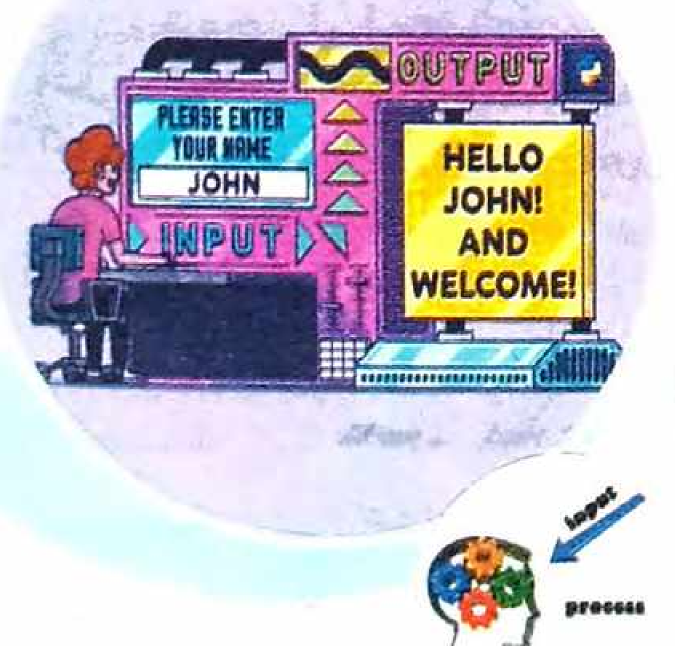
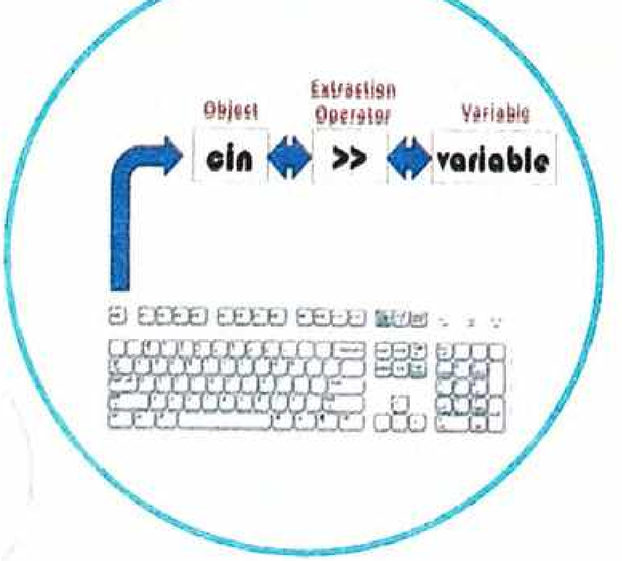
باب نمبر 3

اِن پُٹ اور آؤٹ پُٹ کا عمل

اس باب کی تکمیل کے بعد طلبہ اس قابل ہو جائیں گے کہ وہ:

- آؤٹ پُٹ فنکشنز printf( )، puts( ) اور cout( ) کا استعمال کر سکیں۔
- اِن پُٹ فنکشنز gets()، getchar()، getch()، getche()، scanf() اور cin کو استعمال کر سکیں۔
- اسٹیٹمنٹ ٹرمنیٹر یعنی (semicolon) استعمال کر سکیں۔
- فارمیٹ سپیسیفائر کی تعریف کر سکیں۔
- اسکیپ سیکونس (Escape Sequence) کی تعریف کر سکیں اور پروگرام میں ان کے استعمال کو جان سکیں۔
- حسابی عوامل (Arithmetic Operators) کی تعریف کر سکیں اور پروگرام میں ان کے استعمال کو جان سکیں۔
- اسائنمنٹ آپریٹرز (Assignment Operators) کے استعمال کو جان سکیں۔
- نسبتی آپریٹرز (Relational Operators) کی تعریف کر سکیں اور پروگرام میں ان کے استعمال کو جان سکیں۔
- منطقی آپریٹرز (Logical Operators) کی تعریف کر سکیں اور پروگرام میں ان کے استعمال کو جان سکیں۔
- اسائنمنٹ آپریٹرز اور مساوی آپریٹر کے مابین فرق کر سکیں۔
- یونری (Unary) اور بائنری (Binary) آپریٹرز میں فرق بیان کر سکیں۔
- ثلاثی (Ternary) آپریٹر کی تعریف کر سکیں۔
- آپریٹرز کی آپس میں فوقیت کی تعریف کر سکیں اور اسے سمجھ سکیں۔





NOT FOR SALE

3.1 اِن پُٹ اور آؤٹ پُٹ فنکشنز

C پروگرام تین اہم کام انجام دیتا ہے یعنی وہ ڈیٹا اِن پُٹ کے طور پر حاصل کرتا ہے، ڈیٹا پر کارروائی کرتا ہے اور آؤٹ پُٹ تیار کرتا ہے۔ اِن پُٹ سے مراد ڈیٹا کو وصول کرنا ہے جبکہ آؤٹ پُٹ سے مراد ڈیٹا کا اظہار ہے۔ عام طور پر اِن پُٹ کی بورڈ سے وصول کیا جاتا ہے اور آؤٹ پُٹ اسکرین پر ظاہر ہوتا ہے۔ ہر پروگرامنگ لینگوئج میں اِن پُٹ / آؤٹ پُٹ آپریشن سب سے اہم کام ہوتا ہے۔ C لینگوئج بہت سے اِن پُٹ / آؤٹ پُٹ فنکشن پیش کرتی ہے۔ ڈیٹا حاصل کرنے اور پھر اسے ویری ایبل کو تفویض کرنے کے لئے استعمال ہونے والے فنکشن کو اِن پُٹ فنکشن کہتے ہیں۔ وہ فنکشن جو کمپیوٹر میموری سے ڈیٹا وصول کرنے اور پھر اسے آؤٹ پُٹ ڈیوائسز کو بھیجنے کے لئے استعمال ہوتے ہیں انہیں آؤٹ پُٹ فنکشن کہا جاتا ہے۔

3.1.1 آؤٹ پُٹ فنکشن (Output Functions)

C لینگوئج درج ذیل آؤٹ پُٹ فنکشن کا استعمال کرتی ہے، جیسے:

- printf()
- puts()

جبکہ ++C لینگوئج آؤٹ پُٹ کے لئے درج ذیل کو استعمال کرتی ہے۔

- cout

(a) printf( ) فنکشن

C میں سب سے عام آؤٹ پُٹ فنکشن میں سے ایک printf( ) فنکشن ہے۔ printf( ) اسکرین پر آؤٹ پُٹ ظاہر کرنے کے لئے استعمال ہوتا ہے۔

یہ پیغامات، ویری ایبل کی قیمت اور ریاضی کی عبارتوں (Arithmetic Expressions) کے جوابات کو پرنٹ کرنے کے لئے استعمال ہوتا ہے۔ printf میں "f" فارمیٹڈ کی مختصر شکل ہے اور اشارہ کرتا ہے کہ اسکرین پر چھپی ہوئی عبارت کو فارمیٹ کیا جائے گا۔ فنکشن printf کو print-eff کے نام سے ظاہر کیا جاتا ہے۔ اس کی وضاحت سٹنڈرڈ ہیڈر فائل stdio.h میں کی گئی ہے۔ لہذا printf() کو استعمال کرنے کے لیے ہمیں اپنے پروگرام میں ہیڈر فائل stdio.h شامل کرنا چاہئے۔

printf کا سائنٹکس

```
printf("String");
```

یا

```
printf("format string", var1, var2, var3, ..., varN);
```

format string یہ بتاتا ہے کہ کتنے آرگومنٹس استعمال کئے گئے ہیں اور ان کی اقسام کیا ہیں۔ آرگومنٹ var1، var2، ...، varN وہ ویری ایبل ہیں جن کی format string میں دیئے گئے سپیفائر (specifier) کے مطابق تشکیل اور پرنٹ کی جاتی ہے۔ قیمت ڈیٹا ٹائپ کے مطابق استعمال کیا جاتا ہے۔ آرگومنٹس کی تعداد، ترتیب اور ٹائپ کا format specifier سے مماثل ہونا ضروری ہوتا ہے۔

NOT FOR SALE

پروگرام 3.1 حروف کے ایک سٹرنگ کو اسکرین پر پرنٹ کرنے کا پروگرام

```
# include <stdio.h>
void main()
{
printf("Welcome to Programming Environment\n");
printf("This is my first program in C language\n");
}
```

آؤٹ پٹ

```
Welcome to Programming Environment
This is my first program in C language
```

