[Pick the date]

TECHNICAL EDUCATION

C++

Computer Training Institute | Directed By- **Mr.Murti Nishad**

C++ in Hindi Introduction

- Introduction to **C++ in Hindi**
- *Features of C++ in Hindi*
- Object oriented principles of C++ in Hindi
-

Introduction to C++

C++ एक object oriented programming language है। C++ Bjarne stroustrup के द्वारा develop की गयी थी। C++ के आने से पहले c और Simula 67 दो popular languages थी।

Bjarne stroustrup इन दोनों languages को मिलाकर एक ऐसी language बनाना चाहते थे, जिसमे object oriented programming के सभी features हो। उनकी इसी सोच का परिणाम C++ थी।

C++ को 1979 में bell laboratories में develop किया गया था। C++ एक middle level language है। इसमें high level और low level languages के फीचर्स है। C language के सभी features C++ में पाये जाते है। C++ के 3 important features है।

1. Object oriented
2. Low level as well as high level
3. Easy to learn

Uses of C++

C++ कँहा कँहा यूज़ की जाती है और आप इसे कैसे इस्तेमाल कर सकते है, आइये जानने का प्रयास करते है।

1. सामान्यतः C++ को ज्यादातर उन software में यूज़ किया जाता है जँहा Hardware के ऊपर अधिक control की आवश्यकता होती है। जैसे की device drivers बनाने के लिए C++ का इस्तेमाल किया जाता है।
2. यह एक multipurpose language है। इसे आप बड़े software क्रिएट करने के लिए इस्तेमाल कर सकते है।
3. आज कल C++ को ज्यादातर education के purpose से यूज़ किया जाता है। क्योंकि सभी बड़ी languages के features C++ से लिए गए है। यदि आप C++ को समझ गए है तो बाकि languages को भी आसानी से समझ सकते है।
4. C++ c language का extension है। इसलिए वो सभी software जो c बनाये जा सकते है वो C++ में भी बनाये जा सकते है।
5. यदि आप चाहते है की आपका software low level पर भी उतने ही अच्छे तरीके से काम करे जितना high level पर तो आप C++ यूज़ कर सकते है।
6. Mostly जितनी भी नयी programming language है वो pointer को खुलकर support नहीं करती है। यदि आप pointers के इस्तेमाल से कोई software क्रिएट करना चाहते है तो आप C++ का यूज़ कर सकते है।

Object Oriented Principles

C++ द्वारा support किये जाने वाले object oriented principle के बारे आगे बताया गया है।

Class

Class एक user defined data type होती है। ये C language में structure की तरह ही होती है। लेकिन classes में आप variable के साथ साथ उनसे related functions भी create कर सकते है जिन्हें बाद में objects के द्वारा access किया जा सकता है।

एक class programmer को data centered approach provide करती है। Class के द्वारा आप variables और उनसे related functions को separate कर पाते है। इसके बारे में और अधिक आप C++ – Classes & Objects tutorial से जान सकते है।

Objects

जैसा की आपको पता है class एक user defined data type होती है। Class type के variables को objects कहा जाता है। Object के द्वारा आप class के variables और functions को access कर सकते है।

Abstraction

Abstraction का मतलब होता है की end user को वही functionality show की जाए जिसकी उसको जरुरत है बाकी background functionality hide कर दी जाए। उदाहरण के लिए जब आप एक car को चलाते है तो आप सिर्फ clutch, gear और steering पर ध्यान देते है, engine किस प्रकार काम कर रहा है इससे आपका कोई मतलब नहीं होता है।

Encapsulation

दूसरे शब्दों में encapsulation को data hiding भी कहते है। Encapsulation एक object oriented programming feature है जो

data members (variables) और उनसे related functions को bind करता है जैसे की class होती है।

साथ ही encapsulation के द्वारा data और functions को outside access से protect किया जाता है। Encapsulation properties 3 level (Public, Private, Protected) का protection provide करती है। Encapsulation classes के माध्यम से implement किया जाता है।

Inheritance

Inheritance object oriented programming का concept है जिसमे एक code को दूसरी जगह यूज़ किया जा सकता है। ये principle code re-usability implement करता है। उदाहरण के लिए आप एक class में create किये गए functions को दूसरी class में inherit कर सकते है ऐसा करने से आपको इन functions को दुबारा लिखने की आवश्यकता नहीं होगी। और आप इन्हें दोनों ही classes के objects से access कर पाएंगे।

Polymorphism

Polymorphism का मतलब एक नाम कई काम होता है। इसमें आप एक नाम से कई तरह के tasks implement करते है। C++ में polymorphism function overloading द्वारा implement किया जाता है जिसमें एक नाम के functions को अलग अलग conditions में execute किया जाता है।

Message Passing

Object oriented programming में objects एक दूसरे से communicate करते है जिससे program real life condition को represent करता है।

A Sample C++ program

```
#include<iostream>
 int  main()
 {
     cout<<"Hello Reader";
     return 0;
 }
```

इस उदाहरण को निचे समझाया जा रहा है।

1. सबसे पहली line में header file को include किया गया है। Screen पर output show करने के लिए आपको इस header file की जरुरत पडती है।
2. इसके बाद program का main method शुरू किया गया है।
3. cout function के द्वारा simple message को print किया गया है।
4. return statement के द्वारा function को return किया गया है।

ऊपर दिया गया program निचे दिया गया output generate करता है।

```
Hello Reader
```

इस tutorial में आपने C++ की history, applications और अलग अलग features के बारे में जाना है। किसी भी program में data महत्वपूर्ण होता है। Data पर ही सभी operations perform किये जाते है। C++ में

programming शुरू करने से पहले आपको C++ के विभिन्न data types के बारे में जानना आवश्यक है।

C++ in Hindi – Data Types

Introduction to C++ Data Types

जब भी आप किसी value को store करने के लिए variable create करते है तो आप साथ ही उस variable का data type भी define करते है। Data type से compiler को ये पता चलता है की इस variable में किस type का data store किया जायेगा।

साथ ही compiler data type से ये भी judge करता है की किसी variable को memory में कितना space assign करना है। C++ में data types को 3 categories में divide किया गया है।

