Introduction to C++

==>C++ was invented by Bjarne Stroustrup in 1979 at Bell lab.

initially he called this lang as, c with classes,later in 1983 the name was changed to c++.

==>C++ is a general purpose/multipurpose,high level,compiler based and object oriented programming language.

==>This programming language requires a compiler or interpreter to translate the program into machine language (execute the program).

Example: Java, C#, C,C++, etc.

In c++ we are using compiler as a translator,thats why c++ is called as compiler based programming lang.

A high-level language is further divided into two parts -

a)Procedural Oriented programming language

b)Object-Oriented Programming language

FAQ:

which s/w application we can create using c++?

==>using c++ , we can design operating system(dos,window,unix)

==>using c++,we can design editors(notepad,wordpad,editplus)

==>using c++,we can design commercial application(related to money like (bank program,hotel managment program,supermarket program)

In supermarket,bill should be generated,billing program is designed using c++.Bank transactions application is designed using c++.

==>using c++,we are also able to design database(oracle is designed by c and c++)

==>using c++,we can also design translators(compiler,interpreter)

==>using c++,we can design PC and mobile games. ex:snake game,mario etc.

=========================================================

Function oriented programming lang / Procedural Language:

==>Total program is divided into small programs called as functions. Function is a small program which is ued to do a particular job.Entire C program is collection of functions thats why c++ is called as function oriented programming lang.

Advantages are:

Memory requirement is less

modularity

reusability(write once use many times)

=============================================================================

PROBLEM WITH FUNCTION

==>procedural lang means globally all data stored at one place i.e all data accessible to all function.

==>when program is big and when data is global then to identify which variable is accesed by which function is

very difficult. Any function can access it . so data is not secured because data is freely moving around the function.

==>In c programming lang, data and function is seperately stored and global data is accessible to every function.

==>it is difficult to map real world entities into system design.it is difficult to bind data to the functionality of the system because its functionality is considered seperately

==>debugging a structured program is difficult and time consuming process.since data is shared over the entire program,different functions may be accessing it.to exactly find which function is causing the error is not easy.

====================================================================================

Object oriented programming features

==>class is an extension of C structure

==>structure is collection of heterogenous datatype. structure allows us to store different types of variables at one place under one name.due to structure ,it is very easy to construct object oriented programming.

In oop will going to use class but that class derived/extension of c structure.

==>we can able to declared both primitive and derived datatype at one place using structure.

ex:

I want to store student related data at one place ..

struct stu

{

int id;primitaive

int marks[6];//derived;

};

structure is constructed based on the user requirement thats why structure is called as user defined datatyes...

struct stu

{

int id;primitaive

int marks[6];//derived;

};

int main()

{

struct stu s1,s2;//structure variable stored on stack.

}

here s1 stores 1 student data which is called as 1 object in C++ ,thats why it is called as object oriented .

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*disadvantage with structure\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

==>In c lang, structure data is by default public.structure can be accessed from anywhere within the program.

now to avoid this they have introduced class.

In c++ also, we are having structure.

FAQ:

What is the difference between structure in C and structure in C++?

In C ,inside the structure we can declare only variables(member structure) . while in C++,we can declare member structure +member function.

=============================================================================================

In c++ also structure data is public.that means anyone can access data from anywhere,not secured.

that why they have introduced class in c++.

==========================================

Difference between Structure in C++ and Class in C++

1)In c++ structure, by default all the members and structure data is public, whereas class data is private.

2)using c++ structure ,we cannot define complex programs.But it is possible with the class concepts.

3)class uses the concept of inheritance but structure in c++ never uses the concept of inheritance

====================================================================================

class stu

{

private/protected/public:

datamembers;

memberfunctions;

};

Ex:

In a class, several students are there,every student is having its own data.one student data doesnt belong to another student data.

classname objectname;

stu1 obj;

id ,name,fees----->attributes

To access this variables we have to use function,this function is able to access only student1 data

stu2 obj

id ,name,fees----->attributes,to access this variables we have to use function,this function is able to access only student2 data

i.e data is seperated into no of blocks called object called as object1,object2.

That means object1 contains first student data,object2 contains 2nd student data.

total program is divided into several objects. thats why c++ is called as object oriented programming structure.

