

Key Points on OOPS

OOPS is one of the most used lang in game dev.

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{cout << "Hello world!" ;}
```

```
return 0;
```

```
}
```

OPP - Hello

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

The language was updated 4 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 be found in today's operating systems, graphical user interfaces and embedded systems.

C++ gives a clear structure to program 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 with 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++ Visual 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++ ignores blank lines (whitespace). But we use it to make the code more readable!

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

⇒ using namespace std; - 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 the 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 start 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

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

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

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

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

O/P
Hello world!

I am learning C++.

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++,

Basic
book
Char

Program

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    int myNum = 5;
    float myFloatNum = 5.99;
    double myDoubleNum = 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;
}
```

O/P

```
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 bytes - stores whole numbers, without decimals.

float → 4 bytes → stores fractional no, containing one or more decimal.

Sufficient for storing 6-7 decimal digits.

double → 8 bytes → stores fractional no, containing one or more decimal.

Sufficient for storing 15 decimal digits.

Numeric Types :-

int → whole no without decimal like 38 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 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;  
}  
  
O/P    Type a no : 5  
         Your number is : 5
```

<< - Insertion
operator
>> - Extraction
operator

Create a Simple Calculator:

User will 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)
 float 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 Multiple Variables:-

Declare Many Variables - 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;`

O/P- 67

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

Ex `Cout << "Hello World!"; // This is a Comment.` → first comment

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

`/* 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      Name of variable
(int)     (x or myName)
```

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), \n, \

\n → Insert a backslash character (\)

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++

" → 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 single-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 Multiple Variables :-

You can also assign the same value to multiple 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;  
}
```

Output - 150

C++ Identifiers:-

All C++ variables must be identified with unique names.

These Unique names are called Identifiers.

Identifiers can be shortnames (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; // Ok but not so easy to understand.

int m = 60;

cout << minutesPerHour << "\n";

cout << m; return 0;

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):

Ex:-

```
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;  
}
```

Ex: ^{Since} `const int minutesPerHour = 60;`
`const float PI = 3.14;`

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

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

```
int main() {  
    const int minutesPerHour = 60;  
    const float PI = 3.14;  
    cout << minutesPerHour << endl;  
    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 `get` 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 :- " << (12 -
```

```
1.1) / 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 6-7 decimal digits while double variable has a precision of about 15 digits. Therefore it is refer to the 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;
```

T8

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;`
 `bool isCodingFun = true;`
 `bool isFishTasty = false;`
 `cout << "Coding fun? " << (isCodingFun ? "true" : "false") << endl;`
 `cout << "Is fish tasty? " << (isFishTasty ? "true" : "false") << endl;`
 `return 0;`

O/P
1
0

Boolean values are mostly used for Conditional testing.

Character data types:

Char → Used to store a single character.
must be enclosed by single quotes like `'p'`

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 = 87;`
 `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 ^{text} 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 surrounded 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;
}

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];
 // O/P - 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

String txt = "We are the so called "Viking" from the north;"

Solution to avoid this Problem :-

To use the backslash except characters

- \ Single quotes
- \" Double quotes
- \ Backslash

String length:

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

Eg String `txt` =

```
"A B C D E F G H I J K L M N O P Q R S T U V W X Y Z";
```

Out `<< "The length of the txt string is : " << txt.length();`

```
#include<iostream>
```

```
#include<string>
```

```
using namespace std;
```

```
int main () {
```

```
    string txt = "A — — Z";
```

Out `<< "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";
```

Out `<< "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 $\text{int } x = 100 + 50;$

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

Ex $\text{int sum1} = 100 + 50; \quad // 150(100+50)$
 $\text{int sum2} = \text{sum1} + 250; // 400 (150+250)$
 $\text{int sum3} = \text{sum2} + \text{sum1}; \quad // 800 (400+400)$

Program

```
#include <iostream>
using namespace std;
int main() {
    int sum1 = 100 + 50;
    int sum2 = sum1 + 250;
    int sum3 = sum2 + sum1;
    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;
}
```

O/P-8

Subtraction :-

```
int x=5;
int y=3;
cout << n-y;
return 0;
```

2
3

Multiplication

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

15

Div

```
int x=12;
int y=3;
cout << n/y;
return 0;
```

y

Modulus

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

2
3

Increment

