Key Points on OOPs

OOPs is one of the most used lang in game development.

```
#include <iostream>
using namespace std;
int main() {
cout << "Hello world!";
return 0;
}
```

O/P – Hello

C++ gives programmers a high level of control over system resources and memory.

The language was updated 4 major times in 2011, 2014, 2017 and 2020 to c++11, C++14, C++17, C++20.

Why Use C++?

Most Popular Programming Lang.

C++ can found in today's operating systems, Graphical User interfaces and embedded systems.

C++ gives a clear structure to Programs and allow code to be reused, lowering development cost

Portable

C++ is fun & easy to learn.

C++ is close to C, C#, and Java.

Portable & can be used to develop apps that can be adapted to multiple platforms

C++ Get Started

To start using C++ you need two things.

1. Text editor like Notepad to write C++ code.
2. A compiler like GCC to translate the C++ code into a language that the computer will understand.

C++ with IDE :-

An IDE is used to edit and compile the code.
Popular IDE's include Code::Blocks, Eclipse and Visual Studio.
These are all free and they can be used to both edit and debug C++ code.

Note: web based IDE's can work as well, but functionality is limited.

⇒ C++ ignore blank line. (white space) But we use it to make the code more readable.

⇒ cout is a object used together with the insertion operator to output / print text.

⇒ using namespace std :- (omitting) line can be omitted and replaced with the std keyword, followed by the :: operator for some objects.

⇒ using namespace std ;- It means that we can use names for objects and variables from the standard library.

```
#include <iostream>
int main(){
  std::cout << "Hello world!";
  return 0;}
```

o/p hello world

You can add as many cout objects as you want. However note that it does not insert a new line at the end of the output

```
#include <iostream>
using namespace std;
int main() {
cout<<"Hello world!";
cout<<"I am learning c++";
return 0;
}
```

O/P Hello world! I am learning c++.

New line :- \n

O/P

Hello world!
I am learning c++.

```
#include <iostream>
using namespace std;
int main() {
cout<<"Hello world!\n";
cout<<"I am learning c++";
return 0;
}
```

Note: Two \n character after each other will create a blank line.

O/P

Hello world!

I am learning c++.

```
#include <iostream>
using namespace std;
int main(){
cout<<"Hello world!\n\n";
cout<<"I am learning c++";
return 0;
}
```

Another way to insert a new line, is with the endl manipulator

Example:-

```
#include<iostream>
using namespace std;
int main() {
cout<<"Hello world!"<<endl;
cout<<"I am learning c++";
return 0;
}
```

O/P Hello world!
I am learning c++.

Both \n and endl are used to break lines. However \n is most used.

But what is \n exactly?

The newline character (\n) is called an escape sequence, and it forces the cursor to change its position to the beginning of the next line on the screen. This result is a new line.

Horizontal Tab \t

```
#include<iostream>
using namespace std;
int main() {
cout<<"Hello world!\t";
cout<<"I am learning c++";
return 0;
}
```

Hello world! I am learning c++.

Program

```
#include<iostream>
#include<string>
using namespace std;
int main() {
int myNum=5;
float myFloatNum=5.99;
double myDoubleNumber=9.98;
Char myletter='D';
bool myBoolean=true;
String myString="Hello";
Cout<<"int:"<<myNum<<"\n";
Cout<<"float:"<<myFloatNum<<"\n";
Cout<<"double:"<<myDoubleNum<<"\n";
Cout<<"char:"<<myletter<<"\n";
Cout<<"bool:"<<myBoolean<<"\n";
cout<<"string:"<<myString<<"\n";
return 0;
}
```

OIP

```
int: 5
float: 5.99
double: 9.98
char: D
bool: 1
String: Hello.
```

Basic data types:

boolean - 1 byte → Store true or false value

Char - 1 byte → Store a single character / letter / number, or ASCII value.

