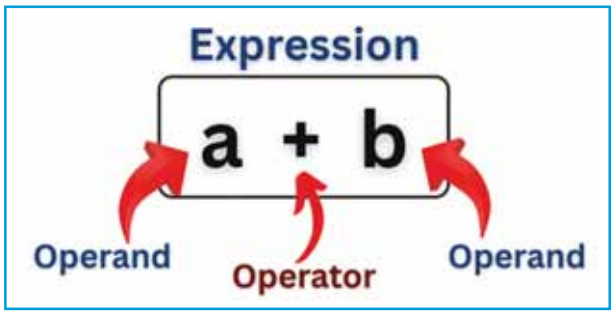
8 ઓપરેટર અને પદાવલીઓ

પરિચય

પાછલા પ્રકરણમાં, આપણે C પ્રોગ્રામિંગના વિવિધ ડેટા પ્રકારો અને સ્ટોરેજ ક્લાસ તેમજ તેમના મહત્ત્વનો અભ્યાસ કર્યો. આ પ્રકરણમાં, C પ્રોગ્રામિંગમાં વપરાતા વિવિધ પ્રકારના ઓપરેટરો અંગે ચર્ચા કરવામાં આવી છે. પ્રોગ્રામર દ્વારા પદાવલીઓ (Expressions) બનાવવા અને પ્રોગ્રામના તાર્કિક પ્રવાહને નિયંત્રિત કરવા માટે ઓપરેટરોનો ઉપયોગ થાય છે. ઓપરેટર C પ્રોગ્રામિંગ ભાષાના મુખ્ય ઘટકો છે, જે ચલ અને મૂલ્યો પર વિવિધ પ્રકારની ક્રિયાઓ કરવાની ક્ષમતા આપે છે. આ પ્રકરણમાં ઓપરેટરોને વિવિધ પ્રકારોમાં વર્ગીકૃત કરવામાં આવ્યા છે જેમ કે - ગણિતીય (Arithmatic), સંબંધસૂચક (Relational), તાર્કિક (Logical), અસાઇનમેન્ટ (Assignment), બિટવાઇઝ (Bitwise) વગેરે. પ્રકરણમાં તેમની વાક્યરચના, પ્રાથમિકતા (precedence) અને સંબંધતા (associativity) ના નિયમોનું વિગતવાર વર્ણન છે. વ્યવહારિક ઉદાહરણો દ્વારા બતાવવામાં આવ્યું છે કે પદાવલીઓ કેવી રીતે રચાય છે, મૂલ્યાંકન થાય છે અને પ્રોગ્રામના તર્ક તથા ડેટા સંચાલન માટે કેવી રીતે ઉપયોગમાં લેવાય છે. ઓપરેટર અને પદાવલીઓની સમજ કાર્યક્ષમ અને તર્કસંગત C પ્રોગ્રામ લખવા માટે અત્યંત જરૂરી છે. તે ગણતરીઓ, નિર્ણય લેવાની પ્રક્રિયા (decision-making) અને લૂપમાં વ્યાપકપણે ઉપયોગમાં લેવાય છે. આ પ્રકરણ ભવિષ્યના વધુ અદ્યતન પ્રોગ્રામિંગ સિદ્ધાંતો સમજવા માટે મજબૂત પાયો પૂરો પાડે છે.

પ્રોગ્રામિંગમાં ઓપરેટરો (Operators) અને પદાવલીઓ (Expressions) શું છે?



આકૃતિ 8.1 : ઓપરેટર અને પદાવલી

પ્રોગ્રામિંગમાં ઓપરેટર (Operators) અને પદાવલીઓ એવા સાધનો છે જે ડેટા પર વિવિધ પ્રકારની ક્રિયાઓ કરવા માટે વપરાય છે. ઓપરેટર એ એક પ્રતીક છે જે પ્રોગ્રામને એક અથવા વધુ મૂલ્યો પર ચોક્કસ ક્રિયા કરવા સૂચવે છે. પદાવલી (Expression) એ મૂલ્યો, ચલો (variables) અને ઓપરેટરોનું સંયોજન છે જે કોઈ પરિણામ આપે છે. એક પદાવલીમાં ઓછામાં ઓછો એક ઓપરેન્ડ (operand) અને શૂન્ય અથવા વધુ ઓપરેટર હોય છે. આકૃતિ 8.1 પદાવલિનું ઉદાહરણ દર્શાવે છે.