```
int main() {
    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;
```

3
O/P 10

Additional assignment optr
+=

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

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

decrement optr

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

15

Multiply - int x=5;
x*=3;
cout << x;

(15)

int x=5;

②
return 0;

Division Assignment

```
int main()
double x = 5;
n / = 3;
cout << x;
return 0;
```

1. 66667

1 =
(1 = n ^ = 3)

```
int main()
int x = 5;
n / = 3;
cout << x;
return 0;
```

6

Modulus

```
int main()
int x = 5;
x % = 3;
cout << x;
return 0;
```

②

x % = 3

```
int main()
int x = 5;
x % = 3;
cout << x;
return 0;
```

0 / P → 1

1 =
(n / = 3)

```
int main()
int x = 5;
x / = 3;
cout << x;
return 0;
```

0 / P → 1

```
int p
divide
divide
}
void quick
{
if(n < 1)
return;
videFurther(
main()
n;
it t1,t2;
ength;
<<"Enter the
> length;
put[length];
i=0; i < leng
l=rand()%10
```

try before se
input, length

ut, length);

fter sorting:
length);

"<<t1/CLOC

>> =
(n >> = 3)

```
int main()
int x = 5;
n >> = 3;
cout << x;
return 0;
```

0 / P

< =
(n << = 3)

```
int x = 5;
&lt; = 3;
cout << x;
return 0;
```

40

C++ Comparison operators

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

The return value of comparison is either 1 or 0, which means true (1) or false (0). These values are known as Boolean values. If you wt

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

25

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;
}

```

John Doe
O/P - 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 " ");

Example

```

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 one 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

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

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

O/P → 0

Return 0 because !(not) is used to reverse the result

O/P → 0

C++ Strings :- String are used for storing text.

A string variable consists a collection of characters surrounded by double quotes,

Ex:- string greeting = "Hello"; Used - String 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 strings 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.

$n < 5 \& n < 10$

#include <iostream>

using namespace std;

int main()

{ int x=5;

int y=3;

cout << (n>3 & n<10); // Return true

return 0;

}

OP \rightarrow !

∴ 5 is greater than 3

and 5 is less than 10

T - T \rightarrow T 0

Logical OR :- Return true if one of the statement is true.
 $n < 5 || n < 4$

#include <iostream>

using namespace std;

int main()

{ int x=5;

int y=3;

cout << (n>3 || x<4); // Return true b/c one of the condition are true (5 is greater than 3)

return 0;

}

OP \rightarrow !

but 5 is not less than 4

Operators :

$= =$ equal to $n = 2y$

```
int main () {  
    int x = 5;  
    int y = 3;
```

```
cout << (n = 2y); // return  
return 0;  
}
```

\neq greater than \geq less than

Not equal to \neq

```
int main () {  
    int x = 5;  
    int y = 3;  
    cout << (n != y);  
    return 0;  
}  
Return 1  
O/P - 1      true
```

```
int main () {  
    int x = 5;  
    int y = 3;  
    cout << (n > y);  
    return 0;  
}  
Return 1  
O/P - 1
```

```
int main () {  
    int x = 5;  
    int y = 3;  
    cout << (n < y);  
    return 0;  
}  
Return 0  
O/P - 0
```

\geq greater than equal to

```
int main () {  
    int x = 5;  
    int y = 3;  
    cout << (n >= y);  
    return 0;  
}  
Return 1  
O/P - 1
```

\leq less than equal to

```
int main () {  
    int x = 5;  
    int y = 3;  
    cout << (n <= 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.

```
for (int i=0; i<10; i++) {  
    if (i==4) {  
        break;  
    }  
    cout << i << endl;  
}
```

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

This example skip the value of 4:

```
for (int i=0; i<10; i++) {  
    if (i==4) {  
        continue;  
    }  
    cout << i << endl;  
}
```

