1

C++ भाषा में प्रोग्राम बनाना
(Programming in C++)

1.1 Getting Started

हम C++ में प्रोग्राम बनाने से पहले एक बने हुये प्रोग्राम का अध्ययन करते हैं। हम इससे यह जान पायेंगे कि C++ प्रोग्राम की संरचना कैसी होगी।

```
// sample.CPP                                          टिप्पणियाँ
// To demostrate Program structure
# include <iostream.h>
 Class XYZ                                             क्लास निर्धारण
  {
   Private :
        int A;                                         क्लास के सदस्य आँकडे
   Public:
        Void getdata (int x)                           क्लास के सदस्य फंक्शन
            {A = x;}
        Void showdata ( )
            {count <<"\ n data is " << A;}
  };
        Void main ( )
            {
              XYZ      OBJ1, OBJ2;                     ऑब्जेक्ट की घोषणा
        OBJ1.getdata (15);                             आँकड़े निर्धारित करने
        OBJ2.getdata (35);                             के लिए सदस्य फंक्शन चलाना
        OBJ1.showdata ( );
        OBJ2.showdata ( );                             आँकडे प्रदर्शित करने के लिए
        }
```

उपरोक्त प्रोग्राम की प्रथम दो पंक्तियां '//' अक्षर से आरम्भ हो रही हैं इनको टिप्पणी पंक्तियां (कमेन्ट लाइन) कहते हैं। ये पंक्तियाँ कम्प्यूटर द्वारा निष्पादित (Execute) नहीं की जाती। ये केवल प्रोग्राम को समझने योग्य बनाने के लिये होती है।

'//' से आरम्भ होने वाली सभी पंक्तियां टिप्पणी पंक्तियां कहलाती हैं। हम किसी भी पंक्ति के पहले '//' अक्षर लगाकर उस पंक्ति को टिप्पणी पंक्ति में बदल सकते हैं। ये पंक्तियां प्रोग्राम का भाग होती है परन्तु न तो कम्पुटर इन्हे पढ़ता है और न ही इनसे कोई कार्य होता है।

अगली पंक्ति # अक्षर से आरम्भ हैं। इसे प्री प्रोसेसर डायरेक्टिव कहते हैं। यह कम्पाइलर को स्त्रोत प्रोग्राम (Source code) में अन्य फाइलों को शामिल करने का निर्देश देता है।

अगली पंक्ति में क्लास को निर्धारण हेतु निर्देश है। इस पंक्ति में Class सुरक्षित शब्द है तथा XYZ क्लास का नाम है। सुरक्षित शब्दों का प्रत्येक भाषा में विशेष अर्थ होता है। कोष्ठक के मध्य क्लास की पूरी परिभाषा है। परिभाषा में प्रथम पंक्ति में Private लिखा है तथा अगली पंक्ति में एक चर A, पूर्णांक (integer) प्रकार का घोषित है। ये कम्पाइलर को बताते है कि A एक चर है जो पूर्णांक प्रकार का है तथा यह Private है। अथार्त यह चर, सदस्य फंक्शन को छोड़कर अन्य फंक्शनों के लिऐ उपलब्ध नहीं होगा। जबकि Public भाग कें ऑकड़े या फंक्शन मुख्य प्रोग्राम या अन्य फंक्शनों के लिए भी उपलब्ध रहते है।

जैसा कि Public भाग में दो फंक्शन getdata व showdata है। इन फंक्शनों को मुख्य प्रोग्राम अथवा अन्य फंक्शन में सीधे उपयोग (Call द्वारा) किया जा सकता है।

अगली पंक्ति में class की परिभाषा समाप्त होती है। अगली पंक्ति Void main ( ) मुख्य प्रोग्राम का आरम्भ बताती है। Void इस बात की सूचना देता है कि यह मुख्य प्रोग्राम किसी प्रकार का आउटपुट नहीं देता। इसी प्रकार main के आगे खाली कोष्ठक यह सूचना देता है कि यह प्रोग्राम बिना पैरामीटर (No argumint) का है।

अगली पंक्ति दो ऑब्जेक्ट उत्पन्न करने का निर्देश देती है । ये ऑब्जेक्ट XYZ क्लास प्रकार के होंगे। वास्तव में जब ऑब्जेक्ट उत्पन्न होते हैं तभी उनके ऑकड़ों को, संग्रहित करने के लिए मैमेरी में जगह मिलती है। इस पंक्ति से दोनों ऑब्जेक्ट के लिए OBJ1.A, OBJ2.A दो चर उत्पन्न होंगे।

अगली दो पंक्तियाँ ऑब्जेक्ट के लिए सदस्य फंक्शनों को चला कर ऑब्जेक्ट के चरों का मान निर्धारित करती है।

अगली दो पंक्तियाँ पुनः ऑब्जेक्ट के दूसरे सदस्य फंक्शनों को चलाकर ऑब्जेक्ट के चरों का मान स्क्रीन पर प्रदर्शित करती है।

अन्तिम पंक्ति में Main प्रोग्राम की समाप्ति का कोष्ठक हैं।

1.2 C++ में काम आने वाले अक्षर (C++ character set)

कम्प्यूटर में लिखे जाने वाले C++ के प्रोग्राम के शब्द, संख्याऐं, समीकरण, आदि जिन अक्षरों से मिल कर बनते है उन्हें C++ भाषा के अक्षरों का समूह (character set) कहते हैं। अक्षरों के समूह को हम इन भागों में बांट सकते है।

(1) अक्षर (Alphabate)

(2) अंक (Digit)

(3) विशेष अक्षर (Special character)

(4) खाली जगह (White space)

(1) अक्षर :– अंग्रेजी वर्णमाला में काम आने वाले अक्षर

A, B, C, D ....Z, a, b, c, d, ...z

(2) अंक :– 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

(3) विशेष अक्षर

,	Comma	&	Ampersand
.	Period	^	Caret
;	Semicolon	*	Asterisk
?	Question mark	-	Minus sign
'	Apostrope	+	Plus sign
"	Quotation mark	<	Less than sign
!	Exclamation mark	>	Greater than
|	Vertical bar	(	Left Parenthesis
/	Slash	)	Right Parenthesis
\	Backslash	[	Left brace
}	Right bracket	_	Underscore
{	Left bracket	~	Tilde Sign
$	Dollar sign	]	Right brace
%	Percent sign	#	Number sign

खाली जगह (White space)

(1) Blank space

(2) Horizontal tab

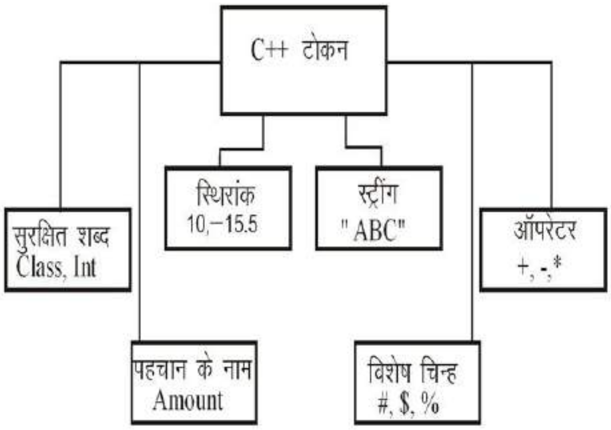
(3) Carriage return

(4) New line

(5) Form feed

1.3 C++ टोकन (C++ Tokens)

किसी भी भाषा में लिखे गये प्रोग्राम की छोटी से छोटी इकाई को उस भाषा के चिन्ह (टोकन) कहते है। C++ भाषा में निम्न प्रकार के चिन्ह है।



1.3.1 पहचान के नाम तथा सुरक्षित शब्द (Identifier and Reserve word)

C++ भाषा के प्रत्येक शब्द या तो सुरक्षित शब्द होते हैं या फिर पहचान के नाम । सभी सुरक्षित शब्द का एक निश्चित अर्थ है जो बदला नहीं जा सकता। ये सुरक्षित शब्द ही किसी प्रोग्राम के मुख्य तत्व होते हैं।

कुछ सुरक्षित शब्दों की सूची निम्नलिखित है। प्रोग्राम में इन्हें हमेशा अंग्रेजी वर्णमाला के छोटे अक्षरों में लिखते हैं।

auto	double	int	struct
break	else	long	switch
case	enum	return	union
char	float	short	void
const	for	sizeof	while
continue	go	static	
do	if	short	

पहचान के नाम (Idintifier)

किसी भी चर, फंक्शन अथवा आव्यूह आदि के नाम को पहचान का नाम कहते हैं। ये नाम हम अपनी इच्छानुसार रखते हैं। ये नाम अक्षर अथवा अंकों से मिलकर बने होते हैं। इनका पहला अक्षर अंग्रेजी वर्णमाला का अक्षर होना चाहिये। नाम वर्णमाला के छोटे, बड़े अथवा दोनों अक्षर से बने हो सकते हैं।

विशेष अक्षर का उपयोग नाम में नहीं करते। '_' (underscore) अक्षर का उपयोग किया जा सकता है।

1.3.2 स्थिरांक (Constants)

स्थिरांक उनको कहते हैं जिनके मान प्रोग्राम के चलने के दौरान बदला न जा सके। जैसे हमें किसी प्रोग्राम में π का मान काम लेना हो और PI = 3.14 होगा।

प्रोग्राम में जहां पर भी PI शब्द का उपयोग होगा वहां पर 3.14 मान काम आयेगा। अतः PI एक स्थिरांक कहलायेगा। आगे हम उन स्थिरांकों के प्रकार व उपयोग करने के तरीके के बारे में पढ़ेंगे।