નીચે પદાવલી (expression) નું એક સરળ ઉદાહરણ આપેલ છે: "5 + 3", જ્યાં -

- 5 અને 3 મૂલ્યો છે. તેમને ઓપરેન્ડ્સ કહેવામાં આવે છે.
- + એ ઓપરેટર છે, જેના કારણે પદાવલીનું મૂલ્યાંકન થઈને પરિણામ 8 મળે છે.

આપણે પદાવલીઓમાં ચલોનો પણ ઉપયોગ કરી શકીએ છીએ. નીચે ચલો વાપરતી પદાવલીનું ઉદાહરણ છે:

```
A = 10
B = 20
Result = A + B
```

અહીં, A અને B ચલના નામો છે; A + B એક પદાવલી છે, જે પરિણામ 30 આપે છે.

હવે, આપણે C પ્રોગ્રામિંગમાં મળતા વિવિધ પ્રકારના ઓપરેટરો વિશે જાણકારી મેળવીશું. મુખ્ય પ્રકારો આ પ્રમાણે છે: ગણિતીય (Arthmatic), સંબંધસૂચક (Relational), તાર્કિક (Logical), બિટવાઇઝ (Bitwise),

અસાઈનમેન્ટ (Assignment), વધારો અને ઘટાડો (Increment & Decrement), અને વિશેષ (Special) ઓપરેટરો. ચાલો ઉદાહરણો સાથે મહત્વના ઓપરેટરોના ઉપયોગને સમજીએ.

ગણિતીય ઓપરેટરો (Arithmatic Operators)

ગણિતીય ઓપરેટરો C પ્રોગ્રામિંગ ભાષામાં ગણિતીય ગણતરીઓ કરવા માટે વપરાય છે. તેઓ આપણને સંખ્યાત્મક મૂલ્યો પર ક્રિયાઓ કરવાની મંજૂરી આપે છે જેથી વિવિધ પ્રકારના પરિણામો પ્રાપ્ત થઈ શકે. ગણિતીય ઓપરેટરોનો ઉપયોગ કરીને આપણે સરવાળા, બાદબાકી, ગુણાકાર, ભાગાકાર, ભાગશેષ જેવી ક્રિયાઓ કરી શકીએ છીએ. આ ઓપરેટરો સરળ ગણનાઓથી લઈને જટિલ અલ્ગોરિધમ સુધીના કાર્ય માટે અત્યંત મહત્વપૂર્ણ છે. આકૃતિ 8.2 ગણિતીય ઓપરેટરની યાદી દર્શાવે છે.

આ ઓપરેટરો બાઈનરી પ્રકારના હોય છે, એટલે કે તેઓ બે ઓપરેન્ડ પર કાર્ય કરે છે.