پروگرام 3.2 printf( ) کی مدد سے 1 2 3 4 پرنٹ کرنے کا پروگرام

```
# include <stdio.h>
void main ( )
{
int a=1,b=2,c=3,d=4;
printf("%d\t%d\t%d\t%d",a,b,c,d);
}
```

آؤٹ پٹ

```
1	2	3	4
```

(b) puts( ) فنکشن

اس کا استعمال اسکرین پر سٹرنگ کو پرنٹ کرنے کے لئے کیا جاتا ہے۔ اس کی تعریف سٹینڈرڈ ہیڈر فائل stdio.h میں موجود ہوتی ہے۔ اس کا سائنٹکس یہ ہے:

puts(string);

- ***string*** کوئی سٹرنگ کانسٹنٹ بھی ہو سکتا ہے یا سٹرنگ ویری ایبل بھی۔
- سٹرنگ کانسٹنٹ کی صورت میں، سٹرنگ کو دوہرے کاموں (Double Quotes) میں بند کیا جائے گا۔
- سٹرنگ ویری ایبل کی صورت میں یہ دوہرے کاموں (Double Quotes) کے بغیر لکھا جائے گا۔

پروگرام 3.3 puts( ) فنکشن کی مدد سے سٹرنگ کو اسکرین پر دکھانا۔

```
#include<string.h>
void main()
{
puts("My name is Khalid ");

}
```

آؤٹ پٹ

```
My name is Khalid
```


NOT FOR SALE

(c) cout آبجیکٹ

++C پروگرام میں آؤٹ پٹ حاصل کرنے کے کے لیے cout کا استعمال کیا جاتا ہے۔ اسے سی آؤٹ پڑھا جاتا ہے اور اس کا مطلب ہے کنسول آؤٹ پٹ۔ اس کو >>(Insertion Operator) کے ساتھ استعمال کیا جاتا ہے، جسے ڈبل اینگل بریکٹ بھی کہا جاتا ہے جیسا کہ شکل نمبر 1.3 میں دکھایا گیا ہے۔ واضح رہے کہ cout>> کو استعمال کرنے کے لیے ہمیں پروگرام میں ہیڈر فائل iostream کو ;using namespace std کے ساتھ لازمی طور پر کوڈ کی پہلی لائن کے طور پر شامل کرنا پڑتا ہے۔

اس کا سائنٹکس یہ ہے:

cout<<Variable/Constant/Expression;

جبکہ

Variable: کوئی بھی ویری ایبل ہو سکتا ہے جس کی قیمت اسکرین پر دکھانا مقصود ہو۔

Constant: مستقل یا کانسٹنٹ کی قیمت بھی اسکرین پر پرنٹ کی جاسکتی ہے۔

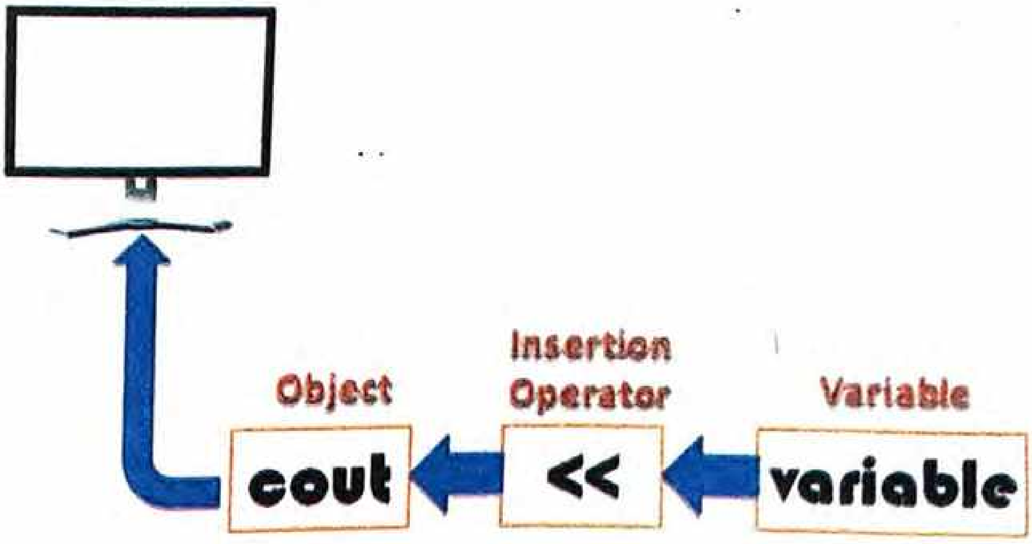
Expression: ریاضی کی کسی عبارت سے حاصل شدہ نتیجہ کو بھی اسکرین پر پرنٹ کیا جاسکتا ہے۔

پروگرام میں ڈبل اینگل بریکٹ کے بعد لکھا ہوا پیغام بطور آؤٹ پٹ کے ظاہر ہو گا۔

مثال کے طور پر:

cout<<"I love my country";

مذکورہ بالا لائن اسکرین پر I love my country دکھائے گی۔ سٹرنگ کانسٹنٹ ہمیشہ دوہرے کاموں میں بند ہوتا ہے۔



شکل نمبر 3.1 cout آبجیکٹ

پروگرام 3.4 دواعداد کے حاصل جمع اور حاصل ضرب کو دکھانے کا پروگرام۔

```
using namespace std;
#include<iostream>
int main()
{
    int a=15, b=22, c;
    cout<<"\nThis program will add and multiply two numbers"<<endl;
    c=a+b;
    cout<<"\nThe sum of "<<a<<" and "<<b<<" is "<<c<<endl;
    cout<<"\n The product of "<<a<<" and "<<b<<" is "<<a*b<<endl;
    return 0;
}
```

آؤٹ پٹ

```
This program will add and multiply two numbers
The sum of 15 and 22 is 37
The product of 15 and 22 is 330
```

یہاں ہم نے endl (اینڈ ایل کے طور پر بولا جاتا ہے) کا استعمال کیا ہے جو بالکل اسکیپ سیکونس n\ کی طرح کام کرتا ہے۔ یہ لائن کو ختم کر دیتا ہے اور کرسر کو اگلی لائن کے آغاز میں لے جاتا ہے۔

3.1.2 اِن پُٹ فنکشنز (Input Functions)

C پروگرامنگ لینگویج اِن پٹ آپریشنز انجام دینے کے لئے بلٹ ان فنکشنز مہیا کرتی ہے۔ اِن پٹ آپریشنز پروگرام چلاتے وقت کی بورڈ سے یوزر سے ویلیو (اِن پٹ) لینے کے لئے استعمال ہوتے ہیں۔ C لینگویج مندرجہ ذیل بنے بنائے اِن پُٹ فنکشن فراہم کرتی ہے۔

- *scanf()*
- *getch()*
- *getche()*
- *getchar()*
- *gets()*

جبکہ C++ لینگویج اِن پُٹ کے لئے درج ذیل کو استعمال کرتی ہے:

- cin

(a) scanf() فنکشن

scanf() فنکشن کا استعمال کی بورڈ سے مختلف ڈیٹا اقسام کے متعدد ڈیٹا ویلیوز کو پڑھنے کے لئے کیا جاتا ہے۔ scanf() فنکشن کی وجہ سے پروگرام آگے نہیں بڑھے گا اور نتائج نہیں دکھائے گا جب تک آپ ویری ایبل کے لیے قیمتیں اِن پُٹ نہیں کرتے۔ scanf() فنکشن کی عمومی شکل درج ذیل ہے۔

scanf ("format specifiers", &var1, &var2, &var3,......... ;).