In oop's they bind variable and function in single unit called as object. when it make private it is accessible only within that funtion.

===================================================================================

Example:

class student

{

int id;//data members

public:

void get()

{

cout<<id;

}

};

get() is the member of the class thats why it is called as member function.

with the help of member function of a class we can access private data of a class.this data cannot be accessible from outside the class called as data hiding. data hiding mostly achieve using private access specifier.

data hiding means insulating the data from external access. outsiders cannot access the data.only member function can access the data.

\*\*class is a collection of data members and member function both are associated together into a single unit called as class is called encapsulation

encapsulation is a process of binding the data and member function together into a single unit called class.

\*\*class provides both data hiding and encapsulation

=============================================================================================

OBJECT

==>In structure,structure variable should be created to access the structure members.

==>In class,class variables are called as object should be created to access the class members.

class stu

{

};

stu s;//object (memory allocation takes place)

==> object s is created from this class stu, that why class is called as blueprint(original copy) to construct the object(xerox copy). thats why to create object we need class,thats why class is a blueprint and object is the instance of the class.

==>object is the physical representation of a class.when object is created then only memory allocation takes place. class never allocates memory i.e logical representation.

==================================================

Difference between C structure and C ++ structure

1)collection of variables collection of variables and function

2)structure members are public structure members can be private,protected,public

----------------------------------------------------------------------------------------------------------------------------------------

similarities between C++ structure and class

1)userdefined userdefined

2)collection of variables & functions collection of variables & functions

3)members are private,protected,public members are private,protected,public

-------------------------------------------------------------------------------------------------------------------------------------------

Differences between C++ structure and class

1)by default members are public by default members are private

2) structure cannot be inherited class can be inherited

3)used to build simple program uses to build complex program,big project

============================================================================

class:

1)it is a userdefined datatypes because we r constructing a class based on primitive and derived datatype as per user requriment.

2)collection of datamembers and member functions

3)it is a container because it contains members.

4)class provides the concept of encapsulation(binding of data memebrs and member function in a single unit),and also provides the concept of data hiding with private access specifier.

5)class is a blueprint to constructs the objects i.e it is a plan before constructing objectts

paper plan(class)---------> real building(object)

6)class is a logical copy to create objects thats why class never occupies memory.

syntax:

class classname//class is keyword

{

access specifier://it specifies how it is going to access in our program

datamember;

member function;

};

Note:

Whenever datamembers and member functions are private, they cannot be accessed directly using objects.

#include<iostream>

using namespace std;

class A

{

int a;

void get()

{

a=100;

cout<<a;//allowed

}//private data can be accessed within member function of same class

};

int main()

{

A obj;

//obj.a=100;//not allowed whenever datamembers and member functions are private cannot be accessed directly using objects.

//obj.get();//not allowed whenever datamembers and member functions are private cannot be accessed directly using objects.

}

======================================================================

#include<iostream>

using namespace std;

class A

{

int a;

public:

void get()

{

a=100;

cout<<a;//allowed

}

};

int main()

{

A obj;

obj.get();

}

Therefore it is better to use one public member

=======================================================================================

OBJECT

To access class members,we should define objects because when class is declared memory is not allocated,when object is defined then only memory is allocated.

ex: int a,b;

int is logical copy occupies no memory

a,b is physical copy ,memory gets allocated

ex:student stu;

stu.get();

stu.put();

thru object we can access class members...

Object is :

1) a variable of type class.

2)an instance/xerox copy of a class

3)it is physical representation of a class.

==========================================================================

Difference between procedural programming and object-oriented programming

1)In procedural programming, the program is divided into small parts called functions.

In object-oriented programming, the program is divided into small parts called objects.

2)It is less secure than OOPs.

Data hiding is possible in object-oriented programming . So, it is more secure than procedural programming.

3)There is no access specifier in procedural programming.

Object-oriented programming has access specifiers like private, public, protected, etc.

4)In procedural programming, data moves freely within the system from one function to another.

In OOP, objects can move and communicate with each other via member functions.

5)There is no code reusability present in procedural programming.

It offers code reusability by using the feature of inheritance.

===============================================================================================

C++ Features

Popular

C++ can be the base language for many other programming languages that supports the feature of object-oriented programming.

