C++ Programming Notes

Contributor: Preeti

[SPM (DU)]

Computer Science Notes

C++

Aeroline
Date: 24/07/17
Page No:

→ Programming Language = Language that is used by programmer to write set of instructions that a computer can understand

→ We create programme to control the behaviour of machine.

→ for low level code, compiler is not required.
— low level → Add X, 05
— Symbolic code.

Categories into 2 level:-

1. low level
2. High level

→ low level — It is closed to hardware programme and written in form of memory & register on computer.
eg — simply language. — Add X, 05
assembly language.
they can convert to machine code without a compiler or interpreter.

→ high level — It is closed to human language these are easier to read, write & maintain.
eg — C++, java, basic, variables, pointer, array, structure.
C ——— medium language.

→ These must be translated into machine language using compilers or interpreters.

→ C is a structural // procedural programming language because to solve a large programm C programming lang. divides the programm into smaller modules and these modules are know are function, method, subroutines etc

→ functions are series of computational steps to be carried out to perform a task.

→ C++ is a object oriented programming language main module of a programme are classes rather than procedures.

→ C++ → main module is class

→ class is a combination of data member & member function.

→ high level language are portable i.e ease of adaptibility of software from one type of machine to another

→ Disadvantage of Structural programming

- does not support polymorphism & inheritance

Date: / /
Page No:

Source Code

↓ Preprocessor

Prepocessed code

↓

compiler. (It transfer source into object code)

↓

object code. (0, 1) (.obj)

↓ linker

executable code. (.exe)

↓ load in memory (loading)

Run.

Secondary Memory / auxiliary — permanent storage.

primary memory — tempory storage.

- C = int C
- Cpp — in C++

linker — It is a computer programme that takes one or more object file & combine them into single executable file. that is it combines the object code. It link object file with a standard library or other object file.

loader — It loads the programme into main memory / primary (RAM)

preprocessor directive. //include <iostream.h>

global variable.
main()
{
}

Variable
1. global
2. local.

{ - opening curly brace

} - closing curly brace.

main() = It is called at the start of the parenthesis programmer.

```
int main()
{
   ——
   ——
   ——
}return 0
```

```
void main()
{
   ——
   ——
   ——
}
```

Body

→ Compiler ~~means~~ takes the code & converts into object code machine code which means binary code.

C++ ... "c" ... function operation.

```
// include <conio.h>
#include <iostream.h>          // include <stdio.h>
void main()
{
cout<<"Welcome in C.S. department";  // printf("");
getch();
}
```

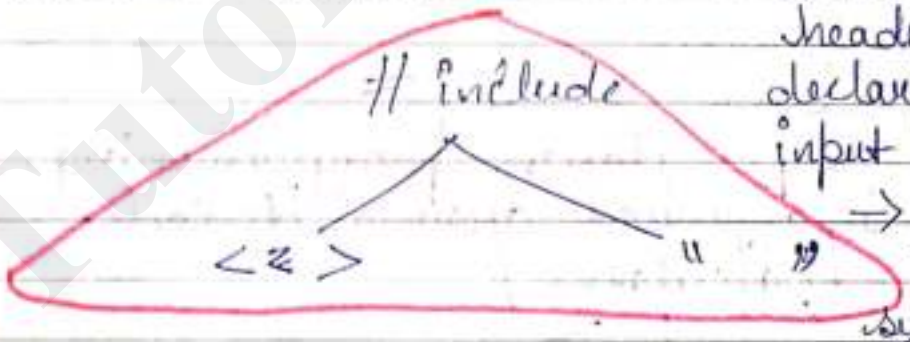
Compile — Alt+F9
Run — Ctrl+F9

→ Header file contains pre-defined function & variables that we may use in our programme

→ cout — is C++ output stream which is declared in iostream.h header file.

→ // include — It is used to include header files which include declaration of basic standard input, output library in C++.

→ // include "abc.h"



→ iostream : standard headerfile which include declaration of basic standard input output library in C++

→ System defined
#include <iostream.h>
Searching begins from system standard directory containing the directory

→ angular bracked — Searching begins from System Standard directory containing the directory
→ include standard library header file

→ user defined header file will be in (" ")

→ " " — searching begins from current directory where source programme was found & it is used to include programmer defined header file

Procedural Programme lang.

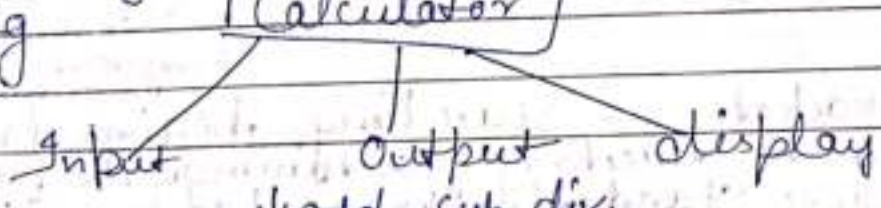
A programme in a procedure, is a list of instructions. The programmer create list of instructions & computer carry them. A procedural programme is divided into functions & each function has a specific purpose

Structural/ Procedural programming approach.

- break down the problem into sub problem.
- Treate problem. pre-exe a program into a new problem. which can further be broken down into smaller problem.
- Solve sub problem.
- It support top-down programming.

→ Programming is organised around its code.

eg



→ Programme is organised around its code.

Problems With Structural Programming

(1) Unrestricted Access

(2)

— Two kinds of data are available.

— local data — created inside a function & exclusively used by function.

— global — it is accessible by any function in programme.

(1) A change in global data necessaties re-writting all function that access that item, When data is modified in large programme. It may not be easy to tell which function access the data. It may be possible that this modification may cause the programme to work incorrectly.

(2) Real World Modelling — These have attributes & behaviours.

1) attributes — characteristics that describe an object.

2) behaviour — action that an object does in response to.

Object Oriented Approach

To combine both data & function into a single unit such a unit is called class. Function of a class are called member function.

```
class c
{
}
```

Characteristics of Ob language

1) class — it is a blue-print or plan. It specifies what data & functions will be included.
2) object — It is an instance of an class it helps physical existence.

Q = Programme of 2 no. add.

26/7/17 C++ lecture-3

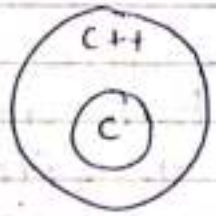
Principles of Object Oriented P

1) There are 3 principles of OOP
1) Encapsulation - It binds together member functions & data member. It binds together code & data. It manipulates and keep it safe from outside interference and misuse. It hides data.

2) Inheritance - If subclass share some common characteristics with class from it is derived. It is a process by which one object acquires (inherit) another properties of another class. Original class i.e - been derived is called base class. A class that inherit properties is called derived class (child class). Derived classes inherit some properties from base class but add new ones of their code. Inheritance provide the idea of reusability.

3) Polymorphism - It means single name multiple forms (1 interface multiple methods)

C language is a subset of C++



~~19/08/17~~

Ch -2 C++ Programme Basics

→ Simple Basic program.

```
# include <iostream.h>
# include <conio.h>
void main()
{
cout << "Hello";
}
return 0;
}
```

→ using namespace std; error declaration syntax error

→ # include — is a preprocessor directives that tells the compiler to insert another file (iostream) into your source file. It includes header files. # include is replaced by the content of the files. Directive is an instructions to the compiles.

→ Header file — They usually have the extension .h. But new C++ standard header file donot have a file extension. It include declaration. Iostream Contains declaration that are needed by Cout identifier and << (insertion operator). It is connected with basic input & output operation

→ Using Namespace std :- Using – U are going to use Std Namespace where feature of C++ standard library such as string or vector are declared. Namespace is a part of program which contains in which certain names are recognised and outside Namespace they are unknown. it says all the programs statement that follow are within Std. Namespace. It defines the scope.

→ cout = It is an object of class ostream which is an standard output stream. Stream refers to flow of data. Outstream Means flow of data to screen / out-put device.

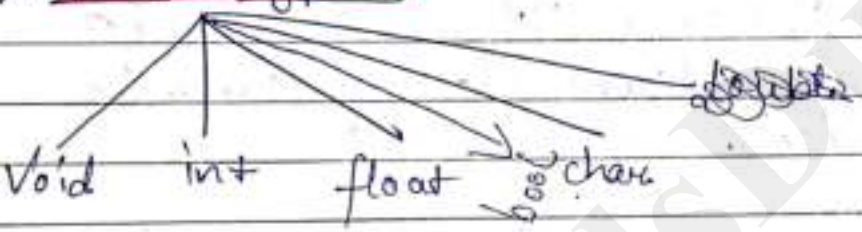
→ << (Insertion operator) or (put-to operator) It directs the content on its right to the object on its left. It directs hellow to cout which sent it to display.