1.3.3 ऑपरेटर (Operators)

ऑपरेटर हम उन संकेत चिन्हों को कहते हैं जो किसी संक्रिया के बारें में संकेत देते हैं। जैसे :– B=3+2 यहां दो संक्रिया चिन्ह है एक '+' व दूसरा '='। ये चिन्ह संकेत देते हैं कि 3 व 2 का जोड़ करना है तथा B के मान को योग के परिणाम के मान के बराबर करना है।

ये विभिन्न प्रकार के होते हैं। जिनका अध्ययन इस अध्याय में आगे करेंगे।

1.4 C++ भाषा के प्रोग्राम का ढ़ाँचा (Structure of a C++ Program)

हम एक साधारण प्रोग्राम लेकर उसका अध्ययन कर चुके हैं। अब पुनः एक छोटे से प्रोग्राम व उसमें काम आने वाले तत्वों का अलग–अलग अध्ययन करते है।

```
# include <iostream.h>
void main ( )
{
cout << " every language has its own grammer";
}
```

इस छोटे से प्रोग्राम में हम main फंक्शन include डायरेक्टिव व हैडर फाइलों के बारे में पढ़ेंगे।

1.4.1 फंक्शन :–

उक्त प्रोग्राम में केवल एक ही फंक्शन है जिसका नाम main है। प्रोग्राम की पहली पंक्ति जो # include से आरम्भ हो रही है वह इस फंक्शन का भाग नहीं है। हमने पहले देखा है कि क्लास के अन्दर भी फंक्शन हो सकते हैं जिन्हें हम उस क्लास का सदस्य फंक्शन कहते हैं,पर मेम्बर फंक्शन के अतिरिक्त स्वतन्त्र फंक्शन भी हो सकते हैं। हमारे प्रोग्राम में फंक्शन का नाम main है तथा इसके आगे कोष्ठक लगे हैं। यदि हम कोष्ठक न लगायें तो कम्पाइलर इसे चर का नाम भी मान सकता है अतः फंक्शन के आगे कोष्ठक लगाना आवश्यक है। इन कोष्ठकों में हम फंक्शन आरग्यूमेन्ट लिखते हैं। फंक्शन के नाम से पूर्व void लिखा है जो यह बताता है कि यह फंक्शन कोई भी मान वापस नहीं करता।

फंक्शन आरग्यूमेन्ट व मान वापस करने के बारे में हम आगे पढ़ेंगे। फंक्शन के नाम की अगली पंक्ति तथा अन्तिम पंक्ति में मझले कोष्ठक है। ये कोष्ठक फंक्शन के आरम्भ व अन्त की सूचना देते हैं। प्रत्येक फंक्शन के निर्देशों को हमेशा मझले कोष्ठक के अन्दर लिखना चाहिये।

प्रत्येक प्रोग्राम का पहला निर्देश जो निष्पादित (execute) होता है वह main फंक्शन का आरम्भ

होता है। किसी भी प्रोग्राम में एक से अधिक फंक्शन हो सकते हैं । परन्तु main फंक्शन का होना आवश्यक है तथा सबसे पहला फंक्शन main ही होना चाहिये जो निष्पादित (execute) हो। यदि किसी प्रोग्राम में main फंक्शन नहीं होगा तो लिंकर गलतियाँ बतायेगा।

1.4.2 # इन्क्लूड डायरेक्टिव (# include directive)

इस प्रोग्राम की पहली लाइन # include< iostream.h> है। यह न तो किसी फंक्शन का भाग है और न ही प्रोग्राम का कथन (statement)। यदि यह किसी फंक्शन का भाग होती तो मझले कोष्ठक के बीच होता और यदि प्रोग्राम का वाक्य हो तो सेमीकालन (;) से खत्म होता। यह पंक्ति # (हैश) चिन्ह से आरम्भ हो रही है। इन्हें प्रीप्रोसेसर डायरेक्टिव कहते हैं। ये पंक्तियां (प्रीप्रोसेसर डायरेक्टिव) कम्प्यूटर को निर्देश देने के बजाय कम्पाइलर को निर्देश देती हैं। कम्पायलर इन निर्देशों की पालना प्रोग्राम के कम्पाइलेशन के समय करता है।

include प्रीप्रोसेसर डायरेक्टिव कम्पाइलर से अन्य फाइलों को हमारे प्रोग्राम के स्त्रोत फाइल (Source Code) में डालने का निर्देश देता है।

1.4.3 हैडर फाइलें (Header Files)

हमें प्रोग्राम बनाते समय कई गणित के फंक्शन जैसे sin, cos, exp आदि का उपयोग करते हैं। C++ भाषा में ये फंक्शन पहले से बने हुये हैं तथा इनको C++ भाषा की Math Library में रखा हुआ है। तथा इन सभी फंक्शनों को इनकी क्रियाओं के अनुसार अलग–2 फाइलों में संग्रहित कर रखा है। जैसे गणित के फंक्शन math.h फाइल में, cin, cout के iostream.h, में आदि इन्हें हम हैडर फाइल कहते हैं। हम यह कह सकते हैं कि हैडर फाइल में सम्बन्धित क्रियाओं की परिभाषाएं व संपादित करने के तरीके संग्रहित है।

जैसे– iostream.h में cout व <<, cin व >> की परिभाषा संग्रहित है। यदि हम प्रोग्राम से पहले iostream.h हैडर फाइल को # include द्वारा शामिल नहीं करेंगे तो कम्पाइलर cout व cin अर्थ नहीं समझ पायेगा।

इसी प्रकार यदि setw( ) फंक्शन प्रोग्राम में काम लें तो iomanip.h हैडर फाइल को शामिल करना आवश्यक है।

1.5 C++ में आगम निर्गम (Input / Output in C++)

1.5.1 cout

हमारे प्रोग्राम में cout << "every language has it's own grammer"; कथन का प्रयोग वाक्य को स्क्रीन पर प्रदर्शित करने के लिये किया गया है। C++ भाषा में cout एक ऑब्जेक्ट है तथा यह मानक निर्गम स्ट्रीम (standard out put stream) से सम्बन्धित है। तथा यह C++ भाषा में पूर्व परिभाषित है। इसका उपयोग अक्षरों या उसके समूह को स्क्रीन पर प्रदर्शित करने अथवा अन्य निर्गम युक्ति (Output device) पर भेजने के लिए किया जाता है।

1.5.2 cin

इसी प्रकार cin भी एक ऑब्जेक्ट है। यह मानक आगम स्ट्रीम से सम्बन्धित है तथा यह C++ में पूर्व परिभाषित है। यह कुँजी पटल द्वारा या अन्य आगम युक्ति (input device) द्वारा कम्प्यूटर में डाले गये आंकडों को ग्रहण कर कम्प्यूटर में संग्रहित करने के उपयोग में आता है।

1.5.3 इन्सर्शन ऑपरेटर ('<<') (Insertion Operator)

इस ऑपरेटर को पुट (put) ऑपरेटर भी कहते हैं। इसका उपयोग cout ऑब्जेक्ट के साथ किया जाता है। इसका कार्य इसके दांयी तरफ के चर के मान को बांयी तरफ के ऑब्जेक्ट (cout) की तरफ भेजना है।

जैसे– नीचे लिखे प्रोग्राम में यह ऑपरेटर बांयी तरफ लिखे संदेश "inter temprature in fahrenhit" को दांयी तरफ अर्थात cout ऑब्जेक्ट की तरफ भेज देता है। यहां यह ध्यान रखना आवश्यक है कि यही ऑपरेटर एक–एक बिट को बांयी तरफ शिफ्ट करता (Left Shift - bit wise) है।

1.5.4 एक्सट्रेक्शन ऑपरेटर ('>>') (Extraction operator)

इस ऑपरेटर को गेट फ्रोम (get from) ऑपरेटर भी कहते हैं। इसका उपयोग cin ऑब्जेक्ट के साथ किया जाता है। इसका कार्य बांयी तरफ लिखे ऑब्जेक्ट के मान को दांयी तरफ लिखे चर में संग्रहित कराना है। जैसे आगे लिखे प्रोग्राम में यह cin ऑब्जेक्ट से प्राप्त मान को ftemp चर में संग्रहित कर देता है।

इस ऑपरेटर का उपयोग एक–एक बिट को दांयी तरफ शिफ्ट करने (bit wise right shift) के लिए भी होता है।

```
# include <iostream.h>
void main ( )
{
    float ftemp,ctemp;
    cout<<"enter temprature in fahrenheit:";
    cin >> ftemp;
    ctemp = (ftemp - 32)* 5/9;
    cout <<" temprature in celsius = " << ctemp;
}
```

1.6 आगम–निर्गम ऑपरेटर्स का एक से अधिक बार उपयोग (Cascading of I/O Operators)

इन्सर्शन ऑपरेटर्स का उपयोग एक पंक्ति में एक से अधिक बार भी किया जा सकता है। जैसे प्रोग्राम के अन्तिम पंक्ति में cout ऑब्जेक्ट के साथ दो बार पुट (<<) ऑपरेटर का उपयोग किया गया है। इस प्रकार उपयोग करने पर बायें से दायें क्रम में क्रिया होगी अर्थात पहले "Temprature in celsius",

cout ऑब्जेक्ट की तरफ भेजेगा उसके बाद Ctemp चर के मान को cout ऑब्जेक्ट की तरफ भेजेगा। इसी प्रकार इन्सर्शन ऑपरेटर >> को भी हम एक पंक्ति में एक से अधिक बार उपयोग कर सकते है।