جبکہ،

"format specifier" فارمیٹ تفصیلات کے ذریعہ ویری ایبل کی شکل متعین کرتا ہے۔ یہ دہرے کاموں میں لکھا جاتا ہے۔

&var1, &var2, & var3,...... ویری ایبل کی فہرست ہے۔ جس میں ایک ویری ایبل کو دوسرے ویری ایبل سے کاما کی مدد سے علیحدہ

NOT FOR SALE

کیا گیا ہے۔ یہ وہ ویری ایبل ہیں جنہیں کوئی قیمت دینی ہے۔ & کی علامت کو scanf کے ساتھ استعمال کیا جاتا ہے جو میموری ایڈریس کو ظاہر کرتی ہے۔

مثال کے طور پر، ڈیٹا ٹائپ انٹیجر کے دو عددی ویری ایبل، n1 اور n2 میں قیمتیں درج کرنے کے لئے ( ) scanf فنکشن ایسے لکھا جائے گا:

scanf("%d,%d", &a, &b);

d% کا فارمیٹ سپیسیفائیر انٹیجر ویری ایبل کے لیے استعمال ہوتا ہے۔

پروگرام 3.5 طالب علم کے رول نمبر، حاصل کردہ نمبر، کل نمبر اِن پُٹ کروا کے فیصد معلوم کرنے کے لیے پروگرام۔

```
#include<stdio.h>
void main()
{
int rn,tm,om;
float pm;
printf("\nEnter Roll number= ");
scanf("%d",&rn);
printf("\nEnter Total marks= ");
scanf("%d",&tm);
printf("\nEnter Obtained marks= ");
scanf("%d",&om);
pm=om/tm*100;
printf("\nPercentage marks=%f",pm);
}
```

آؤٹ پُٹ

```
Enter Roll number = 10
Enter Total marks = 800
Enter Obtained marks = 650
Percentage marks = 81.25
```

(b) getche () ،getch () اور getchar () فنکشنز

getch() ، *getche()* اور *getchar()* اِن پُٹ فنکشنز ہیں جو کی بورڈ سے ایک حرف (Character) کی اِن پُٹ کے لیے استعمال ہوتے ہیں۔ ان فنکشنز میں کسی بھی آرگومنٹ کی ضرورت نہیں ہے۔ ان فنکشنز کے ساتھ صرف خالی بریکٹ ہی استعمال کی جاتی ہے۔

getch( )

() getch کا فنکشن کی بورڈ سے ایک کریکٹر اِن پُٹ کرنے کے لیے استعمال ہوتا ہے لیکن اس فنکشن سے جو کریکٹر اِن پُٹ کیا جاتا ہے وہ سکرین پر دکھائے بغیر اِن پُٹ ہوتا ہے اور ویری ایبل میں اسٹور ہو جاتا ہے۔

NOT FOR SALE

پروگرام 3.6

```
#include<stdio.h>
void main()
{
    char ch;
    printf("Enter Char:");
    cch=getch();
    printf("\n");
    putchar(ch);
}
```

آؤٹ پٹ

```
Enter Char:
j
```

getche()

getche ( ) فنکشن بھی کی بورڈ سے ایک کیریکٹر پڑھتا ہے لیکن echo کے ساتھ۔ یعنی جو حرف اِن پُٹ کیا جاتا ہے وہ اسکرین پر نظر بھی آتا ہے۔

پروگرام 3.7

```
#include<stdio.h>
void main()
{
    char ch;
    printf("Enter Character:");
    ch=getche();
    printf("\n");
    putchar(ch);
}
```

آؤٹ پٹ

```
Enter Char: k
k
```

getchar()

getchar () فنکشن کی بورڈ سے ایک حرف کی اِن پُٹ لیتا ہے لیکن اِن پُٹ کرنے کے بعد Enter key ضرور دبانی پڑتی ہے۔

پروگرام 3.8

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char ch;
    printf( " Enter Char: ");
```

آؤٹ پٹ

```
Enter Char: h
h
```


NOT FOR SALE

```
    ch= getchar();
    printf("\n");
    putchar(ch);
}
```

gets () فنکشن

gets () فنکشن اس قابل ہے کہ یہ کریکٹر سٹرنگ کے درمیان خالی جگہ (White Spaces) کو بھی بطور سٹرنگ لیتا ہے۔ gets فنکشن اسٹرنگ کی ایک لائن یا حرفوں کی ترتیب کو پڑھتا ہے جب تک کہ یوزر Enter key نہ دبائے۔ اس کی عمومی شکل درج ذیل ہے۔

سانٹکس:

gets(Array_Name)*;*

پروگرام 3.9

```
#include<stdio.h>
void main()
{
    char name[20];          // Here [20] specifies the size of the string
    printf("Enter your Name: ");
    gets(name);
    puts(name);
}
```

آؤٹ پُٹ

```
Enter your Name: Muhammad Khalid
Muhammad Khalid
```

gets( ) فنکشن خالی جگہ (White space) کے حروف کے ساتھ ایک سٹرنگ لیتا ہے۔ اس صورت میں، پروگرام چلاتے وقت کی بورڈ سے اسٹرنگ اِن پُٹ کی جائے گی اور جب یوزر Enter بٹن دبائے گا تب اِن پُٹ مکمل ہو گی۔ scanf( ) اور gets( ) کے فنکشنز میں یہ فرق ہے کہ scanf( ) تمام حروف کو اس وقت تک پڑھتا ہے جب تک کہ space نہیں آجاتا یعنی ٹیب یا نئی لائن کا سامنا نہ کرنا پڑے، جب کہ gets () خالی جگہوں اور ٹیب سمیت سب کچھ پڑھے گا جب تک کہ یوزر Enter key نہیں دباتا۔

پروگرام 3.10 کسی شخص کا نام اور عمر (سالوں میں) حاصل کرنے کے لئے ایک پروگرام لکھیں۔ مہینوں میں عمر کا حساب لگائیں اور مہینوں میں اس شخص کی عمر پرنٹ کریں۔

```
#include<stdio.h>
void main( )
{
    int age,age_mon;
    char name[40];
    printf("Enter your Name: ");
    gets(name);
    printf("\nEnter your Age: ");
```

آؤٹ پُٹ

```
Enter your Name: Muhammad Anwar
Enter your Age: 16
Age in months = 12
```

NOT FOR SALE

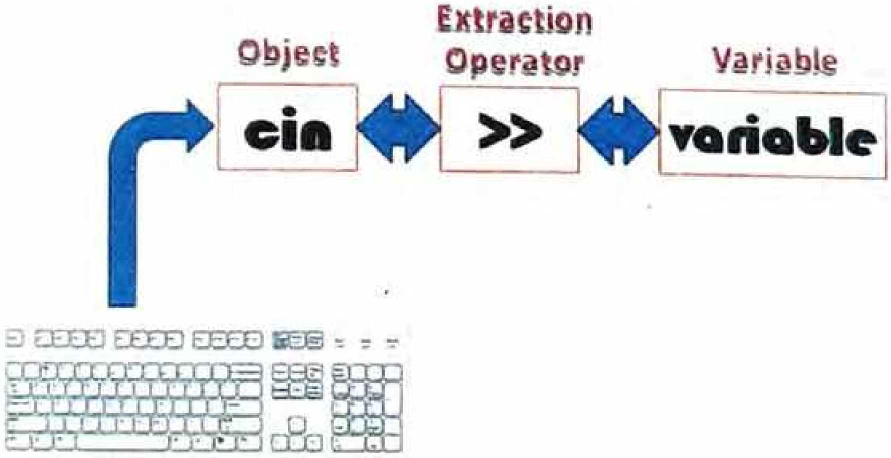
```
scanf("%d",&age);
age_mon=age*12;
printf("\nAge in months = %d ",age_mon);
}
```

(c) cin

کسی ++ C پروگرام میں کی بورڈ سے رن ٹائم پر ان پٹ لینے کے لیے cin کنسول ان پٹ کا استعمال کیا جا سکتا ہے۔ cin کو سی ان پڑھا جاتا ہے۔ <<cin آپریٹر شامل کر کے سٹینڈرڈ ان پٹ کو حاصل کرتا ہے۔ آپریٹر کے بعد ویری ایبل ہونا ضروری ہے جو ان پٹ ڈیٹا کو اسٹور کرنے کے لئے استعمال ہوتا ہے جیسا کہ شکل نمبر 2.3 میں دکھایا گیا ہے۔ یہ پروگرام کو اس وقت تک روکے رکھتا ہے جب تک کی بورڈ سے ان پٹ درج نہیں ہو جاتی۔ cin کو عددی قیمت اور کیریکٹر ان پٹ دونوں کے لیے استعمال کیا جا سکتا ہے۔ cin کو پروگرام میں استعمال کرنے کے لیے ہمیں پروگرام میں ہیڈر فائل iostream کو ;using namespace std کے ساتھ لازمی طور پر کوڈ کی پہلی لائن کے طور پر شامل کرنا پڑتا ہے۔

اس کا سانٹیکس درج ذیل ہے۔

cin>>var1>> var2>>.......;



شکل نمبر 3.2 cin کا استعمال

مثال کے طور پر:

cin>>k;

یہ کی بورڈ سے ایک قیمت حاصل کرے گا اور اسے ویری ایبل k میں محفوظ کرے گا۔

cin>>a>>b>>c;

یہ کی بورڈ سے تین ویلیوز حاصل کرے گا اور اسے ویری ایبل a، b اور c میں محفوظ کرے گا۔

NOT FOR SALE

ایک اہم بات یہ نوٹ کی جائے کہ، یہ لازمی ہے کہ کی بورڈ سے درج کی گئی قیمت ویری ایبل کی ڈیٹا ٹائپ سے مماثل ہونی چاہیے۔ ( )scanf فنکشن کی طرح، cin میں ویری ایبل کے نام کے ساتھ & آپریٹر کی ضرورت نہیں ہوتی ہے۔

پروگرام 3.11

فارن ہائیٹ درجہ حرارت کو سینٹی گریڈ درجہ حرارت میں تبدیل کرنے کے لئے ایک پروگرام لکھیں۔ فارن ہائیٹ درجہ حرارت رن ٹائم پر کی بورڈ سے اِن پُٹ ہونا چاہیے۔