આકૃતિ 8.2 : ગણિતીય ઓપરેટરો

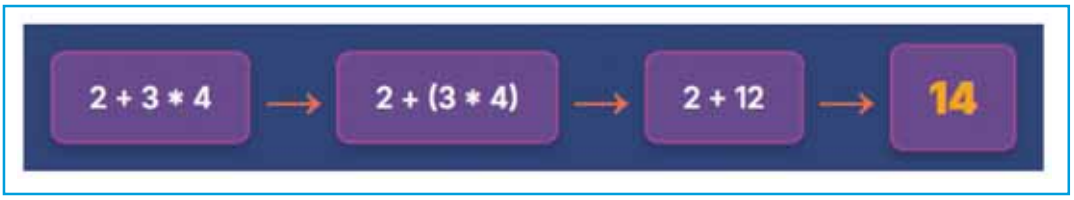
ગણિતીય ઓપરેટરના ઉદાહરણો

નીચેનો C પ્રોગ્રામ બે પૂર્ણાંક પ્રકારના ચલ a અને b પર મૂળભૂત ગણિતીય ઓપરેશન (+, -, , /, %) દર્શાવે છે.

```
/* Program to illustrate use of Arithmatic Operators */
#include <stdio.h>
int main() {
    int a=10, b=3;

    printf("Result of Arithmatic Operations on a and b:\n");

    printf("Addition       : a + b = %d \n", a + b);
    printf("Subtraction    : a - b = %d \n", a - b);
    printf("Multiplication : a * b = %d \n", a * b);
    printf("Division       : a / b = %d \n", a / b);
    printf("Modulus        : a %% b = %d\n", a % b);

    return 0;
}
Result:
Result of Arithmatic Operations on a and b:
Addition       : a + b = 13
Subtraction    : a - b = 7
Multiplication : a * b = 30
Division       : a / b = 3
Modulus        : a % b = 1
```

ગણિતીય ઓપરેટરોના મહત્વપૂર્ણ લક્ષણો

- **પૂર્ણાંક ભાગફળમાં છટણી (Integer Division Truncation) :** જ્યારે ભાગાકાર ઓપરેટર (/) ના બંને ઓપરેન્ડ્સ પૂર્ણાંક હોય, ત્યારે પરિણામ પણ પૂર્ણાંક આવે છે અને કોઈ પણ ભાગાકાર ભાગ (fractional part) દૂર કરવામાં આવે છે. ઉદાહરણ તરીકે, 10 / 3 નું પરિણામ 3 આવે છે, 3.33 નહીં. અપૂર્ણાંક પરિણામ મેળવવા માટે ઓપરેન્ડ્સમાં ઓછામાં ઓછો એક અપૂર્ણાંક પ્રકારનો હોવો જોઈએ, (જેમ કે 10.0 / 3 અથવા 10 / 3.0).
- **મોડ્યુલસ ઓપરેટર (મોડ્યુલસ - %) :** મોડ્યુલસ (ભાગશેષ) ઓપરેટરનો ઉપયોગ માત્ર પૂર્ણાંક ઓપરેન્ડ સાથે જ થઈ શકે છે. તેનો ઉપયોગ અપૂર્ણાંક પ્રકારો (*float* અથવા *double*) સાથે થતો નથી.
- **ઓપરેટરની પ્રાથમિકતા અને સંબંધતા (Operator Precedence and Associativity) :**
  - o પ્રાથમિકતા (Precedence) : ગુણાકાર (*), ભાગાકાર (/), અને ભાગશેષ (%) ઓપરેટરો સરવાળા (+) અને બાદબાકી (–) કરતાં વધુ પ્રાથમિકતા ધરાવે છે. એનો મતલબ છે કે પદાવલીમાં તેઓનું મૂલ્યાંકન પહેલા કરવામાં આવે છે. ઉદાહરણ તરીકે, 2 + 3 * 4 નું મૂલ્યાંકન 14 આવે છે, કારણ કે 3 * 4 પહેલા ગણવામાં આવે છે. (જુઓ આકૃતિ 8.3)



આકૃતિ 8.3 : ઓપરેટર પ્રાથમિકતા

  - o સંબંધતા (Associativity) : જ્યારે પદાવલીમાં સમાન પ્રાથમિકતા ધરાવતા ઓપરેટરો એક સાથે આવે છે, જેમ કે 10 / 2 * 5, ત્યારે તેમનું મૂલ્યાંકન ડાબેથી જમણે (left to right) કરવામાં આવે છે. ઉદાહરણ તરીકે, 10 / 2 * 5 નું મૂલ્યાંકન થાય છેઃ (10 / 2) * 5 = 5 * 5 = 25. તેને આકૃતિ 8.4 માં દર્શાવવામાં આવ્યું છે.