1.6.1 endl व setw का प्रयोग (Use of endl and setw)

setw तथा endl दोनों ही दस्तकार (Manipulator) कहलाते हैं। इनका प्रयोग इन्सर्शन ('<<') ऑपरेटर के साथ किया जाता है। ये आंकडो को सुस्पष्ट रूप से प्रदर्शित करते हैं। यदि हम प्रोग्राम में इनका प्रयोग करे तो iomanip.h हैडर फाइल को आवश्यक रूप से include कर लें।

(1) **endl** :- इस दस्तकार का प्रयोग नई पंक्ति पर जाने के लिए या एक पंक्ति आगे खिसकने के लिए किया जाता है। इसका प्रभाव "\n" (बैकस्लेश अक्षर) के प्रभाव के जैसा ही होता है। आगे लिखे प्रोग्राम में इसका प्रयोग तथा उनका परिणाम दर्शाया गया है।

(2) **setw**:- cout ऑब्जेक्ट के द्वारा जब हम मानों को प्रदर्शित करते हैं। तो हर मान को उतनी ही जगह दी जाती है जितने अक्षर का मान हैं जैसे माना X = 1203 हो तो x का मान प्रदर्शित करने के लिए 4 अक्षर की जगह मिलेगी यदि हम चर के मानों को व उनके मध्य की जगह को इच्छानुसार प्रदर्शित करना चाहें तो setw ( ) मैनिपुलेटर का उपयोग करना पड़ेगा।

हम एक ऐसा प्रोग्राम बनाते हैं जो तीन विद्यार्थियों के नाम व उनके अंक प्रदर्शित करता है।

```
# include <iostream.h>
void main ( )
{
int A = 105, B=1203, C=15;
cout    <<"NAME"<< "MARKS" <<"\n"
        <<"Rajesh" <<A <<'\n"
        << "Raghuveer" << B << "\n"
        <<"Monu" <<C << "\n";
}
```

इस प्रोग्राम का निर्गम (output) इस प्रकार होगा

```
NAME MARKS
Rajesh105
Raghuveer1203
Monu15
```

यह आउटपुट दिखने में भी अच्छा नहीं लग रहा, और न ही इसे आसानी से समझा जा सकता है। यदि इसी प्रोग्राम को इस प्रकार लिखा जाये कि सभी नाम एक सीध में हो तथा उनके अंक दांयी तरफ से एक सीध में हो।

```
# include <iostream.h>
# include <iomanip.h>
void main ( )
{
int A=105, B=1203, C=15;
```

```
cout <<setw(10)<<"NAME"<<setw(6)<<"MARKS"<<endl
    <<setw(10)<<"Rajesh"<<setw(6)<<A<<endl
    <<setw(10)<<"Raghuveer"<<setw(6)<<B<<endl
    <<setw(10)<<"Monu"<<setw(6)<<C<<endl;
}
```

तो निर्गम (output) इस प्रकार होंगे।

NAME	MARKS
Rajesh	105
Raghuveer	1203
Monu	15

setw (n) मैनीपुलेटर प्रदर्शित किये जाने वाले आंकडो अथवा अक्षरों के लिए दी n अक्षर की जगह निश्चित कर देता है। संख्याओं का दांयी तरफ से लिखता है जबकि अक्षरों को बांयी तरफ से। अतः setw मैनीपुलेटर का प्रयोग कर हम Output को सुस्पष्ट बना सकते है।

गलतियाँ (Errors)

प्रोग्राम बनाते समय कई बार टाइप करने में गलती हो जाती है, या किसी कथन को लिखते समय गलत तरीके से लिख देते है या कई अन्य प्रकार की गलतियाँ हो जाती है। ये गलतियाँ कम्पाइलर या लिंकर द्वारा ढूंढी जा सकती है। यदि प्रोग्राम में किसी प्रकार की गलती हो तो कम्पाइलर या लिंकर इस सम्बन्ध में स्क्रीन पर संदेश प्रदर्शित करता है।

कम्पाइलर केवल उन गलतियों को ढूंढ़कर बताता है जो भाषा के कथन लिखने के तरीकों से सम्बन्धित होती है जैसे कथन के अन्त में सेमीकालन न लगाना, किसी चर का घोषित न करना इत्यादि। माना हमने किसी कथन के अन्त में सेमीकालन नहीं लगाया तो कम्पाइलर प्रोग्राम को कम्पाइल करते समय स्क्रीन पर संदेश प्रदर्शित करेगा। जैसे

error C:\EXAM.CPP 8:statement missing; in function main ( )

यह संदेश बताता है कि exam.cpp प्रोग्राम की 8वीं पंक्ति में जो कथन लिखा है उसमें सेमीकॉलन (;) नहीं है।

विन्डो में कर्सर भी उसी लाइन पर होगा जहां संभवतः गलती है या जिस स्थान पर कम्पाइलर की गलती मिली। माउस को क्लिक करके अथवा कुँजी पटल पर कोई भी कुँजी दबाकर, कर्सर को नियत स्थान पर लाकर गलती दूर की जा सकती है।

लिकंर लिकिंग से सम्बन्धित प्रकार की गलतियाँ ढूंढता है जैसे हमारे प्रोग्राम में main ( ) नाम से कोई फंक्शन ही न हो आदि। यदि ऐसा है तो लिंकर निम्न संदेश प्रदर्शित करेगा।

Linker error : undefined symbol_main in module.

1.6.2 रन टाइम त्रुटि (Run time error)

रन टाइम गलती वे गलतियाँ है जो प्रोग्राम के चलाने पर ही ज्ञात की जा सकती है। किसी संख्या को शून्य 0 से भाग देना, या यदि स्टैक काम लिया हो और वो भर गया हो। ये ऐसी गलतियाँ है जो कम्पाइलेशन से ज्ञात नहीं होती। इनको ज्ञात करने के लिए प्रोग्राम को चलाना ही पडेगा।

1.7 सम्पादक का प्रयोग (Use of Editor)

C++ में प्रोग्राम बनाने के लिए जब हम C++ के सॉफ्टवेयर को load करते है तो जो सबसे पहले विण्डो स्क्रीन पर प्रदर्शित होती है उसे C++ भाषा का सम्पादक (Editor) कहते हैं।

The Turbo C++ Screen with edit window

पहली बार यह विण्डो खाली होगी। इसके मीनू बार में File, Edit, Search, Run, Compile, Debug, Project, Option, Windows, Help मीनू होंगें।

प्रोग्राम बनाने के लिए इस सम्पादक में टाइप करना आरम्भ कर दें। जब फाइल में प्रोग्राम टाइप कर चुके हों और संग्रहित करना चाहें तो फाइल मीनू मे से SAVE विकल्प का चयन करें। यहां इच्छानुसार नाम टाइप कर OK बटन का चयन करें। इस प्रकार किसी भी फाइल में प्रोग्राम टाइप कर संग्रहित कर लें। फाइल का विस्तार नाम हमेशा .CPP ही रखें।

तेज गति से टाइप करने हेतु कुछ विशेष कुँजियों का प्रयोग करना आवश्यक है। नीचे सूची में कुछ

कुँजियाँ व उनके उपयोग दिये गये है।

कुंजी	कार्य
↑	कर्सर को एक पंक्ति ऊपर ले जाने के लिये।
↓	कर्सर को एक पंक्ति नीचे ले जाने के लिये।
→	कर्सर को एक स्तम्भ दायें ले जाने के लिये।
←	कर्सर को एक स्तम्भ बायें ले जाने के लिये।
Ctrl + ←	कर्सर को एक शब्द बायें ले जाने के लिये।
Ctrl + →	कर्सर को एक शब्द दायें ले जाने के लिये।
HOME	कर्सर को पंक्ति के प्रथम स्थम्भ पर ले जाने के लिये।
END	कर्सर को पंक्ति के अन्तिम अक्षर पर ले जाने के लिये।
Pgdn	स्क्रीन को एक स्क्रीन नीचे ले जाने के लिये।
Pgup	स्क्रीन पर लिखे अक्षरों को एक स्क्रीन ऊपर ले जाने के लिये।
ENTER	कर्सर को नई पंक्ति पर लाने के लिये।
Backspace	कर्सर के बायें स्थित एक अक्षर को हटाने के लिये।
DEL	कर्सर के ऊपर लिखे अक्षर को हटाने के लिये।
INS	इन्सर्ट मोड ON/OFF करने के लिये।

1.7.1 सम्पादक (Editor) के प्रमुख कमाण्ड (Basic Command of Editor)

C++ भाषा में प्रोग्राम बनाने व निष्पादित (Exequte) करने के लिए सम्पादक (Editer) का उपयोग किया जाता है। यह सम्पादक (Editer) पूर्व में दिखायें चित्र के अनुसार प्रदर्शित होता है इस सम्पादक में तेज गति से कार्य करने के लिए कुछ कमाण्ड निर्धारित हैं। ये सभी कमाण्ड व उनका उपयोग निम्नानुसार है।