1. Basic Data Types – ये वो data types होते है जो ज्यादातर सभी programming languages में पाए जाते है।

2. User Defined Data Types – C++ आपको data type create करने की capabilities provide करती है।
3. Derived Data Types – इस तरह के data types कई data types के combination होते है।

आइये इन data types के बारे में detail से जानने का प्रयास करते है।

Basic Data Types

जैसा की मैने आपको पहले बताया basic data types सभी programming languages में almost common होते है। Basic data types निचे दी गयी 3 categories में होते है।

1. Integer
2. Floating point
3. Character

इनके बारे में निचे detail से दिया जा रहा है।

Integer

Integer category के data types whole number को store करने के लिए यूज़ किये जाते है। Whole numbers वे numbers होते है जिनमें दशमलव और उसके बाद की संख्या नहीं होती है। ये पूर्ण संख्याएं होती है। इस category के data types को निचे table के द्वारा represent किया जा रहा है।

Data Type	Size (In Bytes)	Range
int	2	-32768 To 32767
short int	2	-31768 To 32767
long int	4	-2147483648 To 2147483647
signed int	2	-31768 To 32767
unsigned int	2	0 To 65535

इन data types को इनकी size और range के base पर differentiate किया गया है। आप आवश्यकता के अनुसार कोई भी data type यूज कर सकते है। निचे इसका उदाहरण दिया गया है।

```
int balance = 650000;
```

Floating Point

Floating point data types ऐसी संख्याओं को store करने के लिए यूज किये जाते है जिनमें दशमलव होता है। Floating point data types 3 तरह के होते है, इनके बारे में निचे बताया जा रहा है।

Data Type	Size (In bytes)	Range
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

ऊपर define किये गए data types की size और range अलग अलग है। आप आवश्यकता के अनुसार कोई भी data type यूज़ कर सकते है। इसका उदाहरण निचे दिया जा रहा है।

```
float PI = 3.14;
```

Character

संख्या के बजाय यदि आप किसी अक्षर (character) को store करना चाहते है तो इसके लिए आप इस category के data types यूज़ कर सकते है। इस category के data types के बारे में निचे दिया जा रहा है।

Data Type	Size (In bytes)	Range
char	1	-128 to 127
unsigned char	1	0 to 255
signed char	1	-128 to 127

Character data types को यूज़ करने का उदाहरण निचे दिया जा रहा है।

```
char Name = "V";
```

User Defined Data Types

User defined data types वे data types होते है जिन्हें आप programmer के रूप में define करते है। User defined data types के बारे में आप पहले भी C Language में पढ़ चुके है। C Language में आपको

Struct और Union के बारे में बताया गया था जो की इसी category के data types है।

ये data types C++ में भी पूरी तरह allowed है। साथ ही C++ आपको कुछ नए user define data types create करने के capabilities provide करती है जो की object oriented programming के लिए suitable है। इनके बारे में निचे बताया जा रहा है।

Class

Class भी struct की तरह ही होती है। लेकिन struct के आप object create नहीं कर सकते है और classes object oriented programming का base है। Struct की ही तरह आप classes में भी variables create कर सकते है और साथ ही आप classes में functions भी create कर सकते है। Class के बारे में आपको detail से classes और objects की tutorial में बताया जायेगा।

Enum Data Type

Enum data type एक user define data type होता है जो names के साथ numbers attach कर देता है। इससे code को यूज करना आसान हो जाता है। Enum data type create करने के लिए आप enum keyword यूज करते है। इसका general syntax निचे दिया जा रहा है।

```
enum dataTypeName {value1, value2, value3};
```

enum keyword के बाद आप variable का नाम देते है। सभी values curly braces के अंदर (,) से separate करके लिखी जाती है।

enum keyword के यूज़ से इन names को automatically 0 से लेकर values assign हो जाती है। इसका उदाहरण निचे दिया जा रहा है।

```
enum days {Mon, Tue, Wed, Thi, Fri, Sat, Sun};
```

इसके बाद इस data type का variable create किया जा सकता है। लेकिन उस variable को values सिर्फ वही assign हो सकती है जो नाम आपने enum data type create करते समय दिए थे। हालाँकि C Language में आप directly integer number के द्वारा ही value assign कर सकते है लेकिन C++ में ऐसा नहीं है।

आप integer number द्वारा value assign कर सकते है लेकिन उससे पहले आप data type का नाम bracket में देते है। या फिर ऐसा ना करते हुए आप सीधा नाम भी दे सकते है। इसका उदाहरण निचे दिया जा रहा है।

```
days setDay = (days) 3;
```

Derived Data Types

Derived data types वे होते है जो basic data types से derived होते है। Derived data types कोई नया data type नहीं create करते है, इसके बजाय basic data types में ही कुछ functionality add करते है। C++ में 3 तरह के derived data types पाए जाते है।

1. Arrays
2. Functions
3. Pointers

C++ in Hindi – Variables

- Introduction to **C++ variables in Hindi**
- Creating *C++ variables in Hindi*
- Initializing & displaying C++ variables in Hindi
-

Introduction to C++ Variables

किसी भी तरह के data को computer memory में store करने के लिए आप variables का उपयोग करते है। आसान शब्दों में कहें तो variables किसी box की तरह होते है जिसमें आप values को store करते है।

Technically variables एक computer memory location होती है जिसे एक नाम दिया जाता है और उस memory location पर values store की जाती है। बाद में इसी नाम को use करते हुए आप store की गयी value पर अलग अलग operations perform करते है।

Variables की values change की जा सकती है। उदाहरण के लिए यदि आपने किसी variable में 5 store किया है तो आप इसे change करके 10 कर सकते है। ऐसा आप खुद manually भी कर सकते है या फिर किसी operation के द्वारा भी ऐसा किया जा सकता है।

Creating Variables

C++ में भी C language की तरह ही variables create किये जाते है। Variables create करने का general syntax निचे दिया जा रहा है।

```
data_type variable_name;
```

C++ में variables create करने के लिए सबसे पहले आप ये define करते है की आप किस तरह की value उस variable में store करेंगे। ऐसा आप data type define करके करते है।

Data type से compiler को पता चलता है की किस तरह की value एक particular variable में store की जायेगी, इस जानकारी के base पर compiler जितनी memory required होती है उतनी इस variable को allot करता है।