આકૃતિ 8.4 : ઓપરેટર સંબંધતા

- **પ્રકાર રૂપાંતર (Type Conversion) :** જ્યારે અલગ-અલગ ડેટા પ્રકારના ઓપરેન્ડો ગણિતીય ઓપરેટરો સાથે વપરાય છે, ત્યારે C સ્વયંસંચાલિત પ્રકાર રૂપાંતર (implicit type conversion / type promotion) કરે છે. સામાન્ય રીતે, "નાનો" ડેટા ટાઈપ "મોટા" ડેટા ટાઈપમાં પ્રમોટ કરવામાં આવે છે જેથી ડેટા નુકસાન ટાળવું શક્ય બને. ઉદાહરણ તરીકે, જો *int* અને *float* એક જ પદાવલીમાં સાથે વપરાય છે, તો *int* ને પહેલા *float* માં રૂપાંતરિત કરવામાં આવે છે. જેમ કે,

  float result = 2 + 5.5;

  આ સ્થિતિમાં, 2 ને પ્રથમ 2.0 માં ફેરવવામાં આવે છે, પછી 2.0 + 5.5 = 7.5 ની ગણના થાય છે. આ રીતે, **int result = 2 + 5.5;** માટે, પહેલા 2 ને 2.0 માં રૂપાંતરિત કરવામાં આવે છે, પછી 2.0 + 5.5 = 7.5 થાય છે, અને અંતે 7.5 ને 7 કરવામાં આવે છે કારણ કે result ચલનો પ્રકાર *int* છે.

- **શૂન્યથી ભાગાકાર (Division by Zero) :** C માં શૂન્યથી ભાગાકાર અવ્યખ્યાયિત ક્રિયા છે. જો તમે 10 / 0 અથવા 10.0 / 0.0 કરવા પ્રયત્ન કરો તો તે રનટાઈમ ત્રુટી (error) અથવા અનિર્ધારિત પરિણામ આપે છે, જેને લીધે ઘણીવાર પ્રોગ્રામ તૂટી (crash) પડે છે. તેથી આપણા પ્રોગ્રામમાં શૂન્ય વડે ભાગાકાર (division by zero) જેવી કોઈપણ શક્યતાઓને નિવારવી અત્યંત આવશ્યક છે.

સંબંધસૂચક ઓપરેટરો (Relational Operators)

C પ્રોગ્રામિંગમાં સંબંધસૂચક ઓપરેટરોને તુલનાત્મક ઓપરેટરો (Comparison Operators) તરીકે પણ ઓળખવામાં આવે છે. તે પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવા માટે અગત્યના છે, કારણ કે તે ચલ, અચલ અને પદાવલીઓ વચ્ચે તુલના કરીને નિર્ણય લેવા સક્ષમ બનાવે છે. સંબંધસૂચક ઓપરેટરોનો ઉપયોગ બે મૂલ્યોની તુલના કરવા માટે થાય છે. સંબંધસૂચક પદાવલીનું પરિણામ 1 (સાચું) અથવા 0 (ખોટું) સ્વરૂપે મળે છે. આ ઓપરેટરો સામાન્ય રીતે નિર્ણય વિધાનો (*if, else-if*) અને લૂપ (*for, while, do-while*) માં પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવા માટે સૌથી વધુ ઉપયોગમાં લેવાય છે. ટેબલ 8.1માં C પ્રોગ્રામિંગમાં વપરાતા સંબંધસુચક ઓપરેટરો અને તેમનો ઉપયોગ દર્શાવવામાં આવ્યો છે.