Int → 2 or 4 byte - Store whole numbers, without decimals

float → 4 byte → Store fractional no, containing one or more decimal.
Sufficient for storing 6-7 decimal digits

double → 8 byte → Store fractional no, containing one or more decimals.
Sufficient for storing 15 decimal digits

Numeric Types :-

int → what no without decimal like 35 or 1000
float → when you need a floating point no. (with decimals) like 9.99 or 3.14515.

int :-

```
int myNum = 1000;
Cout << myNum;
```

```
#include <iostream>
using namespace std;
int main() {
int myNum = 1000;
Cout << myNum;
return 0;
}
```

O/P → 1000

float :-

```
float myNum = 5.75;
Cout << myNum;
```

```
#include<iostream>
using namespace std;
int main() {
float myNum = 5.75;
Cout << myNum;
return 0;
}
```

O/P - 5.75

```
#include <iostream>
using namespace std;
int main() {
  int x, y;
 int sum;
 Cout << "Type a number: ";
  Cin >> x;
 Cout << "Type another number: ";
  Cin >> y;

  sum = x + y;
  Cout << "sum is: " << sum;
   return 0;
   }
```

O/P

Type a number - 2
Type another no - 6
sum - 8

```
int x;
Cout << "Type a number: ";
____ >> ___ ;
```

C++ data types :

A variable in c++ must be a specified data type.

Ex
```
int myNum = 5;  // Integer (whole number)
Float myFloatNum = 5.99; // Floating point number
double myDoubleNum = 9.88; Floating Point no.
char myLetter = 'D'; // Character
bool myBoolean = true; // Boolean
String myText = "Hello"; // String
```

Here the user can input a number, which is stored in the variable x. Then we print the value of x.

Ex)
```
int x;
Cout << " Type a number : "; // Type a no & Press Enter
Cin >> x; // Get user input from the keyboard
Cout << " Your number is : " << x;
                // Display the input value
```

Ex
```
# include <iostream>
using namespace std;
int main() {
  int x;
Cout << " Type a number : "; // Type a no & Press Enter
 Cin >> x;
 Cout << " Your number is : " << x; // Get user input from keyboard
 return 0;
 }
```

<< - Insertion operator

>> - Extraction operator

O/P Type a 5
Your number is : 5

Create a Simple Calculator:

User take 2 input then Print sum by Calculating (Adding)

Ex
```
int x, y ;
int sum;
Cout << " Type a number : ";
  Cin >> x;
Cout << " Type another number : ";
 Cin >> y;
  Sum = x + y;
  Cout << " Sum is : " << Sum;
```

You can also declare a variable without assigning the value and assign the value later

Ex
```
int myNum;
myNum = 15;
cout << myNum;
```

```
#include <iostream>
using namespace std;
int main() {
int myNum;
myNum = 15;
cout << myNum;
return 0;
}
```

O/P
15

NOTE: If you assign a new value to an existing variable it will overwrite the previous value.

Ex
```
int myNum = 15; // myNum is 15.
myNum = 10;     // Now myNum is 10.
cout << myNum; // output 10.
```

O/P → 10

Other types :- A demonstration of other data type.

Ex
```
int myNum = 5;              // Integer (whole no. without decimal.
double myFloatNum = 5.99;   // float point no with decimal
char myLetter = 'D';        // character
string myText = "Hello";    // string (text)
bool myBoolean = true;      // Boolean (true or false)
```

Display variables :-

The cout object is used together with the << operator to display variables.

To combine both text and a variable, separate them with the << operator

Ex) int myAge = 35;
cout << "I am" << myAge << "years old.";

```
#include <iostream>
using namespace std;
int main() {
int myAge = 35;
cout << "I am" << myAge << "years old.";
return 0;
}
```

O/P
I am 35 years old.

Add variables Together :-

To add a variable to another variable, you can use the + operator.

Ex-
```
int x = 5;
int y = 6;
int sum = x + y;
cout << sum;
```

```
#include <iostream>
using namespace std;
int main() {
int x = 5;
int y = 6;
int sum = x + y;
cout << sum;
return 0;
}
```

O/P
11

C++ Declare Multible Variables :-

Declare Many Variable :- To declare more than one variable of the same type, use a comma-separated list.