Compiler based

C++ is a compiler-based programming language, which means no C++ program can be executed without compilation. C++ compiler is easily available, and it requires very little space for storage. First, we need to compile our program using a compiler, and then we can execute our program.

Platform Dependent ?

The code which can only be run on same platform on which it was compiled. Compiled code cannot run on any other platforms.such language is called as platform dependent .

ex: C,C++

Here, same machine code which is compiled on windows,cannot be run on Linux or MAC OS.

Rich Library

C++ provides a lot of inbuilt functions that make the development fast. Following are the libraries used in C++ programming are:

<iostream>

<cmath>

<cstdlib>

<fstream>+

Memory Management

C++ provides very efficient management techniques. The various memory management operators help save the memory and improve the program's efficiency. These operators allocate and deallocate memory at run time. Some common memory management operators available C++ are new, delete etc.

Object-Oriented

In C++, object-oriented concepts like data hiding, encapsulation, and data abstraction can easily be implemented using keyword class, private, public, and protected access specifiers. Object-oriented makes development and maintenance easier.

====================================================================

Difference between C and C++

C was developed by Dennis Ritchie between the year 1969 and 1973 at AT&T Bell Labs.

C++ was developed by Bjarne Stroustrup in 1979.

C does no support polymorphism, encapsulation, and inheritance which means that C does not support object oriented programming.

C++ supports polymorphism, encapsulation, and inheritance because it is an object oriented programming language.

C supports procedural programming.

C++ supports object oriented programming paradigms.

Data and functions are separated in C because it is a procedural programming language.

Data and functions are encapsulated together in form of an object in C++.

C does not support data hiding so not secured.

C++ supports data hiding with the help of private access specifier,secured.

Functions in C are not defined inside structures.

Functions can be used inside a structure in C++.

C does not support inheritance.

C++ supports inheritance.

Instead of focusing on data, C focuses on functions.

C++ focuses on data instead of focusing on method or procedure.

C provides malloc() and calloc() functions for dynamic memory allocation, and free() for memory de-allocation.

C++ provides new operator for memory allocation and delete operator for memory de-allocation.

Direct support for exception handling is not supported by C.

Exception handling is supported by C++.

scanf() and printf() functions are used for input/output in C.

cin and cout are used for input/output in C++.

C structures don’t have access modifiers.

C ++ structures have access modifiers.

C does not support reference variables.

C++ supports reference variables.

========================================================================

Types of Variables

There are three types of variables based on the scope of variables in C++

Local Variables

Instance Variables

Static Variables

local variable

==>the variable declared within the parenthesis of any function or any block is

called as local variable.

==>These variables are created when entered into the block or the function is called and destroyed after exiting from the block or when the call returns from the function.

==>The scope of these variables exists only within the block in which the variable is declared. i.e. we can access this variable only within that block.

==>scope and lifetime is within the block/within the function

==>storage is stack section

lifetime means how long the variable exists in the memory

scope means how long the variable or value of that variable can be accessed

in local variable lifetime depends on scope of variable

==============================================================

global variable

==>variable declared outside the function/block is called as global variable

==>initial value is 0

storage is data section

==>lifetime-->thru out the program

==>scope-->thru out the program

Instance variable

1)whenever an instance is created,the variable which will allocate memory is called as instance variable

2)memory for instance variable is alloacted per object

3)changes made to instance variable of one object will not be reflected into another object's instance variable.

=====================================================================================

Boolean: Boolean data type is used for storing Boolean or logical values. A Boolean variable can store either true or false. The keyword used for the Boolean data type is bool.

// C Program to implement Boolean data type

#include<iostream>

using namespace std;

int main()

{

// Boolean data types declared

bool a = true;

bool b = false;

printf("True : %d\n", a);

printf("False : %d\n", b);

cout<<a;

cout<<b;

}

===========================================================================================

Header files available in C++ for Input/Output operations are:

iostream: iostream stands for standard input-output stream. This header file contains definitions of objects like cin, cout, cerr, etc.

Standard output stream (cout): Usually the standard output device is the display screen. The C++ cout statement is the instance of the ostream class. It is used to produce output on the standard output device which is usually the display screen. The data needed to be displayed on the screen is inserted in the standard output stream (cout) using the insertion operator(<<).