સંબંધસૂચક ઓપરેટરનો ઉપયોગ ઉદાહરણ સાથે સમજવા માટે, ચાલો ધારીએ કે આપણા પાસે બે પૂર્ણાંક ચલો a અને b છે. આ બે ચલો પર વિવિધ સંબંધસૂચક ઓપરેટરોના પરિણામો નીચેની કિંમતો પ્રમાણે ટેબલ 8.1માં દર્શાવવામાં આવ્યા છે.

```
int a = 10;
int b = 5;
```

ઓપરેટર	નામ	વર્ણન	C માં ઉદાહરણ (a = 10, b = 5)	પરિણામ
==	Equal to	બે ઓપરેન્ડનું મૂલ્ય સરખું છે તે ચકાસવા	a == b	0 (false)
!=	Not equal to	બે ઓપરેન્ડનું મૂલ્ય સરખું નથી તે ચકાસવા	a != b	1 (true)
>	Greater than	ડાબી બાજુના ઓપરેન્ડનું મૂલ્ય જમણી બાજુના ઓપરેન્ડ કરતા વધારે છે તે ચકાસવા	a > b	1 (true)
<	Less than	ડાબી બાજુના ઓપરેન્ડનું મૂલ્ય જમણી બાજુના ઓપરેન્ડ કરતા ઓછું છે તે ચકાસવા	a < b	0 (false)
>=	Greater than or equal to	ડાબી બાજુના ઓપરેન્ડનું મૂલ્ય જમણી બાજુના ઓપરેન્ડ કરતા વધારે કે સરખું છે તે ચકાસવા	a >= 10	1 (true)
<=	Less than or equal to	ડાબી બાજુના ઓપરેન્ડનું મૂલ્ય જમણી બાજુના ઓપરેન્ડ કરતા ઓછું કે સરખું છે તે ચકાસવા	b <= 5	1 (true)

ટેબલ 8.1 : સંબંધસૂચક ઓપરેટરો

સંબંધસૂચક ઓપરેટરનું ઉદાહરણ

નીચે આપેલો C પ્રોગ્રામ બધા સંબંધસૂચક ઓપરેટરનો ઉપયોગ દર્શાવે છે અને તેમના પરિણામો દર્શાવે છે.

```c
/* Program to illustrate use of Relational Operators */

#include <stdio.h>
int main() {
  int a = 10;
  int b = 5;

  printf("Using a = %d and b = %d \n", a, b);
  printf("---------------------------------\n");

  printf("Is a == b? Result: %d \n", a == b);  /* 10 == 5 is false */
  printf("Is a != b? Result: %d \n", a != b);  /* 10 != 5 is true */
  printf("Is a > b?  Result: %d \n", a > b);   /* 10 > 5 is true */
  printf("Is a < b?  Result: %d \n", a < b);   /* 10 < 5 is false */
  printf("Is a >= b? Result: %d\n",  a >= b);  /* 10 >= 5 is true */
  printf("Is b <= a? Result: %d\n",  b <= a);  /* 5 <= 10 is true */

   return 0;
}
```

```
Result:
Using a = 10 and b = 5
---------------------------------
Is a == b?  Result: 0
Is a != b?  Result: 1
Is a > b?   Result: 1
Is a < b?   Result: 0
Is a >= b?  Result: 1
Is b <= a?  Result: 1
```

યાદ રાખવાના મુખ્ય મુદ્દાઓ

- **== વિરુદ્ધ (verses) = :** નવા શીખનારાઓ પ્રોગ્રામમાં એક ખૂબ જ સામાન્ય ભૂલ એ કરે છે કે શરતમાં સમાનતા ઓપરેટર (==) ને બદલે અસાઈનમેન્ટ ઓપરેટર (=) નો ઉપયોગ કરવો. if (x = 5) એ x ને 5 કિંમત આપે છે ને તે પદાવલી પોતે 5 તરીકે મૂલ્યાંકિત થાય છે (જે શૂન્ય નથી, તેથી સાચું (true) ગણાય છે), જ્યારે *if* (x == 5) એ x ની કિંમત પહેલેથી 5 છે કે કેમ તે તપાસે છે.
- **પરત મૂલ્ય (Return Value) :** સંબંધસૂચક ઓપરેશનનું પરિણામ હંમેશા *int* પ્રકારનું હોય છે: 1 એટલે સાચું (true) અને 0 એટલે ખોટું (false).
- **ડેટા પ્રકારો (Data Types) :** આ ઓપરેટરોનો ઉપયોગ ફક્ત પૂર્ણાંકો જ નહીં, પણ અપૂર્ણાંક સંખ્યાઓ અને અક્ષરો *(char)* ની પણ સરખામણી કરવા માટે થઈ શકે છે. ધ્યાન રાખો કે જ્યારે અપૂર્ણાંક કિંમતોની સમાનતા (==) માટે સરખામણી કરવામાં આવે છે, ત્યારે અપૂર્ણાંક નંબરોની રાઉન્ડિંગ ભૂલો (rounding errors) ને કારણે અણધાર્યા પરિણામો આવી શકે છે. તેથી, ફ્લોટિંગ પોઇન્ટ કિંમતો સાથે (==) નો ઉપયોગ કરતી વખતે સાવચેત રહેવું.

