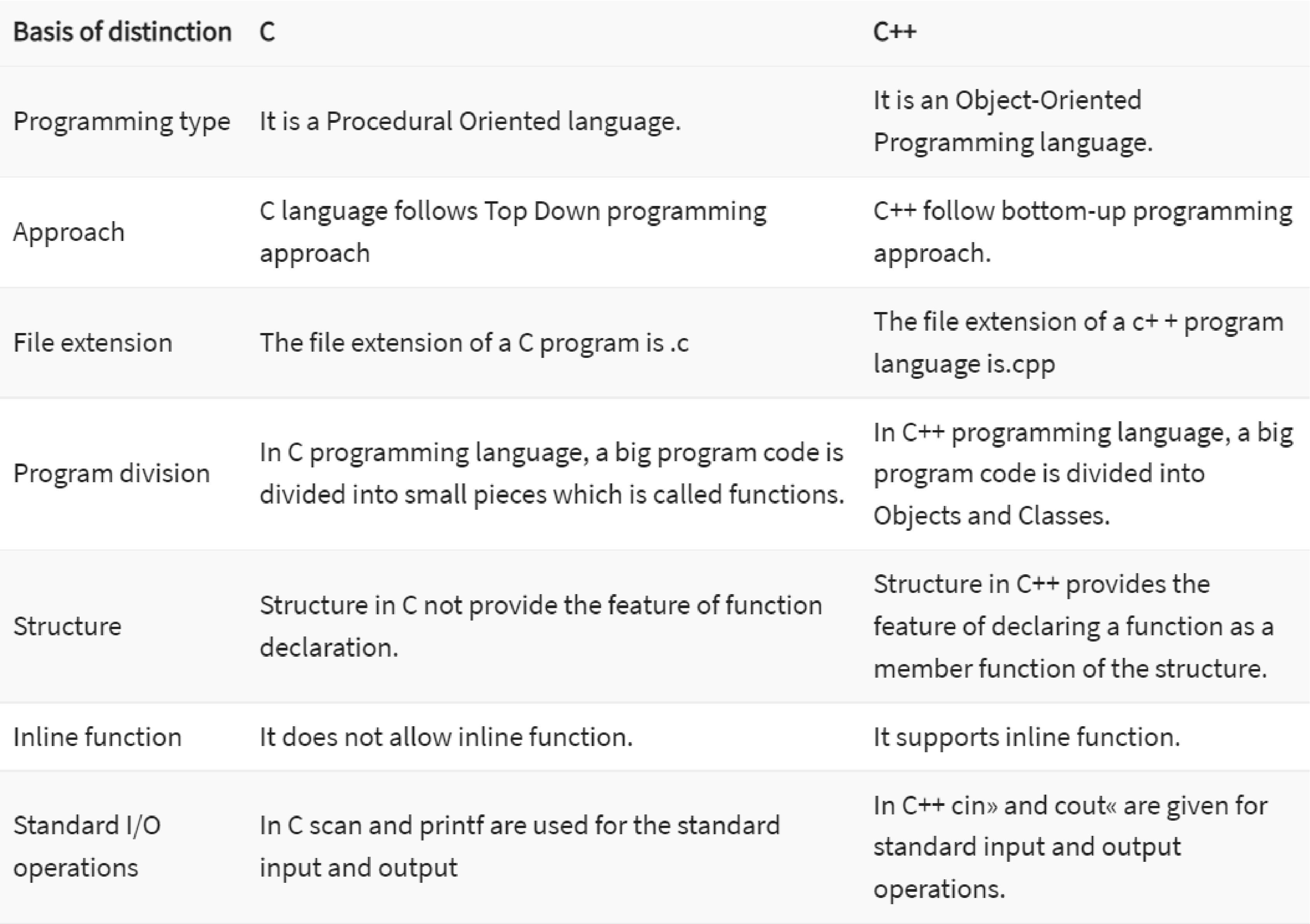
Accessing class members

```
class xyz
{
        int x;
        int y;
  public:
     int z;
     void disp();
};
main()
{
        xyz p;