→ Comments are important part of program they help to understand the program. These are explanatory statements that are included to explain what our program does. Compiler ignores the comment so they do not add to file size. Single line Comment starts from //. Multi line Comment

starts with /* ——— */

⇒ Variable — It has a symbolic name and it is given a variety of values. It is a name memory location that is used to stored a value. All variables (Var. Type & Var. Name) must be declared before they are used

⇒ Data Types :-

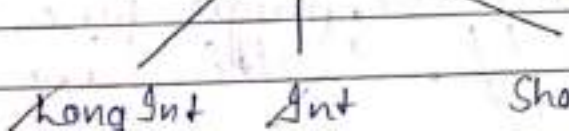


integer — A no. without fraction part is called integer data type. Range of int -2147483648 to

1) 32 bit system in int occupies four bytes of memory but earlier version of windows or MS Dos Contain 2 Byte of memory

Rang of Integer = -2147483648 to +2147483647
(-2^{31} to $2^{31}-1$)

Int data type



Range

Long always occupies 4 Byte = Same as int.
Short " " 2 Byte = -32768 to +32767
= -2^{15} to $2^{15}-1$

→ Signed Integer = It can take Both +ve & -ve values
A signed Datatype save 1 bit for signed and remaining bit for magnitude. By default datatype
0 = + are signed. signed int is same as
1 = - int.

→ Unsigned ⇒ Can take Only Non-Negative Values.
Unsigned int ⇒ Range (0 - 4294967295)
Unsigned short ⇒ (0 - 65535)
" long ⇒ Same as int
Range short ⇒ 0 to 2^{16}
int long ⇒ 0 to $2^{32}-1$

→ Size ∴ cout << sizeof (x) data type.

→ Character Datatype (Char) = stores integer that are in range from (-128 to 127) only 1 byte of memory

char a;
char a = 'A' get converted into ASCII Value

Signed Char = -2^7 to 2^7-1
Unsigned Char = 0 to 255

ASCII
0 - 48 9 - 57
1 - 49
2 - 50

A = 65
Z = 90

a = 97
b = 98
z = 122

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Floating Point Type – A no. with a fractional part to represent real no. To represent real no. we use floating data type.

- float ⇒ occupies 4 byte of memory, float is always signed. Range $(3.4 \times 10^{-38}$ to $3.4 \times 10^{+38})$
- Out of 32 bit - 1 bit is for signed., 8 bit for exponent & 23 bit for fraction. 0 – 22 for exponent fraction, 23 – 30 is for exponent, 31 is for signed
- double ⇒ Occupies 8 byte of Memory, 64 bit of memory $(-1.7 \times 10^{-308}$ to $+1.7 \times 10^{308})$ with precision of 15 digits long double is compiler dependent but is same as double.
- long double is compiler dependent but is same as double.
- Boolean (bool) = 1 bit memory = It can only has 2 values i.e true or false. 1 bit of storage
- Void – It has no values and no operations It

is used to specify type of function. Both the set of values and operations are empty.

→ Declaration perform two task. the (1)st one it tells the compiler about the name of variable. (2)what the datatype of variable is.

→ Definition = Assigning Values to a variable.

```
int a  // declaration of a
a = 0  // definition or initialization.
```

To create a constant of type long using letter 'L'

```
long a = 3.2L
```

→ Rules to define ~~identifier/variable~~ (1)Identifier may ~~constan~~ contain letter, digit & Underscore character.
(2)They must begin with a letter or underscore
(3)It should not be a Keyword.
(4)Variables name are case-sensitive i.e uppercase & lowercase letter is different.
(5) White space is not allowed.

```
int 1a X
int __a;
```

→ Keyword — a pre-define word ~~with~~ special meaning is called Keyword there are 73 Keyword in C++. (int, float, main, break, continue)

Output Variations

- endl - Causes a line feed to be inserted so that subsequent text is displayed on next line
- Input with Cin - is standard input stream that it represents the data of coming from Keyboard. >> (Extraction Operation) It takes the value on its left and places in the variable on its right

```
cin >> a >> b ---case coding;
```

- Escape Sequences - It starts with '\' back slash. It causes \ escape from normal way characters are interpreted.

```
\n = new line
\t = tab
\b = backspace
\\ = \
\' = '
\" = "
\r = return
```

Q. \\n - what is output

Aerolins
Date: / /
Page No:

Output display

```
# include <iostream.h>
# include <conio.h>
long a = 3.2L;
void main()
{
clrscr();
cout <<a;          → Output 3
getch();
}
```

```
# include <iostream.h>
# include <conio.h>
void main()
{
clrscr();
cout <<"hello\nall";      — (1)
cout <<"hello\tall";      — (2)
cout <<"hello\ball";      — (3)
cout <<"hello\\all";      — (4)
cout <<"hello\?all";      — (5)
cout <<"hello\"all";      — (6)
cout <<"hello\rallo";     — (7)   allo
cout <<"
cout <<"hello\\n all";    — (8)
cout <<"hello\\t all";    — (9)
cout <<"hello\\b all";    — (10)
```

```
cout << "hello \\\ all ";  — (11)
cout << "hello \\' all ";  — (12)
cout << "hello \\r all ";  — (13)
cout << "_____";
cout << "hello \\\n all ";  — (14)
cout << "hello \\\t all ";  — (15)
cout << "hello \\\b all ";  — (16)
         "hello \\\\ all ";  — (17)
         "hello \\\' all ";  — (18)
         "hello \\\" all ";  — (19)
         "hello \\\r all ";  — (20)
getch();
}
```

Output

① hello
all

② hello all

③ hellall

④ hello\ all

⑤ hello' all

⑥ hello" all

⑦ allo allo •

⑧ hello\n all

⑨ hello \t all

⑩ hello \b all

⑪ hello \\ all •

⑫ hello \' all

⑬ hello \r all

⑭ hello\
all

⑮ hello\ all

⑯ helloall

⑰ hello \\ all

⑱ hello \' all

⑲ hello \" all

⑳ allo\ •

Date: / /
Page No:

```
cout << " hello \\\\nroad." ;  — (21)
cout << " hello. \\\\t all " ;  — (22)
cout << " hello \\\\b all " ;  — (23)
cout << " hello \\\\ all " ;  — (24)
cout << " hello \\\\' all " ;  — (25)
cout << " hello \\\\x all " ;  — 26
```

Ans)
(21) hello \\n all
(22) hello \\t all
(23) hello \\b all
(24) hello \\ ll
(25) hello \\' all
(26) hello \\x all

Q = `cout << " hello \\" all " ;`
`cout << " hello \\\\" all " ;`] error occur

Correct Version.
1) "hello \\"
2) "hello \\\\"

Program

Q-1 W.A.P to swap the two no. with using 3 variable.

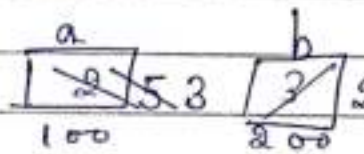
Ans)
```
void main()
{
int a, b, t;
cout << "enter a & b";
cin >> a >> b;
t = a;
a = b;
b = t;
cout << "the swap value is" << a << b;
cout << "The new value of a = " << a;
cout << " The new value of b = " << b;
getch();
return 0;
}
```

cout << "before swapping";
cout << "a = " << a;
cout << "\n b = " << b;

order of writing when int main is used.

Q-2 Without using a third variable.

```
void main()
{
a = a+b;
b = a-b
a = a-b
cout << " the new value are " << a << b;
getch()
return 0;
}
```



In multiple conditions we use else if.

~~Expressions~~ Operators. — 2/8/17

1) Arithmetic — Arithmetic Assignment Operators.
2) Logical — Increament operator ⇒ (a+1=2)
3) Relational
4) Bitwise

⇒ pre — change then use.
post — use then change.

int i = 0

-a/b = -ve
a/-b = -ve

Relational operation >, <, >=, <=, !=, ==
↳ use to compare the value.

= use to assign a value
⇒ && (logical) — ampersand
|| — or
! — not

Ex ⇒ Expression
const int x = 5;
x = x + 7;
error : wrong constant value can't be changed

Logic Operator

Q =
```
# include <iostream.h>
# include <conio.h>
Void main ()
{
clrscr ();
int c = 1, a = 5, b = 5;
① if (c = b + 5 && a == 5)
{
Cout << a << endl << b << endl << c ;
}
getch ();
}
```

Output = 5 ← ①
5
10

Warning — Possibly incorrect assignment.

② if (c = b + 5)

Output → 5
5
10

③ if (a == 5 || (c = b + 5))

output → 5
5
10

(4) if (c = b+5) || a == 5)

output = 5
5
10

(5) if (a == 5 && (c = b+5))

output = 5
5
10

Left Shift or Right Shift

1) Left shift (<<) operator.

int i = 2
j = i << 2
j = 8

0000 0010
0000 0100 — 1 shift
0000 1000 — 2 shift

Left operands value is moved left by the no of bits specified by right operand for each left shift higher order bit is shifted is shifted out & and a zero is brought on the right.