Data types के बारे में और अधिक आप C++ Data Types की tutorial से पढ सकते है।

Data type define करने के बाद variable का एक unique नाम define किया जाता है। आप एक नाम के 2 variables नहीं create कर सकते है।

जैसा की आपको पता है की C++ एक case sensitive language है, इसलिए upper case और lower case variables अलग अलग माने जाते है। उदाहरण के लिए age और Age दो different variables माने जायेंगे।

आइये अब C++ में variables create करना एक उदाहरण के माध्यम से समझने का प्रयास करते है।

```
int age;
```

ऊपर दिए गए उदाहरण में integer type का variable create किया गया है। इस variable का नाम age है।

Initializing Variables

Variables में values store करना variable intialization कहलाता है। किसी भी variable में value store करवाने के लिए सबसे पहले आप उस variable का नाम लिखते है, इसके बाद assignment operator लगाकर आप वह value लिखते है जिसे आप इस variable में store करना चाहते है।

इसका general syntax निचे दिया जा रहा है।

```
variable_name = value;
```

आइये अब इसे एक उदाहरण से समझने का प्रयास करते है। मान लीजिये आप ऊपर create किये गए variable को value assign करवाना चाहते है तो ऐसा आप इस प्रकार कर सकते है।

```
age = 29;
```

आप चाहे तो variable create करते समय भी value assign कर सकते है। इसका उदाहरण निचे दिया जा रहा है।

```
int age = 29;
```

यदि आप user से value input करवाना चाहते है तो इसके लिए आप cin input statement का प्रयोग करते है। Input और output statements के बारे में और अधिक आप C++ I/O System की tutorial से पढ सकते है। इसका general syntax निचे दिया जा रहा है।

```
cin>>variable-name;
```

उदाहरण के लिए आप age variable की value user से input करवाना चाहते है इसके लिए आप इस प्रकार statement लिखेंगे।

```
cin>>age;
```

जब ऊपर दिया गया statement execute होगा तो console window खुलेगी जिसमें user अपनी value type कर सकता है। User के enter press करने पर लिखी गयी value variable को assign हो जाएगी।

Displaying Variables

किसी भी variable की value console screen पर display करने के लिए आप cout statement का प्रयोग करते है। इसका general syntax निचे दिया जा रहा है।

```
cout<<variable-name;
```

उदाहरण के लिए यदि आप age variable की value print करना चाहते है तो ऐसा आप इस प्रकार कर सकते है।

```
cout<<age;
```

Scope of Variables

C++ में variables उनके scope के अनुसार 2 प्रकार के होते है ।

1. Global Variables
2. Local Variables

Global Variables

जो variables पुरे program में कँही भी access/use किये जा सकते है वे global variables कहलाते है। ऐसे variables को program की शुरुआत में ही define किया जाता है। इसका उदाहरण निचे दिया जा रहा है।

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


// Global variable num
int num=25;

void display()
{
     // Accessing global variable in local scope
     cout<<"Number is : "<<num;
}

int main()
{
     display();
     return 0;
}
```

ऊपर दिया गया program निचे दिया गया output generate करता है।

```
Number is : 25
```

Local Variables

Local variables ऐसे variables होते है जो किसी block में create किये जाते है, जैसे की कोई function या control statement block आदि। ऐसे variables सिर्फ उस block में ही access/use किये जा सकते है जिनमें इन्हें define किया जाता है। इसका उदाहरण निचे दिया जा रहा है।

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


int myFunction()
{
     // Local variable num
     int num=10;
     cout<<num;
}

int main()
{
    cout<<"Hello!";

    // Error: Accessing local variable in global scope
    cout<<num;

    return 0;
}
```

ऊपर दिए गए उदाहरण में myFunction() function में create किये गए variable को main() function में access किया जा रहा है। ऐसा करने पर error generate होगी क्योंकि num एक local variable और उसे

myFunction() में ही use किया जा सकता है। ये program निचे दिया गया output generate करता है।

C++ in Hindi – Constants

- Introduction to **C++ constants in Hindi**
- Defining *C++ constants in Hindi*
- Example of C++ constants in Hindi
-

Introduction to C++ Constants

C++ में constants और variables एक समान ही होते है। Constants को भी variables की तरह ही define किया जाता है, इनके data type भी variables के जैसे ही होते है और memory allocation भी variables की तरह ही होता है।

एक constant और variable में फर्क सिर्फ इतना होता है की variable की value run time में किसी भी operation द्वारा change की जा सकती है लेकिन constant की value run time में किसी भी तरह change नहीं की जा सकती है।

उदाहरण के लिए आपने एक variable Num create किया है जिसको आपने 0 value assign की है। Program के run होने पर आप इसमें 2+2 add करके Num variable में store करवाते है।

```
int Num = 0;


Num = 2+2;   // Num now have 4
```

इसके बाद Num variable की value 4 हो जाती है जिसे आप print करवाते है। ऐसा आसानी से हो जाता है क्योंकि Num एक variable है और इसकी value आसानी से change हो सकती है।

लेकिन यदि ऊपर दिए गए उदाहरण में Num एक constant होता है और उसकी value को 2+2 expression द्वारा change करने का प्रयास किया जाता तो ऐसा होने पर program में error generate होती है।

Constant को value उन्हें create करते समय ही assign की जाती है। एक बार program run होने के बाद constant की value को change नहीं किया जा सकता है।

C++ में 2 प्रकार के constants होते है। इनके बारे में निचे बताया जा रहा है।

Literal Constants

एक literal constant direct value होती है। जैसे की 5, 5.4, hello world आदि। Program में जब भी इनकी आवश्यकता होती है तो इन्हे directly use किया जाता है। उदाहरण के लिए कोई variable Num है की जिसकी value 10 है। आप Num variable की value को 2 से divide करके result variable में result store करना चाहते है।

```
int Num, Result; Num = 10;// 2 is literal, used directly in program
Result = Num/2;
```

ऐसे में आप Num/2 इस प्रकार लिखेंगे। यँहा पर 2 एक literal constant (direct value) है। इसे directly program में define किया गया है।

Literal constants की के साथ सबसे बडी problem यह होती है की यदि आपने इन्हे program में कई बार use किया है और बाद में आपको इन्हे change करने की आवश्यकता होती है तो आपको उन्हें सभी जगह पर ढूंढ के manually change करना होगा। इस problem का solution Symbolic constants होते है।