कमाण्ड	–	उपयोग
F2	–	फाइल को भण्डारित (Save) करने के लिये।

F3	–	भण्डारित (Save) फाइल को खोलने के लिये।
F4	–	प्रोग्राम को कम्पाइल करने के बाद कर्सर पर जाने के लिये।
Alt+X	–	सम्पादक (Editer) से बाहर निकलने के लिये।
Alt+BSP	–	किये गये कार्य को मिटाने के लिए (Undo)।
Shift+Alt+BSP	–	मिटाये गये कार्य को पुनः लाने के लिए (Redo)।
Shift+Del	–	चयनित भाग को हटाने व क्लिप बोर्ड पर भण्डारित करने के लिये।
Ctrl+INS	–	चयनित भाग की प्रतिलिपि करने के लिए व क्लिप बोर्ड पर भण्डारित करने के लिए (Copy)।
Shift+INS	–	क्लिप बोर्ड पर भण्डारित भाग को पुनः प्रोग्राम में लाने के लिए (Paste)।
Ctrl+Del	–	चयनित भाग को हटाने के लिए।
Ctrl+L	–	किसी शब्द को ढूंढते समय जब एक बार वह शब्द मिल जाये तो अगले स्थान पर उस शब्द को ढूंढने के लिए।
Ctrl+F9	–	प्रोग्राम को निष्पादित (Execute) करने के लिए।
Alt+F7	–	कम्पाइल करते समय पिछली गलती (Previous Error) पर कर्सर को ले जाने के लिए।
Alt+F8	–	कम्पाइल करते समय अगली गलती (Next Error) पर कर्सर को ले जाने के लिए।

1.8 कम्पाइलिंग एवं लिंकिग (Compiling and linking)

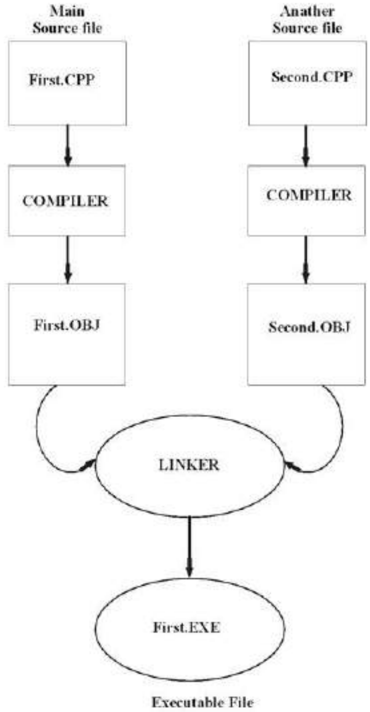
जो प्रोग्राम हम सम्पादक (editor) में टाइप कर संग्रहित (Save) करते हैं वह स्त्रोत फाइल (Source file) कहलाती है। यह एक ASCII फाइल होती है। अर्थात इस फाइल में लिखे प्रोग्राम को चला नहीं सकते। इस प्रोग्राम को चलाने के लिए इसे चलने योग्य (Executable) फाइल में परिवर्तित करना होगा। यह परिवर्तन दो चरणों में होता है।

(1) कम्पाइलिंग

(2) लिंकिंग

कम्पाइलिंग (Compiling)

कम्पाइलिंग का अर्थ स्त्रोत फाइल (Source file) को ऑब्जेक्ट (object) file में बदलना। कम्पाइलिंग के दौरान प्रोग्राम को मशीन भाषा (Binary form) में बदला जाता है। यह मशीन भाषा में लिखा प्रोग्राम ही कम्प्यूटर पर चल सकता है। पर चलाने से पूर्व लिकिंग करना भी आवश्यक है।



लिंकिंग (Linking)

लिंकिंग का अर्थ है कि यदि एक से अधिक ऑब्जेक्ट फाइलें हैं तो इन सबको एक साथ करना। कई बड़े–2 प्रोग्रामों में एक से अधिक फाइलें होती है। जिन्हें एक साथ करना आवश्यक है। यदि एक से अधिक ऑब्जेक्ट फाइल न हो तो भी लिंकिंग आवश्यक है। हो सकता है कि हमने # include डायरेक्टिव के द्वारा हैडर फाइल या लाइब्रेरी फाइल का उपयोग किया हो। हैडर फाइल या लाइब्रेरी फाइल को मुख्य फाइल के साथ जोड़ने के लिए भी लिंकिंग आवश्यक है।

चित्र में कम्पाइलिंग व लिंकिंग में मध्य सम्बन्ध बताया गया है। लिकिंग करने के बाद ही प्रोग्राम चलने योग्य होगा। साधारणतयाः चलने योग्य फाइल एक से अधिक ऑब्जेक्ट फाइलों से मिलकर बनती है।

कम्पाइल करने के लिए फाइल को खोलकर संग्रहित (Save) करने के बाद फाइल मीनू में

Compile to OBJ form विकल्प का चयन करें। ऐसा करने से कम्पाइल विन्डो प्रदर्शित होंगी। यदि प्रोग्राम में किसी प्रकार की गलती नहीं है तो

0 Error, Success : Press any key

संदेश प्रदर्शित होगा। इस प्रकार जो ऑब्जेक्ट फाइल बनेगी उसका नाम वही होगा जो स्त्रोत फाइल का है। केवल विस्तार नाम .obj होगा।

लिंक करने के लिए कम्पाइल मीनू में से Link Exe File विकल्प का चयन करें। इस प्रकार ऑब्जेक्ट फाइल अन्य ऑब्जेक्ट फाइल से मिलकर चलने योग्य (Executive) फाइल बनती है। इसका विस्तार नाम .exe होगा।

प्रोग्राम को चलाना (Execution of Program)

चलने योग्य (Executable) फाइल को चलाने के लिए Run मीनू में से Run विकल्प का चयन करे या कुँजी पटल पर Ctrl + F9 कुँजीयाँ दबायें। ऐसा करने पर प्रोग्राम चलेगा तथा परिणाम देखने के लिए कुँजी पटल पर Alt + F5 दबायें या विण्डो मीनू में से User Screen विकल्प का चयन करें।

1.9 C भाषा की आगम–निर्गम क्रियाए (Input-Output operations from C Language)

सी भाषा में जो आगम निर्गम क्रियाएें उपलब्ध हैं लगभग सभी C++ भाषा में भी उपलब्ध है। मुख्यतः ये क्रियाएें स्ट्रींग को पढ़ने या प्रिन्ट करने के लिऐ उपयोग में लायी जाती हैं।

(1) gets ( ) :-

gets ( ) -फंक्शन stdio.h हैडर फाइल के अन्दर होता है। यह फंक्शन एक स्ट्रींग को पढ़ने के काम आता है।

उदाहरण : एक चर char name [20]डिक्लेयर किया गया है हम इस चर का मान उपयोगकर्ता द्वारा कुँजी पटल के माघ्यम से पढने के लिऐ निम्न प्रोग्राम लिखेंगे।

```
#include <stdio.h>
# include <conio.h>
   void main()
{
   char name[20];
   printf ("\n enter the name");
   gets(name);
   puts (name);
   getch ( );
}
```

2) puts ( ) :-

puts ( ) -फंक्शन भी gets() की तरह ही कार्य करता है।

इसका उपयोग किसी स्ट्रींग के मान को प्रिन्ट करने के लिऐ किया जाता है। तथा जब भी हम उस फंक्शन का उपयोग करेंगे तो stdio.h हैडर फाइल का प्रयोग करेंगे।

अभ्यास प्रश्न

वस्तुनिष्ठ प्रश्न

निम्नलिखित प्रश्नो में सही विकल्प का चुनाव कीजिए

1 निम्न में से C++ भाषा के टोकन नहीं है :-
   (अ) स्थिरांक   (ब)ऑपरेटर
   (स) ऑब्जेक्ट   (द) विशेष चिन्ह

2 iostream.h फाइल है :-
   (अ) लाईब्रेरी फाइल   (ब) प्री प्रोसेसर डायरेक्टिव
   (स) हैडर फाइल   (द) कमेन्ट लाईन

3 इन्सर्शन ऑपेरटर (<<) का उपयोग होता है :-
   (अ) cout के साथ   (ब) cin के साथ
   (स) # के साथ   (द) उपरोक्त मे से कोई नहीं

4 प्री प्रोसेसर डायरेक्टिव निर्देश देते हैं :-
   (अ) CPU को   (ब) कम्पाइलर को
   (स) उक्त दोनों को   (द) उपरोक्त में से कोई नहीं

5 setw एक :-
   (अ) प्री प्रोसेसर डायरेक्टिव है।   (ब) ऑब्जेक्ट है
   (स) मेन्युपुलेटर है   (द) उपरोक्त मे से काई नहीं

अतिलघुत्तरात्मक प्रश्न

1 हैडर फाइल से आप क्या समझते है ?
2 प्री प्रोसेसर डायरेक्टिव का क्या उपयोग है ?
3 मेन्युपुलेटर किसे कहते है ?
4 सम्पादक (Editor) से आप क्या समझते है ?
5 कमेन्ट लाईन से आप क्या समझते है ? इनका उपयोग प्रोग्राम में किस प्रकार करते है समझाइयें।

निबन्धात्मक प्रश्न

1 C++ में काम आने वाले अक्षर (character) कौन कौन से है ।
2 C++ के टोकन से आप क्या समझते है ? विभिन्न टोकनो को समझाइये।
3 cin,cout, >> ,<< का C++ भाषा में क्या उपयोग है ? समझाइये।
4 कम्पाइलिंग व लिंकिंग से क्या समझते है ? सचित्र समझाइये।
5 पहचान के नाम (Identifire) व सुरक्षित शब्द (Reserve Word) से आप क्या समझते है ? उदाहरण सहित समझाइये।