→ Size of operator

```
int x;
cout<<sizeof(x);
```

4 → 00000100 (1's comp)
11111011 (1's comp)
+ 1 (2's comp)
−4 → 11111100

2) Right shift (>>) operand
left operand value is moved right by no of bits specified by right operand.

	S	C
0+0 = 0	0	0
0+1 = 1	0	0
1+0 = 1	0	0
1+1 = 0	0	1

→ Conditional Operator.

? Smessage true : message false

Comma Operator.

→ Left side of comma operator is always evaluated as void i.e expression on right side become the value of comma separated expression

```
int a=5, b=10;
a=(b=9, c=10);   ——10
```

(b=9 not evaluate; c=10 evaluate)

output

```
a=(9,10,15);   ——15
```

→ Precedence Table.

Unary > binary > ternary

Highest	( ) , [ ] , .	→ L to R
Unary	! ~ ++ -- Size of & * (Pointer) → R to L	
Arithmetic	+ - * / %	→ L to R

Left shift / Right shift >> , << — L to R

Relational Operator >, >= , < , <= , == , - L to R

Equality == , != = L to R

Bitwise and , or , not

Logical && , || — L to R

Conditional ? : — L-R

Assignment Operator. = , += , -= , *= => R-L

Comma operator — L-R

Q - Reverse of No. => Rewrite

Ans) void main()
{ rev = 0
int n;
cout << "enter the no";
cin >> n;
while (n > 0)
{
int d = n % 10; ——> rev = rev * 10 + d
n = n / 10; } getch(); }

Q- To print pattern.

```
int n; cout << "\n enter the no. upto which u want pattern";
for (int i=0 ; i<n; i++)
{
for (int j=0 ; j<=i ; j++)
{
cout << "\* ";
}
cout << endl;
}
getch ();
}
```

```
*
**
***
```

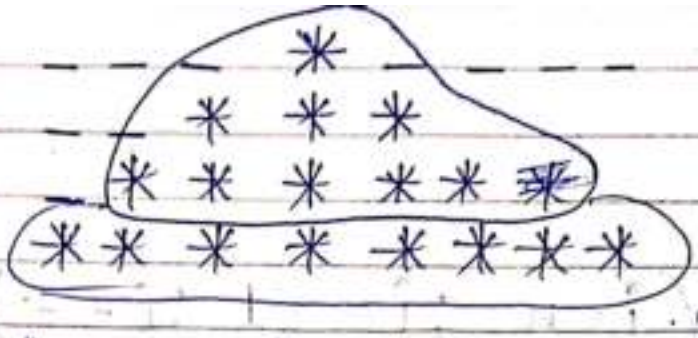
or

```
(i=1 ; i<=n; i++)
(j=1 ; j<=i ; j++)
```

Q- Sum of the digits

```
int n, s=0;
cout << "enter the no";
cin >> n;
while (n>0)
    {int d = n % 10;
     n = n / 10;
     s = s+d;
    }
cout << s; getch(); }
```

Q =



n = no. of rows

```
for(i=1 ; i<=n ; i++)
{
for(j=n-1 ; j>=i ; j--)
{
cout<<" ";
}
for (k=1 ; k<=(2*i-1) ; k++)
{
cout<<"*";
}
cout<<"\n";
}
```

Q =

```
*  *  *  *  *
   *  *  *
      *
```

n = 3

i = 3 ; i >= 1 ; i--

```
for (i=n ; i>=1 ; i--)
{
for (j=(2*i-1); j>=1 ; j--)
{
cout<<"*";
}
for (k=n-1; k>=i ; k--)
cout<<" ";
}
```

my pattern

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Q =

```
* * * * *
 * * *
  *
```

Ans)

```
for (i=n ; i>=1 ; i--)
{
  for (j=n ; j>i ; j--)
  {
    cout << " ";
  }
  for (k=1 ; k<(2*i) ; k++)
  {
    cout << "*";
  }
  cout << "\n";
}
```

n = 3

→ ma
po

Output

```
* * * * *
 * * *
  *
```

DIAMOND PATTERN.

Ans)

```
for (i=1 ; i<=n ; i++)
{
for (j=i ; j<=n ; j++)
{
cout << " ";
}
for (k=1 ; k<=(2*i-1) ; k++)
{
cout << "*";
}
cout << "\n";
}
for (i=(n-1) ; i>=1 ; i--)
{
for (j=n ; j>i ; j--)
{
cout << " ";
}
for (k=1 ; k<(2*i) ; k++)
{
cout << "*";
}
cout << endl;
} getch(); }
```

Structure

Structure – It is a collection of simple variables. In a structure can be of different type int, float, long, char etc. Data items in a structure are called members of structure.

Syntax / Declaration

```
struct name
{
datatype variable;
 ___
 ___
};
```

∵ when a structure is declared memory is not allocated to it.

Examples

```
struct student
{
int roll;
int age;
float marks;
};
```

(members of structure)

When a structure is declared memory is not allocated to it.

⇒ Defining a variable

Structure name / variable name

student S1;
student S1, S2;

⇒ S1 is a variable of type structure. Size of a structure variable will be equal to the size of all make members of a structure.

In above case 4 – int roll
4 – int age
4 – float marks

12 byte total.

Accessing a structures member.

– Once a structure variable is define its member can be excessed using (.) dot operator.

⇒ Syntax :-
Structure variable . member name;
Initializing Member func.

S1.age = 22;
S1.roll = 12
S1.marks = 22
Cin >> S1.age;
cout << S1.roll;

Another Syntax:

```
struct Student
{
int roll;
int age;
float marks;
} S1, S2, S3;
```

Initializing Member Variables.

Another Syntax.

```
S1.roll = 12
S1.age = 22
S1.marks = 22
```

OR

```
Student S1 = {5, 20, 22};
```

Structure Assignment

One structure variable can be assigned to another.

```
Student S1 = {22, 12, 225};
Student S3;
S3 = S1;
```

values of S1 is also present in S3.

Note — One structure variable can be assigned to another variable only when they are of same structure type.

① One structure variable can not be compared to another structure variable. & an arithmetic & relational operators can't be used with structural variable. It can be performed on individual member of structure variable.

S3 = S1 + S2 - X

S3.marks = S1.marks + S2.marks ✓

if (S1 > S2) X
if (S1.age > S2.age) ✓

Nested Structures.

```
Struct address
{
  int zipcode;
  int house no;
long int phone no;
};
Struct student
{
  int roll;
  int age;
  int marks;
  address a1;
};
void main()
{
  student S1;
  S1.age;
  S1.a1.zipcode = 110085;
  S1.a1.house no = 68;
```

16/8/17

Date: / /
Page No:

C++

Structure & Classes are same except for default excess level structure is bydefault public but class is bydefault private. In C structures are extensively use for data only but in C++ they are used for both data & function.

Enumeration — It is set of name integer constants

Syntax

```
enum enumerator name
{
value 1, value 2 ---- value n
};
```

each of the value stands for an integer value

eg -

```
enum day_week
{
SUN, MON, TUE
};
```

(0, 1, 2)

```
void main()
{
cout << SUN << MON
```

(SUN value is 0, MON → 1)

Default Value for the values in enumerated list is Zero. Rest have the value 1 more that the previous one.

If you want to specify / assign any Specific Value than you should specifically assign value.

eg :- enum day_week
{
SUN=1, TUES, WED ;
};
↳ Increment by 1 by previous value

⇒ Declaring Enum Variable :-

enumeration name variable.
day_week $d_1, d_2, d_3 - - - d_n$;

2) Arithmetic & Relational Operation can be performed on Enumerator. d1 = MON + TUE = 1+2 = 3

3) # include <conio.h>
enum direction
{
North, South,
East,
West;
};
Void main ()
{
direction d;

d = EAST;
cout << d;
getch();
}

Check
{
int x;
x = West;
cout x;
for checking

don't know ask mom.

Date: / /
Page No:

⇒ Union = In union a memory location that is shared by 2 or more diff. type of variable.

```
Union U
{
int a;
char c;
double d;
};
```

groups
A FUNCTION ~~PROVIDES~~ A NO. OF PROGRAM STATEMENTS INTO A UNIT AND GIVE IT A NAME.

WHY WE NEED FUNCTION
1. Reduce Program Size.
2. Maintaining and Debugging is easy.
3. Understanding, Coding & Testing Of Separate functions Are Easier
4. Same Code Can be Used Multiply Times.