```
using namespace std;
#include<iostream>
int main( )
{
   float ftemp, ctemp;
   cout<<"\n Please input a value for Fahrenheit temperature: ";
   cin>>ftemp;
   ctemp = (ftemp-32) * 5 / 9;
   cout<<"\nCelsius equivalent of Fahrenheit :"<< ftemp<<" is "<<ctemp;
   return 0;
}
```

آؤٹ پُٹ

```
Please input a value for Fahrenheit temperature: 98.6
Celsius equivalent of Fahrenheit : 98.6 is 37
```

3.1.3 اسٹیٹمنٹ ٹرمینیٹر (Statement Terminator)

C کے ہر پروگرام میں ایک اسٹیٹمنٹ کو دوسری اسٹیٹمنٹ سے جدا کرنے کے لیے اسٹیٹمنٹ کے اختتام پر سیمی کالن (;) کا استعمال کیا جاتا ہے۔ یہ سیمی کالن کمپائلر کو بتاتا ہے کہ ایک اسٹیٹمنٹ اس جگہ ختم ہو گئی ہے۔ اگر ایک اسٹیٹمنٹ کو سیمی کالن سے ختم نہیں کیا جاتا تو، C کا کمپائلر کمپائل کرنے کے عمل کے دوران Error Message دے گا اور پروگرام کمپائل نہیں کیا جائے گا۔

مثال کے طور پر،

```
printf("Hello, World! \n"); // Semicolon is used to show end of statement
```

3.1.4 فارمیٹ سپیسیفایرز (Format Specifiers)

فارمیٹ سپیسیفائر اِن پُٹ اور آؤٹ پُٹ مقاصد کے لئے C میں استعمال ہوتا ہے۔ اس کے استعمال کرنے سے کمپائلر یہ سمجھ سکتا ہے کہ ()scanf فنکشن اور ()printf فنکشن میں ویری ایبل کس قسم کا ڈیٹا لیتا یا پرنٹ کرتا ہے۔

ذیل میں فارمیٹ سپیسیفایئر کی ایک فہرست دی گئی ہے۔

NOT FOR SALE

Format specifier	Purpose
%d or %i	Signed integer
%ld	long integer
%f	floating-point (decimal notation)
%g	floating-point (exponential notation)
%e	floating-point (Exponential notation)
%c	single character
%s	String

یہ بنیادی فارمیٹ سپیسیفائر ہیں۔ ہم فارمیٹ سپیسیفائر کے ساتھ کچھ اور حصے بھی شامل کر سکتے ہیں۔

ایک مائنس علامت (-) بائیں سیدھ (left alignment) کو بتاتا ہے۔

% کے بعد کا نمبر فیلڈ کی کم سے کم چوڑائی متعین کرتا ہے۔ اگر سٹرنگ چوڑائی سے کم ہے تو، یہ خالی جگہوں سے پُر ہو گا۔

ایک پیریڈ (. فیلڈ) کی چوڑائی کو پریسین سے الگ کرنے کے لئے استعمال کیا جاتا ہے۔

انٹیجر فارمیٹ سپیسیفائر (%d, %ld and %i)

%d کا فارمیٹ سپیسیفائر انٹیجر اور %ld کا فارمیٹ سپیسیفائر لانگ انٹیجر کو پرنٹ کرنے کے لیے استعمال کیا جاتا ہے۔

پروگرام 3.12 انٹیجر فارمیٹ سپیسیفائر کے استعمال کو ظاہر کرتا ہے۔

```
#include<stdio.h>
void main()
{
    int a, b;
    a = 14;
    b = a / 2;
    printf("a=%d\n",a);        // Here %d will be replaced with the value of a
    printf("b=%d\n",b);        // Here %d will be replaced with the value of b
}
```

آؤٹ پٹ

```
a=14
b=7
```

فلوٹنگ پوائنٹ فارمیٹ سپیسیفائر (%f, %e and %g)

%f کا فارمیٹ سپیسیفائر فلوٹنگ پوائنٹ اعداد کو نقطہ اعشاریہ کے ساتھ دکھاتا ہے۔ فارمیٹ سپیسیفائر %e فلوٹنگ پوائنٹ نمبر کو ایکسپونینشل نوٹیشن (Exponential Notation) میں ان پٹ کرنے اور دکھانے کے لیے استعمال کیا جاتا ہے۔

%g کا فارمیٹ سپیسیفائر بھی %e کی طرح ہی کام کرتا ہے۔

NOT FOR SALE

پروگرام 3.13 میں فلوٹ فارمیٹ سپیسیفائر کا استعمال دکھایا گیا ہے۔

```
#include<stdio.h>
void main()
#include<conio.h>
void main()
{
int a;
float b;
a = 15;
b = a / 2;
printf("a=%d\n",a);            // Here %d will be replaced with the value of a
printf("b=%f\n",b);            // Here %f will be replaced with the value of b
printf("b=%e\n",b);            // Here %e will be replaced with the value of b //in exponential
}
```

آؤٹ پُٹ

```
a=15
b=7,500000
b=7.5e+0
```

کریکٹر فارمیٹ سپیسیفائر %c (Character Format Specifier)

کریکٹر فارمیٹ سپیسیفائر ایک وقت میں صرف ایک کریکٹر کو اِن پُٹ کرنے کے لیے استعمال کیا جاتا ہے۔ پروگرام 3.14 میں %c کے استعمال کو دکھایا گیا ہے۔

پروگرام 3.14

```
#include<stdio.h>
void main()
{
    char ch='A';
    printf("Character=%c\n",ch);    // Here %c will be replaced with the value of ch
}
```

آؤٹ پُٹ