0 1 2 3 4 5 6 7 8 9
i+1:
i+2:
i+3:
i+4:
i+5:
i+6:
i+7:
i+8:
i+9:
i+10:
i+11:
i+12:
i+13:
i+14:
i+15:
i+16:
i+17:
i+18:
i+19:
i+20:
i+21:
i+22:
i+23:
i+24:
i+25:
i+26:
i+27:
i+28:
i+29:
i+30:
i+31:
i+32:
i+33:
i+34:
i+35:
i+36:
i+37:
i+38:
i+39:
i+40:
i+41:
i+42:
i+43:
i+44:
i+45:
i+46:
i+47:
i+48:
i+49:
i+50:
i+51:
i+52:
i+53:
i+54:
i+55:
i+56:
i+57:
i+58:
i+59:
i+60:
i+61:
i+62:
i+63:
i+64:
i+65:
i+66:
i+67:
i+68:
i+69:
i+70:
i+71:
i+72:
i+73:
i+74:
i+75:
i+76:
i+77:
i+78:
i+79:
i+80:
i+81:
i+82:
i+83:
i+84:
i+85:
i+86:
i+87:
i+88:
i+89:
i+90:
i+91:
i+92:
i+93:
i+94:
i+95:
i+96:
i+97:
i+98:
i+99:
i+100:
i+101:
i+102:
i+103:
i+104:
i+105:
i+106:
i+107:
i+108:
i+109:
i+110:
i+111:
i+112:
i+113:
i+114:
i+115:
i+116:
i+117:
i+118:
i+119:
i+120:
i+121:
i+122:
i+123:
i+124:
i+125:
i+126:
i+127:
i+128:
i+129:
i+130:
i+131:
i+132:
i+133:
i+134:
i+135:
i+136:
i+137:
i+138:
i+139:
i+140:
i+141:
i+142:
i+143:
i+144:
i+145:
i+146:
i+147:
i+148:
i+149:
i+150:
i+151:
i+152:
i+153:
i+154:
i+155:
i+156:
i+157:
i+158:
i+159:
i+160:
i+161:
i+162:
i+163:
i+164:
i+165:
i+166:
i+167:
i+168:
i+169:
i+170:
i+171:
i+172:
i+173:
i+174:
i+175:
i+176:
i+177:
i+178:
i+179:
i+180:
i+181:
i+182:
i+183:
i+184:
i+185:
i+186:
i+187:
i+188:
i+189:
i+190:
i+191:
i+192:
i+193:
i+194:
i+195:
i+196:
i+197:
i+198:
i+199:
i+200:
i+201:
i+202:
i+203:
i+204:
i+205:
i+206:
i+207:
i+208:
i+209:
i+210:
i+211:
i+212:
i+213:
i+214:
i+215:
i+216:
i+217:
i+218:
i+219:
i+220:
i+221:
i+222:
i+223:
i+224:
i+225:
i+226:
i+227:
i+228:
i+229:
i+230:
i+231:
i+232:
i+233:
i+234:
i+235:
i+236:
i+237:
i+238:
i+239:
i+240:
i+241:
i+242:
i+243:
i+244:
i+245:
i+246:
i+247:
i+248:
i+249:
i+250:
i+251:
i+252:
i+253:
i+254:
i+255:
i+256:
i+257:
i+258:
i+259:
i+260:
i+261:
i+262:
i+263:
i+264:
i+265:
i+266:
i+267:
i+268:
i+269:
i+270:
i+271:
i+272:
i+273:
i+274:
i+275:
i+276:
i+277:
i+278:
i+279:
i+280:
i+281:
i+282:
i+283:
i+284:
i+285:
i+286:
i+287:
i+288:
i+289:
i+290:
i+291:
i+292:
i+293:
i+294:
i+295:
i+296:
i+297:
i+298:
i+299:
i+300:
i+301:
i+302:
i+303:
i+304:
i+305:
i+306:
i+307:
i+308:
i+309:
i+310:
i+311:
i+312:
i+313:
i+314:
i+315:
i+316:
i+317:
i+318:
i+319:
i+320:
i+321:
i+322:
i+323:
i+324:
i+325:
i+326:
i+327:
i+328:
i+329:
i+330:
i+331:
i+332:
i+333:
i+334:
i+335:
i+336:
i+337:
i+338:
i+339:
i+340:
i+341:
i+342:
i+343:
i+344:
i+345:
i+346:
i+347:
i+348:
i+349:
i+350:
i+351:
i+352:
i+353:
i+354:
i+355:
i+356:
i+357:
i+358:
i+359:
i+360:
i+361:
i+362:
i+363:
i+364:
i+365:
i+366:
i+367:
i+368:
i+369:
i+370:
i+371:
i+372:
i+373:
i+374:
i+375:
i+376:
i+377:
i+378:
i+379:
i+380:
i+381:
i+382:
i+383:
i+384:
i+385:
i+386:
i+387:
i+388:
i+389:
i+390:
i+391:
i+392:
i+393:
i+394:
i+395:
i+396:
i+397:
i+398:
i+399:
i+400:
i+401:
i+402:
i+403:
i+404:
i+405:
i+406:
i+407:
i+408:
i+409:
i+410:
i+411:
i+412:
i+413:
i+414:
i+415:
i+416:
i+417:
i+418:
i+419:
i+420:
i+421:
i+422:
i+423:
i+424:
i+425:
i+426:
i+427:
i+428:
i+429:
i+430:
i+431:
i+432:
i+433:
i+434:
i+435:
i+436:
i+437:
i+438:
i+439:
i+440:
i+441:
i+442:
i+443:
i+444:
i+445:
i+446:
i+447:
i+448:
i+449:
i+450:
i+451:
i+452:
i+453:
i+454:
i+455:
i+456:
i+457:
i+458:
i+459:
i+460:
i+461:
i+462:
i+463:
i+464:
i+465:
i+466:
i+467:
i+468:
i+469:
i+470:
i+471:
i+472:
i+473:
i+474:
i+475:
i+476:
i+477:
i+478:
i+479:
i+480:
i+481:
i+482:
i+483:
i+484:
i+485:
i+486:
i+487:
i+488:
i+489:
i+490:
i+491:
i+492:
i+493:
i+494:
i+495:
i+496:
i+497:
i+498:
i+499:
i+500:
i+501:
i+502:
i+503:
i+504:
i+505:
i+506:
i+507:
i+508:
i+509:
i+510:
i+511:
i+512:
i+513:
i+514:
i+515:
i+516:
i+517:
i+518:
i+519:
i+520:
i+521:
i+522:
i+523:
i+524:
i+525:
i+526:
i+527:
i+528:
i+529:
i+530:
i+531:
i+532:
i+533:
i+534:
i+535:
i+536:
i+537:
i+538:
i+539:
i+540:
i+541:
i+542:
i+543:
i+544:
i+545:
i+546:
i+547:
i+548:
i+549:
i+550:
i+551:
i+552:
i+553:
i+554:
i+555:
i+556:
i+557:
i+558:
i+559:
i+560:
i+561:
i+562:
i+563:
i+564:
i+565:
i+566:
i+567:
i+568:
i+569:
i+570:
i+571:
i+572:
i+573:
i+574:
i+575:
i+576:
i+577:
i+578:
i+579:
i+580:
i+581:
i+582:
i+583:
i+584:
i+585:
i+586:
i+587:
i+588:
i+589:
i+590:
i+591:
i+592:
i+593:
i+594:
i+595:
i+596:
i+597:
i+598:
i+599:
i+600:
i+601:
i+602:
i+603:
i+604:
i+605:
i+606:
i+607:
i+608:
i+609:
i+610:
i+611:
i+612:
i+613:
i+614:
i+615:
i+616:
i+617:
i+618:
i+619:
i+620:
i+621:
i+622:
i+623:
i+624:
i+625:
i+626:
i+627:
i+628:
i+629:
i+630:
i+631:
i+632:
i+633:
i+634:
i+635:
i+636:
i+637:
i+638:
i+639:
i+640:
i+641:
i+642:
i+643:
i+644:
i+645:
i+646:
i+647:
i+648:
i+649:
i+650:
i+651:
i+652:
i+653:
i+654:
i+655:
i+656:
i+657:
i+658:
i+659:
i+660:
i+661:
i+662:
i+663:
i+664:
i+665:
i+666:
i+667:
i+668:
i+669:
i+670:
i+671:
i+672:
i+673:
i+674:
i+675:
i+676:
i+677:
i+678:
i+679:
i+680:
i+681:
i+682:
i+683:
i+684:
i+685:
i+686:
i+687:
i+688:
i+689:
i+690:
i+691:
i+692:
i+693:
i+694:
i+695:
i+696:
i+697:
i+698:
i+699:
i+700:
i+701:
i+702:
i+703:
i+704:
i+705:
i+706:
i+707:
i+708:
i+709:
i+710:
i+711:
i+712:
i+713:
i+714:
i+715:
i+716:
i+717:
i+718:
i+719:
i+720:
i+721:
i+722:
i+723:
i+724:
i+725:
i+726:
i+727:
i+728:
i+729:
i+730:
i+731:
i+732:
i+733:
i+734:
i+735:
i+736:
i+737:
i+738:
i+739:
i+740:
i+741:
i+742:
i+743:
i+744:
i+745:
i+746:
i+747:
i+748:
i+749:
i+750:
i+751:
i+752:
i+753:
i+754:
i+755:
i+756:
i+757:
i+758:
i+759:
i+760:
i+761:
i+762:
i+763:
i+764:
i+765:
i+766:
i+767:
i+768:
i+769:
i+770:
i+771:
i+772:
i+773:
i+774:
i+775:
i+776:
i+777:
i+778:
i+779:
i+780:
i+781:
i+782:
i+783:
i+784:
i+785:
i+786:
i+787:
i+788:
i+789:
i+790:
i+791:
i+792:
i+793:
i+794:
i+795:
i+796:
i+797:
i+798:
i+799:
i+800:
i+801:
i+802:
i+803:
i+804:
i+805:
i+806:
i+807:
i+808:
i+809:
i+810:
i+811:
i+812:
i+813:
i+814:
i+815:
i+816:
i+817:
i+818:
i+819:
i+820:
i+821:
i+822:
i+823:
i+824:
i+825:
i+826:
i+827:
i+828:
i+829:
i+830:
i+831:
i+832:
i+833:
i+834:
i+835:
i+836:
i+837:
i+838:
i+839:
i+840:
i+841:
i+842:
i+843:
i+844:
i+845:
i+846:
i+847:
i+848:
i+849:
i+850:
i+851:
i+852:
i+853:
i+854:
i+855:
i+856:
i+857:
i+858:
i+859:
i+860:
i+861:
i+862:
i+863:
i+864:
i+865:
i+866:
i+867:
i+868:
i+869:
i+870:
i+871:
i+872:
i+873:
i+874:
i+875:
i+876:
i+877:
i+878:
i+879:
i+880:
i+881:
i+882:
i+883:
i+884:
i+885:
i+886:
i+887:
i+888:
i+889:
i+890:
i+891:
i+892:
i+893:
i+894:
i+895:
i+896:
i+897:
i+898:
i+899:
i+900:
i+901:
i+902:
i+903:
i+904:
i+905:
i+906:
i+907:
i+908:
i+909:
i+910:
i+911:
i+912:
i+913:
i+914:
i+915:
i+916:
i+917:
i+918:
i+919:
i+920:
i+921:
i+922:
i+923:
i+924:
i+925:
i+926:
i+927:
i+928:
i+929:
i+930:
i+931:
i+932:
i+933:
i+934:
i+935:
i+936:
i+937:
i+938:
i+939:
i+940:
i+941:
i+942:
i+943:
i+944:
i+945:
i+946:
i+947:
i+948:
i+949:
i+950:
i+951:
i+952:
i+953:
i+954:
i+955:
i+956:
i+957:
i+958:
i+959:
i+960:
i+961:
i+962:
i+963:
i+964:
i+965:
i+966:
i+967:
i+968:
i+969:
i+970:
i+971:
i+972:
i+973:
i+974:
i+975:
i+976:
i+977:
i+978:
i+979:
i+980:
i+981:
i+982:
i+983:
i+984:
i+985:
i+986:
i+987:
i+988:
i+989:
i+990:
i+991:
i+992:
i+993:
i+994:
i+995:
i+996:
i+997:
i+998:
i+999:
i+1000:
i+1001:
i+1002:
i+1003:
i+1004:
i+1005:
i+1006:
i+1007:
i+1008:
i+1009:
i+1010:
i+1011:
i+1012:
i+1013:
i+1014:
i+1015:
i+1016:
i+1017:
i+1018:
i+1019:
i+1020:
i+1021:
i+1022:
i+1023:
i+1024:
i+1025:
i+1026:
i+1027:
i+1028:
i+1029:
i+1030:
i+1031:
i+1032:
i+1033:
i+1034:
i+1035:
i+1036:
i+1037:
i+1038:
i+1039:
i+1040:
i+1041:
i+1042:
i+1043:
i+1044:
i+1045:
i+1046:
i+1047:
i+1048:
i+1049:
i+1050:
i+1051:
i+1052:
i+1053:
i+1054:
i+1055:
i+1056:
i+1057:
i+1058:
i+1059:
i+1060:
i+1061:
i+1062:
i+1063:
i+1064:
i+1065:
i+1066:
i+1067:
i+1068:
i+1069:
i+1070:
i+1071:
i+1072:
i+1073:
i+1074:
i+1075:
i+1076:
i+1077:
i+1078:
i+1079:
i+1080:
i+1081:
i+1082:
i+1083:
i+1084:
i+1085:
i+1086:
i+1087:
i+1088:
i+1089:
i+1090:
i+1091:
i+1092:
i+1093:
i+1094:
i+1095:
i+1096:
i+1097:
i+1098:
i+1099:
i+1100:
i+1101:
i+1102:
i+1103:
i+1104:
i+1105:
i+1106:
i+1107:
i+1108:
i+1109:
i+1110:
i+1111:
i+1112:
i+1113:
i+1114:
i+1115:
i+1116:
i+1117:
i+1118:
i+1119:
i+1120:
i+1121:
i+1122:
i+1123:
i+1124:
i+1125:
i+1126:
i+1127:
i+1128:
i+1129:
i+1130:
i+1131:
i+1132:
i+1133:
i+1134:
i+1135:
i+1136:
i+1137:
i+1138:
i+1139:
i+1140:
i+1141:
i+1142:
i+1143:
i+1144:
i+1145:
i+1146:
i+1147:
i+1148:
i+1149:
i+1150:
i+1151:
i+1152:
i+1153:
i+1154:
i+1155:
i+1156:
i+1157:
i+1158:
i+1159:
i+1160:
i+1161:
i+1162:
i+1163:
i+1164:
i+1165:
i+1166:
i+1167:
i+1168:
i+1169:
i+1170:
i+1171:
i+1172:
i+1173:
i+1174:
i+1175:
i+1176:
i+1177:
i+1178:
i+1179:
i+1180:
i+1181:
i+1182:
i+1183:
i+1184:
i+1185:
i+1186:
i+1187:
i+1188:
i+1189:
i+1190:
i+1191:
i+1192:
i+1193:
i+1194:
i+1195:
i+1196:
i+1197:
i+1198:
i+1199:
i+1200:
i+1201:
i+1202:
i+1203:
i+1204:
i+1205:
i+1206:
i+1207:
i+1208:
i+1209:
i+1210:
i+1211:
i+1212:
i+1213:
i+1214:
i+1215:
i+1216:
i+1217:
i+1218:
i+1219:
i+1220:
i+1221:
i+1222:
i+1223:
i+1224:
i+1225:
i+1226:
i+1227:
i+1228:
i+1229:
i+1230:
i+1231:
i+1232:
i+1233:
i+1234:
i+1235:
i+1236:
i+1237:
i+1238:
i+1239:
i+1240:
i+1241:
i+1242:
i+1243:
i+1244:
i+1245:
i+1246:
i+1247:
i+1248:
i+1249:
i+1250:
i+1251:
i+1252:
i+1253:
i+1254:
i+1255:
i+1256:
i+1257:
i+1258:
i+1259:
i+1260:
i+1261:
i+1262:
i+1263:
i+1264:
i+1265:
i+1266:
i+1267:
i+1268:
i+1269:
i+1270:
i+1271:
i+1272:
i+1273:
i+1274:
i+1275:
i+1276:
i+1277:
i+1278:
i+1279:
i+1280:
i+1281:
i+1282:
i+1283:
i+1284:
i+1285:
i+1286:
i+1287:
i+1288:
i+1289:
i+1290:
i+1291:
i+1292:
i+1293:
i+1294:
i+1295:
i+1296:
i+1297:
i+1298:
i+1299:
i+1300:
i+1301:
i+1302:
i+1303:
i+1304:
i+1305:
i+1306:
i+1307:
i+1308:
i+1309:
i+1310:
i+1311:
i+1312:
i+1313:
i+1314:
i+1315:
i+1