Function Declaration

It is a statement that identifies a func. with a name, a list of arguments it accepts and type of data it returns it is terminated with a semicolon

Syntax
returntype functionname (datatype variable, datatype variable)

```
int sum (int a, int b);
void fun (float f, double);
```

=> The function declaration is also called function prototype / signature.

void ab(); function that accepts no argument & no return value

1. function Declaration — outside main or header file
2. " Definition — below main
3. function Calling — inside main

=> func. Def.

When a function is defined, a space is allocated for that in memory. It includes two parts

1) Declarator — It Must be Matched with function prototype.
2) function Body — Combination of statements that define a function and it is delimited by. { }

=> Declarator is not terminated by semicolon.

Syntax

```
int sum (int a, int b)
{
 int c;
 c = a+b;
 cout << "Sum = ";
 return c;
}
```

⇒ Argument Name in the function definition & function declaration need not be same. However the data type of argument must be matched.

Calling of A function.

Syntax

```
sum (a, b);
```

17/8/17

⇒

```
# include <iostream.h>
# include <conio.h>
int sum (int a, int b); — f" decl"
void main ()
{
 int a, b, c;
 cout << "enter 2 no";
 cin >> a >> b;
 c = sum (a, b); — function calling
 cout << "sum = " << c;
 getch ();
}
```

```
⇒ int sum (int a, int b)
{
 int c = a+b;
 return c;
}
```

⇒ Elemenating the declaration

→ func. declaration can be elemenated by placing func. definition before the func. call.

```
void display()
{
cout << "Hello";
}
void main()
{
display();
getch();
}
```

⇒ Passing Arguments To functions.

Argument — data i.e passed from a program to func. definition.

Parameter — Variable Used With a func. to hold argument values.

Date: / /
Page No:

Two Ways of Passing Arguments to function.

a) Call by value b) Call by reference.

1) Value of the variables are passed by calling function to called function. The function used copy of Arguments.

In Call by Value – Any Modification to the value of parameter will not effect argument value.

2) Passing Structure as Argument

→ Entire structure can be passed as argument to function.

```
struct Employ
{
int id;
float sal;
};
void display (employ e)
{
cout << e.id << e.sal;
}
int main ()
{
employee e1;
e1 = {31, 50000}
display (e1);
return (0);
}
```

```
Struct employee
{
int id;
float sal;
void display (int a)
{
cout <<a;
}};
int main ()
{
employee e;
e.sal = 5.5;
e.id = 1;
display (e); - return 0;
}
```

⇒ Returning Values from function.

Return statement is used to ~~determine~~ terminate the execution of a function & return control to called function.
When a () returns a value datatype of this value must be specified.

1) By - default return type of fun. is int.
2) A fun. can return only one value / argument to return multiple argument we used call by reference.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Returning A Structure

```
struct cord
{
int x, y;
};
cord add_cord (cord C1, cord C2)
{
cord C3;
C3.x = C1.x + C2.x;
C3.y = C1.y + C2.y;
return C3;
}
void main ()
{
clrscr ();
cord c, d, e;
cout << "enter the values of c = ";
cin >> c.x;
cin >> c.y;
cout << "enter the values of d = ";
cin >> d.x;
cin >> d.y;
e = add_cord (c, d);
cout << e.x << endl << e.y;
getch();
}
```

or

```
cord c, d, e
c = {1, 3};
d = {4, 5};
cout << addcord(c,d);
} OR
cordsum = add.
          cord(c,d);
cout << sum;
```

```
{cord sum = add_
            cord(c,d);
cout << sum;
```

Reference

Reference is an alias (alternative) a diff. name for a variable

Syntax

datatype &varname

⇒
```
int a = 5;
int &b = a;
  b = 9;
cout << a << b;
```

Output = 9, 9

1) Once a reference is created it can't be made reference to another object. Reference can't be initialized.

```
int &b =      X
int &b = a    ✓
```

2) Reference Can't be null

⇒
```
int main()
{
  int a = 15;
  int &b = a;
  cout << a << "\t" << b << endl
```

Output	
15	15
30	30
28	28

```
b = 30;
cout << a << "\t" << b << endl;
  b--;
  a--;
cout << a << "\t" << b << endl;
getch();
return 0;
}
```

→ A variable can have multiple reference. Changing value of one will affect all of them.

#Call Of Reference

1) The Address of Argument is copied into the parameter. changes made to the parameter will have effect an argument also.

Program

```
void swap (int &x, int &y);
void swap (int &x, int &y)
{
int temp;
temp = x;
 x = y;
 y = temp; }
```

```
void main()
{
int a = 100, b = 200;
cout << "Before swap a = " << a;
cout << "          "  b = " << b;
swap (a, b);
cout << "After swap a = " << a;
cout << "After swap b = " << b;
getch();
return 0;
}
```

⇒ **Default Argument** — It must be the last argument of the func;

```
float area (float a, float pi = 3.14)
int main()
{
float area_circle;
int r;
cout << "enter r";
cin >> r;
area_circle = area (r);
getch();
return 0; }
```

```
float area ( float r, float pi=3.14)
{
int a;
c = pi*r*r;
return c;
}
```

It is an argument whose value is provided in fun. declaration it is automatically assigned by compiler if the caller fun. doesn't provide a value for argument with default values. It must be the last argument of the argument list.

```
X int fun (int a=2, int b = 3, int c);
  int fun (int a , int b = 8, int c = 3); ✓
```

Scope And Storage Class

A storage class defines the scope (visibility) and lifetime of variable or fun. within a C++ program.

There are following storage classes can be used within a C++ lang.

1) Auto
2) Register
3) Static
4) & Extern

1) The Auto storage class is the default storage for all local variables

int x ; ✓
auto int x ; ✓

2) Register – The register storage class is used to define local variables that should be stored in a register instead of RAM. This means that the variable has max. size equal to register size. register int x;

3) The Static Storage Class instruct the compiler to keep a local variable in existence during the life time of program. Instead of creating & distroying it each time it comes into and goes out of scope.

Page No.

Therefore making local variables static allows them to maintain their values b/w func calls.

The static modifier may also be applied to global variables when this is done, it causes that variable scope to be restricted to the file in which it is declare.

In C++ where static is used on a class data member it causes only one copy of that member to be shared by all objects of its class.

Static have default value 0.

25/08/17

=>
```
# include <iostream.h>
void func (void); — declaration.
static int count = 10; // Global variable.
                       static
void main ()
{
while (count - -){ - 9 .
{
func();
}
}
void func (void)
{
static int i = 5; // local static variable.
i++;
```

```
cout.<<" i is "<<i;
cout << " and count is " <<count <<endl;
}
```

Output

i is 6 and count is 9
i is 7 and count is 8
8 and count 7

Static don't Initialize more than 1

9	6
10	5
11	4
12	3
13	2
14	1
15	0

3) Extern Storage – The extern storageclass is used to give a reference of a global variable that is visible to all the program file.

When you use extern, the variable can't be initialize at all it does us point a variable name. at a storage location that has been previously define.

when you have multiple file and you define a global variable or func. which will be used in other file also, then extern will be used in another file to give reference of define variable or function.

main.cpp (first program)

```
#include <iostream.h>
int count;
extern void w();
main()
{
count = 5;
w();
}
```

S.cpp (2nd program)

```
#include <iostream.h>
extern int count;
void w()
{
cout << "count is:" << count
}
```

do not run.

28/8/13

1 X 0 ! = base case / terminate condition

Recursion :- A function calling itself is called recursion. It divides a bigger problem in sub problems.

Factorial Programme :

```
int fact (int);
void main ()
{
int n, factorial;
cout << "Enter the no. =";
cin >> n;
factorial = fact (n);
cout << "factorial =" << factorial;
getch ();
}
int fact (int num)
{
if (num == 1)
return 1;
else
return num * fact (num - 1);
}
```

Eg = 5 ! = 5 X 4 !

fun (5) = 5 * fun (4

5 X 4 X 3 X 2 X 1

Array

It is a collection of homogeneous elements.

int a[10]; - 'a' is an array of size 10.

In array there is contiguous memory allocation

Datatype arrayname [size]; → Syntax.

Maximum No. of Element Array Will Hold.

int Marks[5]; #define size[10];

Array Index is used to refer a memory location.
Array Index always starts from zero.

Accessing Array Element

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

Date: / /
Page No.