        p. x =0;     //error . x is private
        p. z=10;         //ok ,z is public
        p.disp();                //can access
}
```

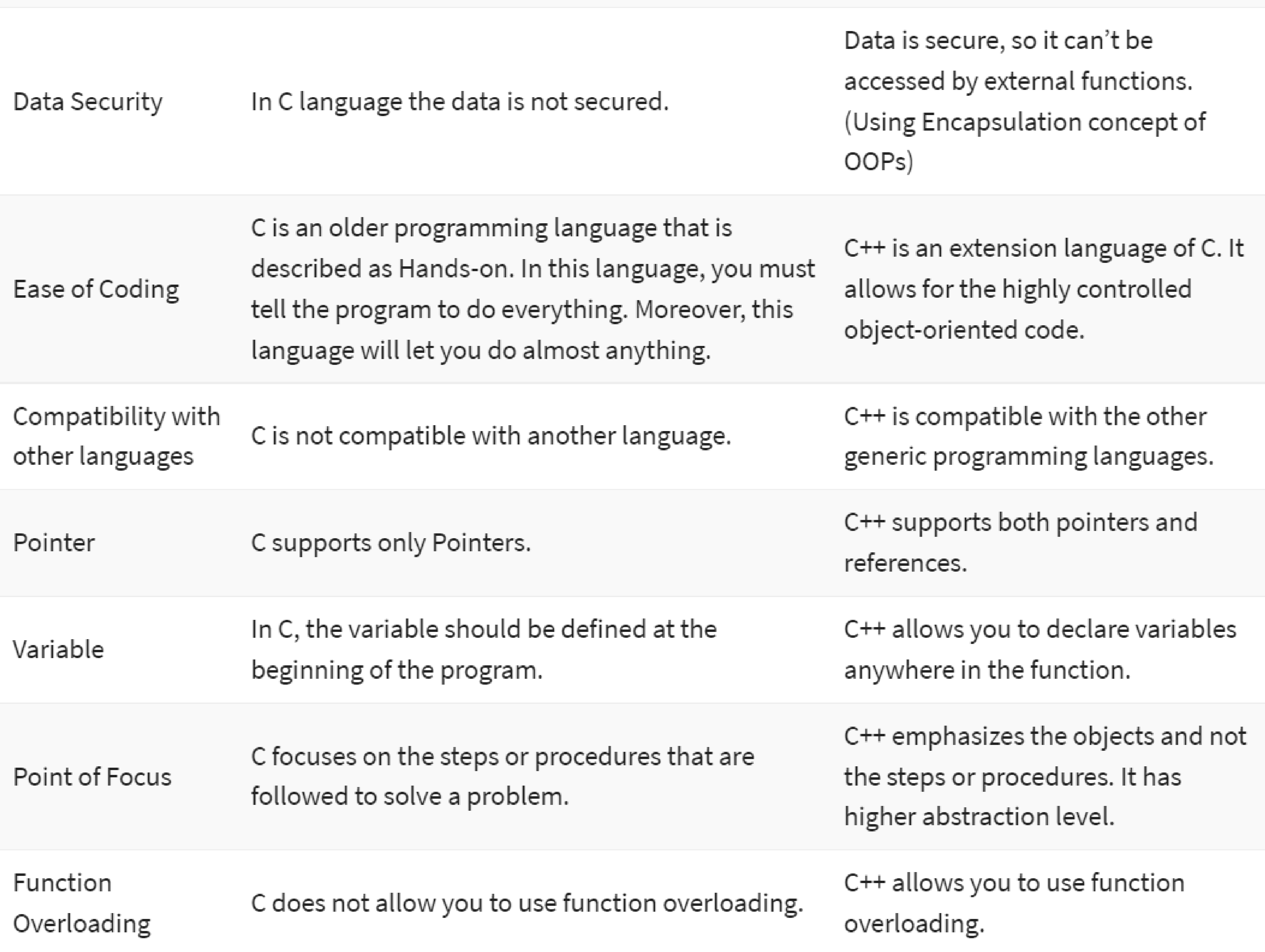
Class Point ?