તાર્કિક ઓપરેટરો (Logical Operators)

વિવિધ સમસ્યા-ઉકેલ પરિસ્થિતિઓમાં, આપણને ઘણીવાર આપણા પ્રોગ્રામમાં બહુવિધ શરતોને જોડવાની જરૂર પડે છે. C પ્રોગ્રામિંગમાં તાર્કિક ઓપરેટરોનો ઉપયોગ કરીને, આપણે ઇચ્છિત પરિણામો પ્રાપ્ત કરવા માટે આ શરતોને અસરકારક રીતે ભેગી કરી શકીએ છીએ.

દાખલા તરીકે, ચાલો એક એવી પરિસ્થિતિ જોઈએ જ્યાં આપણે વિદ્યાર્થીના ગુણ અને હાજરીના આધારે તેણે પરીક્ષા પાસ કરી છે કે નહીં તે નક્કી કરવાની જરૂર છે. પાસ થવા માટેના માપદંડો આ પ્રમાણે છે:

- માર્ક્સ 40 કે તેથી વધુ હોવા જોઈએ.
- હાજરી 75% કે તેથી વધુ હોવી જોઈએ.

આ કિસ્સામાં, આપણે આ બે શરતોને જોડવા અને આપણી સમસ્યાનું નિરાકરણ લાવવા માટે તાર્કિક AND (&&) ઓપરેટરનો ઉપયોગ કરી શકીએ છીએ.

તાર્કિક ઓપરેટરો પ્રોગ્રામમાં નિર્ણય લેવા માટે ખૂબ જ ઉપયોગી છે, જે આપણને વધુ જટિલ શરતોને નિયંત્રિત કરવામાં સક્ષમ બનાવે છે. તે બુલિયન પદાવલી (boolean expressions) માં પણ મદદ કરે છે, જેનું મૂલ્યાંકન સાચા (true) અથવા ખોટા (false) માં થાય છે. ધ્યાન રાખો કે C પ્રોગ્રામિંગમાં, કોઈપણ બિન-શૂન્ય (non-zero) અને બિન-નલ (non-null) મૂલ્યોને સાચા તરીકે ગણવામાં આવે છે, જ્યારે શૂન્ય (zero) ને ખોટા તરીકે ગણવામાં આવે છે. ટેબલ 8.2 તાર્કિક ઓપરેટરો અને C પ્રોગ્રામિંગમાં તેના ઉપયોગનો સારાંશ આપે છે.

ઓપરેટર	નામ	ઉપયોગ	ઉદાહરણ (ધારોકે A=1, B=0)	પરિણામ
&&	Logical AND	બંને ઓપરેન્ડ સાચાં હશે તો સાચું (true) આપશે.	A && B (1 && 0)	0
||	Logical OR	કોઈ એક ઓપરેન્ડ સાચો હશે તો સાચું (true) આપશે.	A || B (1 || 0)	1
!	Logical NOT	ઓપરેન્ડની સ્થિતિ બદલશે. (true to false and vice versa)	!A (!1)	0

ટેબલ 8.2 : તાર્કિક ઓપરેટરો

તાર્કિક AND અને તાર્કિક OR ઓપરેટરોનું મૂલ્યાંકન નીચે આપેલ ટ્રૂથ ટેબલ (Truth Table) મુજબ થાય છે, જે ટેબલ 8.3 માં દર્શાવ્યું છે. ટેબલ 8.4 માં તાર્કિક NOT ઓપરેટરનું મૂલ્યાંકન આપેલ છે.