Ex
```
int x = 5, y = 6, z = 50;
cout << x + y + z;
```

```
#include <iostream>
using namespace std;
int main() {
int x = 5, y = 6, z = 50;
cout << x + y + z;
return 0;
}
```

O/P - 61

Ex: `Cout << "Hello World!"; // This is a Comment.` → for Short Comments

C++ Multiline Comments:- `/* */` → for long Comments

```
/* The code will Print the words Hello World! is amazing.
Cout << "Hello World!";
```

```
#include <iostream>
using namespace std;
int main() {
/* this is amazing */
Cout << "Hello World!";
return 0;
}
```

Output

Hello World!

Variables:- char → store single characters

Syntax:-

```
type VariableName = Value;
```

- type (int) ← Type
- VariableName (x or myName) ← Name of variable
- = ← Equal sign is used to assign values to the variable.

Ex1.

Create a variable called myNum of type int and assign it the value 15:

```
int myNum = 15;
Cout << myNum;
```

```
#include <iostream>
using namespace std;
int main() {
int myNum = 15;
Cout << myNum;
return 0;
}
```

O/P → 15

Escape Sequence (\t), \\, \"

\\ → Insert a backslash character (\):-

```
#include <iostream>
using namespace std;
int main() {
cout << "Hello world!\\I am learning c++";
return 0;
}
```

O/P

Hello world!\I am learning c++

\" → Insert a double quote character

```
#include <iostream>
using namespace std;
int main() {
cout << "They call him \"Johnny\".";
return 0;
}
```

O/P

They call him "Johnny".

c++ Comments:- To make code more readable.
It can also be used to prevent execution when testing alternative code.
Comments can be singled-lined or multi-lined.

Single line comments:- Comments start with 2 forward slashes (//)

Any text b/w // and the end of the line is ignored by the compiler (will not be executed)

```
// This is a comment
cout << "Hello world!";
```

O/P Hello world!

return 0;
}

One Value to Multible Variables :-

You can also assign the same value to multible variables in one line:

Example:

```
int x, y, z;
x = y = z = 50;
cout << x + y + z;
```

```
#include <iostream>
using namespace std;
int main () {
int x, y, z;
x = y = z = 50;
cout << x + y + z;
return 0;
}
```

O/P -150

C++ Identifiers :-

All C++ variables must be identified with unique names.
These Unique names are called Identifiers.
Identifiers can be short names (like x and y) or more descriptive name (age, sum, total volume)

NOTE: It is recommended to use descriptive names in order to create understandable & maintainable code.

Ex
```
//Good
int minutesPerHour = 60;
// ok, but not so easy to understand what am actually in
int m = 60;
```

Ex
```
#include <iostream>
using namespace std;
int main () {
int minutesPerHour = 60;   // Good name
int m = 60;                // ok but not so easy to understand.
cout << minutesPerHour << "\n";
cout << m;
return 0;
}
```

O/P 60 60

120000

Constants :-

When you do not want others to change existing variable values, use the const keyword (this will declare the variable as "constant", which means unchangeable and read only):

Ex1

```
Const int myNum = 15; //myNum will always be 15.
myNum = 10;           // Error: assignment of read-only variable 'myNum'
```

```
#include <iostream>
using namespace std;
int main() {
Const int myNum=15;
myNum=10;
Cout << myNum;
return 0;
}
```

In function 'int main()';
6.9: error assignment of read only variable 'myNum'
6 | myNum = 10;

Ex: Const int (Since) minutesPerHour = 60;
Const float PI = 3.14;

```
#include <iostream>
using namespace std;
int main() {
Const int minutesPerHour=60;
Cout << minutesPerHour << "\n";
Cout << PI;
return 0;
}
```

O/P → 60
3.14

C++ User Input :-

Cout → is used to output (Print) values.
Cin → to get user input.
Predefined variable that reads data from the keyboard with the extraction operator (>>).

double:- double myNum = 19.99;
cout << myNum;

```
#include <iostream>
using namespace std;
cout<<"Time taken to run this program is :- "<<(t2-
t1)/CLOCKS_PER_SEC<<endl;
int main() {
double myNum = 19.99;
Cout<< myNum;
return 0;
}
```

O/P - 19.99

float vs double:-

The Precision of floating point values indicates how many digits the value can have after the decimal point. The Precision of float is only six or seven decimal digits while double variable have a Precision of about 15 digits. Therefore it is safer to use double for most Calculations.