- Data members?
- Functions?

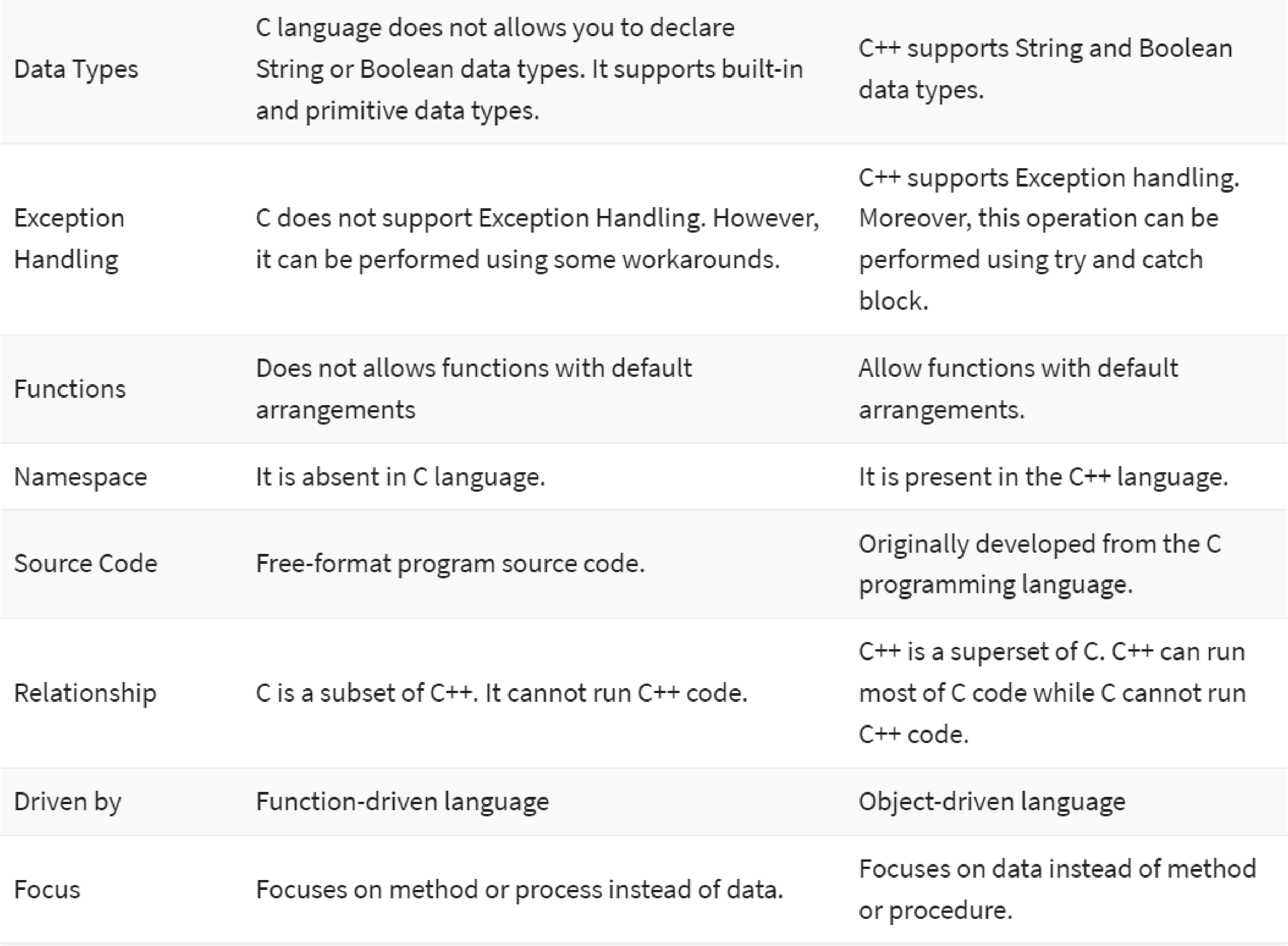
- List all the differences between C and C++



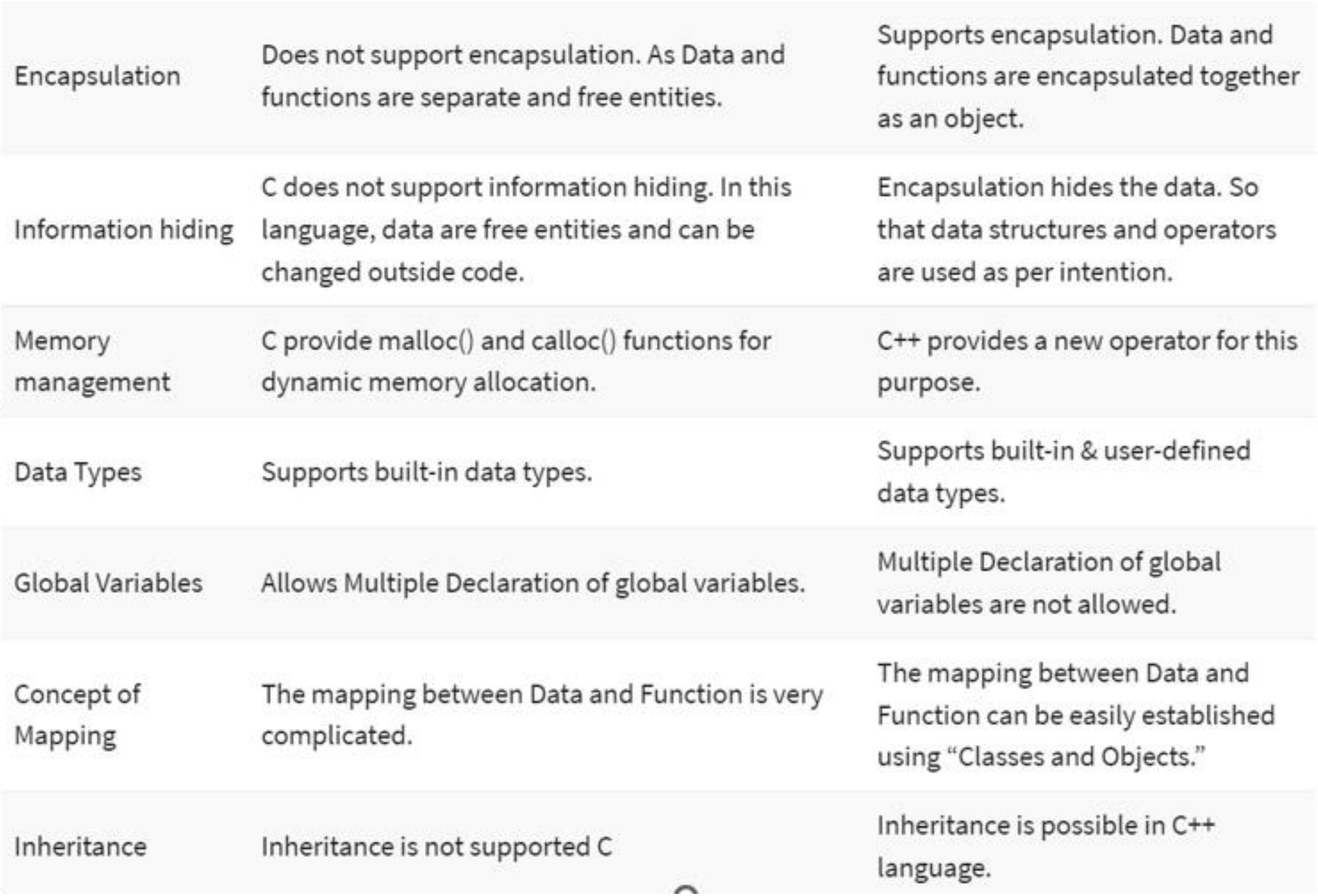
Basis of distinction	C	C++
Programming type	It is a Procedural Oriented language.	It is an Object-Oriented Programming language.