==>ostream class availabe in iostream.h thats why we have to include this headerfile

standard input stream (cin): Usually the input device in a computer is the keyboard. C++ cin statement is the instance of the class istream and is used to read input from the standard input device which is usually a keyboard.

The extraction operator(>>) is used along with the object cin for reading inputs. The extraction operator extracts the data from the object cin which is entered using the keyboard.

Constants:

const keywords is used to define the constant value that cannot change during program execution. It means once we declare a variable as the constant in a program, the variable's value will be fixed and never be changed. If we try to change the value of the const type variable, it shows an error message in the program.

1. Const variable

It is a const variable used to define the variable values that never be changed during the execution of a program. And if we try to modify the value, it throws an error.

Syntax

const data\_type variable\_name;

Example: Program to use the const keyword in C++

Let's create a program to demonstrate the use of the const keyword in the C++ programming language.

#include <iostream>

using namespace std;

int main ()

{

// declare the value of the const

const int num = 25;

num = num + 10;

return 0;

}

Note:Constants cannot appear on LHS of assignment operator(=)

========================================================================

1) Constant pointer to an integer

#include <iostream>

using namespace std;

int main ()

{

int x = 10, y = 20;

int\* const ptr = &x; //const integer ptr variable point address to the variable x

//ptr = &y;

\*ptr = 15;

cout << " The value of x: " << x << endl;

cout << " The value of ptr: " << \*ptr << endl;

}

Output

The value of x: 15

The value of ptr: 15

In the above program, pointer ptr pointing to the address of int variable 'x' and ptr variable cannot be changed their address once it initialized, but the pointer ptr can change the value of x.

2. Pointer to a constant integer

#include <iostream>

using namespace std;

int main ()

{

int x = 10, y = 20;

const int\* ptr = &x;

ptr = &y;

cout << ptr<< endl;

ptr=&x;

cout << ptr<< endl;

// \*ptr = 15;

cout << " The value of ptr: " << \*ptr << endl;

}

3)Constant pointer to a Constant Integer

#include <iostream>

using namespace std;

int main ()

{

int x = 10, y = 20;

const int\* const ptr = &x;

//ptr = &y;

cout << ptr<< endl;

//ptr=&x;

cout << ptr<< endl;

// \*ptr = 15;

cout << " The value of ptr: " << \*ptr << endl;

}

=================================================

Reference variable..

==>A reference variable is an alias, that is, another name for an already existing variable.

==>In c++ we have pass by reference. but why?

==>In call by address,we have to declared formal parameter as a pointer and pointer holds address.

problem is we can initialize any variable address in future.First variable address is passed,later we can change address inside pointer. Means we can initialize another variable address into this pointer, due to this which variable address it is going to store, makes confusion. That’s why pointer is not secured.

==>To overcome the above limitations, reference datatype is introduced.

2)A variable can be declared as a reference by putting '&' in the declaration.

syntax:

data\_type &refname = existing variable;

ex:

int num=10;

int &refnum=num;

==>num is an integer variable, located at some address which is initialized to a value 10.

==>refnum is a another name for variable num, therefore refnum is a reference to num.

3)We can access the contents of the variable through either the original variable name or the reference name.

ex:

int i = 17;

int &r = i;

r is the reference name for the original variable name i.

#include<iostream>

using namespace std;

int main()

{

int num=10;

int &refnum=num;

cout<<num<<" "<<refnum<<endl;

num=100;

cout<<num<<" "<<refnum<<endl;

cout<<&num<<" "<<&refnum<<endl;

}

4)No seperate memory is allocated for a reference.Its just an alternative name given to the existing variable.

5)You cannot have NULL reference. Reference should always be initialized with some variable.

reference must be initialized at the time of the declaration.

int &refnum;

&refnum=num;//not allowed

The above code will throw a compile-time error as refnum is not assigned at the time of declaration.

6)Once a reference is initialized to an object, it cannot be refer to another object i.e it is secured.

#include <iostream>

using namespace std;

int main()

{

int x=11; // variable initialization

int z=67;

int &y=x; // y reference to x

int &y=z; // y reference to z, but throws a compile-time error.

return 0;

}

==>reference is a rigid connection.