A (true/false)	B (true/false)	A && B (Logical AND)	A || B (Logical OR)
0 (false)	0 (false)	0	0
0 (false)	1 (true)	0	1
1 (true)	0 (false)	0	1
1 (true)	1 (true)	1	1

ટેબલ 8.3 : તાર્કિક AND અને OR નું ટ્રૂથ ટેબલ

A (true/false)	!A
0	1
1	0

ટેબલ 8.4 : તાર્કિક NOT (!) નું ટ્રૂથ ટેબલ

તાર્કિક ઓપરેટરનું ઉદાહરણ

નીચેનો C પ્રોગ્રામ તાર્કિક AND ઓપરેટરનું ઉદાહરણ પ્રસ્તુત કરે છે.

```
/* C program to illustrate use of AND operator. */

#include <stdio.h>
int main() {
    int marks = 50, attendance = 80;

    if (marks >= 40 && attendance >= 75)
        printf("Student has passed the exam.");
    else
        printf("Student has not passed the exam.");
    return 0;
}
Result:
Student has passed the exam.
```

આ પ્રોગ્રામમાં બંને પદાવલીઓ (marks >= 40) અને (attendance >= 75) નું મૂલ્યાંકન સાચું (true) થાય છે, તેથી તે આઉટપુટ તરીકે "Student has passed the exam." પ્રિન્ટ કરશે.

નીચેનો C પ્રોગ્રામ તાર્કિક OR ઓપરેટરનું ઉદાહરણ પ્રસ્તુત કરે છે.

```
/* C program to illustrate use of OR operator. */
#include <stdio.h>
int main() {
    int a = -5, b = 10;
    if (a > 0 || b > 0)
        printf("At least one is positive.");
    return 0;
}
Result:
At least one is positive.
```

નીચેનો C પ્રોગ્રામ તાર્કિક NOT ઓપરેટરનું ઉદાહરણ પ્રસ્તુત કરે છે.

```
/* C program to illustrate use of NOT operator. */
#include <stdio.h>
int main() {
    int isAdmin = 0;
    if (!isAdmin)
        printf("User is not an administrator.");
    else
       printf("User is an administrator.");
    return 0;
}
Result:
User is not an administrator.
```

અસાઇનમેન્ટ ઓપરેટરો (Assignment Operators)

C પ્રોગ્રામિંગમાં અસાઇનમેન્ટ ઓપરેટરોનો ઉપયોગ ચલને મૂલ્ય આપવા માટે થાય છે. સૌથી સામાન્ય અસાઇનમેન્ટ ઓપરેટર છે = (સમાન ચિહ્ન).

મૂળ (Basic) અસાઇનમેન્ટ ઓપરેટર (=)

અસાઇનમેન્ટ ઓપરેટર = નો ઉપયોગ ડાબી બાજુના ચલને જમણી બાજુનું મૂલ્ય આપવા માટે થાય છે. ઉદાહરણ:

```
int a;
a = 10;        /* This statement assigns 10 to variable a */
```

શૉર્ટ-હેન્ડ (Short-hand) અસાઇનમેન્ટ ઓપરેટરો

શૉર્ટ-હેન્ડ અસાઇનમેન્ટ ઑપરેટર એ ગણિતીય ઓપરેટરને અસાઇનમેન્ટ ઓપરેટર સાથે જોડે છે. આનો ઉપયોગ કોડને વધુ સંક્ષિપ્ત બનાવવા માટે થાય છે. આ પ્રકારના ઓપરેટર સામાન્ય રીતે લૂપ અને ગણિતીય ગણતરીઓમાં વપરાય છે.