Symbolic Constants

Literal constant की तरह symbolic constant कोई value नहीं होती है। Symbolic constant एक नाम होता है जो एक memory location को represent करता है जँहा पर value stored रहती है।

उस नाम के द्वारा आप value को access कर पाते है जैसा की आप variables के साथ करते है। लेकिन जैसा की आपको पता है आप उस value को change नहीं कर सकते है।

Symbolic constant को आप program में कितनी भी बार use कर सकते है। जब आप इसकी value change करते है तो वह value program में automatically change हो जाती है। इसलिए literal constants की अपेक्षा symbolic constants अधिक उपयोगी होते है।

Defining C++ Constants

C++ में literal constants को आप आसानी से use कर पाते है लेकिन symbolic constants को आप पहले define करते है और उसके बाद use कर करते है। C++ में symbolic constants को आप दो प्रकार से define कर सकते है। इनके बारे में निचे बताया जा रहा है।

Using #define Preprocessor

C++ में constants को #define preprocessor के द्वारा define करने का general syntax निचे दिया जा रहा है।

```
#define constant-name value
```

जब आप #define preprocessor द्वारा constant define करते है तो ऐसा आप program की शुरआत में header files को include करने के बाद करते है। जब आप #define preprocessor के द्वारा constant define करते है तो किसी प्रकार का data type नहीं define करते है।

सबसे पहले आप #define preprocessor को define करते है। इसके बाद आप constant का नाम देते है और उसके बाद space दे कर उसकी value देते है।

Using const Keyword

C++ में constant define करने के लिए const keyword को use करने का general syntax निचे दिया जा रहा है।

```
const-keyword data-type constant-name = value;
```

आप const keyword के प्रयोग से किसी particular type (int, float, char आदि) का constant create कर सकते है।

सबसे पहले आप const keyword define करते है। इसके बाद आप वह data type define करते है जिसका constant आप create करना चाहते है। इसके बाद आप constant का नाम लिखते है और उसके बाद assignment operator (=) लगाकर वह value लिखते है जिसे आप उस constant name के द्वारा access करना चाहते है।

Example of C++ Constants

C++ में constants के use को निचे उदाहरण द्वारा समझाया जा रहा है।

```
#include<iostream>
using namespace std; int main()
{
   // Defining constant.
   const int Tax = 5;    int Bill, Total;    cout<<“Please enter item amount”;
   cin>>Bill;    // Tax can not be changed while calculating bill.
   Total =Bill+((Bill*Tax)/100);    cout<<“Total billing amount with tax is : “<<Total;

}
```

ऊपर दिया गया उदाहरण निचे दिया गया output generate करता है।

Please enter item amount : 300
Total billing amount with tax is : 315

C++ in Hindi – Operators

- Introduction to **C++ operators in Hindi**
- Different C++ operators in Hindi
-

Introduction to C++ Operators

किसी भी C++ program में data पर operations perform किये जाते है। इसके लिए सबसे पूर्व data को variables के द्वारा computer memory में store किया जाता है। इसके बाद C++ द्वारा supported operators को use करते हुए उस data पर operations perform किये जाते है।

उदाहरण के लिए आप दो variables जिनमे integer values store की गयी है उनके बीच addition operation perform करते है। ऐसा आप addition operator (+) द्वारा करेंगे।

C++ आपको अलग अलग operations perform करने के लिए अलग अलग type के operators provide करती है। जिन्हें आप variables के साथ प्रयोग करते है।

उदहारण के लिए यदि आप program में arithmetic operations जैसे की addition, subtraction, division आदि perform करना चाहते है तो इसके लिए arithmetic operators +, -, / आदि का प्रयोग करते है।

उसी प्रकार यदि आप program में logic (data को compare करना) perform करना चाहते है तो उसके लिए relational operators जैसे की <, >, = आदि use करते है।

Operators के साथ define किये जाने वाले variables operands कहलाते है। उदाहरण के लिए आपने 2 variables a और b create किये है। अब आप इन दोनों के बीच addition operation perform करके result c variable में store करना चाहते है। आप इस प्रकार code लिखते है।

```
c = a + b;
```

ऊपर दिए गए code में a और b operands है। क्योंकि इन दोनों को addition (+) operator के साथ define किया गया है। इस code में c भी एक operand है क्योंकि इसे assignment (=) operator के साथ define किया गया है।

C++ में दो तरह के operators पाए जाते है। इनके बारे में निचे बताया जा रहा है।

- Binary – ऐसे operators जो दो operands के साथ operation perform करते है binary operators कहलाते है। उदाहरण के लिए सभी arithmetic operators binary operators होते है।
- Unary – ऐसे operators जो एक operand के साथ operation perform करते है unary operator कहलाते है। उदाहरण के लिए increment (++) और decrement (–) operators unary operators होते है। इन्हे एक ही variable (operand) के साथ use किया जाता है।

C++ में available विभिन्न operators के बारे में आगे detail से बताया जा रहा है।

Arithmetic Operators

Arithmetic operators वे operators होते है जिनके द्वारा arithmetic operations perform किये जाते है। C++ द्वारा supported arithmetic operators की list निचे दी जा रही है।

Operator	Explanation	Syntax
+ (Addition)	Adds a and b	a + b
- (Subtraction)	Subtract b from a	a - b
* (Multiplication)	Multiply a and b	a * b
/ (Division)	Divide a by b	a / b
% (Modulus)	Divide a by b and return remaining value	a % b

Relational Operators

Relational operators data को compare करने या data के बीच relation define करने के लिए प्रयोग किये जाते है। ये operators true और false के रूप में boolean value return करते है। C++ में available relational operators की list निचे दी जा रही है।

Operator	Explanation	Syntax
> (Greater than)	Checks whether a is greater than b	a > b
< (Lesser than)	Checks whether a is lesser than b	a < b
>= (Greater than & equal)	Checks whether a is greater than & equal to b	a >= b
<= (Lesser than & equal)	Checks whether a is lesser than & equal to	a <= b
== (Equal)	Checks whether a is equal to b	a == b
!= (Not equal)	Checks whether a is not equal to b	a != b

Logical Operators

Logical operators program में logic perform करने के लिए प्रयोग किये जाते है। ये operators conditional expressions के बीच प्रयोग किये जाते है और boolean value true या false return करते है। C++ में available logical operators के बारे में निचे दिया जा रहा है।