7)Reference can pass as a parameter to function.

==>References can also be passed as a function parameter. It does not create a copy of the argument and behaves as an alias for a parameter. It enhances the performance as it does not create a copy of the argument.

Pass by reference will reduce the complexity generated by pointers in pass by address.

#include <iostream>

using namespace std;

void swap(int&, int&);

int main()

{

int a,b;

cout<<"enter values of a & b\n"<<endl;

cin>>a>>b;

swap(a,b);

cout<<"after swapping "<<endl;

cout<<"values of a & b is"<<a<<b<<endl;

}

void swap(int &p,int &q)

{

int temp;

temp=p;

p=q;

q=temp;

}

==>In c++,reference is also used to share data between functions i.e pass by reference.

ex:

void main()

{

int a=100;

int &b=a;

int &c=b;

cout<<a<<b<<c;

c=200;

cout<<a<<b<<c;

}

=====================================================================================================

Difference between pointer and reference;

1)pointer is a flexible connection. it can point to any variable at a given point of a program.

Reference is a rigid connection,reference once created,cannot refer to any other object.

2)pointer has to be dereferenced using \* operator to access the value at the address pointed by it.

A reference has direct connection as it is another name for the same memory location.

3)Reference cannot be null whereas pointer can point to null to indicate that they are not pointing to any valid thing. .

FAQ:

#include <iostream>

using namespace std;

int main()

{

int\* ptr = NULL;

int& ref = \*ptr;

cout << ref << '\n';

}

//ERROR

===============================================================================

STATIC VARIABLE

==>sometimes only one copy of a variable is required between all the objects of the same class,rather than having each object maintaining its own copy.in such case,that data members is declared as static.only one copy of static variable is created on data section and that copy is shared between all the object of the same class.Therefore static variable is called as class variable.

==>static data members are not part of the object,they are stored seperately.

==>Variable declared with static keyword are called as static variable.

==> Static Variable stored on data section

==>changes to the static variable affects all the instance of the same class.

==>It is initialized before any object of this class is created, even before the main starts.

==>Static member function are only declared in a class declaration, not defined. They must be explicitly defined outside the class using the scope resolution operator.

Static variables in a Function: When a variable is declared as static, space for it gets allocated for the lifetime of the program. Even if the function is called multiple times, space for the static variable is allocated only once and the value of the variable in the previous call gets carried through the next function call. This is useful for i application where the previous state of function needs to be stored.

#include <iostream>

using namespace std;

void demo()

{

static int count = 0;

cout << count << " ";

count++;

}

int main()

{

for (int i = 0; i < 5; i++)

demo();

return 0;

}

==>static variable is used to count the number of object created.

STATIC FUNCTION

1)function declare with static keyword is called as static function

2)static function is invoked by class name

3)syntax:

classname::static function name();

Memory efficient: Now we don't need to create instance for accessing the static members, so it saves memory.

4)This pointer is not passed to the static function.therefore static mem fun cannot access non static data member.

5)static function access only static data but a non static function can access both static as well as non static

Accessing static data member

i)if the static data member is public in the class then it can be accessed outside the class using classname::staticmembers;

2)if it is a private variable then it can be accessed either through non static member function and static member function.

=============================================================================

The reason we need Static member :

1)Static members are used to store information that is shared by all objects in a class.

2)keep track of the quantity of newly generated objects of a specific class type using a static data member as a counter.

==================================================================================

#include<iostream>

using namespace std;

class Complex

{

int real;

int img;

static int x;

public:

void accept();

void setReal(int);

int getImg()const;

void show()const

{

cout<<"complex number is "<<real<<"+"<<img<<"i"<<endl;

}

Complex(int r,int i)

{

real=r;

img=i;

}

};

void Complex::accept()

{

cout<<"enter real and img\n";

cin>>real>>img;

}

void Complex::setReal(int r )

{

real=r;

}

int Complex::getImg()const

{

return img;

}

int main()

{

const Complex d1(1,2);

//d1.accept();//function calling

d1.show();

//d1.setReal(10);//setter function

d1.show();

cout<<"imaginary part is "<<d1.getImg();

cout<<"size of object is "<<sizeof(d1);