Scientific No:-

A floating point number can also be a scientific number with an "e" to indicate the Power of 10.

Ex
```
float f1 = 35e3;
double d1 = 12E4;
Cout<< f1;
Cout<< d1;
```

```
#include <iostream>
using namespace std;
int main(){
float f1 = 35e3;
double d1 = 12E4;
cout<<f1<<"\n";
cout<<d1;
return 0;
}
```

O/P - 35000
120000

Boolean Types: A Boolean datatype is declared with the bool keyword and can only take the values true or false.

True - 1 false - 0

Ex:
```
#include <iostream>
using namespace std;
int main()
{
bool isCodingFun = true;
bool isFishTasty = false;
cout << isCodingFun; //output 1 (true) << "\n";
cout << isFishTasty; //output 0 (false)
return 0;
}
```

O/P
```
1
0
```

Boolean values are mainly used for conditional testing.

Character data types:

Char → Used to store a single character.
must be surrounded by single quotes like 'A' 'a' 'c'

Ex:-
```
char myGrade = 'B';
cout << myGrade;
```

```
#include <iostream>
using namespace std;
int main() {
char myGrade = 'B';
cout << myGrade;
return 0;
}
```

O/P
B

Alternative you can use ASCII value to display certain characters

Ex:
```
char a=65, b=66, c=67;
cout << a;
cout << b;
cout << c;
```

Ex
```
#include <iostream>
using namespace std;
int main(){
Char a=65, b=66, c=67;
Cout<<a;
Cout<<b;
Cout<<c;
return 0;
}
```

O/P
ABC

String- This type is used to store a sequence of character (text).
This is not a built-in type, but it behaves like one in its most basic usage.

String values must be surrounding by double quotes.
Create a variable of type string & assign it a value.

Ex
```
String greeting = "Hello";
Cout << greeting;
```

To use string you must include <String Library>

Ex
```
# include <string>   //include string library.
string greeting = "Hello"; // Create a string variable
Cout <<greeting;          // output string value
```

```
# include <iostream>
# include <string>
using namespace std;
int main() {
string greeting = "Hello";
Cout <<greeting;
return 0;
}
```

O/P → Hello

Access strings :-
You can access the characters in a string by referring to its index number inside square brackets [].
This prints the first character in mystring.

Ex
```
string myString = "Hello";
cout << myString[0];
// O/P - H
```

Note :- String indexes start with 0; [0] is the first character [1] is the second character etc.

```
string myString = "Hello";
cout << myString[1];
//output - e
```

Change string characters : Refer to the index no. & use single quotes.

Ex
```
string myString = "Hello";
myString[0] = 'J'
cout << myString;   // O/P - Jello instead of Hello
```

String special characters :-
C++ misunderstand this string and generate an error

```
Strintxt = "We are the so called "Vikings" from the north.";
```

Solution to avoid this Problem :-
To use the backslash escape character

\'	'	Single quotes
\"	"	double "
\\	\	Backslash

String Length:

To get the length of a string, use the length() function.

Eg string txt =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";
Cout << "The length of the txt string is :" << txt.length();

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string txt = "A ---- Z";
Cout << "The length of the txt string is :" << txt.length()
                                                    ();
Return 0;
  }
```

O/P The length of the txt string is : 26

NOTE :- You might see some C++ Programs that use the size()
function to get the length of a string.
This is just an alias of length().
You can use which you want

```
String txt = "A ------ Z";
Cout << "The length of the text String is :" << txt.size();
  return 0;
  }
```

O/P The length of the text string is : 26

String Concatenation :-

Operators are used to perform operations on variables & two values.

+ → operator to add together two values

Ex int x = 100 + 50;