Operator	Explanation	Syntax
&& (AND)	returns true if both conditions are true	cond1 && cond2
|| (OR)	returns true if any of the two condition is true	cond1 || cond2

! (NOT)	returns true if condition is not true, Unary operator	!cond

Bitwise Operators

Bitwise operators program में bit operations perform करने के लिए प्रयोग किये जाते है। सबसे पहले variables की values को bits में convert किया जाता है और उसके बाद उन पर operations perform किये जाते है। ये operators true और false return करते है। C++ में available bitwise operators की list निचे दी जा रही है।

Operator	Explanation	Syntax
& (Binary AND)	Returns bits which are identical in a and b	a & b
| (Binary OR)	Returns all bits from a and b	a | b
^ (XOR)	Returns bits which are in a but not in b	a ^ b
~ (Binary Ones compiment)	Unary operator, Change 0 bits to one and 1 bit to 0	~a
<< (Left Shift)	Shifts bits of a to left by number given on right side of <<	a << number
>> (Right Shift)	Shifts bits of a to right by number given on right side of >>	a >> number

Assignment Operators

Program में assignment operations perform करने के लिए assignment operators प्रयोग किये जाते है। C++ में available assignment operators के बारे में निचे दिया जा रहा है।

Operator	Explanation	Syntax
= (Assignment)	Assign value of a to b	a = b
+= (Plus and assignment)	Add value of a & b then assign result to a	a += b
-= (Minus and assignment)	Subtract value of b from a then assign result to a	a -= b
*= (Multiply and assignment)	Multiply a & b then assign result to a	a *= b
/= (Divide and assignement)	Divide a by b then assign result to a	a /= b
%= (Modulus and assignment)	Divide a by b then assign remaining value to a	a %= b

Conditional Operator (?:)

C++ आपको conditional operator provide करती है। यह operator ? और : के combination से बनता है। यह operator if else statement का short रूप होता है। इस operator का general syntax निचे दिया जा रहा है।

```
condition ? true-part : false-part
```

जैसा की आप ऊपर दिए गए syntax में देख सकते है सबसे पूर्व condition define की जाती है। इसके बाद question mark (?) symbol लगाया जाता है।

इस symbol के बाद वह statement लिखा जाता है जो condition true होने पर use किया जाएगा। इसके बाद colon (:) symbol लगाया जाता है। Colon symbol के बाद वह code लिखा जाता है जो condition के true होने पर use किया जाएगा।

इस operator को मुख्यतः condition के आधार पर variables को value assign करने के लिए प्रयोग किया जाता है।

sizeof() Operator

C++ में sizeof operator किसी भी variable की size return करता है। इसका syntax निचे दिया जा रहा है।

```
sizeof(variable-name);
```

जिस variable की size आप पता करना चाहते है उसका नाम brackets में लिखा जाता है।

* (Pointer) Operator

Pointer variable define करने के लिए pointer operator का प्रयोग किया जाता है। इस operator का general syntax निचे दिया जा रहा है।

```
data-type *variable-name;
```

Pointer variable का प्रयोग किसी दूसरे operator का address store करने के लिए किया जाता है।

& (address of) Operator

C++ में address of operator किसी भी variable का address return करता है। इस operator का syntax निचे दिया जा रहा है।

```
pointer-variable = &variable-name;
```

Address of operator को उस variable के नाम से पूर्व define किया जाता है जिसका variable आप access करना चाहते है। Address को किसी pointer variable में store किया जाता है।

C++ in Hindi – Flow Control

- Introduction to **C++ flow control in Hindi**
- Types of flow control statements in Hindi

Introduction to C++ Flow Control

Normally जब भी कोई C++ program execute होता है तो पहला दूसरा तीसरा ऐसे ही sequence में सारे statements execute होते है। Program execution की यह sequence program का execution flow कहलाती है।

इस execution flow को आप control कर सकते है। आप चाहे तो किसी statement के execution को skip कर सकते है, किसी statement को एक से ज्यादा बार execute करवा सकते है या फिर program में एक statement से दूसरे statement पर jump कर सकते है।

Program के execution flow को control करने के लिए C++ आपको कुछ built in flow control statements provide करती है। C++ आपको 3 प्रकार के flow control statements provide करती है।

1. Selection Statements
2. Looping Statements
3. Jump Statements

इन statements के बारे में निचे detail से दिया जा रहा है।

Selection Statements

ये वो statements होते है जो किसी statement को condition के base पर execute करते है। जब condition match होती है तो statement execute हो जाता है नहीं तो उसे skip कर दिया जाता है। Selection

statements 4 प्रकार के होते है। निचे इनके बारे में detail से बताया जा रहा है।

If Statement

If statement एक condition block होता है। Condition true होने पर इस block में दिए गए सभी statements execute हो जाते है। यदि condition true नहीं है तो इस block का कोई भी statement execute नहीं होता है। If statement का general syntax निचे दिया जा रहा है।

```
if(condition)
{
    // Statements to be executed when condition is true.
}
```

If statement को निचे उदाहरण के माध्यम से समझाया जा रहा है।

```
if(5>3)
{
     cout<<"5 is greater than 3";
}
```

If-else Statement

If else statement में if statement के साथ else block भी जोड़ दिया जाता है। Else block में वे statements होते है जो condition के false होने पर execute होंगे। इसका general syntax निचे दिया जा रहा है।

```
if(condition)
{
     // Statements to be executed when condition is true.
}
else
{
     // Statements to be executed when condition is false.
}
```

If else statement को निचे उदाहरण के माध्यम से समझाया जा रहा है।

```
if(5>3)
{
     cout<<"5 is greater than 3";
}
else
{
     cout<<"5 is less than 3";
}
```

Nested If

जब आप एक if block में दूसरा if block define करते है तो वह nested if कहलाता है। ऐसे आप कितने भी if block define कर सकते है। इसका general syntax निचे दिया जा रहा है।

```
if(condition)
{
       if(condition)
       {


       }

}
```

इसका उदाहरण निचे दिया जा रहा है।

```
if(num>0)
{
      if(num<2)
      {
                cout<<"Num is 1";
      }
}
```