Searching An Element in Array

```
#include <iostream.h>
int a[5], i;
void main()
{
for (i=0 ; i<5 ; i++)
cout << "\n enter the elements";
cin >> a[i]; ——→ cout << "enter the element to search" << n;
for (i=0 ; i<5 ; i++)
{
if (a[i] == x)
} cout << " No. found at " << i;
}
getch();
}
```

→ If array is initialised at the time of declaration we can omit size.
→ If we assign few values to array remaining values are said to 0. → If we assign values more than array size than it will generate compiler error.
→ Initialization & declaration of array can't be separate

Multiplication

```
void main()
{
int m1[3][3], m2[3][3], m3[3][3], sum=0, i, j, k;
cout << "enter first matrix element of m1";
for (i=0 ; i<3 ; i++)
{
for (j=0 ; j<3 ; j++)
{
cin >> m1[i][j];
}
}
cout << "Enter 2nd matrix";
for (i=0; i<3; i++)
{
for (J=0; J<3; J++)
{
cin >> m2[i][j];
}
cout << "multiplying two matrix";
for (i=0; i<3; i++)
{
```

```
}
for (j=0, J<3, j++)
{
Sum=0;
for(k=0, k<3, k++)
{
sum= sum + m1 [i][k] * m2 [k][J];
}
m3[i][j]=sum;
}
cout <<" multiplication of 2m".
for (i=0, i<3, i++)
{ for ( j=0, J<3, j++)
{
cout << m3 [i] [j] <<" \t";
}
cout <<" \n";
}
getch();
}
```

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

(2) 13/07/17 C++ Strings

→ If the entered value exceeds the array size then it may lead to crashing of system.

'\0' = null character.

O = 48
A = 65
} Ascii Value.

∵ char s[10];
cout << "enter the string"
cin >> s;

→ set w ( ); cin >> setw(9) >> s;

→ '>>' extraction operator considers a space as a terminator. So any thing entered after space will not be consider as part of string

→ To read space we use ~~cin.get()~~

cin.get (str, size, delimiter)
↓ By default new line.

eg. cin.get (s, 10);
cout << s;
cin.get (s, 10, ' ')

eg char s[] = "Hello";
char s[] = {'H', 'E', 'L', 'L', 'O'};

Programme to count size of string

→ while (s[i] != '\0')
{
i++;
}
cout << "size of array" << i;

char s[20];
cout << "enter string";
cin >> s;
int i = 0;

Coping of one string into another.

```
#include <string.h>
char S1[10], S2[10];
cout <<"enter first string";
cin >> S1;
int len = strlen(S1);
for (i=0 ; i< len ; i++)
{
 strcpy (S2 , S1);
}
S2[i] = '\0';
cout << S;
}
```

or `S2[i] = S1[i];`

or

```
while (i < len)
{
S2[i] = S1[i];
}cout << S2;
```

Concentenate two string

S1 - Hello
S2 - Hi HelloHi

```
char S[50], i;
cin >>S; int i=0;
while (S[i] != '\0')
{
S[i] = S[i] -32;
}
```

A → 65 ASCII Value
a → 97

(5) Checking uppercase or lowercase..

```
if ( s[i] >= 'A' || s[i] <= 'Z')
```

(6)
```
int main ()
{ getline (cin, s4);
string s1;
string s2 ("Hello World");
string s3 (s2);
string s4 (5, 'A');
string s5 (s2, 6);
string s6 ("Hello", 3);
string s7 ("Hello", 33, 95);
cout << "Value of s1" << s1;
.
.
.
cout << "Value of s7" << s7;
}
```

System defined class.

Ch - 7

C++ - class
C - void main.

(7) Concatenation

```
int main ()
{
string s1 ("Hi");
string s2 ("Hello");
```

Set w = iomanip.h

Inbuit func. & their purpose.

* strcpy(S1,S2) → Copy string S2 into string S1.
* Strcat(S1,S2) → concatenate (append) string S2 onto the end of string S1.
* strlen(S1) → Return the length of S1
* strcmp(S1,S2) → return 0 if S1 & S2 are the same, < 0 if S1 less than S2 & > 0 if S1 > S2
* strchr (S1, ch) – return a pointer to the 1st occurence of character Ch in the string S1.
* strstr(S1, S2); – return a pointer to the 1st occurence of string S2 in the string S1.

```
#include <iostream.h>
#include <string.h>
int main()
{
char string str1[10] = "Hello";
     str2[10] = "World";
     str3[10];
int len;
strcpy(str3, str1);
cout << str3;
strcat(str1, str2); cout << str1; }
```

```
len = strlen(str1);
cout << len << endl
}
```

C++

```
#include <iostream.h>
#include <conio.h>
int main()
{
string str1 = "Hello";
"      str2 = "World";
"      str3 ;
int len;
str3 = str1;
cout<<"str3="<<str3<<"str1"<<str1;
str3 = str1 + str2;   -> incase of string operator overloading
cout << "str3 = "<< str <<endl;
len = str3.size();
cout <<"str3 = "<<len <<endl;
return 0;
}
```

=> Array of strings

1) Using 2D array (Both C & C++)
2) Using string keyword (only in C++)
3) Using vectors (only C++) (not in syllabus)

```
#include <iostream.h>{String in 2D}
int main()
{char colour[4][10] = {"Blue", "Red", "Orange",
                       "Yellow"};
for (int i=0; i<4; i++)
cout << colour[i] << "\n"
return 0;
}
```

*No. of strings and ~~size~~ size of a string both the values are fixed.

A 2-D array is allocated whose 2-dimension. is equal to max. size string which causes wastage of space.

```
#include <iostream.h>
#include <string.h>
int main()
{
string colour[4] = {"Blue", "Red", "Orange",
                    "Yellow"}
for(int i=0 ; i<4 ; i++)
cout << colour[i] << "\n";
getch();
}
```

Hello World

⇒ → str1.compare(str2)
↕
strcmp(str1, str2)
} Both have same header file.

→ strcat(str1, str2)
↕
str3 = str1 + str2;
↕
str1.append(str2);
} Both have same header file

str1.append(str2, pos1, len2);

→ swap(str1, str2);

objects & class

class is a user-defined data type which is combination of member data & fun.

```
class classname
{
___  → access specifier  (Private / Public / Protected.)
___    data member
___    member function
};
```

Three types of access specifier :-
1) Private 2) Public 3) Protected.

Public – Allow data & fun. to be accessible by other parts of program i.e outside the class.

Private → Private are accessible only by members of class i.e within the class. It is basically used for inheritance.

⇒
```
class box
{
public:
int l;b,h;
void getdata()
{cout<<"enter l,b & h";
cin>>l>>b>>h;
}
void displaydata()
{cout<<'l,b & h are :"<<l<<b<<h;};
void main()
{
box a;
a.getdata();
a.displaydata(); getch(); }
```

C++

=> Object is the instance of class

outside the class

Syntax

return type classname :: func. name()

{

}

```
class cont call
{
public :
int a, b;
void add (int a, int b);
void Sub (int a, int b);
};
void cal :: add (int a, int b)
{
cout << "sum = " << a + b;
}
void cal :: Sub (int a, int b)
{
cout << " Sub = " << a - b;
}
void main ( )
{
cal c;
c.add (10, 5);
c. Sub (10, 5); getch ( ); }
```

Constructor is automatically called when an object is created. It is a special member func. of a class.

* It has the same name of the class. It must be a public member.

* No return value.
* Default constructors are called when constructors are not defined for the classes.

```
class ~~employee~~ example
{
int a,b;
public:
Example()
{
a = 10;
b = 80;
cout << "In Constructor \n";
}
void display()
{
cout << "values =" << a << "\t" << b;
};
void main()
{ example
~~employee~~ e;
e.display();
getch();
}
```

int a
class (b)
b.a;

Constructor has 3 types.

1. Default constructor
2. Parametrized constructor
3. Copy constructor.

~~Copy Constructor~~ ⇒

Default ⇒ Without any argument

Parameterised ⇒ Constructor that has arguments

Copy ⇒ When an object is used to initialise another object or copy of existing instance need to be created

Syntax ⇒ `classname (const classname &object)`

```
{
  body of Constant;
}
```

- Example (Const Example &e1)

```
{
  a = e1.a;
  b = e1.b;
}
```

→

```
void main()
{example e2(e1); getch();}
```

Always pass object by reference by using copy constructor.

Destructor

It is called when a object is destroyed. It has same name as of class but preceded by (~) sign. It has no return type. Can't have parameters.

Switch Case Using Do-while Loop

```
void main ()
{
int a, c, b, s, d, m;
char ch;
do
{
cout << "\n 1. sum";
cout << "\n 2. diff";
cout << "\n 3. Mult";
cout << "\n 4. Exit";
cout << "enter ur choice";
cin >> c;
switch (c)
Case 1:
cout << "enter 2 no's for addi";
cin >> a >> b;
c = a + b;
Cout << "sum" << c;
break;
Case 2: ==
Case 3: ==
Case 4:
exit(0);
default : cout<<"      ";
}
cout<<" Do u want to
continue (y/n) ";
cin >> ch;
} while (ch == 'Y' ||
ch == 'y');
getch ();
}
```

Explicit:- use only with constructor. A constructor with 1 argument is implicitely a conversion constructor (from basic to class). To prevent implicit conversion we use explicit keyword

```
class A
{
public:
 int i;
 explicit A (int j)
 {e = -1;
 }
 };
```

A ob(5);
A ob = 5; X → conversion constructor

Objects as Argument

It is used by value or by reference.

```
class A
{
void add (A a1, A a2)
{
A a3;
a3.a = a1.a + a2.a
cout << a3.a;
};
void main()
{
A a1(5);
A a2(6);
A a3;
a3.add(a1, a2); }
```

```
class A
{int a; public:
A (int a1)
{a = a1;
}
```

Static Data Member variable.