}

===========================================================

#include<iostream>

using namespace std;

class Complex

{

int real;

int img;

static int cnt;

public:

void accept();

void setReal(int);

int getImg();

void show()

{

cout<<"complex number is "<<real<<"+"<<img<<"i"<<" "<<cnt<<endl;

}

static int getCnt()

{

cnt=5;

return cnt;

//return real;

}

};

//datatype classname::static variable name;

int Complex::cnt;//memory allocation takes place

void Complex::accept()

{

cout<<"enter real and img\n";

cin>>real>>img;

}

void Complex::setReal(int r )

{

real=r;

}

int Complex::getImg()

{

return img;

}

int main()

{

Complex c1;

c1.accept();

c1.show();

// cout<<"value is "<<Complex::getCnt();

cout<<"value is "<<c1.getCnt();

}

==========================================================

Constructor:

Sequence after creation of object

1)memory allocated by the compiler

2)constructor invocation by compiler implicitly

3)memory initialization by constructor

==>Constructor in C++ is a special method that is invoked automatically at the time of object creation. It is used to initialize the data members of newly created objects.

==>The constructor in C++ has the same name as the class or structure. It constructs the values i.e. provides data for the object which is why it is known as constructor.

==>Constructor do not return value, hence they do not have a return type not even void.

==>Constructor cannot be constant member function.

The prototype of the constructor looks like

<class-name> (list-of-parameters);

Constructor can be defined inside the class declaration or outside the class declaration

a. Syntax for defining the constructor within the class

<class-name>(list-of-parameters)

{

//constructor definition

}

b. Syntax for defining the constructor outside the class

<class-name>: :<class-name>(list-of-parameters)

{

//constructor definition

}

Types of Constructors

Default Constructors:

==>The constructor which doesn’t take any argument. It has no parameters. It is also called as no argument constructor.

==>If constructor is not defined in the source code by the programmer, then the compiler defined the default constructor implicitly during compilation and initilizes data members of class to garbage values.

==>There can be only one no argument constructor per class.

#include <iostream>

using namespace std;

class Complex

{

int real,img;

public:

void show();

Complex();

Complex(int,int);

};

void Complex ::show()

{

cout<<"complex no is "<<real<<"+"<<img<<"i"<<endl;

}

Complex::Complex()

{

cout<<"in default constructor\n";

//real=10;

//img=20;

cout<<"enter value of real\n";

cin>>real;

cout<<"enter value of img\n";

cin>>img;

}

Complex::Complex(int r,int i)

{

cout<<"in parametrized constructor\n";

real=r;

img=i;

}

int main()

{

Complex c1,c2;

c1.show();

c2.show();

Complex c3(5,6);

//c3.real=50; allowed when real is publicly declared

c3.show();

}

#include <iostream>

using namespace std;

class Complex

{

int real,img;

static int cnt;

public:

void show();

Complex();

Complex(int,int);

static int getCnt();

};

int Complex::cnt; //cnt=100;

int Complex::getCnt()

{

return cnt;

}

void Complex ::show()

{

cout<<"complex no is "<<real<<"+"<<img<<"i"<<" "<<cnt<<endl;

}

Complex::Complex()

{

cout<<"in default constructor\n";

real=10;

img=20;

cnt++;

}

Complex::Complex(int r,int i)

{

cout<<"in parametrized constructor\n";

real=r;

img=i;

cnt++;

}

int main()

{

Complex c1,c2;

Complex c3(5,6);

cout<<"no of obj created is "<<Complex::getCnt();

}

===================================================================================

#include<iostream>

using namespace std;

class item

{

public:

static int x;

int number;

public:

void getdata(int a)

{

number=a;

number++;

x++;

}

void getcount()

{

x=10;

cout<<"value is "<<x<<endl;

}

void getcount\_1()

{

x++;

cout<<"value is "<<number<<endl;

cout<<"value is "<<x<<endl;

}

};

int number=9;

int item::x;

int main()

{

item a;

a.getcount();

a.getdata(100);

a.getcount\_1();

cout<<number<<endl;

cout<<a.number;

}

==========================================================

Function Overloading

==>In C++, We can have more than one constructor in a class with same name, as long as each has a different list of signatures.This concept is known as Constructor Overloading

==>Two or more function having same function name but different signatures is called as function overloading.