```
#include <iostream>
using namespace std;
int main(){
int x = 100 + 50;
cout << x;
return 0;
}
```

O/P
150

Ex int sum1 = 100+50; // 150 (100+50)
int sum2 = sum1+250; // 400 (150+250)
int sum3 = sum2+sum2; // 800 (400+400)

Program

```
# include <iostream>
using namespace std;
int main () {
int sum1 = 100+50;
int sum2 = sum1 +250;
int sum 3 = sum2 + sum 2
cout << sum1 << "\n";
cout << sum2 << "\n";
cout << sum3;
return 0;
}
```

O/P 150
400
800

① Arithmetic operators :-

```
# include <iostream>
using namespace std;
int main() {
int x = 5;
int y = 3;
cout << x + y;
return 0;
}
```

Op-8

Subtraction :-

```
int x=5;
int y=3;
Cout<<x-y;
return 0;
}
```

2

Multiply

```
int x=5;
int y=3;
Cout<<x*y;
return 0;
}
```

15

Div

```
int x=12;
int y=3;
Cout<<x/y;
Return 0;
}
```

4

Modulus

```
int main(){
int x=5;
int y=2;
Cout<<x%y;
return 0;
}
```

1

Increment

```
intmain(){
int x=5;
++x;
Cout<<x;
return 0;
}
```

O/P 6

Decrement

```
int x=5;
--x;
cout<<x;
return 0;
}
```

4

C++ Assignment operators:

Assignment operators are used to assign values to variable.

= to assign value.

```
int x=10;
Cout<<x;
return 0;
}
```

O/P 10

Additional assignment oper +=

Ex
```
int x=10;
x+=5;
```

```
int main(){
int x=10;
x+=5;
Cout<<x;
return 0;
}
```

15

Multiply -

```
int x=5;
x*=3;
Cout<<x;
```

(15)

decrement oprtr

```
int x=8;
x-=3;
Cout<<x;
return 0;
```

(2)

```
int ...
x=5;
return 0;
```

Division Assignment

```
int main(){
double x=5;
x /=3;
Cout<<x;
return 0;
```

o/p → 1.66667

Modulus

```
int main(){
int x=5;
x%=3;
Cout<<x;
return 0;
}
```

(2)

&=

```
int main(){
int x=5;
x&=3
Cout<<x;
return 0;
}
```

o/p → 1

|=

x|=3

```
int main(){
int x=5;
x=3;
Cout<<x;
return 0;
}
```

o/p → 7

^=

x^=3

```
int main(){
int x=5;
x^=3;
Cout<<x;
return 0;
}
```

(6)

>>=

x>>=3

```
int main(){
int x=5;
x>>=3;
Cout<<x;
return 0;
}
```

o/p

<<=

x<<=3

```
int x=5;
x<<=3;
Cout<<x;
return 0;
}
```

40

C++ Comparison operators

Comparion operators are used to compare 2 values or (variable). This is important in Programming b/c it helps us to find ans & make decisions.

The return value of Comparion is either 1 & 0, which mean true (1) or false (0). These values are known as Boolen values. ~~& you will~~

Ex
```
int x=5;
int y=3;
Cout<<(x>y); // returns 1 (true) because 5 is greater than 3.
```

Ex String firstName = "John";
String lastName = "Doe";
String fullName = firstName + lastName;
Cout << fullName;

Program
```
#include <iostream>
#include <string>
using namespace std;
int main() {
string firstName = "John";
string lastName = "Doe";
string fullName = firstName + lastName;
Cout << fullName;
return 0;
}
```
O/P – JohnDoe (John Doe)

NOTE - Here we added a space after firstName to create a space b/w John and Doe on output. However you could also add a space with quotes (" " or ' ');

Examples
```
String firstName = "John";
String lastName = "Doe";
String fullName = firstName + " " + lastName;
Cout << fullName;
```
" " or ' '
+

Program
```
#include <iostream>
#include <string>
using namespace std;
int main() {
string firstName = "John";
string lastName = "Doe";
string fullName = firstName + " " + lastName;
Cout << fullName;
return 0;
}
```
O/P → John Doe

Append:- A string in C++ is actually an object which contains functions that can perform certain operations on string. For example you can also concatenate strings with the append() function.

Ex
```
string firstName = "John";
string lastName = "Doe";
string fullName = firstName.append(lastName);
cout << fullName;
```