If a member variable is declared as static then only one copy of that variable exists & all objects of that class shares that variable.

Static Data Member is declared within class but it is defined outside the class. It means static members are associated with class not with object

```
Class A
{public:
int n;
static int m;
public & public
A()
{
n = 3;
m++;
}
A(int a1, int b1)
{
n = a1;
m = b1;
}
void display()
{
cout << m << "\t" << n;
}
};
int A::m = 0;
void main()
{
A a1(50,60);
A a2(70,80);
a1.display();
a2.display();
A a3,a4;
a3.display();
}
```

Output	
2	3
2	3
4	3
4	3

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

C++

Static Member function

The static member fun. can be called, even if no objects of the class exist & the static fun. are accessed using only the class name & scope resolution operator. A static member fun. can only access static data member, other static member fun. & any other fun. from outside the class.

```
class Box
{
public:
static int ob;
int l, b, h;
Box (int a1, int b1, int c1)
{
cout << " constructor called ";
l = a1;
b = b1;
h = c1;
ob++;
}
int volume ()
{
return l*b*h;
}
```

```
static int getcount()
{return ob;
}};
int Box :: ob = 0;
int main()
{
```

```
cout << "Initial stage count" << Box::getcount()
     << endl;

Box Box1 (3, 1, 1);
Box Box2 (8, 5, 2);
cout << "final stage count = " << Box::getcount()
     << endl;
Box1.getcount() << endl;
return 0;
}
```

Static fun. can only use static variable.

Restrication On Static Member fun.

1) Static fun. does not have This Pointer.

```
static int disf()
{
cout << this -> l; X
}
```

2) Can directly refer to static memory only.
3) A fun. can't have both static & non-static version in same class.

3) Can't be declared const or volatile or virtual

4) **friend function** – A non-member func. of class that has access to public, private & protected members of a class, public. To create/declare a friend fun. include its prototype within class. It can be declared either in public or private sections.

*A friend functions can be a member of another class

```
class B;  // forward declaration
class A
{ int m;
public:
A(int i)
{
m = i;
}
friend void add(A a, B b);
};
class B
{ int m;
public:
 B(int j)
{m=j; }  friend void add(A a, B b);
};
```

```
friend void add (A a , B b)
{
int sum = a.m + b.m;
cout << "Sum = "<< sum;
}
void main()
{
A a(5);
B b(7);
add (a, b);
}
```

---

2)

```
class B;
class A
{
int m;
public:
void add (A a , B b);
};
class B
{
int n;
public:
friend void A :: add (A a , B b);
};
void A :: add (A a, B b)
{
```

```
int c;
c = a.m + b.n;
cout << "sum = " << c;
}
void main()
{
A a1, a2(5);
B b1(7)
a1.add(a1, b1);
}
```

friend class

A class can be a friend of another class that have access to public & protected members of a class.

```
class rectangle
{
  int l, b;
  public:
  rectangle()
  {l = 10;
   b = 20;
  }
  friend class square;
};
class square
{int s;
public:
```

A friend class of rect

* In order to access the private & protected member of class rectangle into square class we must explicitly passed an object of rectangle class to the member fun. of a square class.

```
Square ()
{
s = 5;
}
void display (Rectangle Rect)
{
cout << Rect.l;
cout << Rect.b;
cout << s;
}};
void main ()
{
Rectangle R;
Square S;
S.display (R);
}
```

★ Similar to passing an object as arg. argument. Bhi the difference it as an object we are passing as argument is of diff. class (rectangle) & calling object is of diff. class square.

friend function Ex.

```
class A
{
int m, n;
Public:
friend void display (A a);
int setval (int a, int b)
{
m = a;
n = b;
```

```
void display (A a)
{
cout << a.m;
cout << a.n;
a.m = 5;
}
void main()
{
A a1;
a1.setval(7, 2);
display(a1);
}
```

Q = Coding for checking Prime No.

```
void main()
{
int n;
cout << "enter n";
cin >> n;
for (int i = 2; i <= n/2; i++)
{if (n%i == 0)
 {
  c = 0;
  break;
 }
 else
 {
  cout << "not prime";
 }
}
if (c == 1)
cout << "no is prime";
else
cout << "no is not prime";
```

*fibonacci Series Using Recursion:

```
int fibonacci (int);
void main ()
{
int num
cout << "enter the no. of terms in series";
cin >> num
cout << "\n fibonacci series is :\n";
for (int i=0 ; i< num; i++) or for (int i= 1 ; i<= num; i++)
{
cout << fibonacci (i) << "\n";
}
return 0;
}
int fibonacci (int n)
{
if (n == 0)
return 0;
else if (n == 1)
return 1;
else
return (fibonacci (n-1) + fibonacci (n-2));
}
```

or

```
if (n < 2)
return n;
```

0 1 1 2 3 5

0

→ **Data** : Raw facts which is given to the computer for processing..

→ Data is represented in binary form (0 or 1)

→ getch() is used to hold the output on screen.

Q = factorial.

```
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr();
int i,n;
cout<<"enter the no. you want factorial of:";
cin>>n;
for(int i=1;i<=n;i++)
{
i = i*i;
}
cout<<"factorial is"<<i;
}
```

⇒ Bitwise operator

p	d	p & d	p | d	!p	!d
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

Q = Pattern reverse

```
* * *
* *
*
```

```
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr();
int n;
cout <<"\n enter the no. upto which you want pattern";
cin >>n;
for (int i=n ; i>=0 ; i--)
{
for (int j=1 ; j<=i ; j++)
{
cout <<" \t* ";
}
cout <<"\n";
}
getch();
}
```

Q = Pattern reverse

→ 0 space 3*
1 space 2*
2 space 1*

```
* * *
 * *
  *
```

```
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr();
int n;
```

```
cout <<"\n enter the no. upto which you want pattern";
cin >>n;
for (int i=n ;i>=0; i--)
{
for (int j=1 ;j<=n-i ;j++)
{
cout <<" ";
}
for (int k=1 ;k<=i ;k++)
{
cout <<"* ";
}
cout <<"\n";
}
getch ();
}
```

Q = Palindrome

```
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr ();
int n,p,rev=0;
p=n;
while (n>0)
{d = n%10;
 n = n/10;
 rev = rev*10 +d;
}if (rev ==p) ⟶
```

```
⟶ cout<<"palindrome";
else
cout <<"not palindrome";
getch ();
}
```

Array

```
Q = int a[5];
    for (i=0 ; i<5 ; i++)
    cin >> a[i];
    max = a[0];
    for (i=0 ; i<5 ; i++)
    {
    if (a[i] > max)
    max = a[i];
    }
    cout << max;
```

2-D array

	Col 1	Col 2	Col 3
Row 0	1	2	3
Row 1	4	5	6
Row 2	7	8	9

8/9/17

```
int a[3][3] = {1,2,3,4,5,6,7}
int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}}
int a[][] = {1,2,3,4,5,6,7,8,9};
int a[3][] = X
```

↳ column must be specified

```
int a[][] = {1,2,3,4,5,6}  error.
```

#

```
void main()
{
int a[3][3],i,j;
for (i=0 ; i<3 ; i++)
```

```
{
for (j=0 ; j<3 ; j++)
{
cout << "enter elements = ";
cin >> a[i][j];
}
for (i=0 ; i<3 ; i++)
{
for (j=0 ; j<3 ; j++)
{
cout << a[i][j] << "\t";
}
cout << "\n";
}
```

Addition

```
for (int i=0 ; i<3 ; i++)
{
for (int j=0 ; j<3 ; j++)
{
cout << "enter matrix A:";
cin >> A[i][j];
cout << "\n";
}}
for (i=0 ; i<3 ; i++)
{
for (j=0 ; j<3 ; j++)
{
cout << "enter matrix B:";
cin >> B[i][j];
cout << "\n"; }}
```

```
for (i=0 ;i<3 ;i++)
{
for (j=0 ;j<3 ;j++)
{
c[i][j] = a[i][j] + b[i][j];
}}
for (i=0 ;i<3 ;i++)
{
for (j=0 ;j<3 ;j++)
{
cout << c[i][j] << "\t";
}
cout << "\n";
}
```

Transpose

```
void main ()
{
int a[10][10], b[10][10], c[10],[10], i, j, m, c;
cout << "Enter rows & col";
cin >> m >> c;
for (i=0 ;i<m ;i++)
{
for (j=0 ;j<c ;j++)
{cin >> a[i][j];
}}
for (i=0 ;i<m ;i++)
{
for (j=0 ;j<c ;j++)
{
cout << a[i][j] << "\t";
}
cout << "\n";
}
for (i=0 ;i<m ;i++)
{
for (j=0 ;j<c ;j++)
{
c[i][j] = a[j][i];
}}
cout << "After transpose";
for(i=0
for (j=0
cout << c[i][j];
```

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Inheritance

Process of creating new class from an existence class is called inheritance. The new class is also called subclass / derived class / child class and the existing class is known as base class / parent class / superclass. ~~derive~~

Derived class ~~has~~ inherits all the members (data & function) of base class and adds its own functionality to them.