उत्तरमाला

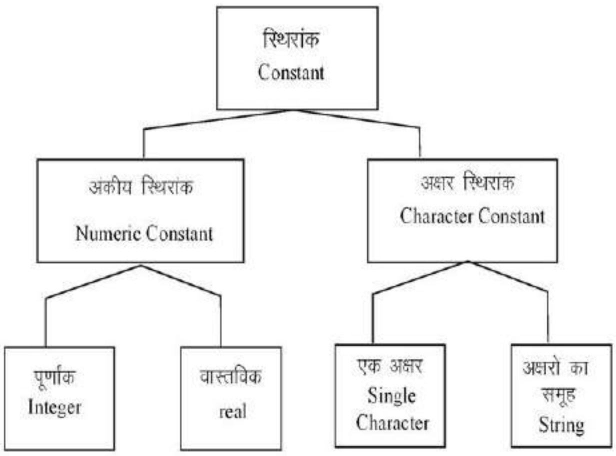
1 (स)	2 (स)	3 (अ)
4 (ब)	5 (स)	

2

आँकडों के प्रकार, चर, स्थिरांक (Data Type, Variable, Constant)

2.1 स्थिरांक (Constants)

स्थिरांक उन्हें कहते हैं जिनका मान प्रोग्राम के निष्पादन के दौरान बदला न जा सके। C++ में निम्न प्रकार के स्थिरांक होते है।



2.1.1 पूर्णांक स्थिरांक (Integer Constant) :– बिना दशमलव चिन्ह के अंकों का समूह जिसका मान प्रोग्राम के निष्पादन के दौरान बदला ना जा सकें, को हम पूर्णांक स्थिरांक कहते हैं। ये निम्न प्रकार के होते हैं :–

(1) डेसीमल (Decimal) पूर्णांक स्थिरांक।
(2) आक्टल (Octal) पूर्णांक स्थिरांक।
(3) हेक्साडेसिमल (Hexadecimal) पूर्णांक स्थिरांक।

डेसिमल पूर्णांक स्थिरांक (Decimal Integer Constant)

डेसिमल पूर्णांक स्थिरांक में केवल 0 से 9 तक के अंकों का उपयोग किया जा सकता है।

आक्टल पूर्णांक स्थिरांक (Octal Integer Constant)

आक्टल पूर्णांक स्थिरांक में 0 से 7 तक के अंकों का उपयोग किया जा सकता है। स्थिरांक से पहले अक्षर 'O' लिखना आवश्यक है।

हैक्साडेसिमल पूर्णांक स्थिरांक (Hexadecimal Integer Constant)

हैक्साडेसिमल पूर्णांक स्थिरांक में 0 से 9 तक के अंक व A, B, C, D, E व F अक्षरों का उपयोग कर सकते हैं। हैक्साडेसिमल में लिखे पूर्णांक स्थिरांक के पूर्व "Ox" अक्षरों का प्रयोग आवश्यक है।

किसी भी पूर्णांक स्थिरांक के पहले '+' अथवा '–' का चिन्ह लगाया जा सकता है। इनके अतिरिक्त किसी भी विशेष अक्षर (Special character) का प्रयोग नहीं किया जा सकता।

सही डेसिमल पूर्णांक स्थिरांक	गलत डेसिमल पूर्णांक स्थिरांक	गलती
123	15 750	खाली जगह नहीं होनी चाहिए
–321	20,000	कॉमा (,) का प्रयोग नहीं करना
0	$1000	$ चिन्ह नहीं होना चाहिये।
+ 78	10.5	दशमलव चिन्ह नहीं हो।

सही ऑक्टल पूर्णांक स्थिरांक	गलत ऑक्टल पूर्णांक स्थिरांक	गलती
O37	O37.15	दशमलव चिन्ह नहीं हो।
O2	O380	अंक 8 का प्रयोग नहीं हो।
O435	O390.5	अंक 9,व दशमलव चिन्ह नहीं होना चाहिए।
O551	370	प्रथम अक्षर 0 होना चाहिये।

सही हैक्साडेसिमल पूर्णांक स्थिरांक	गलत हैक्साडेसिमल पूर्णांक स्थिरांक	गलती
OX2	OX2.0	दशमलव चिन्ह नहीं होना चाहिये।
OX9F	OXG2	G अक्षर नहीं होना चाहिये।
OXbcd	abc	प्रथम अक्षर Ox होना चाहिये

पूर्णांक स्थिरांक का अधिकतम मान कम्प्यूटर पर निर्भर करता है। यदि कम्प्यूटर 16 bit का है तो 32767 अधिकतम मान हो सकता है और यदि कम्प्यूटर 32 bit का है तो अधिकतम मान 2, 147, 483, 647 हो सकता है। पूर्णांक स्थिरांक के अन्त में U. L. या UL अक्षर लगाकर बड़ी संख्याएं भी संग्रहित की जा सकती है। इन अक्षरों के अर्थ निम्नानुसार है।

U - Unsigned

L - Long integer

UL - Unsigned long integer

2.1.2 वास्तविक स्थिरांक (Floating Point Constant)

पूर्णांक स्थिरांक के द्वारा उन मानों को प्रदर्शित नहीं किया जा सकता। जो मान एक से छोटा हो या एक का छोटा भाग जैसे 0.56, 0.23, 0.5, 0.32 आदि। इनको प्रदर्शित करने के लिये वास्तविक संख्याओं का प्रयोग करते हैं।

पूर्णांक स्थिरांक व वास्तविक स्थिरांक के मध्य केवल इतना सा अन्तर है कि वास्तविक स्थिरांक में दशमलव (.) चिन्ह हो सकता है जब कि पूर्णांक स्थिरांक में नहीं।

उदाहरण

0.00083, -0.85, 768.39, +220.0

दशमलव चिन्ह के पहले अथवा बाद में अंक का होना आवश्यक नहीं है। उदाहरण के लिये

245., .115, -.159 आदि भी सही वास्तविक पूर्णांक है।

वास्तविक संख्याओं को वैज्ञानिक तरीके से घातांक रूप में भी लिख सकते हैं। जैसे 2,00,000 को हम 2×10^5 या 2.0e+ 5 भी लिख सकते हैं। कम्प्यूटर में संख्याओं को घातांक रूप में लिखने के लिऐ निम्न प्रारूप है।

Mantissa e Exponant

मैन्टीसा भाग पूर्णांक संख्या या वास्तविक संख्या हो सकती है। इसके बाद अक्षर 'e' का होना आवश्यक हैं तथा Exponent भाग एक पूर्णांक संख्या होगी। Exponent तथा अक्षर 'e' के मध्य + अथवा – चिन्ह हो सकता है।

उदाहरण -

1.2 e -4, 0.95 e 4, 1.9 e + 50, 1.2 e -2

वास्तविक संख्याओं को घातांक रूप में हम तब लिखते हैं जब संख्या बहुत अधिक बड़ी अथवा बहुत अधिक छोटी हो।

2.1.3 अक्षर स्थिरांक (Character Constant)

ये दो प्रकार के होते हैं। पहला एक अक्षर वाले स्थिरांक और दूसरा अक्षरों के समूह का स्थिरांक।

(1) एक अक्षर स्थिरांक (Single Character Constant)

एक अक्षर का स्थिरांक केवल एक अक्षर का बना होता है जो कि '' (Single quote) चिन्हों के अन्दर देने होते हैं।

उदाहरण

'C', 'P', '9', 'x' आदि।

यहाँ '9' अक्षर स्थिरांक व 9 संख्या दोनों भिन्न–2 हैं। प्रत्येक अक्षर स्थिरांक का एक निश्चित ASCII मान होता है।

जैसे 'd' का ASCII मान 100 है। 'a' का ASCII मान 97 है। यदि चूंकि प्रत्येक अक्षर स्थिरांक का ASCII मान संख्या होती है अतः इन पर अंकगणितीय गणना भी की जा सकती है।

2.1.4 अक्षरों का समूह स्थिरांक (String Constant)

जैसा नाम से ज्ञात होता है ये स्थिरांक एक अक्षर के न होकर एक से अधिक अक्षरों का समूह होता है। ये स्थिरांक " " (double quotes) चिन्हों के मध्य देने होते हैं। ये किसी भी अक्षर से मिलकर बने हो सकते हैं, चाहे अंक हो, अक्षर या विशेष अक्षर।

उदाहरण

"RAMESH",	"2005",	"C"
"Wel Come",	"200+5+6"	

यहां ध्यान देने की बात है कि 'C' व "C" दोनों अलग–2 हैं जहां 'C' एक अक्षर स्थिरांक है वही "C" अक्षरों का समूह स्थिरांक है। 'C' एक ASCII मान होगा जब कि "c" का नहीं। शब्दों या शब्द समूहों पर क्रिया करने के लिए अक्षरों का समूहों का उपयोग किया जाता है।

2.1.5 बैकस्लेश अक्षर स्थिरांक (Backslash Character Constant)

C व C++ भाषा में निर्गम कार्य (Output Operation) के लिए कुछ निश्चित बैक स्लेश अक्षर स्थिरांक है इनका विशेष अर्थ होता है। जैसे "\n" का अर्थ नई पंक्ति है। अन्य बैक स्लैश अक्षर स्थिरांक व उनके परिणाम नीचे सूची में दिये गये हैं।

"\a"	audible alert
"\b"	back space
"\f"	form feed
"\n"	New line
"\r"	Carriage return
"\t"	horizontal tab
"\v"	vertical tab
"\' "	single quote
"\" "	double quote
"\?"	question mark
"\\"	back slash
"\0"	null character

2.2 आँकड़ो के प्रकार, चर, स्थिरांक

आँकड़ो के प्रकार (Data Type)

हम प्रोग्राम में विभिन्न प्रकार के आँकड़ों का उपयोग करते हैं जैसे integer, float, Array आदि। प्रोग्राम में आँकड़ों का प्रकार बताते समय हमें यह ध्यान रखना आवश्यक है कि हमारे आँकड़ो का अधिकतम मान क्या हो सकता है। तथा ये आँकड़े किस प्रकार हमारी स्मृति (Memory) में कम से कम स्थान घेरे। जैसे integer प्रकार के आँकड़े केवल 2 बाइट जगह घेरते है। परन्तु इनसे अधिकतम 32767