Switch Case

Switch case एक ऐसा block होता है जिसमें आप अपनी choice pass करते है। Switch block में define किये गए जिस case से आपकी choice match हो जाती है उसी case के statements execute हो जाते है। इसका general syntax निचे दिया जा रहा है।

```
switch(choice)
{
   case 1:
                // Statements to be executed;
                break;
  case 2:
                // Statements to be executed;
                break;
  case 3:
               // Statements to be executed;
               break;
  default:
               // Statements to be executed;
               break;


}
```

जब आपकी choice से कोई भी case match नहीं करता है तो default case execute हो जाता है। सभी cases के statements के बाद break statement लगाना अनिवार्य है। इसे निचे उदाहरण के माध्यम से समझाया जा रहा है।

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


int main()
{
    int choice=4;

    // Passing case in switch
    switch(choice)
    {
case 1:
                    cout<<"This is case 1";
                    break;
case 2:
                    cout<<"This is case 2";
                    break;
 case 3:
                    cout<<"This is case 3";
                    break;
 default:
                cout<<"Please enter number between 1 to 3";
                    break;
    }

     return 0;
}
```

ऊपर दिया गया program निचे दिया गया output generate करता है।

```
Please enter number between 1 to 3
```

Looping Statements

एक loop statement ऐसा block होता है जिसमें define किये गए blocks एक से अधिक बार execute किये जाते है। यदि आप किसी statement को एक से अधिक बार execute करवाना चाहते है तो इसके लिए आप एक loop use कर सकते है। C++ में looping statements 3 प्रकार के है। इनके बारे में निचे दिया जा रहा है।

While loop

While loop एक ऐसा block होता है जिसमें दिए गए statements तब तक बार बार execute होते है जब तक की दी गयी condition true होती है। जैसे ही condition false होती है loop terminate हो जाता है। यदि condition false ना हो तो loop infinitely चलता रहेगा, इसलिए आप loop control variable को increase करते है।

Loop control variable condition में define किया जाता है। हर loop iteration में इसे increase किया जाता है ताकि condition false हो सके और loop terminate हो जाए। While loop का general syntax निचे दिया जा रहा है।

```
while(condition)
{
      // Statements to be executed;
      // X++, where X is loop control variable;
}
```

While loop को निचे उदाहरण के माध्यम से समझाया जा रहा है।

_____

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


int main()
{
        int num=0;

        // While loop iterating until num is less than 5
        while(num<5)
        {
            cout<<"Best Hindi Tutorials"<<"n";

            // Incrementing num to reach the loop termination
condition
            num++;
        }

        return 0;
}
```

ऊपर दिए उदाहरण में जैसे ही num की value 5 होगी loop terminate हो जाएगा। ये program निचे दिया गया output generate करता है।

```
Best Hindi Tutorials
Best Hindi Tutorials
Best Hindi Tutorials
Best Hindi Tutorials
Best Hindi Tutorials
```

Do While Loop

एक do while loop भी while loop की तरह ही होता है। Do while loop में ख़ास बात ये है की ये कम से कम एक बार जरूर execute होता है फिर चाहे condition true हो या फिर flase हो। साथ ही do while loop में

condition loop के आख़िर में check की जाती है। Do while loop का general syntax निचे दिया जा रहा है।

```
do
{
    statements to be executed;
    X++;  // Where X is loop control variable
}while(condition)
```

जैसा की आप ऊपर दिए गए syntax में देख सकते है condition के check होने से पहले do block में दिए गए statements execute होंगे और इसके बाद condition check होगी।

यदि condition false होती है तो loop terminate हो जायेगा नहीं तो do block वापस execute होगा। Loop control variable को do block में ही increase किया जायेगा। आइये अब इसे एक उदाहरण के माध्यम से समझने का प्रयास करते है।

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


int main()
{

    int num=0;

    // Checking condition after executing statements
    do
    {
       cout<<"Best Hindi Tutorials";
       num++;
```

```
    }while(num>10);

    return 0;

}
```

ऊपर दिया गया program निचे दिया गया output generate करता है।

```
Best Hindi Tutorials
```

For Loop

सभी loops में for loop सबसे simple loop माना जाता है। For loop में loop control variable का initialization, condition और increment तीनों एक ही statement में लिखे जाते है। इसके बाद block में वे statements लिखे जाते है

जिन्हें आप condition true होने पर execute करना चाहते है। For loop का general syntax निचे दिया जा रहा है।

```
for(initialization;condition;increment)
{
     // Statements to be executed when condition is true.
}
```

आइये अब for loop के use को एक उदाहरण के माध्यम से समझने का प्रयास करते है।

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


int main()
{
    int num;

    // 3 times iterating for loop
    for(num=0;num<=2;num++)
    {
          cout<<"Best Hindi Tutorials"<<"n";
    }

    return 0;
}
```

ऊपर दिया गया program निचे दिया गया output generate करता है।

```
Best Hindi Tutorials
Best Hindi Tutorials
Best Hindi Tutorials
```

Jump Statements

Jump statements वे statements होते है जो execution control को एक statement से दूसरे statement को pass कर देते है। C++ में jump statements 3 प्रकार के होते है। इनके बारे में निचे दिया जा रहा है।

Break Statement

Break statement loop को terminate करने के लिए use किया जाता है। जब भी आप किसी block में break statement use करते है तो इसे

execute होते है ही program का control उस block से बाहर आ जाता है। इसका general syntax निचे दिया जा रहा है।

```
break;
```

Break statement के use को निचे उदाहरण के माध्यम से समझाया गया है।

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


int main()
{
    int num;
    for(num=0;num<5;num++)
    {
          if(num==3)
          {
              cout<<"Loop terminated by break statement";

              // Terminating loop on 4th iteration
              break;
          }
          else
          {
               cout<<"Best Hindi Tutorials"<<"n";
          }
   }

    return 0;
}
```

ऊपर दिए गए उदाहरण में जैसे ही loop की तीसरी iteration आती है loop break statement द्वारा terminate कर दिया जाता है। ये program निचे दिया गया output generate करता है।

```
Best Hindi Tutorials
Best Hindi Tutorials
Best Hindi Tutorials
Loop terminated by break statement
```

Continue Statement

Continue statement loop की iteration को skip करने के लिए use किया जाता है। Continue statement के use से loop उस iteration को skip करके next iteration को execute करता है। इसका general syntax निचे दिया जा रहा है।