```
#include<iostream>
#include<string>
using namespace std;
int main() {
string firstName = "John";
string lastName = "Doe";
string fullName = firstName.append(lastName);
cout << fullName;
return 0;
}
```

John Doe

Note:- C++ uses the + operator for both addition and concatenation.

No are added. Strings are concatenated.

Ex
```
int x = 10;
int y = 20;
int z = x + y; // z will be 30 (an integer)
```

! Logical not :- Reverse the result, returns false if the result is true

```
! (x<5
 && x < 10));
```

```
#include<iostream>
using namespace std;
int main() {
int x = 5;
int y = 3;
Cout << (! (x>3 && x<10));
return 0;
}
```

O/P → 0

Return false (0) because ! (not) is used to reverse the result o/p.

C++ Strings :- String are used for storing text.
A String variable consist a collection of characters surrounded by double quotes.

E.g) string greeting = "Hello"; Used - Strig library.

```
#include<iostream>
#include<string>
using namespace std;
int main() {
string greeting = "Hello";
Cout << greeting;
return 0;
}
```

O/P → Hello

String Concatenation :- The + operator can be used between strig to add them together to make a new String. This is called Concatenation.

Logical operators:

True - 1
false - 0

Operator Logical and

Return true if both statements are true.

x<5 && x<10

```
#include<iostream>
using namespace std;
int main(){
int x=5;
int y=3;
cout<<(x>3 && x<10); // Return true
return 0;
}
```

∵ 5 is greater than 3 and 5 is less than 10

T _ T → T (1)

O/P → 1

Logical OR :- Return true if one of the statement is true.

x<5 || x<4

```
#include<iostream>
using namespace std;
int main(){
int x=5;
int y=3;
cout<<(x>3 || x<4); // Return true b/c one of the condition are true (5 is greater than 3 but 5 is not less than 4
return 0;
}
```

O/P → 1

Operators :-

== equal to x==2y

```
int main(){
int x = 5;
int y = 3;
Cout << (x==2y); // Return 0 bcs 5 is not equal to 1
return 0;
}
```

O/P → 0

Not equal to != :-

```
int main(){
int x = 5;
int y = 3;
Cout << (x != y);
return 0;
}
```

Return 1 true

O/P → 1

> greater than

```
int main(){
int x = 5;
int y = 3;
Cout << (x > y);
return 0;
}
```

Return 1 true

O/P 1

< Less than

```
int main(){
int x = 5;
int y = 3;
Cout << (x < y);
return 0;
}
```

Return 0 false

O/P → 0

>= greater than equal to

```
int main(){
int x = 5;
int y = 3;
Cout << (x >= y);
return 0;
}
```

Return 1 true

O/P → 1

<= Less than equal to

```
int main(){
int x = 5;
int y = 3;
Cout << (x <= y);
return 0;
}
```

Return 0 → false

O/P - 0

Break and Continue :-

Break:- It was used to "jump out" of a switch statement.
Can also be used to jump out of a loop.

Ex
```
for (int i=0; i<10; i++) {
    if (i==4) {
    break;
    }
    cout<<i<<"\n";
}
```

O/P
0
1
2
3

Continue:- The continue statement breaks one iteration (in the loop) if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 4:

Ex
```
int main() {
for (int i=0; i<10; i++) {
    if (i==4) {
        continue;
    }
    cout<<i<<"\n";
}
return 0;
}
```

O/P:-
0
1
2
3
4
5
6
7
8
9

Break and Continue in while loop :-

```
int i=0;
while (i<10) {
cout<<i<<"\n";
i++;
if (i==4) {
break;
}
}
```

O/P
0
1
2
3

Continue Example

```
int i=0;
while (i<10) {
if (i==4) {
i++;
continue;
}
cout<<i<<"\n";
i++;
}
```

OP
0
1
2
3
5
6
7
8
9