मान ही संग्रहित कर सकते है। Long int प्रकार के आँकडे लगभग 2 अरब मान तक संख्या संग्रहित कर सकते है । पर वे जगह भी 4 बाइट की घेरते हैं। अतः हमें आवश्यकतानुसार आँकड़ों के प्रकार का चयन करना चाहिये। C++ भाषा में आँकड़े निम्नलिखित प्रकार के होते हैं।

(1) प्राइमरी, बिल्ट–इन या फण्डामेण्टल प्रकार के आँकड़े ।

(2) यूजर डिफाइन्ड प्रकार के आँकड़े ।

(3) डेराइव्ड प्रकार के आँकड़े।

प्राइमरी प्रकार के आँकड़ों का प्रारूप पूर्व निर्धारित होता है अतः इनका प्रयोग करते समय केवल इनके प्रकार का नाम ही बताना होता है अन्य गुणधर्म पूर्व निर्धारित होते हैं जैसे int, float, char आदि। जबकि यूजर डिफाइन्ड में सभी गुणधर्म इन्हें परिभाषित करते समय बताने होते है । जैसे ये आँकड़े किन–2 से मिलकर बने होंगे, इनके क्या मान होंगे, इनका नाम क्या होगा आदि। जैसे कि क्लास में हम यह बताते है कि क्लास में किस–किस प्रकार के आँकड़े होंगे, इसका नाम क्या होगा आदि। डेराइव्ड प्रकार के आंकड़े जो प्राथमिक प्रकार के आँकड़ो से मिलकर बने होते हो जैसे एरे, स्ट्रक्चर आदि।

2.2.1 प्राइमरी प्रकार के आँकड़े (Primary Data Type)

जैसा हम पढ़ चुके हैं कि इन आँकड़ों का प्रारूप पूर्व निर्धारित होता है। इनके विभिन्न प्रकार, आवश्यक स्मृति जगह (Memory Space) व इनके द्वारा संग्रहित अधिकतम मान निम्न सूची में दिया है।

आँकडो का मान जो ये ग्रहण कर सकते है।	आवश्यक स्थान	अधिकतम मान
int	2 byte	-32768 से + 32767
short int	1 byte	-128 से + 127
long int	4 byte	-2,147, 483, 648 से +2,147, 483, 647
unsigned int	2 byte	0 से 65535
unsigned short int	1 byte	0 से 255
unsigned long int	4 byte	0 से 4, 294, 967, 295
signed char	1 byte	-128 से + 127
unsigned char	1 byte	0 से 255
float	4 byte	3.4e - 38 से 3.4 e + 38
double	8 byte	1.7e - 308 से 1.7e +308
long double	10 byte	3.4e - 4932 से 3.4e + 4932

प्राइमरी प्रकार के आँकड़े मुख्यतः चार प्रकार के ही होते हैं।

(1) char

(2) int

(3) float

(4) double

परन्तु इनसे पूर्व long, short, signed व unsigned लगाकर इनकी संग्रहण क्षमता कम या ज्यादा

कर सकते हैं। एक साथ एक से अधिक शब्द भी लगा सकते हैं । जैसे signed long int में long व signed दोनो शब्द लगे हैं।

char :-

किसी भी एक अक्षर को character प्रकार का आंकड़ा कहते है। ये 1 BYTE स्थान घेरते है।

int :-

पूर्णांक प्रकार के आकड़े, जिनमें दशमलव चिन्ह का प्रयोग न हो, को integer प्रकार के ऑकड़े कहते हैं। ये मेमोरी में 2 BYTE स्थान घेरते हैं। इनके पहले Signed, long, unsigned शब्द लगाकर इनकी संग्रहण क्षमता को कम या ज्यादा किया जा सकता है।

float :-

वास्तविक संख्याओं को floating प्रकार के ऑकड़े कहते हैं। इनमें दशमलव चिन्ह का होना आवश्यक है। इनके प्रारूप के बारे में हम स्थिरांक पढ़ते समय पढ़ चुके हैं। यदि इनमें संग्रहित संख्या की शुद्धता बढ़ाना हो तो float के स्थान पर double प्रकार के ऑकड़ों का प्रयोग करें। float प्रकार के ऑकड़े जहां दशमलव चिन्ह के बाद 6 अंक तक संग्रहित कर सकते है वही double प्रकार के ऑकड़े दशमलव के बाद 14 अंक तक की संख्या को ग्रहण कर सकते हैं।

यूजर डिफाइन्ड प्रकार के ऑकडे व डिराइव्ड प्रकार के ऑकडों के बारे में हम आगे अध्ययन करेंगे।

2.2.2 const:-

यह एक सुरक्षित शब्द (Reserved Word) है। इसका प्रयोग किसी भी ऑकड़े को परिभाषित करते समय आंकड़े के प्रकार से पूर्व किया जाता है। यह इस बात की जानकारी देता है कि आंकड़े स्थिरांक प्रकार के हैं । अतः इसका मान पूरे प्रोग्राम में बदला नहीं जा सकता। यदि हम प्रोग्राम में इस मान को बदलने का प्रयास करेंगे तो कम्पाइलर गलती का संदेश प्रदर्शित कर देगा।

उदाहरण के लिए निम्न प्रोग्राम

```
# include <iostream.h>
void main ( )
{
    float rad, area;
    const float PI = 3.14159;
    cout <<"Enter radius of circle";
    cin >>rad;
    area = PI * rad * rad;
    cout << "area of circle is"<<area <<"\n" ;
}
```

इस प्रोग्राम के अन्दर **PI** का मान float प्रकार को घोषित है। परन्तु float शब्द के पूर्व const शब्द यह भी घोषित करता है कि **PI** का मान पूरे प्रोग्राम में 3.14159 ही रहेगा, बदलेगा नहीं।

इस प्रकार स्थिरांकों के पूर्व const शब्द का प्रयोग करने से प्रोग्राम को पढ़कर यह ज्ञात किया जा सकता है कि कौन–2 से स्थिरांक का प्रयोग प्रोग्राम में किया गया है। यदि इसका मान भूलवश बदलने हेतु प्रोग्राम में कथन लिखा हो तो कम्पाइलर इस कथन पर गलती का संदेश दिखा देता है।

2.2.3 चर (Variables)

ऑकड़ों को संग्रहित करने हेतु जो नाम प्रयोग में लेते हैं उन्हें हम चर कहते है। चर, प्रोग्राम के चलने

के दौरान विभिन्न मान ग्रहण कर सकते हैं। चर के नाम प्रोग्राम बनाते समय प्रोग्रामर द्वारा दिये जाते है। ये नाम इस प्रकार दिये जाने चाहिये कि जो आँकड़े ये ग्रहण करे उसका अर्थ जाना जा सके।

जैसे :–

COUNTER, Length, classize, total आदि।

किसी भी चर का नाम, अंग्रेजी वर्णमाला के अक्षर, अंक या '-' (underscore) अक्षर से मिलकर बना हो सकता है। किसी भी चर का नाम देते समय निम्न नियमों का ध्यान रखा जाता है।

(1) नाम का पहला अक्षर अंग्रेजी वर्णमाला के अक्षरों में से होना चाहिये।

(2) नाम साधारणतया 31 अक्षरों का हो सकता है। हालांकि कुछ कम्पाइलर केवल प्रथम 8 अक्षरों को ही महत्व देते हैं।

(3) अंग्रेजी वर्णमाला के बड़े अक्षर व छोटे अक्षर अलग–2 माने जाते हैं। जैसे AMIT व amit अलग–2 होंगे।

(4) नाम कोई सुरक्षित शब्द (key word) नहीं हो सकता।

(5) नाम के मध्य खाली स्थान न हो।

सही नामों के कुछ उदाहरण ramesh, X1, school_name, sum1, distance x_y आदि।

गलत नाम	कारण
char	सुरक्षित शब्द का उपयोग नहीं होगा।
price$	$ चिन्ह का उपयोग नहीं होगा।
group one	खाली जगह नहीं होगी।

2.2.4 चरों की घोषणा (Decleration of Variables)

चर का नाम निश्चित करने के बाद इन्हें प्रोग्राम में उपयोग करने से पूर्व घोषित करना होता है। यह घोषणा कम्पाइलर को दो बातें बताती है।

(1) चर का नाम

(2) चर किस प्रकार के आँकड़े ग्रहण करेगा या चर को स्मृति (Memory) में कितनी जगह की आवश्यकता होगी।

प्राथमिक प्रकार के चरों की घोषणा निम्न प्रकार से की जा सकती है।

आंकड़े का प्रकार चर1, चर2, ...;

यहां चर 1, चर 2, ... आदि उन चरों के नाम है जो कि दिये गये प्रकार के है। यदि ये एक से अधिक चर हो तो इनके मध्य कोमा (,) होना आवश्यक है। पंक्ति के अन्त में ; (semicolon) होना आवश्यक है।

उदाहरण :

```
int     xyz;
int     p, q, sum;
float   noofst, average;
```

एक पंक्ति में एक ही प्रकार के आँकड़ों को घोषित कर सकते हैं। दूसरे प्रकार के लिए अलग पंक्तियां प्रयोग करें।

2.2.5 चरों को प्रारंभिक मान देना (Initialisation of Variables)