Inheritance allow code reused ability.

The Syntax of a derived class

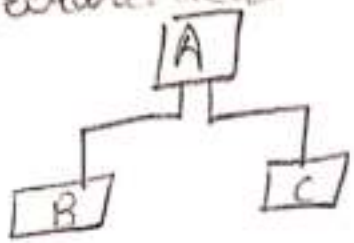
```
class A          → Parent class / base class
{
  ____;
  ____;
  ____;
};
class B : Public A → Access-by-default specifier is private
{
  ____
  ____
  ____
};
```

→ child class / derived class.

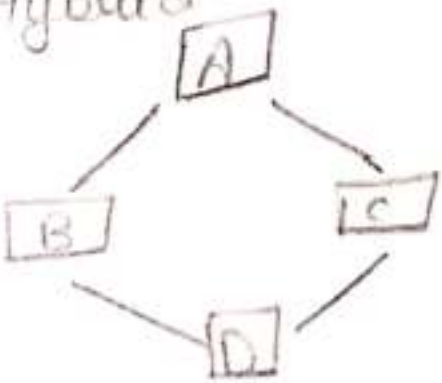
Syntax of derived class

By default access specifier is private.

Hierarchical



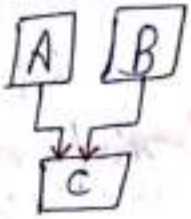
Hybrid



Inheritence type	Base Access type	Derived access type
Private	private public protected protected.	inherited but inaccessible private Private
Protected	private protected public	inherited but inaccessible Protected protected.
Public	Private Protected public	inherited but inaccessible protected public.

Access	public	protected	private
Same class	yes	yes	yes
Derived class.	yes	yes	No.
Outside class	yes.	No.	No.

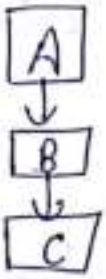
Multiple



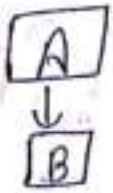
Types of inheritance

1. Single inheritance (Single level)
2. Multilevel inheritance.
3. Multiple
4. Hierarchical inheritance.
5. Hybrid. (combination of Multiple & Hierarchical)

Multilevel



Single



Single level Inheritance

Example:

```
class A
{
public :
int a,b;
void getdata ()
{
cout <<"Enter a,b";
cin >>a>>b;
}
void display ()
{
cout <<" a="<<a;
cout <<" b="<<b;
} ;
class B : public A
{
Public :
int c,d;
void get()
{ cout <<"Enter c & d";
cin >>c;
cin>>d;
}
Void display ()
{
```

```
display ();
{ cout <<" c = "<<c;
cout <<" d = "<<d;
}
};
void main ()
{ B b1;
```

= Inline func. on the last page

```
b1.a =5;
b1.b = 6
b1.c = 7
b1.d = 8;
b1.get ();
} b1.disp();
```

Constructor and destructor in inheritance →

1) Constructors are executed in ordered of derivation but destructor will be executed in reverse order.

Program (1)

```
class A
{
public:
A()
{
cout << "A";
}
~A()
{
cout << "~A";
}
};
class B : public A
{
public:
B()
{
cout << "\nB";
}
~B()
{
cout << "~B";
}
};
```

```
void main()
{
B b;
}
```

output
A
B
B
A

2) If base class has default constructor then there may be or not a default constructor for derived class.

2) If base class has parameterized constructor then there must have a constructor to pass argument in derived class. E.g. we create object of derived class only in inheritance.

3) Syntax

derived class name (argument) : base class name (argument)

Program 9

```
class A
{
protected:
 int i;
public:
A (int m)
{
i = m;
}
};
class B : public A
{
int n;
public:
B (int p, int q)
                : A(p)
{
n = q;
}
void display ()
{
cout << n << i;
}};
void main ()
{
B b (3, 10);
b. display ();
}
```

3) If base class constructor and derived class has not have, even then derived class has to declared one for base class

eg

```
class B: public A
{
public:
  B(int n) : A(n)
  {
  }
  void display()
```

Ambiguity in Multiple Inheritance

Prog - 3

```
class A
{
public:
int i;
void display() —①
{
cout << "i = " << i;
}};
```

```
class B
{
public;
int k, j;
void display() —②
{
cout << k << j;
}};
class C : public A, public B
{ public:
void print()
{
cout << i << j; }};
cout << k;
void main()
{
C c;
c.print();
c.display();
```

derived class obj . base class :: function()

```
void main()
{
C c;
c.A:: display();
c.B :: display();
c.print();
}
```

If derived class and base class has same var. name then derived class var. hide base class var.

eg :-

Prog 4.

```
class A
{
public:
int i;
 void disp()
{
cout<<i;
};
class B : public A
{public:
int i;
public:
void display()
{ cout<<i;
};
void main()
{
B b;
};
```

Syntax

```
b.i=5;
b.A::i=9;
```

```
b.disp();
b.display();
```

7) If both the class has constructor which has same variable.

Syntax

A::i = 9;

Polymorphism (Single name multiple forms)

1) The word polymorphism means having many forms it is defined as the ability of a msg to be displayed in more than 1 form.

2) Polymorphism is mainly divided into 2 types
1) Compile time polymorphism.
2) Run time polymorphism.

3) Compile time polymorphism :- this type of polymorphism is achieved by func overloading or operator overloading.

4) Run time polymorphism :- this type of polymorphism is achieved by func. overriding

5) ~~Function~~ overloading :- when there are multiple func with same name but diff. parameters then these func are said to be overloaded. functions can be overloaded by change in no. of arguments or/And types of arguments.

prog 5

```
class G1:
{
public:
void func (int x)
{
cout << "Value of x is" << x << endl;
}
void func(double x)
{
cout << "Value of x is " << x << endl;
}
void func (int x, int y)
{
cout << "Value of x" << x << "y = " << y << endl;
};
void main()
{
G1 g;
g.func(7);
g.func(9.132);
g.func(85, 64);
}
```

6) C++ also provide option to overload operators
for eg:- we can make the operator (+) for a string class to concatenate 2 strings. We know that this is a add. operator to add 2 operands so single operator (+) when placed b/w int operands, adds them when place b/w string it concantenate.

• Prog – 6 •

Syntax

classname operator operator used (classname keyword const & obj);

```
class complex
{public: private:
int real, image;
public:
complex (int r, int i)
 {real = r;
  image = i;
 };
complex operator +(complex const& obj)
{
Complex res;(4,5);
res.real = real + obj.real;
res.image = image + obj.image;
return res;
}
void print()
{
cout << real << "+i" << image << endl;
};

void main()
{
complex c1(10,5);
complex c2(2,4);
complex c3 = c1 + c2; or, c3 = c1.+(c2)
c3.print(); }
```

Overriding

```
class shape
{
public:
int width, height;
shape (int a, int b)
{
width = a;
height = b;
}virtual
int area ()
{
Cout << " parent class area" <<endl;
return 0;
}
}
class rectangle : public shape
{
Public :
Rectangle (int a, int b):shape (a,b)
{ }
int area ()
{
Cout << "Rectangle class area" <<endl;
return (width * height);
};
```

```
Class Triangle: public
 shape
{
Public:
Triangle (int a, int b):
{ } shape(a,b)
int area()
{
Cout <<" Triangle class area"
<<endl;
return (width * height/2);
};
int main ()
{
Shape * S;
rectangle rec(10,7);
Triangle tri(10,5);
S = &rec;
S → area ();
S = &tri;
S → area ();
return 0;
}
```

```
#int *p;
 int x =5;
  p = &x;
```

cout << p;
cout << p;
cout << *p;

output
200 5 100

p	x
200	5
100	200

output
parent class area.
parent class area.

The reason for incorrect output is that the call of the func. area is being set once while a compiler as a version defined in the base class. This is ~~clas~~ called static resolution of the func. call or static linkage. The func. call is fixed before the program is executed. This is also called early binding bez. the area func. is said during the compilation of the program.

Lets make a slite modification in our program and precced the definition in (in front of) Shape. same class. with the key word virtual so that it execute the correct output.