class demo temp (0,0); //Parameter
temp.a = a + obj.a; //Value sum in temp
temp.b = b + obj.b;
return temp;

You can write void
return the
but now we
One class
has
demo

} ;
};

Void main ()

```
Class obj;
obj ob(10,20), ob1(30,40), ob2(0,0); //Create objects
ob2 = ob + ob1; //Add value
ob2.show(); //Show sum
getch();
}
```

A=40 B=60

10 20 30 40
 60

Implicit parameter
through this pointer
Explicit value pass through
object -> object ->

If the name
is fil name
starting by
&fil.name

for the file
&fil.name;

++ i
fil.name

le Not For

le Name.

is it the
&fil.name;

if fil.name

demo
demo
for default constructor
the include <conio.h>
A
clear demo
,
int a, b;
public:
demo() // default constructor
{

} ;
demo(int x, int y)

} ;
a = x;
b = y;

}

fe a program
llection of
ffected group
le organization
y which it can
le Allocation
ganization,
t is, record
e can only be
location:
location struc
s own name
the "m" block
block, the
cation.:
ration, each
e disk. The
pointer to

void main()

new obj & pass value.

clrscr();

demo obj(10,20); demo obj(30,40); demo obj;
obj = obj + obj;
obj.show();
getch();

A=40

B=60

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

Avoid compiler confusion.