यदि हम चाहें तो चर के प्रकार की घोषणा के साथ–2 चर को आरम्भिक मान भी दे सकते हैं।
ऑंकड़ों के प्रकार चर 1 = मान, चर 2 = मान ....;
उदाहरण –

int x = 20, y = 39, z = 15;

किसी भी चर के प्रारम्भिक मान देना initialisation कहलाता है।
इस प्रकार हम चर की घोषणा के साथ आरम्भिक मान भी दे सकते हैं।
चर को आरम्भिक मान देने का अन्य तरीका भी है। इसमें पहले चरों की घोषणा करते हैं तथा बाद में इनके प्रारम्भिक मान देते हैं।

```
int x, y, z;
x=20;
y=39;
z=15;
```

इस प्रकार भी हम चर को आरम्भिक मान दे सकते हैं।

2.3 असाइनमेन्ट कथन (Assignment Statement)

किसी भी चर में कोई मान ग्रहण कराने के लिए जो कथन लिखा जाता है उसे असाइनमेन्ट कथन कहते हैं। इस कथन में '=' (बराबर) का चिन्ह का उपयोग किया जाता है। किसी भी चर में मान ग्रहण कराने का निम्न तरीका है–

चर का नाम = मान ;

उदाहरण

```
initial = 0;
Mid = 30;
Average = 25.29;
Yes = p + q ;
```

इन कथनों का अर्थ इस प्रकार है कि '=' चिन्ह के दायीं ओर लिखे मान अथवा अभिव्यक्ति का मान '=' चिन्ह के बायीं तरफ लिखे चर में ग्रहण करावें। अतः

Yes = p + q कथन में पहले चर p व q का मान जोडने के पश्चात् कुल आये परिणाम को yes चर में ग्रहण कराया जायेगा।

कथन x = x + 1

में दायीं ओर के चर x के मान में 1 जोड़ने के बाद जो मान आया उसे बायीं ओर के चर x में ग्रहण करावें। माना कि x का मान 5 हो तो इस कथन के निष्पादित होने के बाद x का मान x+1 = 6 हो जायेगा।

2.4 ऑपरेटर्स एवं अभिव्यक्ति (OPERATORS & EXPRESSIONS)

ऑपरेटर चिन्ह एवं कथन (Operator Sign and Statement)

C व C++ भाषा में विभिन्न प्रकार के ऑपरेटर (Operators) काम में लिये जाते है। जैसे +, -, *, = आदि। ऑपरेटर चिन्ह वे चिन्ह होते हैं जो कम्प्यूटर को आँकड़ों पर अंकगणितीय या तार्किक क्रियाएं करने के बारे में निर्देश देते हैं।

ऑपरेटर निम्न प्रकार के होते हैं।

(1) अर्थमेटिक ऑपरेटर (Arithmetic Operator)

(2) रिलेशनल ऑपरेटर (Relational Operator)

(3) लॉजिकल ऑपरेटर (Logical Operator)

(4) इन्क्रीमेन्ट / डिक्रिमेन्ट ऑपरेटर (Increament / Decreament Operator)

(5) कन्डीशनल ऑपरेटर (Conditional Operator)

2.4.1 अर्थमेटिक ऑपरेटर :-

C/C++ भाषा में निम्न अर्थमेटिक ऑपरेटर उपलब्ध हैं। ये ऑपरेटर अंकगणितीय संक्रियाएें अर्थात् संख्याओं पर संक्रिया करने के लिये उपयोग में लाये जाते हैं।

ऑपरेटर	कार्य
+	जोड़ने के लिए या धनात्मक चिन्ह के लिए।
−	घटाने के लिए या ऋणात्मक चिन्ह के लिये।
*	गुणा करने के लिए।
/	भाग देने के लिए।
%	शेषफल ज्ञात करने के लिए।

भाग देते समय यदि संख्याएं पूर्णांक (integer) प्रकार ही है तो भागफल में दशमलव के बाद की संख्याएं हटा दी जाती है। जैसे 16 / 5 = 3

जिन चरों अथवा मानों पर संक्रिया होती है उन्हें हम ऑपरेण्ड कहते हैं।

यहाँ 16 व 5 पर संक्रिया हो रही है अतः इनको ऑपरेण्ड कहेंगे तथा इन दोनों ऑपरेण्ड पर भाग संक्रिया हो रही है।

यदि किसी अभिव्यक्ति में सभी ऑपरेण्ड पूर्णांक प्रकार के हो तो उस अभिव्यक्ति को पूर्णांक अभिव्यक्ति (integers expression) कहते है। इस संक्रिया का परिणाम भी पूर्णांक प्रकार का होगा।

उदाहरण के लिये माना a=28, व b=5 हो तो

a - b = 23
a + b = 33
a * b = 140
a / b = 5
a % b = 3

शेषफल संक्रिया के परिणाम का चिन्ह प्रथम ऑपरेण्ड (भाज्य) के चिन्ह के बराबर होता है अर्थात भाज्य ऋणात्मक है तो शेषफल भी ऋणात्मक होगा। यदि भाज्य धनात्मक है तो शेषफल भी धनात्मक

होगा। यदि किसी अभिव्यक्ति में सभी ऑपरेण्ड वास्तविक ओपरेन्ड हो तो उस अभिव्यक्ति को वास्तविक अभिव्यक्ति कहते है। वास्तविक संख्याओं में दशमलव के बाद 6 अंक ही आते है। अतः संक्रिया का परिणाम लगभग समान होता हैं। जैसे

$x = 1.0/3.0 = 0.333333$

$y = 2.0/3.0 = 0.666667$

शेषफल ऑपरेटर (%) वास्तविक संख्याओं पर कार्य नहीं करता। यह केवल पूर्णांक संख्याओं पर ही कार्य करता है।

2.4.2 रिलेशनल ऑपरेटर

हम प्रोग्राम में कई बार दो चरों की तुलना करते है। तथा उनके सम्बन्ध के आधार पर निर्णय लेते हैं। जैसे दो वस्तुओं के मूल्य की तुलना करना आदि। तुलना करने हेतु जो संक्रिया चिन्ह (ऑपरेटर) काम में आते है उन्हें रिलेशनल ऑपरेटर कहते हैं। C++ में निम्न छः रिलेशनल ऑपरेटर है।

ऑपरेटर	अर्थ
<	से छोटा है।
<=	से बराबर या छोटा है।
>	से बड़ा है।
>=	से बराबर या बडा है।
==	के बराबर है।
! =	के बराबर नहीं है।

किसी अभिव्यक्ति में रिलेशनल ऑपरेटर हो तो इस अभिव्यक्ति को रिलेशनल अभिव्यक्ति कहते है। किसी भी रिलेशनल अभिव्यक्ति का परिणाम सही (1) या गलत (0) होता है।

जैसे

$10 < 20$ = सही (1)

$20 < 10$ = गलत (0)

एक साधारण रिलेशनल अभिव्यक्ति में एक रिलेशनल ऑपरेटर व दो ओपरेन्ड (operand) होते है। रिलेशनल अभिव्यक्ति का निम्न प्रारूप होता है।

AE1 R.O. AE2

यहां R.O. एक रिलेशनल ऑपरेटर है। तथा AE1 व AE2 दोनो अर्थमेटिक कथन है, जो स्थिरांक या चर या इनका समूह हो सकते हैं। यदि किसी रिलेशनल अभिव्यक्ति में, अर्थमेटिक ऑपरेटर हो तो पहले अर्थमेटिक ऑपरेटर चलेंगे उसके बाद रिलेशनल ऑपरेटर।

उदाहरण

माना $x = 2$, तथा $y=5$ हो तो

$x+9 > y*2$

में पहले $x + 9$ व $y*2$ के मान पहले निकलेगा उसके बाद $11 > 10$ कथन चलेगा जिसका परिणाम सही (1) आयेगा।

2.4.3 लॉजिकल ऑपरेटर

C व C++ भाषा में तीन लॉजिकल ऑपरेटर हैं, जो निम्न है।

ऑपरेटर	अर्थ
& &	Logical AND
||	Logical OR
!	logical NOT

लॉजिकल ऑपरेटर & & (AND) व || (OR) का प्रयोग एक से अधिक रिलेशनल अभिव्यक्ति को मिलाने के लिये किया जाता है।

जैसे :–

(a > b) & & (a = = 100)

यदि किसी अभिव्यक्ति में दो या दो से अधिक रिलेशनल अभिव्यक्ति को मिलाकर लिखा जाये तो उसे लॉजिकल अभिव्यक्ति कहते है। लॉजिकल अभिव्यक्ति का परिणाम भी सही (1) या गलत (0) होगा। जैसा ट्रूथ टेबल में बताया गया है && (AND) ऑपरेटर के प्रयोग वाले अभिव्यक्ति का परिणाम सही (1) केवल तभी होगा जब && ऑपरेटर के दोनों तरफ के रिलेशनल अभिव्यक्ति का मान सही (1) हो। जबकि || (OR) ऑपरेटर वाले अभिव्यक्ति का परिणाम गलत (0) केवल तभी सही होगा जबकि दोनों तरफ की रिलेशनल अभिव्यक्ति गलत (0) हो

उदाहरण

if (Marks > 60 && age < 14)

इस कथन का मान सही केवल तब होगा, जबकि Marks, 60से बड़ा है तथा age 14 से छोटी हो।

लॉजिकल अभिव्यक्ति का प्रारूप

RE1	RE2	AND &&	OR ||
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

TRUTH TABLE

2.4.4 इन्क्रीमेन्ट व डिक्रिमेन्ट ऑपरेटर (Increment and Decrement Operator)