» Virtual func. is a func. in a base class class that is declared using the keyword virtual. A ~~virtual~~ Defined in a base class a virtual func. with another func. in a derived class singnals to the compiler that we don't want static linkage for this func.

What we do want is the selection of the fun. to be called at any given point in the program to be based on the kind of object for which it is called this type of operation is referred as dynamic linkage or late binding.

Pure virtual function.

It is possible that you want to include a virtual function in a base class so that it may be redefined in a derived class to suit the objects of that class but there is no meaningful definition you could give for the fun. in the base class.

Eg In (1) → Program.

```
class shape
{
  virtual int area() = 0;
};
```

We can change the virtual func. area in the base class to the following. The "=0" tells the compiler that the func. has no body and above virtual function will be called pure virtual func.

9/5/17

C++

Operator overloading is an important concept in C++. It is a type of polymorphism. In which an operator is overloaded to give user-defined meaning to it. Overloaded operator is used to perform operation on user defined datatype. Almost any operator can be overloaded in C++ however. There are few operators which can't be overloaded. Operators that are not overloaded are as follows:-

1) :: - scope resolution
2) sizeof()
3) . (dot operator) - member selector
4) * - star operator.
5) ternary operator

Operator Overloading Syntax :- Returntype

```
Returntype classname::operator operator symbol(argument list)
{
    function body
}
```

(operator - reserved keyword; operator to be overloaded.)

→ Operator overloading can be done by implementing a func. which can be member func., non-member func., friend func.

→ Operator overloading func. can be a member func. if the left operand is an object of that class, but if the left operand is diff. then operator overloading func. must be a non member func.

→ Operator overloading func. can be made friend func. if it needs access to the private & protected member of class.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Restriction On Operators Overloading

1) Following are some restrictions while implementing operators overloading :-
2) Precedence & associativity of an operator can't be change.
3) Arity (No. of operands) can't be change it means unary operators remains unary & binary operators remain binary. No new operator can be created, only existing operators can be overloaded.
4) Can't predefined the meaning of a procedure, you can't change how Integers are added.

23/oct

Ch-15 → Exception Handling

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstances that arises while a program is running, Such as an attempt to divide by 0. Exceptions provide a way to transfer control from one part of a program to another in C++ exception handling is built upon 3 keywords try, catch & throw.

→ Try

A try block identifies a block of code for which particular exception will be achieved. It is followed by one or ~~match~~ more catch block.

→ Catch

A program catches an exception with an exception handles at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.

→ Throw

A program throws an exception when a problem generates. This is done using a throw keyword

Exception - 1

```
#include <iostream.h>
#include <conio.h>
void main()
{
int a,b,x;
cout<<"enter a &b";
cin >>a>>b;
x = a-b;
try
{
 if (x!=0)
}}
cout <<"result:"<<(a/a-b);
}
else
{
throw (x); }}
catch (int i)
{
cout <<"divide by zero";
}
getch();}
```

Output

```
Enter a &b 4
5
result : -4
```

⇒ Multiple — Exception - 2

```
#include <iostream.h>
#include <conio.h>
void test (int x)
{
try
```

```
}
if (x>0)
{
throw x;
}
else
{
throw 'x'; }}
catch (int no.)
{
cout <<"caught an integer "<<no.;
}
catch (char ch)
{
cout <<"caught a character"<<ch;
}
cout <<"next statements"; }
int main ()
{
cout <<"multiple catch statements";
cout <<"x is greater than 0";
test (5);
cout <<"x is less than 0";
test (-5);
getch ();
return 0;
}
```

→ There is a special catch block called catch all (catch .....) that can be used to catch all types of exception.

```
⇒int main()
 {
 try
 {
 throw 10; }
 catch (char * c)
 {
 cout << "caught is char" << e;
 }
 catch (....)
 {
 cout << "default exception \n";
 }
 return 0; }
```

→ Implicit type conversion

Implicit type conversion does not happen for primitive type. In the following program 'a' is not implicitly converted to int.

```
⇒int main()
 {
 try
 {
 throw 'a';
 }
```

```
catch (int x)
{
cout << "caught integer" << x;
}
catch (.....)
{
cout << "default exception"; } return 0; }
```

⇒ In C++, try-catch block can be nested Also an exception can be rethrown using throw.

```
int main()
{
try
{
try
{
throw 20;
}
catch (int x)
{
cout << "handle partially"; }
throw; // rethrowing an exception }}
catch (int x)
{
cout << "handle remaining";
}
return 0;
}
```

→ A function can also rethrow a func. using same "throw" (keyword). A function can handle a part and can ask the caller to handle remaining.

→ When an exception is thrown all objects created inside the enclosing try block are destructed before the control is transferred to catch block.

```
class test
{
public:
test()
{
cout << "constructor of test" << endl;
}
~test()
{
cout << "destructor of test" << endl;
}};
int main()
{
try
{
test t;
throw 10;
}
catch(int i)
{
cout << "caught :" << i << endl; } }
```

→ Operator overloading
Assignment operator

```
#include <iostream.h>
class distance.
{
private: ~~public~~ : private:
 int feet;
 int inches;
public:
distance ()
{
feet = 0;
inches = 0;
}
distance (int f, int i)
{
feet = f;
inches = i;
}
Void operator = (const distance & D)
{
feet = D. feet;
inches = D. inches;
}
void display ()
{
```

```
cout << "F:" << feet << " i:" << inches << endl;
};
int main()
{
distance D1(11,10), D2(5,11);
cout << "first distance " << D1.display();
cout << "second distance " << D2.display();
D1 = D2;
cout << "first distance is the assignment";
D1.display();
return 0;
}
```

ex:

```
class check
{
private:
 int i;
 public:
 check()
 {
  i=0;
 }
check operator ++()
{
check temp;
++i;
temp.i = i;
return temp;
}
void display()
{
cout << "i= " << i << endl;
}
};
int main()
{
check obj, obj1
obj.display();
obj1.display();
obj1 = ++obj;
obj.display();
obj1.display();
getch();
}
```

Check post increment

⇒ all above is same from previous program.

```
check operator ++ (int)
{
check temp;
i++;
temp.i = i;        or temp.i = i++;
return temp;
}
```

~~obj1 = obj++;~~

```
void display ()
{
cout << "i=" << i << endl;
};
int main ()
{
chck obj, obj1;
obj.display ();
obj1.display ();
obj1 = obj++;
obj.display ();
obj1.display ();
return 0;
```

obj1.display ();

When increment operator is overloaded in prefix form then operator ++() is called. but when increment operator is overloaded in postfix form then operator ++(int) is called. The int inside this bracket gives information to the compiler that it is the post fix version of operator.

Do not confuse this int indicate integer.

In decrement just change the sign - operator, < operator

=> class distance.

```
{
private:
int feet;
int inches;
public:
distance()
{
feet = 0;
inches = 0;
}
distance(int f, int i)
{
feet = f;
inches = i;
}
void display()
{
cout << "F = " << feet << "I"
<< inches << endl;
}
Distance operator - ()
{
feet = - feet ;
inches = - inches ;
}
bool operator < (const distance &d)
{
if (feet < d.feet)
```

```
;
return true;
}
if (feet = d.feet && inches < d.inches)
{
return true;
} else
return false;
};
int main()
{
distance D1(11,10), D2(5,11);
if (D1 < D2)
{
cout << "D1 is less than D2" <<endl;
}
else
{
cout << "D2 is less than D1" <<endl;
}
return 0;
}
```

Output D1 is less than D2

Catching Base & Derived Classes as Exception

1) If both base & derived class are count as exception then catch block of derived class must appear before the base class.

2) If we put base class first then the derived class can catch block will never be reached.

```
class Base
{
};
class derived: public Base
{
};
int main()
{
derived d;
try
{
throw d;
}
catch(base b)
{cout << "caught base exception";
}
catch(derived d)
{
cout << "caught derived exception";
}
}
```

1) The following code cause base exception

2) If we change the order of catch statements. then both catch statements become reachable.

3) first write the prog in base derived } order

4) then derived base } order

```
catch (derived d)
{cout << " caught derived }
```

```
catch (Base b)
{
cout<< "caught base exception";
}
getch();
return 0;
}
```

Inline functions

functions
- function declaration
- Calling
- definition.

```
void add (int, int);
void main ()
{
int a, b;
cin >> a>>b;
add (a,b);
}
inline void add (int a, int b)
{
int c = a+b;
cout c;
}
```

Macro

```
#include <iostream.h>
#include <conio.h>
#define SQUARE(x) x*x
inline float square(float I)
{
return (I*I);
}
void main()
{
clrscr();
int p=3, q=3, r, s;
r = SQUARE(++p);
s = SQUARE(++q);
cout << "r = " << r;
cout << "\ns = " << s;
getch();
}
```

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well