Approach	C language follows Top Down programming approach	C++ follow bottom-up programming approach.
File extension	The file extension of a C program is .c	The file extension of a c+ + program language is.cpp
Program division	In C programming language, a big program code is divided into small pieces which is called functions.	In C++ programming language, a big program code is divided into Objects and Classes.
Structure	Structure in C not provide the feature of function declaration.	Structure in C++ provides the feature of declaring a function as a member function of the structure.
Inline function	It does not allow inline function.	It supports inline function.
Standard I/O operations	In C scan and printf are used for the standard input and output	In C++ cin» and cout« are given for standard input and output operations.



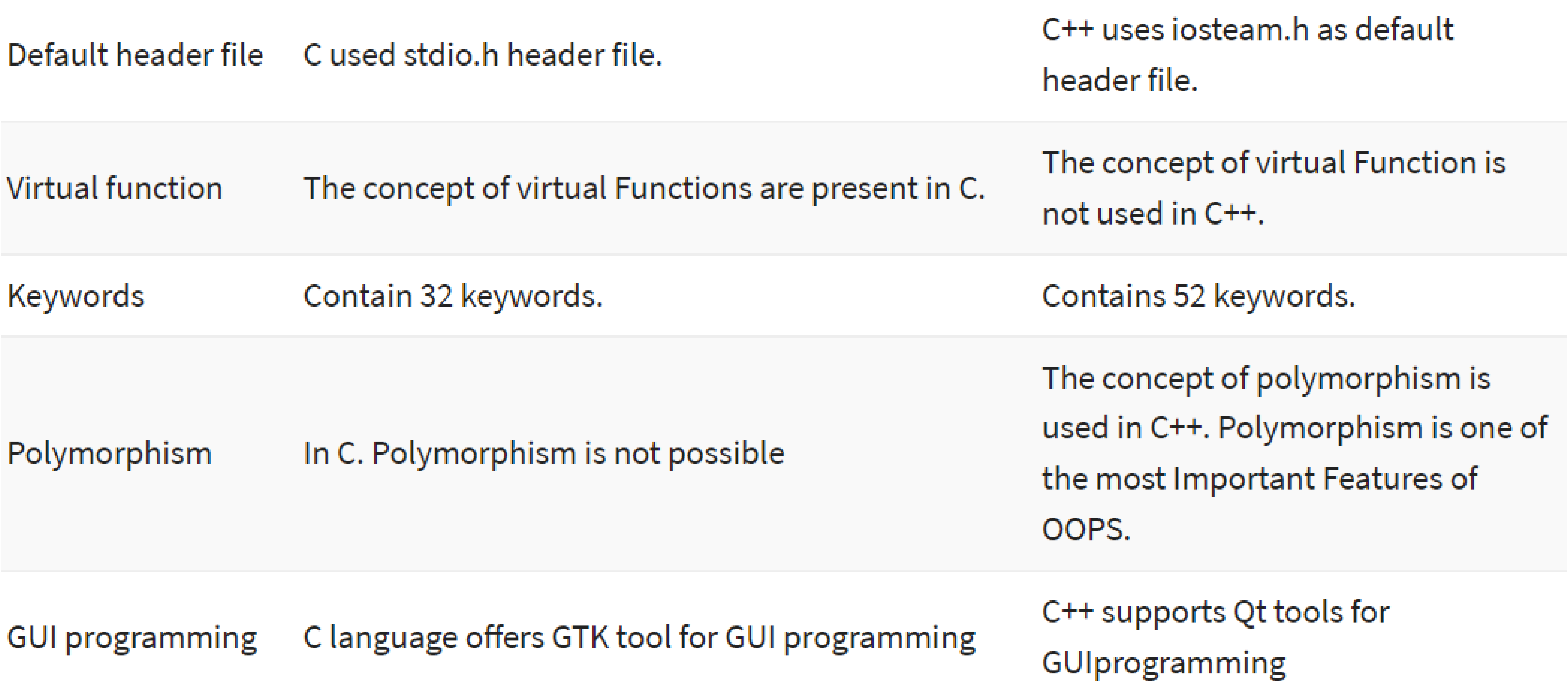
Data Security	In C language the data is not secured.	Data is secure, so it can’t be accessed by external functions. (Using Encapsulation concept of OOPs)
Ease of Coding	C is an older programming language that is described as Hands-on. In this language, you must tell the program to do everything. Moreover, this language will let you do almost anything.	C++ is an extension language of C. It allows for the highly controlled object-oriented code.
Compatibility with other languages	C is not compatible with another language.	C++ is compatible with the other generic programming languages.
Pointer	C supports only Pointers.	C++ supports both pointers and references.
Variable	In C, the variable should be defined at the beginning of the program.	C++ allows you to declare variables anywhere in the function.
Point of Focus	C focuses on the steps or procedures that are followed to solve a problem.	C++ emphasizes the objects and not the steps or procedures. It has higher abstraction level.
Function Overloading	C does not allow you to use function overloading.	C++ allows you to use function overloading.



Data Types	C language does not allows you to declare String or Boolean data types. It supports built-in and primitive data types.	C++ supports String and Boolean data types.