C व C++ भाषा में दो विशेष ऑपरेटर इन्क्रीमेन्ट व डीक्रिमेन्ट ऑपरेटर उपलब्ध हैं जो अन्य भाषा में नहीं हैं। इंक्रीमेन्ट ऑपरेटर 1 जोड़ने के लिए व डीक्रिमेन्ट ऑपरेटर 1 घटाने के लिए उपयोग में लिये जाते हैं। दोनों ऑपरेटर एक ओपरेन्ड पर ही लागू होंगे। अतः इन्हें यूनेरी ऑपरेटर कहते हैं।

उदाहरण :–

+ + x या x + + का अर्थ x = x + 1 है।

तथा - - x या x - - का अर्थ x = x - 1 है।

यदि ऑपरेटर का प्रयोग चर से पहले करें तो पहले मान बढ़ेगा फिर अन्य संक्रिया होगी। जबकि ऑपरेटर चर के बाद में हो तो पहले अन्य संक्रिया होगी फिर मान बढ़ेगा। जैसे

x = 10 व y = ++x तो y = 11 होगा

जबकि

x = 10, y = x++ हो तो y = 10 होगा

2.4.5 कन्डीशनल ऑपरेटर (Conditional Operator)

जिस प्रकार अन्य भाषाओं में if - then - else कथन का प्रयोग किया जाता है। उसी प्रकार की परिस्थिति के लिए C/C++ भाषा में "?" एक कंडीशनल ऑपरेटर (Conditional Operator) है।

इसका प्रारुप निम्न है–

<Condition> ? <if True>: <else>;

Rexp1 ? exp2 : exp3;

जहां Rexp1, exp2 व exp3 तीनों कथन है। यह ऑपरेटर निम्न प्रकार कार्य करेगा।

यदि Rexp1 कथन का परिणाम सही (1) हो तो exp2 कथन सम्पादित होगा।

जबकि Rexp1 का परिणाम गलत (0) हो तो exp3 कथन सम्पादित होगा

उदाहरण

माना p = 2

q = 8

a = (p>q) ? p : q;

यहां Rexp1 अर्थात p>q है। इसका परिणाम गलत (0) है अतः exp3 अर्थात् q का मान a मे संग्रहित होगा। जो कि 8 है अतः a चर का मान 8 होगा।

यदि Rexp1 अर्थात् p>q का परिणाम सही (1) होता तो exp2 अर्थात p का मान a में संग्रहित होगा जो कि 2 है। अतः चर a का मान 2 होगा। किसी भी कथन में ऑपरेटर की पूर्ववर्तिता निम्न है।

ऑपरेटर	पूर्ववर्तिता
++(post),- - (post)	1
++(pre),- - (pre),-(unary minus)	2
*,/,%	3
+,-	4
<,<=,>,>=	5
==,!=	6
&&	7
||	8
? : conditional expression)	9

उदाहरण

(1) यदि a=2 है तो C=a*++a के मान की गणना करो।

उत्तर :– ऑपरेटर की पूर्ववर्तिता के अनुसार पहले हम ++ गणना करेंगे जिससे उसकी मान (value) 3

आयेगी अब 3 को 3 से गुणा करेंगे, जिससे उसका उत्तर 9 आयेगा।

2.6 अभिव्यक्ति (Expressions)

अभिव्यक्ति का निर्माण आपरेटर, चर (variable) तथा नियतांक (constant) को मिलाकर किया जाता है।

यह +, -, *, / कई तरह के हो सकते है। लेकिन हम यहां पर साधारणतया दो तरह के अभिव्यक्ति दर्शाते हैं।

(i) अंकगणित अभिव्यक्ति (Arithmetic expression)

(ii) तर्कसंगत अभिव्यक्ति (Logical expression)

2.6.1 अंकगणित अभिव्यक्ति (Arithmetic expression)

अंकगणित अभिव्यक्ति वे होते है जो +, -, *, / चिन्ह के प्रयोग द्वारा बनते है।

यहां पर कुछ सार्थक अंकगणित अभिव्यक्ति दर्शाये गये है।

int x, y, z;

float p, q, r;

तब

z/x, x+y, y/z+x, p/r+x-y - सार्थक

p%r - निरर्थक है, क्योंकि % (Modules) चिन्ह float के साथ प्रयुक्त नहीं करते हैं।

2(3*x+y) - निरर्थक है, क्योंकि 2 और 3 के बीच ऑपरेटर नहीं है।

उदाहरण (1) निम्नलिखित अभिव्यक्ति को उससे संबंधित C++ अभिव्यक्ति में लिखिये

(i) $d = b^2-4ac$

(ii) u + ft

(iii) $(2a + b)^2$

उत्तर d = b*b - 4*a*c;

u + f * t

(2*a+b)*(2*a+b)

2.6.2 तर्कसंगत अभिव्यक्ति (Logical expression)

वह अभिव्यक्ति जिसका उत्तर हां (True) या ना (False) में आता है। तर्कसंगत अभिव्यक्ति कहते है।

कुछ सार्थक उदाहरण यहाँ पर दिये गये है।

p > q

!x>y

(p+q)>s

(!q) && (r||p)

2.7 व्यंजक में टाइप का स्वतः परिवर्तन
(Automatic type Conversion in expression)

अपने आप परिवर्तित होने वाले टाइप (Type) वे हैं जो (Compiler) कम्पाइलर द्वारा अपने आप परिवर्तित हो जाते है।

उदाहरण

```
# include <iostream.h>
# include <conio.h>
void main ( )
{
int a=2;
char ch = 'a';
cout <<(a+ch);
getch ( );
}
```

उपरोक्त प्रोग्राम (program) को जब हम कम्प्यूटर पर रन (Run) करते है तो इसका आउटपुट (output) 99 आता है क्योंकि (char) करैक्टर अपने आप ASCII टेबिल के द्वारा integer में परिवर्तित हो जाता है।

उदाहरण

```
# include <iostream.h>
void main ( )
{
int x=7, y;
y=x/2;
cout <<y;
}
```

उपरोक्त प्रोग्राम को कम्प्यूटर पर रन करने पर आउटपुट आता है 3. अगर हम वेरियबल y को int से float में परिवर्तित कर देते हैं। और फिर आउटपुट देखते है तो आउटपुट 3.5 आता हैं ना कि 3 । अगर हम 3.5 आउटपुट लाना चाहते है तो हम type casting का प्रयोग करेंगे।

2.8 टाइप कास्टिंग (Type Casting)

टाइप कास्टिंग अभिव्यक्ति वह होता है जो कि किसी भी अभिव्यक्ति (expression) को विशिष्ट टाइप में परिवर्तित करने के काम आती है।

टाइप कास्टिंग लिखने की रचना इस प्रकार है।

(Data Type) Expression

उपरोक्त प्रोग्राम में अगर हम टाइप कास्टिंग का प्रयोग करते हैं तो परिवर्तित प्रोग्राम इस प्रकार दर्शाया जायेगा।

```
# include <iostream.h>
# include <conio.h>
void main ( )
{
int x=7; float y;
y=(float) x/2;
cout <<y;
getch ( );
}
```

अब यह प्रोग्राम 3.5 आउटपुट देगा।

कुछ सार्थक उदाहरण निम्न प्रकार है।

(i)
```
int p = 6;
float q;
q=(float) p + 5.7;
```
(ii)
```
int c;
c=(int) 3.5421;
```

2.9 C++ शार्ट हैण्ड (C++ Short hands)

a = a + b को हम शार्ट हैण्ड में a + = b लिखेंगे।

a = a - b को हम शार्ट हैण्ड में a - = b लिखेंगे।

a = a * b को हम शार्ट हैण्ड में a * = b लिखेंगे।

a = a / b को हम शार्ट हैण्ड में a / = b लिखेंगे।

a = a % b को हम शार्ट हैण्ड में a % = b लिखेंगे।

अभ्यास प्रश्न

वस्तुनिष्ठ प्रश्न (Obective Type Questions)

1.. कम्प्यूटर की मैमोरी के अंदर फ्लोट डाटा टाइप लेता है।
   (अ) 2 बाइट्स (ब) 4 बाइट्स
   (स) 8 बाइट्स (द) उपरोक्त में से कोई भी नहीं
2. बैक स्लेश अक्षर "\"" का विशेष अर्थ होता है।
   (अ) कैरिज रिर्टन (Carriage Return) (ब) होरिजोंटल टैब (Horizontal Tab)
   (स) बैक स्लैश (Back Slash) (द) डबल कोट (Double quote)
3. लॉजिकल आपरेटर && अर्थ होता है।
   (अ) लॉजिकल NOT (ब) लॉजिकल OR
   (स) लॉजिकल AND (द) लॉजिकल NAND
4. C++ भाषा में कितने रिलेशनल आपरेटर होते है।
   (अ) 4 (ब) 5
   (स) 6 (द) 8

लघुत्तरात्मक प्रश्न (Short Answer Type Questions)

1. अभिव्यक्ति कितने प्रकार की होती हैं ?
2. चरों की घोषणा कम्पाइलर को क्या बातें बताती है।?

3. चरों का नाम देते समय किन नियमों का ध्यान रखा जाता हैं ?

निबन्धात्मक प्रश्न (Essay Type Questions)

1. C++ भाषा में उपलब्ध प्राइनरी प्रकार के आंकड़ो पर निवंध लिखिए।
2. C++ भाषा में उपलब्ध आपरेटरों को उदाहरण देकर समझाइए ।
3. C++ भाषा में उपलब्ध अभिव्यक्ति को समझाइए ?

उत्तरमाला

1 (ब) 2 (द) 3 (स) 4 (स)