```
demo temp(0,0); //Parameterize Cons (value Pass)
temp.a = a+obj.a; //value stored in temp
temp.b = b+obj.b;
return temp;
}
};
void main()
{
clrscr();
demo ob(10,20), ob1(30,40), ob2(0,0); //Cons Parameterized
ob2 = ob+ob1; //Add value
ob2.show(); //show fun
getch();
}
```

You can write void return type but here we write class type i.e. demo

implicit Pass value through this Pointer
Explicitly value Pass through object →

O/P

A=40 B=60

Value Passing Result

10 30 20 40
40 60

demo
demo
for default Constructor

```
#inc
#include<conio.h>
class demo
{
int a,b;
public:
demo() // default Constructor
{
}
demo(int x, int y)
{ a=x;
b=y;
}
```

```
void show()
 {
  cout<<"A"<<a<<" "<<"B"<<b<<endl;
 }
demo operator+(demo obj)
 {
 demo temp;
 temp.a = a+obj.a;
 temp.b = b+obj.b;
  return temp;
 }
);
   void main()
  {
   clrscr();
   demo ob(10,20), ob1(30,40), ob2;
    obj = ob+ob1;                    A   B
    ob2.show();                      40  60
    getch();
    }

void show()
  } cout <<
  )
 friend demo operator+(demo &obj, demo &obj2);
key  };
 demo operator+(demo &obj, demo &obj2)
    demo temp; //Clr obj
    temp.a = obj.a+obj2.a;
a'A value she in temp  temp.b = obj.b+obj2.b;
    return temp;
    3
```

friend fun (outside of the clss

clss

object (fun) + → operator

both value explicitly Pass in friend fun.) So 2 reference Pass

```
void main()
{
clrscr();
demo ob(10,20); demo ob1(30,40); demo ob2;
ob2 = ob+ob1;
ob2.show();
getch();
}
```

Make obj & Pass value.

A=40 B=60

This keyword :- Used when instance variable & local variables both are same.

Avoid Compiler Confusion.

```
#include <iostream.h>
#include <conio.h>
class A
{
int a,b;  // Instance variable (variable of a class)
public:
A(int x, int y)  // Parameterized Constr.
{ a=x; b=y;
}
void show()
{
cout<<a<<" "<<b<<endl;
}
};
void main()
{
}
```

type of

default sec. of class & directly access the obj

Naming Collision Compiler Confuse

```
A(int a, int b)
{ a=a; b=b;
}
```

```
int main() {
for (int i=0 ; i<=10; i=i+2){
  Cout<<i<<"\n";
}
  return 0;
}
```

O/P
0
2
4
6
8
10

Nested Loops:- It is also Possible to place a loop inside another loop. This is called a nested loop.

The inner loop will be executed one time for each iteration of the outer loop.

Ex
```
// Outer loop
int main() {
  for (int i=1; i<=2; ++i) {
    Cout << "Outer: " << i << "\n";
    // executes 2 times.
    // Inner loop
    for (int j=1; j<=3; ++j) {
      Cout << " Inner: " << j << "\n"; // execute 6 times (2*3)
    }
  }
```

O/P
Outer: 1
Inner: 1
" 2
" 3
Outer: 2
Inner: 1
" 2
" 3

For each loop:- There is also a "for-each loop" which is used exclusively to loop through elements in an array.

Syntax:-
```
for (type variableName : arrayName) {
  // Code block to be executed
}
```

NOTE:- output all element in an array, using a "for each loop",
```
int myNumbers[5] = {10, 20, 30, 40, 50};
for (int i : myNumbers) {
  Cout << i << "\n";
```

O/P-
10
20
30
40
50

Syntax :-

```
while (Condition) {
  // Code to be executed
}
```

```
int i=0;
while (i<5) {
cout << i << "\n";
i++;
}
```

O/P → 0 1 2 3 4

Note:- Do not forget to increase the variable used in condition, otherwise the loop will never end!.

C++ Do/While Loop :-

do/while loop is a variant of the while loop. This loop will execute the code block once before checking if the condition is true then it will repeat the loop as long as the condition is true

Syntax:

```
do {
// Code to be executed
}
while (Condition);
```

```
int i=0;
do {
cout << i << "\n";
i++;
} while (i<5);
```

O/P → 0 1 2 3 4

For loop :-

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop.

Syntax:-

```
for (Statement 1; Statement 2; Statement 3) {
  // Code to be executed.
}
```

St-1 → St1 is executed (One time) before the execution of the code block.

St-2 → Define the condition for executing the code block

St-3 → is executed (Every time) after the code block has been executed

Ex

```
for (int i=0; i<5; i++) {
  cout << i << "\n";
}
```

O/P → 0 1 2 3 4

Syntax :-

Short hand if-else (Ternary operator) :-

It consists of 3 operands. It can be used to replace multiple lines of code with a single line.
It is often used to replace if else statement.

Syntax

```
variable = (condition) ?
expTrue : expfalse;
```

Switch :-

```
switch (exp) {
  case x:
   // Code block
  break;
  case y;
   // Code block
  break;
  default:
  // Code block
  }
```

Default keyword :- default
It specify some code to run if there is no case match.

Break keyword :-
Break out of the switch block.
when the match is found, and the job is done, its time for a break. There is no need for more testing.

C++ loop :- Loop can execute a block of code as long as a specified condition is reached.

Loop :- Save time, Reduce error, Make code more readable

C++ while Loop :-
loops through a block of code as long as a specified cond is true

```
while (condition) {
  // Code execute
  }
```

C++ Conditions and If statements

C++ supports the usual logical conditions from the mathematics

less than $a < b$

less than or equal to: $a <= b$

greater than: $a > b$

greater than or equal to: $a >= b$

Equal to $a == b$

Not equal to: $a != b$.

If :- To specify a block of code to be executed if a specified condition is true.

else :- To specify a block of code to be executed, if the same condition is false.

else if :- To specify a new condition to test, if the first cond. is false. (If 2 else if false → else executes)

switch :- To specify many alternative blocks of code to be executed.

if → ✓ If ✗ → Generate an error

① if :- (if condition) {
// code executed
}

② else
If (Cond) {
// Code execute if cond true
} else {
// code executed if the cond is false.

③ else if :-
If (Cond) {
// block of code exec.
} else if (Code
// Code execute if 1st false & Code true.
} else {
// Code execute if cond1 is false and Cond2 is false
}

getline() function to read a line of text. It takes cin as the first Parameter, and the String variable as second

Ex

```
String fullName;
Cout << "Type your full name: ";
getline (cin, fullName);
Cout << " Your name is: " << fullName;
```

C++ Math:

C++ has many fxns that allows you to perform mathematical tasks on numbers.

Max and Min:

The max(x,y) fxn can be used to find the highest value of x & y.

Ex1
```
Cout << max(5,10);
#include <iostream>
using namespace std;
int main() {
Cout << max(5,10);
return 0;
}
```
O/P-10

min(x,y) fxn can be used to find the lowest value of x and
```
Cout << min(5,10);
```
O/P-5

C++ <cmath> header

Other function, such as sqrt (square root), round (round a no.) and log (natural logarithm) can be found in <cmath> header file

Ex2
```
#include <cmath>
Cout << sqrt(64);
Cout << round(2.6);
Cout << log(2);
```
O/P
8
3
0.693147

if cond1 is true and cond2 is false

Ex String txt = "We are the so called \"Vikings\" from the north.";

String txt = "It \'s alright.";
Cout << txt;

\n & tab (

String txt = "hello \n world!!!";
Cout << txt;
Return 0;
}

O/P hello
world!

int main() {
String txt = "hello \t world!!!";
Cout << txt;
return 0;
}

O/P Hello world!

96

User input strings:

extraction operator >> on Cin → to store a string entered by the user.

Ex string firstName;
Cout << "Type your first name: ";
cin >> firstName; // get user input from the keyboard
cout << "Your name is: " << firstName;
// Type your first name: John
// Your name is: John

Cin consider a space (whitespace, tabs etc) as a terminating character which means that it can only store a single word.

NOTE: we expect to print "John Doe" but it only print "John" that's why when working with string, we often use the

0.693147