Exception Handling	C does not support Exception Handling. However, it can be performed using some workarounds.	C++ supports Exception handling. Moreover, this operation can be performed using try and catch block.
Functions	Does not allows functions with default arrangements	Allow functions with default arrangements.
Namespace	It is absent in C language.	It is present in the C++ language.
Source Code	Free-format program source code.	Originally developed from the C programming language.
Relationship	C is a subset of C++. It cannot run C++ code.	C++ is a superset of C. C++ can run most of C code while C cannot run C++ code.
Driven by	Function-driven language	Object-driven language
Focus	Focuses on method or process instead of data.	Focuses on data instead of method or procedure.



Encapsulation	Does not support encapsulation. As Data and functions are separate and free entities.	Supports encapsulation. Data and functions are encapsulated together as an object.
Information hiding	C does not support information hiding. In this language, data are free entities and can be changed outside code.	Encapsulation hides the data. So that data structures and operators are used as per intention.
Memory management	C provide malloc() and calloc() functions for dynamic memory allocation.	C++ provides a new operator for this purpose.
Data Types	Supports built-in data types.	Supports built-in & user-defined data types.
Global Variables	Allows Multiple Declaration of global variables.	Multiple Declaration of global variables are not allowed.
Concept of Mapping	The mapping between Data and Function is very complicated.	The mapping between Data and Function can be easily established using "Classes and Objects."
Inheritance	Inheritance is not supported C	Inheritance is possible in C++ language.



Default header file	C used stdio.h header file.	C++ uses iosteam.h as default header file.
Virtual function	The concept of virtual Functions are present in C.	The concept of virtual Function is not used in C++.
Keywords	Contain 32 keywords.	Contains 52 keywords.
Polymorphism	In C. Polymorphism is not possible	The concept of polymorphism is used in C++. Polymorphism is one of the most Important Features of OOPS.
GUI programming	C language offers GTK tool for GUI programming	C++ supports Qt tools for GUIprogramming