```
continue;
```

Continue statement को निचे उदाहरण के माध्यम से समझाया गया है।

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


int main()
{
      for(int num=0;num<5;num++)
      {
            if(num==2)
            {
                  cout<<"Third iteration skipped!"<<"n";

                  // Skipping third iteration of loop
```

```
                continue;
            }
            else
            {
                cout<<"Best Hindi Tutorials"<<"n";
            }
      }

       return 0;
}
```

ऊपर दिया गया program निचे दिया गया output generate करता है।

```
Best Hindi Tutorials
Best Hindi Tutorials
Third iteration skipped!
Best Hindi Tutorials
Best Hindi Tutorials
```

Go to Statement

Goto statement program में एक point से किसी दूसरे point पर jump करने के लिए use किया जाता है। ये jump किसी label पर किया जाता है, इसलिए जब भी आप program में कँही jump करना चाहते है तो पहले label define करते है। एक label define करने का general syntax निचे दिया जा रहा है।

```
label-name:
```

जैसा की आप ऊपर दिए गए syntax में देख सकते है, सबसे पहले आप label का नाम define करते है और उसके बाद colon लगाते है। किसी भी label

पर jump करने के लिए आप goto statement define करते है। इसका general syntax निचे दिया जा रहा है।

```
goto label-name;
```

Goto statement के use को निचे उदाहरण के माध्यम से समझाया गया है।

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


int main()
{

     cout<<"Hello Reader!";
     goto bhtLabel;


     if(5>3)
     {
          cout<<"This will not be executed!";
     }

     bhtLabel:
      cout<<"Hello Reader Again!";

     return 0;
}
```

ऊपर दिए गए उदाहरण में control if block को skip करते हुए सीधा bhtLabel पर jump कर रहा है। ये program निचे दिया गया output generate करता है।

```
Hello Reader!
Hello Reader Again!
```

C++ in Hindi – Arrays

- Introduction to **C++ arrays in Hindi**
- Declaring *C++ arrays in Hindi*
- Initializing C++ arrays in Hindi
-

Introduction to C++ Arrays

एक array similar type (data type) की values का collection होता है। ये values sequence में होती है और contiguous memory locations (एक के बाद एक series में) में store की जाती है। ये सभी values एक ही नाम से represent और access की जाती है जिसे array का नाम कहते है। Array का नाम आप array declare करते समय define करते है।

Array की size fixed होती है और ये compile time पर (program के run होने से पहले) ही array को declare करते समय define की जाती है। जितनी array की size होती है आप उतने ही elements array में store कर सकते है।

उदाहरण के लिए आप बहुत से integer numbers एक नाम से store करने के लिए एक integer array create कर सकते है। C++ में आप char type के variable में एक बार में एक ही character store कर सकते है।

लेकिन मान लीजिये आप किसी व्यक्ति का नाम store करना चाहते है तो आप एक character array create कर सकते है और उसमे आवश्यकता के अनुसार characters store कर सकते है। इसी प्रकार आप float type का array बनाकर उसमे float numbers store कर सकते है।

C++ में arrays create करने के 2 steps है।

- Declaring an array
- Initializing an array

इन दोनों steps के बारे में नीचे detail से दिया जा रहा है।

Declaring An Array

इस step में आप array का नाम, उसका type और उसकी size define करते है। Array declaration का general syntax नीचे दिया जा रहा है।

```
data-type array-name [size];
```

सबसे पहले आप array का type define करते है जैसे की int और char आदि। इसके बाद आप array का एक unique नाम देते है। इसके बाद आप array की size angular brackets में define करते है। इसको नीचे उदाहरण के द्वारा समझाया जा रहा है।

```
char HelloArray[5];
```

ऊपर दिया गया statement HelloArray नाम से array declare करता है। इस array में आप कोई भी 5 characters store कर सकते है।

Array को declare करते ही उसकी size के according continuous memory locations allot कर दी जाती है। इन memory location को index numbers से identify किया जाता है। Array की index हमेशा zero से शुरू होती है। इसलिए array का first element 0th index पर store होता है। इसी प्रकार array का second element first index पर store होता है।

यदि आपके array की size N है तो first element [0] index पर और last element [N-1] index पर stored रहता है। यानी की ऊपर दिए गए उदाहरण में HelloArray में जब आप values store करवाएंगे तो पहली value [0] index पर और आखिरी value [4] index पर stored होगी।

Initializing an Array

इस part में आप array में initial values store करवाते है। हालाँकि ऐसा आप array को declare करते समय भी कर सकते है। इसके लिए आप size declare करने के बाद assignment operator (=) लगाकर curly braces में values को comma से separate करके लिखते है।

```
data-type array-name[size] =
{value1,value2,value3,….,valueN};
```

इसका उदाहरण नीचे दिया जा रहा है।

```
char HelloArray[5] = {‘H’,’E’,’L’,’L’,’O’};
```

जैसा की आप ऊपर दिए गए उदाहरण में देख सकते है HelloArray को declare करने के साथ ही initialize भी किया गया है।

आप चाहे तो हर एक element को अलग से भी store करवा सकते है। इसके लिए आप index numbers को यूज करते है। सबसे पहले आप array का नाम लिखते है और उसके बाद वो index number [] angular brackets में लिखते है जिस location पर आप value store करना चाहते है।

इसके बाद आप assignment operator लगाकर value लिखते है। यदि value character है तो उसे quotes में लिखा जायेगा। इसका syntax नीचे दिया जा रहा है। इसका syntax नीचे दिया जा रहा है।

```
array-name[index-number] = value;
```

उदाहरण के लिए आप यदि HelloArray में इस तरीके से value store करना चाहते है तो ऐसा आप इस प्रकार कर सकते है।

```
HelloArray[0] = 'H' ;
HelloArray[1] = 'E';
HelloArray[2] = 'L';
HelloArray[3] = 'L';
HelloArray[4] = 'O';
```

Taking Array Elements From User

आप चाहे तो array elements user से भी input करवा सकते है। इसके लिए आप loop का इस्तेमाल करते है। Loop को array की size के बराबर ही चलाया जाता है। फिर उस loop के अंदर आप cin>> statement का प्रयोग करते हुए array elements को user से enter करवाते है।

मान लीजिये आप ऊपर दिए गए HelloArray के elements user से input करवाना चाहते है। ऐसा आप इस प्रकार कर सकते है।