==> In Function Overloading Function name should be the same and the arguments should be different.

==>The criteria to overload a constructor is to differ the number of arguments or the type of arguments. We dont consider return type as a criteria to overload a constructor.

Why Function Overloading?

Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the function such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you to understand the behavior of the function because its name differs.

The parameters should follow any one or more than one of the following conditions for Function overloading:

1)Parameters should have a different type

add(int a, int b)

add(double a, double b)

#include <iostream>

using namespace std;

void add(int a, int b)

{

cout << "sum = " << (a + b);

}

void add(double a, double b)

{

cout << endl << "sum = " << (a + b);

}

int main()

{

add(10, 2);

add(5.3, 6.2);

}

2)Parameters should have a different number

add(int a, int b)

add(int a, int b, int c)

#include <iostream>

using namespace std;

void add(int a, int b)

{

cout << "sum = " << (a + b);

}

void add(int a, int b, int c)

{

cout << endl << "sum = " << (a + b + c);

}

int main()

{

add(10, 2);

add(5, 6, 4);

}

3)Parameters should have a different sequence of parameters.

add(int a, double b)

add(double a, int b)

#include<iostream>

using namespace std;

void add(int a, double b)

{

cout<<"sum = "<<(a+b);

}

void add(double a, int b)

{

cout<<endl<<"sum = "<<(a+b);

}

int main()

{

add(10,2.5);

add(5.5,6);

}

------------------------------------

int add();

float add();

//not overloaded

How does Function Overloading work?

Exact match:- (Function name and Parameter)

If a not exact match is found:–

->Char are promoted to an int.

->Float is promoted to double

If no match is found:

->C++ tries to find a match through the standard conversion.

ELSE ERROR

=============================================================================================

This Keyword

Each object gets its own copy of data members and all objects share a single copy of member functions.

Then now question is that if only one copy of each member function exists and is used by multiple objects, how are the proper data members are accessed and updated?

The compiler supplies an implicit pointer along with the names of the functions as 'this'.

==>whenever an object invokes member function,the address of that object is passed implicitly to the function as a hidden parameter . this pointer holds the address of the current object that invokes the fuction.

==>It is a constant pointer. The type of this pointer for the object cdate d1 will be

cdate \* const this; //it is a constant pointer to cdate object.

==>This pointer is used to retrieve objects address. In c to find out variable address,we use & symbol.in c++,to know current object address, this pointer is used.

==>this pointer is use to distinguish our data members from local variables when both are declared with the same name.To identify datamembers this pointer is used.

==>every non static members of c++ is having one hidden parameter called as this. static members never contains this.

Let's see the example of this keyword in C++ that refers to the fields of current class.

#include<iostream>

using namespace std;

class test

{

int a,b;

public:

void show()

{

a=10;

b=20;

cout<<"obj add"<<this<<endl;//print current class obj adddress (1000)

cout<<"a" <<this->a<<endl;//cout<<"a" <<a<<endl;

cout<<"b"<<this->b<<endl;//cout<<"b"<<b<<endl;

}

};

int main()

{

test t;

cout<<&t<<endl;//1000(object's address) assume this=1000

t.show();//implicitly this pointer will be passed to show()

}

========================

#include<iostream>

using namespace std;

class test

{

int a,b;

public:

void show(int a,int b)

{

this->a=a;

this->b=b;

}

void display()

{

cout<<a<<b;

}

};

int main()

{

test t;

t.show(10,20);

t.display();

}

-----------------------------------------------------------------------------------------------------------------------

#include<iostream>

using namespace std;

class date

{

int dd,mm,yy;

public:

void show();

date(int,int,int);

date();

};

date::date()

{

this->dd=this->mm=this->yy=0;

}

date::date(int dd,int mm,int yy)

{

this->dd=dd;

this->mm=mm;

this->yy=yy;

}

void date::show()

{

cout<<" date is "<<this->dd<<"/"<<this->mm<<"/"<<yy<<endl;

}

int main()

{

date d1(1,1,1);

d1.show();

date d2(2,2,2);

d2.show();

}

================================================================================