```
Character=A
```

3.1.5 اسکیپ سیکونسز (Escape Sequences)

اسکیپ سیکونسز ایسے خاص حروف ہیں جو پرنٹ تو نہیں ہوتے لیکن آؤٹ پُٹ کی شکل میں ترمیم کرنے کے لئے استعمال ہوتے ہیں۔ یہ حرفوں کا مجموعہ ہیں جو بیک سلیش \ کے ساتھ شروع ہوتے ہیں، اس کے بعد حرف یا خاص علامت ہوتی ہے۔ آؤٹ پُٹ کو فارمیٹ کرنے کے لئے اسکیپ سیکونسز کا استعمال کیا جاتا ہے۔ یہ اکیلے یا سٹرنگ کانسٹنٹ کے درمیان استعمال ہوتے ہیں۔ انہیں ہمیشہ دوہرے کاموں میں بند کیا جاتا ہے۔ آؤٹ پُٹ سٹرنگ میں کہیں بھی اسکیپ سیکونسز استعمال کی جاسکتی ہے۔

NOT FOR SALE

عام طور پر استعمال ہونے والے اسکیپ سیکوئنسز یہ ہیں:

اسکیپ سیکوئنسز	مقصد	مثال	آؤٹ پٹ
\n	نئی لائن (پرنٹنگ کنٹرول کو اگلی لائن کے آغاز پر منتقل کرتا ہے)	printf("Hello\nWorld");	Hello World
\t	افقی ٹیب (کرسر کو ایک ٹیب آگے بڑھاتی ہے۔)	printf("Hello\tWorld");	Hello World
\a	بیل (الرٹ) کمپیوٹر کے اندرونی سپیکر میں بیپ کی آواز پیدا کرتی ہے۔	printf("Hello \a how are you?");	Beep after Hello and before how are you?
\b	Backspace (پرنٹ کے بعد کرسر اپنی جگہ واپس منتقل کرنے کے لیے استعمال کیا جاتا ہے)	printf("Hello\bWorld ");	"Hello" is printed but \b deletes 'o'. Then "World" is printed. So "HellWorld" is displayed on the screen.
\f	فارم فیڈ (خالی فارم فیڈ کرنے کے بعد آؤٹ پٹ کو پرنٹر پر پرنٹ کرنے کے لیے استعمال کیا جاتا ہے)	printf("Hello\fWorld");	HelloWorld After printing "Hello", the computer will include a blank paper and then print the remaining output. It is used during printing.
\'	سنگل کوٹ (سنگل کوٹ ، کو پرنٹ کرنے کے لیے استعمال کیا جاتا ہے)	printf("\'Hello World\'");	'Hello World'
\"	دوہرا کوٹ (دوہرے کوٹ " کو پرنٹ کرنے کے لیے استعمال کیا جاتا ہے۔)	printf("\"Hello World\"");	"Hello World"
\\	Backslash (بیک سلیش پرنٹ کرنے کے لیے استعمال کیا جاتا ہے)	printf(\\Hello World\\);	\Hello World\

3.2 آپریٹرز (Operators)

آپریٹرز خصوصی علامتیں ہیں جو کمپائلر کے ذریعے ڈیٹا پر مختلف قسم کے آپریشن دیتی ہیں۔ C لینگویج میں استعمال ہونے والے بہت سے آپریٹرز ہیں۔ ان میں حسابی (Arithmetic) آپریٹرز، اسائنمنٹ (Assignment) آپریٹرز، ریلیشنل (Relational) آپریٹرز اور لاجیکل (Logical) آپریٹرز شامل ہیں۔

NOT FOR SALE

3.2.1 حسابی آپریٹرز (Arithmetic Operators)

حسابی آپریٹرز وہ علامات ہیں جو ڈیٹا پر ریاضی کے عوامل کے لئے استعمال ہوتی ہیں۔ C لینگویج مختلف آپریٹرز کو مختلف حسابی عمل انجام دینے کے لئے معاونت کرتی ہے جیسے جمع، تفریق، تقسیم، ضرب وغیرہ۔ C میں تمام حسابی آپریٹرز کی فہرست مندرجہ ذیل ہے۔

آپریشن	علامت	تفصیل
جمع	+	دو قیمتوں کو جمع کرتا ہے۔
تفریق	–	ایک قیمت کو دوسری سے تفریق کرتا ہے۔
ضرب	*	دو قیمتوں کو ضرب دیتا ہے۔
تقسیم	\	ایک قیمت کو دوسری قیمت سے تقسیم کرتا ہے۔
ماڈولس	%	دو انٹیجر کی تقسیم سے حاصل شدہ باقی ماندہ قیمت معلوم کرتا ہے۔

تمام حسابی آپریٹرز صرف عددی قیمتوں کے ساتھ ہی عمل کر سکتے ہیں۔ فرض کریں ہمارے پاس A اور B دو ویری ایبلز ہیں جہاں A = 8 اور B = 3 ہے۔ A اور B پر مندرجہ ذیل حسابی عوامل استعمال ہو سکتے ہیں۔

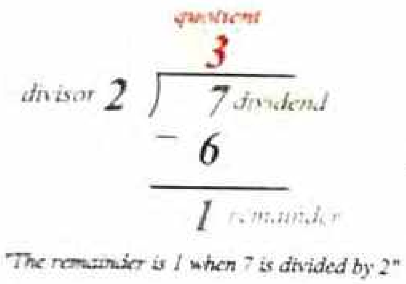
آپریٹر	مثال(int A=8, B=3)	نتیجہ
+	A+B	11
–	A–B	5
*	A*B	24
\	A/B	2 (Integer Division)
%	A%B	0

ماڈولس آپریٹرز کے حوالے سے چند اہم نکات درج ذیل ہیں۔

- C میں ماڈولس آپریٹر فیصد علامت % آپریٹر کے ذریعہ ظاہر کیا جاتا ہے۔
- ماڈولس آپریٹر کو ریمینڈر آپریٹر بھی کہا جاتا ہے۔
- ماڈولس آپریٹر صرف دو آپرینڈز کے درمیان کام کرتا ہے۔
- ماڈولس آپریٹر صرف انٹیجر پر ہی کام کرتا ہے۔
- اگر ماڈولس آپریٹر میں 0 کو کسی عدد سے تقسیم کیا جائے تو اس کا حاصل ہمیشہ 0 ہو گا۔ مثال کے طور پر 0%7 کا حاصل 0 ہو گا۔
- ایکسپریشن 5%7 میں، 5 عدد 7 سے تقسیم نہیں ہو سکتا، اس کا نتیجہ 5 ہو گا۔

NOT FOR SALE

تقسیم کے آپریٹر کا کام



شکل نمبر 3.3 تقسیم کا عمل

ڈویژن آپریٹر کا عمل دوسرے آپریٹرز سے مختلف ہے۔ ڈویژن آپریشن کا نتیجہ ہمیشہ ایک عدد ہوتا ہے اگر شمار کنندہ (Dividend) اور مخرج (Divisor) انٹیجر ہوں۔ اس صورت میں کسری (Fractional) حصہ ختم (Truncate) کر دیا جاتا ہے۔ مثال کے طور پر، 7/2 کا نتیجہ 3 ہے، جیسا کہ شکل نمبر 3.3 میں دکھایا گیا ہے۔

3.2.2 اسائنمنٹ آپریٹر = (Assignment Operator)

اسائنمنٹ آپریٹر کو ویری ایبل (میموری لوکیشن) کو قیمت تفویض کرنے کے لئے استعمال کیا جاتا ہے۔ C میں ایک ہی اسائنمنٹ آپریٹر ہے۔ C میں یہ = علامت اپنے دائیں جانب موجود عبارت (expression) کو حل کر کے اس کی قیمت بائیں جانب موجود ویری ایبل کو تفویض کرتا ہے۔ یہ ویری ایبل کو عددی یا حرف کی قیمت تفویض کرنے کے لئے استعمال کیا جاتا ہے۔ تاہم، اس آپریٹر کے ساتھ اسٹرنگ ڈیٹا کسی ویری ایبل کو تفویض نہیں کیا جا سکتا ہے۔ اسائنمنٹ سٹیمنٹ کا فارمیٹ درج ذیل ہے۔

Variable = Character_constant/Variable/Expression;

مثال کے طور پر نیچے اسائنمنٹ ٹیبل پر غور کریں۔

Operation	Description
x = 10	Assigns 10 to variable x
y = (5 + 4) / 2	Evaluates expression (5+4)/2 and assign result to y
z = x + y	Evaluates x + y and assign result to z
5 = x	Error, you cannot re-assign a value to a constant
10 = 10	Error, you cannot re-assign a value to a constant

اسائنمنٹ آپریٹر کے دائیں جانب (RHS) کانسٹنٹ، ایکسپریشن یا ویری ایبل ہونا ضروری ہے جبکہ بائیں ہاتھ کی طرف (LHS) ایک ویری ایبل (درست میموری لوکیشن) ہونا چاہئے۔

3.2.3 کمپاؤنڈ اسائنمنٹ آپریٹرز (Compound Assignment Operator)

C کمپاؤنڈ اسائنمنٹ آپریٹرز فراہم کرتی ہے جو اسائنمنٹ آپریٹر اور حسابی آپریٹرز کا مجموعہ ہے۔ کمپاؤنڈ اسائنمنٹ آپریٹرز حسابی عوامل کو بہت آسانی سے حل کرنے کی سہولت فراہم کرتے ہیں۔

کمپاؤنڈ اسائنمنٹ آپریٹر کا عمومی سائنٹکس درج ذیل ہے۔

NOT FOR SALE

سائنٹکس:

variable op= expression;

فرض کریں ہمارے پاس A اور B کے دو ویری ایبلز ہیں تو 8 = A اور 3 = B ہے۔ A اور B پر کمپاؤنڈ اسائنمنٹ آپریٹرز استعمال کیے جاسکتے ہیں۔

نتیجہ	(int A=8, B=3)مثال	آپریٹر
11	A+=B or A=A+B	+=
5	A–=3 or A=A–3	–=
56	A*=7 or A=A*7	*=
2 (Integer Division)	A/=B or A=A/B	/=
3	A%=5 or A=A%5	%=
	Value of b will be assigned to a	a=b

پروگرام 11 ایسا پروگرام جو انٹیجر پر تمام کمپاؤنڈ آپریٹرز کو استعمال کرتا ہے۔

```
# include <stdio.h>
void main ()
{
    int a;
    a=10;
    printf("value of a :%d\n",a);
    a+=5;
    printf("value of a after a+=5:%d\n",a);
    a-=5;
    printf("value of a after a-=5:%d\n",a);
    a*=2;
    printf("value of a after a*=2:%d\n",a);
    a/=2;
    printf("value of a after a/=2:%d\n",a);
    a%=2;
    printf("value of a after a%=2:%d",a);
}
```

آؤٹ پٹ