```
cout<<"Please enter five character elements";
for(int i=0; i<=5;i++)
{
    cin>>HelloArray[i];
}
```

Displaying Array Elements

Array elements को आप दो तरह से display कर सकते है। इन दोनों तरीकों के बारे में नीचे detail से दिया जा रहा है।

Displaying One Array Element at a Time

जिस प्रकार आप एक बार में एक array element को initialize करते है उसी प्रकार आप एक element को display भी कर सकते है। उदाहरण के लिए यदि आप HelloArray के elements को एक एक करके display करना चाहते है तो ऐसा आप इस प्रकार कर सकते है।

```
cout<<"First element is :"<<HelloArray[0];
cout<<"Second element is :"<<HelloArray[1];
cout<<"Third element is :"<<HelloArray[2];
cout<<"Fourth element is :"<<HelloArray[3];
cout<<"Fifth element is :"<<HelloArray[4];
```

Displaying Complete Array

यदि आप सभी array elements को एक साथ display करना चाहते है तो इसके लिए आप loop का इस्तेमाल करते है। Loop को array की size के बराबर चलाया जाता है। उदाहरण के लिए यदि आप सभी HelloArray elements को एक साथ display करना चाहते है तो ऐसा आप इस प्रकार कर सकते है।

```
cout<<"Array elements are";
for(int i=0;i<=5;i++)
{
      cout<<HelloArray[i];
}
```

C++ में array के use का निचे एक complete उदाहरण दिया जा रहा है।

```
#include <iostream>
using namespace std;
int main()
{
   // Array declaration
   char HelloArray[5];
   cout<<"Please enter five character elements";

   // Reading array elements at runtime
   for(int i=0; i<=5;i++)
   {
       cin>>HelloArray[i];
   }
   cout<<"Array elements are";


    // Printing array elements to screen

   for(int i=0;i<=5;i++)
   {
       cout<<HelloArray[i];
   }
   return 0;
}
```

ऊपर दिया गया program निचे दिया गया output generate करता है।

```
Please enter five character elements
5
4
3
2
1
Array elements are : 5  4  3  2  1
```

Two Dimensional Arrays

अभी तक आपने array में जितने भी elements input करवाये वो एक list की form में store हुए है। इस प्रकार के arrays को one dimensional arrays कहा जाता है। इस प्रकार के arrays में elements list के रूप में stored होते है और उसी रूप में display भी किये जाते है।

यदि आप चाहे तो data को tabular form में भी store कर सकते है और बाद में उसे एक table के रूप में present कर सकते है। ऐसे arrays को two dimensional arrays कहा जाता है।

Creating Two Dimensional Array

Two dimensional arrays को row और column के संदर्भ में declare किया जाता है। इसका general syntax निचे दिया जा रहा है।

```
data-type array-name [row][column];
```

जैसा की आप ऊपर दिए गए syntax में देख सकते है first angular brackets में rows की सँख्या define की जाती है और दूसरे angular bracket में columns की सँख्या define की जाती है।

उदाहरण के लिए यदि आप एक array declare करना चाहते है जिसमे 3 rows हो और 3 ही columns हो तो ऐसा आप इस प्रकार कर सकते है।

```
int NumArray[3][3];
```

Initializing Two Dimensional Arrays

Two dimensional arrays में value input करवाना बेहद आसान है। इसे और आसानी से करने के लिए आपको सबसे पहले अपने दिमाग में एक table

की form बनानी चाहिए। इसके बाद आप जिस row के जिस column में value input करना चाहते है उनकी संख्या लिखते है।

उदाहरण के लिए यदि आप ऊपर दिए गए NumArray में पहली row के पहले column में value input करना चाहते है तो ऐसा आप इस प्रकार करेंगे।

```
NumArray[1][1] = 101;
```

इसी प्रकार यदि आप पहली row के दूसरे column में value input करवाना चाहते है तो ऐसा आप इस प्रकार करेंगे।

```
NumArray[1][2] = 102;
```

यदि आप user से input लेना चाहते है तो इसके लिए आप 2 loops का इस्तेमाल करते है। पहला लूप rows को iterate करता है और दूसरा लूप columns को iterate करता है। इसका उदाहरण निचे दिया जा रहा है।

```
cout<<"Please enter array elements";
// It will iterate rows, i represents row number

for(int i=0;i<3;i++)
{
    // It will iterate columns, j represents column number
    for(int j=0;j<3;j++)
    {
          // Enter value in j number column of i number row
          cin>>NumArray[i][j];
     }
}
```

Displaying Two Dimensional Array Elements

यदि आप एक बार में एक element display करना चाहते है तो ऐसा आप इस प्रकार कर सकते है।

```
// Will display first row first column value
cout<<NumArray[1][1];
```

यदि आप सभी array elements को display करना चाहते है तो इसके आपको 2 loops इस्तेमाल करने होंगे। इसका उदाहरण निचे दिया जा रहा है।

```
cout<<"Array elements are :"<<endl;
for(int i=0;i<3;i++)
{
   for(int j=0;j<3;j++)
  {
       cout<<NumArray[i][j];
  }
}
```

Two dimensional array का complete उदाहरण निचे दिया जा रहा है।

```
#include <iostream>
using namespace std;
int main()
{


// Two dimensional array declaration

int NumArray[3][3];
cout<<"Please enter array elements";
// For loop iterating rows of two dimensional array

for(int i=0;i<3;i++)
{
    // For loop iterating columns of two dimensional array
    for(int j=0;j<3;j++)
    {
          // Enter value in j number column of i number row
          cin>>NumArray[i][j];
     }
}
cout<<"Array elements are :"<<endl;
for(int i=0;i<3;i++)
{
   for(int j=0;j<3;j++)
  {
       // Printing out two dimensional array elements
       cout<<NumArray[i][j]<<"t";
  }
  cout<<endl;
}

return 0;
}
```

ऊपर दिया गया program निचे दिया गया output generate करता है।

```
Please enter array elements
9
8
7
6
5
4
3
2
1
Array elements are :
9  8  7
6  5  4
3  2  1
```