#include <iostream.h>

#include <conio.h>

class A

{ int a,b; } *object* variable (variable of a class)

deafult set of class & directly access the obj

public :

A (int x, int y) // Parameterized constructor

{} *Name* *copying* *constructor* *parameter* *constructor*

a=x; b=y;

void show()

{ cout << "Address"; }

};

void main()

{

main();

for

ar

;

void P

for (i

coun

;

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

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

```
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)
}
```

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

Re
We
take
Time

Symbolic
variableName; anyName) {
 for (type variableName : anyName) {
 // code block to be executed
 }
}

While all elements in an array, using a "for each loop"
) output all numbers[i] = {{10, 20, 30, 40, 50}};
 for (int i = 0; i < numbers.length; i++) {
 int num = numbers[i];
 System.out.println("Number " + num);
 }
}

Syntax:-

```
int i=0;
while (i<5) {
    //Code to be executed
}
```

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

C++ DoWhile Loop:-

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

Syntax:-

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

For loop:-

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

Syntax:-

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

St-1 → St-1 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 (everytime) after the code block has been executed

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

Syntax :-

Short hand if - else (ternary operator) :-

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

Syntax

Variable = (Condition)?
code True : code False;

Switch :-

```
switch (exp) {  
    case X:  
        // Code block  
    break;  
    case Y:  
        // Code block  
    break;  
    default:  
        // Code block  
}
```

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

Break Keyword :-

Break out of the switch block.
When the match is found,
and our job is done, its time
for a break. There is no need
for nesting.

C++ loop can execute a block of code as long as a
specified condition is needed.

Loop - Save time, Reduce error, More code reuse needed

C++ while loop :-
Loops through a block of code
as long as a specified
condition 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: $a \leq b$

greater than: $a > b$

greater than or equal to: $a \geq b$

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 condition is false (if 2nd condition is true → else if condition)

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

if → ✓ If → Generate an error

if → ✓
if condition { ② else
 if (cond) { ③ elseif if
 // code execute if (cond)
 // code true {
 }
}

3 }
 } else {
 // code execute
 if (cond is
 true)
 }
}

else {
 // code execute
 if cond is
 true
}

getLine() function to read a line of text. It takes one
line of text as input and returns it as a string variable as output.

13

Spring fullName;

Look >> "Type your full name: ";

name (in full name);

Your name is: " < < Full Name;"

C++ Meth:

C++ has many features that allow you to perform mathematical terms on numbers.

May and June

The met (my) function can be used to find the highest

13 Count << max (5,10);

```
#include <iostream>
using namespace std;
int main () {
    cout << max (5,10);
    return 0;
}
```

min(y) fun can be used to find the lowest value of x and
Count < min(510); (O/P-5)

Cat (*Catodon*) redens (Savart (Square root), round (sound after function), much on *Isognathus*) can be found in

a no) and no - .
, mehn) header file

old
Ez At include < Cmeth >
& Sqrt(64),
Cant < Sqr(2, c); } old
Cant < Sqr(2, c); } old
0.693147

1st card 1st game
and card 2 in play

e.g. `String txt = "we are the so called \"Vikings\" from the north."`;

`String txt = "2t \\'s always . ' "`
`cout < txt;`

```
in & out  
String txt = "Hello \n word!"  
cout < txt;  
return 0;
```

0/p - hello
world!

```
int main()  
{  
    String txt = "Hello \t word!"  
    cout < txt;  
    return 0;  
}
```

0/p Hello world!

User input strings?
extraction operator `>>` on cin → to take a string entered by the user.

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

cin Consider a space (whitespace, tabs etc)
as a terminating character which means treat it
(an empty space a single word).
Note we expect to print "John Doe", but it only prints "John",
that's why when working with string, we often use the