```
Value of a. 10
Value of a after a+=5: 15
Value of a after a– =5: 10
Value of a after a*=2: 20
Value of a after a/=2: 10
Value of a after a%=2: 0
```

3.2.4 اضافہ / انکریمنٹ آپریٹر ++ (Increment Operator)

انکریمنٹ آپریٹر دوہری جمع کی علامت ++ سے ظاہر کیا جاتا ہے۔ اس کا استعمال عددی ویری ایبل کی قیمت میں 1 کا اضافہ کرنے کے لئے کیا جاتا ہے۔ انکریمنٹ آپریٹر صرف ویری ایبل کے ساتھ استعمال کیا جاسکتا ہے۔ اسے کانسٹنٹ یا ایکسپریشن کے ساتھ استعمال

NOT FOR SALE

نہیں کیا جاسکتا۔

مثال کے طور پر اگر کسی ویری ایبل x کی قیمت میں 1 کا اضافہ کرنا ہو تو اس کو ایسے لکھیں گے:

$$x = x + 1;$$

انکریمنٹ آپریٹر ++ استعمال کر کے یہی کام ایسے کرے گا:

$$x++;$$

یہ نوٹ کرنا ضروری ہے کہ x++ یا x = x + 1 یا x + = 1 ایک ہی کام کرتے ہیں۔

انکریمنٹ آپریٹر ویری ایبل سے پہلے یا بعد میں لکھا جاسکتا ہے۔ اگر یہ ویری ایبل سے پہلے لکھا گیا ہے تو، اسے پریفکس انکریمنٹ آپریٹر کہا جاتا ہے۔ اگر یہ ویری ایبل کے بعد لکھا گیا ہے، تو یہ پوسٹ فکس انکریمنٹ آپریٹر کے نام سے جانا جاتا ہے۔

پری فکس انکریمنٹ آپریٹر (Prefix Increment Operator)

پریفکس انکریمنٹ آپریٹر ویری ایبل کی موجودہ قیمت میں 1 کا اضافہ کر دیتا ہے اور پھر اس کی قیمت کو عبارت میں استعمال کیا جاتا ہے۔

مثال کے طور پر،

پروگرام 3.16

```
# include <stdio.h>
void main ( )
{
        int a,b,c,sum;
        a=1;
        b=1;
        c=3;
        \sum= a+ b+ (++c);
        printf("Sum = %d:\n",sum);
        printf("c=    %d",c);
}
```

آؤٹ پٹ

```
sum=6
c=4
```

مندرجہ بالا پروگرام میں c کی قیمت میں 1 کا اضافہ ہو گا اس سے پہلے کہ یہ عبارت میں استعمال ہو۔ اس لیے پروگرام کے اختتام پر sum کی قیمت 6 ہو گی اور c کی قیمت 4 ہو گی۔

پوسٹ فکس انکریمنٹ آپریٹر (Postfix Increment Operator)

جب انکریمنٹ آپریٹر کسی بھی عبارت میں پوسٹ فکس انداز میں استعمال کیا جاتا ہے، تو اس میں ویری ایبل کی قیمت میں، عبارت میں استعمال ہونے کے بعد، 1 کا اضافہ کرے گا۔

NOT FOR SALE

مثال کے طور پر، اگر مذکورہ بالا مثال میں، انکریمنٹ آپریٹر پوسٹ فکس انداز میں استعمال ہوتا ہے تو نتیجہ مختلف ہوگا۔ اسٹیٹمنٹ کچھ ایسے ہوگی:

Sum = a + b + c ++;

ایسی صورت میں، c کی موجودہ قیمت کو عبارت میں استعمال کرنے کے بعد اس میں 1 کا اضافہ کیا جائے گا۔ اس طرح عمل درآمد کے بعد، Sum 5 کے برابر ہوگی اور c کی قیمت 4 ہوگی۔

3.2.5 ڈیکریمنٹ آپریٹر -- (Decrement Operator)

ڈیکریمنٹ آپریٹر کو دوہری مائنس -- علامت سے ظاہر کیا جاتا ہے۔ یہ ایک عددی ویری ایبل کی قیمت سے 1 گھٹانے کے لئے استعمال ہوتا ہے۔ ڈیکریمنٹ آپریٹر صرف ویری ایبل کے ساتھ استعمال کیا جاسکتا ہے۔ اسے کانسٹنٹ یا ایکسپریشن کے ساتھ استعمال نہیں کیا جاسکتا۔

مثال کے طور پر، ویری ایبل x کی قیمت سے 1 کم کرنے کے لئے، ہم درج ذیل اسٹیٹمنٹ کو ایسے لکھیں گے:

--x; یا x--;

x-- یا x=x-1 یا x-=1 ایک ہی طرح کام کرتے ہیں۔

پریفکس ڈیکریمنٹ آپریٹر (Prefix Decrement Operator)

پریفکس ڈیکریمنٹ آپریٹر ویری ایبل کی موجودہ قیمت سے 1 کم کر دیتا ہے۔ پھر اس کی قیمت کو عبارت میں استعمال کیا جاتا ہے۔

مثال کے طور پر:

پروگرام 3.17

```
#include <stdio.h>
void main ()
{
      int a, b, c, sum;
      a=1;
      b=1;
      c=3;
      sum= a  + b + (-- c);
      printf("sum=  %d\n",sum);
      printf("c =  %d",c);
}
```

آؤٹ پٹ

```
sum=4
c=2
```

مندرجہ بالا پروگرام میں c کی قیمت میں 1 کی کمی کر دی جائے گی اور اس کے بعد اسے عبارت میں استعمال کیا جائے گا۔ اس طرح اسٹیٹمنٹ کے چلنے کے بعد sum کی قیمت 4 اور c کی قیمت 2 ہوگی۔

پوسٹ فکس ڈیکریمنٹ آپریٹرز (Postfix Decrement Operator)

جب کسی عبارت میں ڈیکریمنٹ آپریٹر پوسٹ فکس انداز میں استعمال کیا جائے تو یہ ویری ایبل کی قیمت کو پہلے عبارت میں استعمال کرے گا اور پھر اس کے بعد اس کی قیمت میں 1 کو گھٹائے گا۔

مثال کے طور پر، اگر مذکورہ بالا مثال میں، ڈیکریمنٹ آپریٹر پوسٹ فکس انداز میں استعمال ہوتا ہے تو نتیجہ مختلف ہوگا۔ اس صورت میں اسٹیٹمنٹ کچھ ایسے لکھی جائے گی:

sum = a+ b + c--;

اس صورت میں، c کی موجودہ قیمت کو عبارت میں استعمال کرنے کے بعد c کی قیمت سے 1 گھٹایا جائے گا۔ اس طرح عمل درآمد کے بعد، sum کی قیمت 5 کے برابر ہوگی اور c کی قیمت 2 ہوگی۔

3.2.6 نسبتی آپریٹرز (Relational Operators)

نسبتی یا ریلیشنل آپریٹرز کو پروگراموں میں کسی بھی قسم کی شرط (Condition) کی وضاحت کرنے کے لئے استعمال کیا جاتا ہے۔ ایک ریلیشنل آپریٹر دو قیمتوں کا موازنہ کرتا ہے۔ اس کا نتیجہ درست (True) یا غلط (False) کی صورت میں ہوتا ہے۔ ریلیشنل آپریٹرز کو بعض اوقات موازنہ (comparison) آپریٹرز کہا جاتا ہے کیونکہ وہ ٹیسٹ کنڈیشن میں استعمال ہوتے ہیں جن کا نتیجہ ہاں یا ناں میں ہوتا ہے۔ C++ / C درج ذیل ریلیشنل آپریٹرز فراہم کرتا ہے:

Operator	Symbol	Form	Operation
Greater than	>	x > y	true if x is greater than y, false otherwise
Less than	<	x < y	true if x is less than y, false otherwise
Greater than or equals	>=	x >= y	true if x is greater than or equal to y, false otherwise
Less than or equals	<=	x <= y	true if x is less than or equal to y, false otherwise
Equality	==	x == y	true if x equals y, false otherwise
Inequality	!=	x != y	true if x does not equal y, false otherwise

ریلیشنل آپریٹرز کا استعمال (Use of Relational Operators)

اگر x = 10، y = 20 اور z = 5 ہوں تو ریلیشنل آپریٹرز کا آؤٹ پٹ ذیل کے جدول میں دکھایا گیا ہے:

Form	Output Returned
x > y	False
y < z	False
x == z	False
x <= y	True
z >= y	False
y != x	True

NOT FOR SALE

3.2.7 لاجیکل یا منطقی آپریٹرز (Logical Operators)

ایک سے زیادہ کنڈیشن ایکسپریشن کا جائزہ لینے کے لئے لاجیکل آپریٹرز استعمال کیے جاتے ہیں۔C میں تین لاجیکل آپریٹرز ہیں:

- AND &&
- OR ||
- NOT !

(a) اینڈ آپریٹر && (AND Operator)

AND آپریٹر کے لئے استعمال ہونے والی علامت && ہے۔ یہ دو شرائط کا موازنہ کرنے کے لئے استعمال ہوتا ہے۔ اگر دونوں شرائط کا نتیجہ True ہے تو مجموعی نتیجہ True آئے گا اور اگر شرائط میں سے کوئی بھی غلط (False) ہے تو مجموعی نتیجہ False رہے گا۔

Condition 1	Operator	Condition 2	Result
False	&&	False	False
False	&&	True	False
True	&&	False	False
True	&&	True	True

مثال: فرض کریں کہ ہمارے پاس دو ویری ایبل A = 100 اور B = 50 ہیں۔

اس (A> 10) && (B> 10) کا نتیجہ درست (True) ہے، کیونکہ دونوں ہی شرائط درست (True) ہیں۔

اس (A> 50) && (B> 50) کا نتیجہ غلط (False) ہے، کیونکہ پہلی شرط کا نتیجہ درست (True) ہے لیکن دوسری شرط کا نتیجہ غلط (False) ہے۔

(b) OR آپریٹر || (OR Operator)

OR آپریٹر کے لئے استعمال ہونے والی علامت || ہے یہ دو شرائط کا موازنہ کرنے کے لئے استعمال ہوتا ہے۔ اگر کوئی ایک شرط بھی درست (True) ہو تو اس کا نتیجہ True ہو گا۔ اگر دونوں شرائط کا نتیجہ غلط (False) ہے تو اس کا نتیجہ غلط (False) ہو گا۔

Condition 1	Operator	Condition 2	Result
False	||	False	False
False	||	True	True
True	||	False	True
True	||	True	True

مثال: فرض کریں کہ ہمارے پاس دو ویری ایبل A = 100 اور B = 50 ہیں۔

اس (A> 50) || (B> 50) کا نتیجہ درست (True) ہو گا کیونکہ ایک شرط کا نتیجہ درست (True) ہے۔ کیونکہ ایک شرط صحیح ہے۔

NOT FOR SALE

(c) NOT آپریٹر! (NOT Operator)

NOT آپریٹر کے لئے! علامت استعمال ہوتی ہے۔ یہ کسی شرط کے نتیجے کو الٹنے کے لئے استعمال ہوتی ہے۔ اگر شرط کا نتیجہ غلط(False) ہے تو اس کا نتیجہ صحیح(True) ہو گا اور اگر شرط کا نتیجہ صحیح(True) ہے تو اس کا نتیجہ غلط(False) ہو گا۔

Operator	Condition	Result
!	True	False
!	False	True

مثال: فرض کریں کہ ہمارے پاس دو ویری ایبل A = 100 اور B = 50 ہیں۔

حالت !(A == B) صحیح ہے، کیونکہ A B کے برابر نہیں ہے۔

3.2.8 اسائنمنٹ آپریٹر (=) اور مساوی آپریٹر (==) میں فرق

اسائنمنٹ آپریٹر = ایک ویری ایبل کی قیمت مقرر کرنے کے لئے استعمال کیا جاتا ہے جبکہ مساوی آپریٹر == دو ایک جیسی قسم کی قیمتوں کا موازنہ کرنے کے لئے استعمال ہوتا ہے۔

مثال کے طور پر:

```
x=11;
 y=a;
c='s';
```

مذکورہ بالا اسٹیٹمنٹ میں، ویری ایبل x کی قیمت 11 مقرر کی گئی ہے، y کو ویری ایبل a کی قیمت تفویض کی گئی ہے اور c کو 's' کیریکٹر تفویض کیا گیا ہے۔

ریلیشنل آپریٹر == کے استعمال سے شرائط بنائی جاتی ہیں تا کہ کمپیوٹر کوئی بھی فیصلہ کر سکے۔

مثال کے طور پر:

```
b  ==  15;
x  ==  y;
```

پہلی ایکسپریشن میں، اگر B کی قیمت 15 کے برابر ہے تو شرط کا نتیجہ صحیح ہے ورنہ یہ غلط ہے۔ دوسری حالت میں، مساوی آپریٹر کو جانچ پڑتال کے لئے استعمال کیا جاتا ہے کہ آیا x کی ویلیو y کی ویلیو کے برابر ہے، اگر ایسا ہے تو شرط کا نتیجہ صحیح ہے ورنہ غلط ہے۔

3.2.9 یونری (Unary) آپریٹر اور بائنری (Binary) آپریٹر میں فرق

(a) یونری آپریٹرز (Unary Operators)

آپریٹر کی ایک قسم جو سنگل اوپیرینڈ (رقم) کے ساتھ کام کرتی ہے، یونیری آپریٹر کہلاتا ہے۔

مندرجہ ذیل آپریٹرز یونری آپریٹر ہیں:

-, ++,--

NOT FOR SALE

مندرجہ بالا آپریٹرز ایک اوپرینڈ (رقم) کے ساتھ استعمال ہوتے ہیں۔

```
-a;
n++;
--x;
```

(b) بائنری آپریٹر (Binary Operators)

آپریٹر کی ایک قسم جس کو اپنا عمل مکمل کرنے کے لیے دو رقموں کی ضرورت ہو اسے بائنری (Binary) آپریٹر کہا جاتا ہے۔

بائنری آپریٹرز یہ ہیں: +,-,*, /, %

کچھ مثالیں یہ ہیں:

```
x+y;
a-b;
y/z;
x*t;
```

3.2.10 مشروط/کنڈیشنل آپریٹر ? (Conditional Operators)

کنڈیشنل آپریٹر ایک ایسا سٹرکچر ہے جو فیصلے کرنے کے لیے استعمال ہوتا ہے۔ یہ بالکل سادہ سے *if-else* کی جگہ استعمال کیا جاتا ہے۔ اس کو ٹرنری آپریٹر بھی کہا جاتا ہے کیونکہ یہ تین آپرینڈز کے بغیر اپنا عمل مکمل نہیں کر سکتا۔

سائنٹکس:

(Condition)?true-case statement: false-case statement;

Conditions یہ کوئی بھی ریلیشنل یا لاجیکل عبارت ہو گی۔

true-case یہ اس وقت چلے گا جب شرط کا نتیجہ True ہو گا۔

false-case یہ اس وقت چلے گا جب شرط کا نتیجہ False ہو گا۔

مثال:

فرض کریں ہمارے پاس ایک ویری ایبل a = 5 ہے تو درج ذیل اسٹیٹمنٹ

x= (a>50)? 1:0;

اگر شرط a>50 درست ہے تو x کو 1 ویلیو ملے گی، اگر شرط غلط ہے تو x کو 0 ویلیو ملے گی۔ جیسا کہ a چھوٹا ہے 50 سے لہٰذا x = 0۔

3.2.11 آپریٹرز اور ان کی فوقیت (Operators and their Precedence)

آپریٹرز کی فوقیت اصل میں وہ قواعد ہیں جو اس چیز کا تعین کرتے ہیں کہ کسی بھی عبارت میں کون سا عمل پہلے انجام پائے گا۔
4 + 2 * 3 جیسے ایکسپریشن کو حل کرنے کے لیے ہمیں یہ پتہ ہونا چاہئیے کہ آپریٹرز کیا کرتے ہیں، اور ان کو لاگو کرنے کے لئے صحیح ترتیب کیا ہے۔ جس ترتیب میں آپریٹرز کو کسی عبارت میں استعمال کیا جاتا ہے اس کو آپریٹر ترجیح (Operator Precedence) کہا جاتا ہے۔ عام حسابی ترجیحی قواعد (جس میں ضرب کا عمل جمع سے پہلے کیا جاتا ہے) کا استعمال کرتے ہوئے، ہم جانتے ہیں کہ دی گئی عبارت 4+(2*3) کا نتیجہ 10 آئے گا۔

NOT FOR SALE

C میں، تمام آپریٹرز کو ایک خاص سطح کی ترجیح دی جاتی ہے۔ جن کی ترجیح زیادہ ہے ان کو پہلے حل کیا جاتا ہے۔ یہ نوٹ کرنا ضروری ہے کہ ضرب اور تقسیم کی ترجیح جمع اور منفی سے زیادہ ہے۔ کمپائلر ان ترجیحات کا استعمال کر کے اس بات کا تعین کرتے ہیں کہ عبارت کو کیسے حل کرنا ہے۔

3*2+4 کچھ (2*3)+4 کی صورت میں حل کی جائے گی کیونکہ ضرب کے عمل کو جمع پر فوقیت حاصل ہے۔

آپریٹرز کو ذیل میں زیادہ ترجیح سے لے کر کم ترجیح تک درج کیا گیا ہے۔

Precedence	Operators	Type
1	!,-,++,--	Logical NOT, Unary Operators
2	*, / , %	Multiplication, Division & Modulus
3	+, -	Addition & Subtraction
4	<, >, <=, >=	Relational operators
5	= =, !=	Relational operators
6	&&	Logical AND
7	||	Logical OR
8	=	Assignment operator

پروگرام 3.18 C لینگویج میں استعمال ہونے والے تمام آپریٹرز کی فوقیت واضح کرنے والا پروگرام۔

```
#include <stdio.h>
void main()
{
    int a = 20;
    int b = 10;
    int c = 15;
    int d = 5;
    int e;
    e = (a + b) * c / d;     //( 30 * 15 ) / 5
    printf("Value of (a + b) * c / d is : %d\n", e );
    e = ((a + b) * c) / d;   //(30 * 15 ) / 5
    printf("Value of ((a + b) * c) / d is  : %d\n" , e );
    e = (a + b) * (c / d);   //(30) * (15/5)
    printf("Value of (a + b) * (c / d) is  : %d\n", e );
    e = a + (b * c) / d;     // 20 + (150/5)
    printf("Value of a + (b * c) / d is  : %d\n" , e );
}
```

آؤٹ پٹ

```
Value of (a + b) * c / d is : 90
Value of ((a + b) * c) / d is  : 90
Value of (a + b) * (c / d) is  : 90
Value of a + (b * c) / d is  : 50
```


NOT FOR SALE

اہم نکات

- پروگرامنگ میں، اِن پُٹ وہ ڈیٹا ہوتا ہے جو کمپیوٹر کو پروسیسنگ کے لئے دیا جاتا ہے اور آؤٹ پُٹ وہ ہوتا ہے جو ڈیٹا پر کارروائی کے بعد کمپیوٹر تیار کرتا ہے۔
- اسائنمنٹ / اِن پُٹ اسٹیٹمنٹ کے ذریعہ ڈیٹا ویری ایبل کو تفویض کیا جاسکتا ہے۔
- فارمیٹ سپیسیفائر ان پُٹ اور آؤٹ پُٹ فنکشن میں استعمال ہوتے ہیں۔
- ایسکیپ سیکوئنس سے مراد حرفوں کا مجموعہ ہوتا ہے جس کا آغاز بیک سلیش / سے ہوتا ہے۔ اس کے بعد صرف خصوصی علامت ہوتی ہے۔
- scanf( ) فنکشن پروگرام کے نفاذ کے دوران ویری ایبل کے لیے values کو ان پُٹ کرنے کے لئے استعمال ہوتا ہے۔
- پروگرام پر عمل درآمد کے دوران کی بورڈ سے کسی ایک کیریکٹر کو پڑھنے کے لئے getche ()، getch() ، اور getchar() فنکشن استعمال ہوتے ہیں۔
- سیمی کالن ; ہر اسٹیٹمنٹ کے آخر میں استعمال ہوتا ہے۔
- آپریٹر علامتیں ہیں جو ڈیٹا پر مختلف کارروائیوں کے لئے استعمال کی جاتی ہیں۔
- C لینگویج کمپاؤنڈ اسائنمنٹ آپریٹرز فراہم کرتی ہے جو اسائنمنٹ آپریٹر اور حسابی آپریٹرز کا مجموعہ ہے۔
- انکریمنٹ آپریٹر کی نمائندگی ڈبل پلس ++ کے ذریعہ کی جاتی ہے۔
- ڈیکریمنٹ آپریٹر کی نمائندگی ڈبل مائنس -- کے ذریعہ کی جاتی ہے۔
- ریلیشنل آپریٹرز کو پروگراموں میں شرائط لکھنے کے لئے استعمال کیا جاتا ہے۔
- آپریٹر کی ایک قسم جو سنگل اوپیرینڈ کے ساتھ کام کرتی ہے، یونیری آپریٹر کہلاتا ہے۔
- آپریٹر کی ایک قسم جو دو رقموں کے ساتھ کام کرتی ہے اسے بائنری آپریٹر کہا جاتا ہے۔
- آپریٹر کی ایک قسم جو تین رقموں کے ساتھ کام کرتی ہے اسے ٹرنری آپریٹر کہا جاتا ہے۔

مشق

سوال نمبر 1۔ موزوں جواب کا انتخاب کریں۔

i. ایسے فنکشنز جو ڈیٹا لیتے اور پھر انہیں کسی ویری ایبل میں اسٹور کرتے ہیں ________ کہلاتے ہیں۔

(الف) آؤٹ پُٹ فنکشنز (ب) سٹرنگ فنکشنز (ج) عددی فنکشنز (د) اِن پُٹ فنکشنز

ii. ان میں کون سا فنکشن آؤٹ پُٹ کو سکرین پر دکھانے کے لیے استعمال ہوتا ہے؟

(الف) printf() (ب) scanf() (ج) gets() (د) getchar()

NOT FOR SALE

iii. کون سا فنکشن پروگرام کے چلنے کے دوران کسی ویری ایبل کی قیمت رکھنے کے لیے استعمال ہوتا ہے؟

(الف) printf() فنکشن (ب) cout فنکشن

(ج) stdio.h فنکشن (د) scanf() فنکشن

iv. C لینگویج میں ہر اسٹیٹمنٹ کا اختتام ________ سے کیا جاتا ہے۔

(الف) فل سٹاپ (ب) ڈبل کوٹ (ج) سیمی کالن (د) کامہ

v. ذیل میں کون سا فارمیٹ سپیسیفائر انٹیجر کے لیے استعمال ہوتا ہے؟

(الف) %s (ب) %c (ج) %d (د) %f

vi. آؤٹ پٹ پر نئی لائن ڈالنے کے لیے کون سا اسکیپ سیکونس استعمال کیا جاتا ہے؟

(الف) \new (ب) \t (ج) \n (د) \line

vii. ان میں کون سا آپریٹر ماڈولس آپریٹر کہلاتا ہے؟

(الف) + (ب) % (ج) / (د) ◦

viii. کون سا آپریٹر ویری ایبل کی قیمت میں ایک عدد کا اضافہ کر دیتا ہے؟

(الف) '+' آپریٹر (ب) اسائنمنٹ آپریٹر

(ج) ڈیکریمنٹ آپریٹر (د) انکریمنٹ آپریٹر

ix. کون سا آپریٹر ویری ایبل کی قیمت میں عبارت میں استعمال ہونے کے بعد، 1 کا اضافہ کرے گا؟

(الف) پوسٹ فکس آپریٹر (ب) پری فکس آپریٹر

(ج) ماڈولس آپریٹر (د) بائنری آپریٹر

x. کون سا آپریٹر ہے جس کا نتیجہ True ہوگا اگر اس کی دونوں شرائط کا نتیجہ True ہو؟

(الف) AND آپریٹر (ب) OR آپریٹر

(ج) NOT آپریٹر (د) اوپر کے تمام

سوال نمبر 2۔ ذیل میں دیئے گئے مختصر سوالات کے جوابات تحریر کریں۔

i. فارمیٹ سپیسیفائر کیا ہے؟ مثالیں دیں۔

ii. اسکیپ سیکونس کیوں استعمال کیے جاتے ہیں؟ مثالیں دیں۔

iii. gets() فنکشن کس مقصد کے لیے استعمال ہوتا ہے؟

NOT FOR SALE

iv. ()getche اور ()getch فنکشنز میں فرق واضح کریں۔

v. ذیل میں دی گئی عبارتوں کو حل کریں۔

(الف) 9-5*(6+2) (ب) 60/10*24+3 (ج) 001%50-100%3

vi. اسائنمنٹ آپریٹر اور کمپاؤنڈ اسائنمنٹ آپریٹر میں فرق بیان کریں۔

سوال نمبر 4۔ آؤٹ پُٹ فنکشن کیا ہے؟ آؤٹ پُٹ فنکشن کی مختلف اقسام کو تفصیل سے بیان کریں۔

سوال نمبر 6۔ ان پُٹ فنکشن میں استعمال ہونے والے فارمیٹ سپیسیفائر اور اسکیپ سیکونس کی تفصیل سے وضاحت کریں۔

سوال نمبر 8۔ ایک پروگرام لکھیں جو مثلث کا قاعدہ اور اونچائی ان پُٹ کرے اور پھر مثلث کا رقبہ معلوم کر کے پرنٹ کرے۔

سوال نمبر 10۔ ایک پروگرام لکھیں جو طالب علم کا نام اور ایڈریس ان پُٹ کرے اور اسے اسکرین پر پرنٹ کرے۔ اس مقصد کے لیے ()gets اور ()puts کے فنکشن استعمال کریں۔

NOT FOR SALE