નીચે આપેલ વિધાન જુઓ, જેમાં += શૉર્ટ-હેન્ડ ઓપરેટરનો ઉપયોગ થયો છે, જે + ઓપરેટરને = ઓપરેટર સાથે જોડે છે:

```
a += 5;
```

તે નીચેના બરાબર છે:

```
a = a + 5;
```

+= ઓપરેટર જમણી બાજુના ઓપરેન્ડનું મૂલ્ય ડાબી બાજુના ઓપરેન્ડમાં ઉમેરે છે અને પરિણામ ફરીથી ડાબી બાજુના ઓપરેન્ડને આપે છે. ટેબલ 8.5માં વિવિધ શૉર્ટ-હેન્ડ અસાઇનમેન્ટ ઓપરેટરો, તેમનો ઉપયોગ અને ઉદાહરણો દર્શાવવામાં આવ્યા છે.

ઓપરેટર	અર્થ	ઉદાહરણ	ના બરાબર
+=	Add and assign	a += 5;	a = a + 5;
-=	Subtract and assign	a -= 3;	a = a - 3;
*=	Multiply and assign	a *= 2;	a = a * 2;
/=	Divide and assign	a /= 2;	a = a / 2;
%=	Modulus and assign	a %= 3;	a = a % 3;

ટેબલ 8.5 : શૉર્ટ-હેન્ડ અસાઇનમેન્ટ ઓપરેટરો

શૉર્ટ-હેન્ડ અસાઇનમેન્ટ ઓપરેટરનું ઉદાહરણ

```
/*Program to illustrate use of Short-hand assignment operators */
#include <stdio.h>
int main() {
    int a = 10;
    a += 5;     /* a = 10 + 5 = 15  */
    a *= 2;     /* a = 15 * 2 = 30  */
    printf("Final value of a: %d", a);
    return 0;
}
Result:
Final value of a: 30
```

બિટવાઇઝ ઓપરેટરો (Bitwise Operators)

C પ્રોગ્રામિંગમાં બિટવાઇઝ ઓપરેટરોનો ઉપયોગ પૂર્ણાંક પર બિટ સ્તરે (bit level) કામગીરી કરવા માટે થાય છે. આ ઓપરેટરો પૂર્ણાંકને બિટ્સની શ્રેણી (0 અને 1) તરીકે ગણે છે અને દરેક બિટ પર અલગથી ક્રિયા કરે છે. આ રીત કેટલીક કામગીરી માટે ખૂબ જ કાર્યક્ષમ સાબિત થાય છે, ખાસ કરીને જ્યારે ફ્લેગ્સ (flags) સાથે કામ કરવું હોય, હાર્ડવેર રજિસ્ટર (hardware registers) સંચાલિત કરવા હોય અથવા અલ્ગોરિધમ્સને કાર્યક્ષમ કરવા હોય. ટેબલ 8.6માં બિટવાઇઝ ઓપરેટરો અને તેમનો ઉપયોગ દર્શાવવામાં આવ્યો છે.

ઓપરેટર	ઉપયોગ
&	બિટવાઇઝ AND
|	બિટવાઇઝ OR
^	બિટવાઇઝ Exclusive OR
~	બિટવાઇઝ NOT (આ એક યુનરી ઓપરેટર છે)
<<	ડાબે ખસેડવું (Left shift - as per bits specified)
>>	જમણે ખસેડવું (Right shift - as per bits specified)

ટેબલ 8.6 : બિટવાઇઝ ઓપરેટરો

બિટવાઇઝ AND, બિટવાઇઝ OR, બિટવાઇઝ Exclusive OR, ડાબે ખસેડવું (Left Shift) અને જમણે ખસેડવું (Right Shift) બાઈનરી ઓપરેટરો છે, કારણ કે તેઓની ક્રિયા માટે બે ઓપરેન્ડની જરૂર હોય છે. બિટવાઇઝ NOT ઓપરેટર એક યુનરી (unary) ઓપરેટર છે, કારણ કે તેની ક્રિયા માટે એક જ ઓપરેન્ડ પૂરતો છે. આ કોર્સમાં બિટવાઇઝ ઓપરેટરો વિશે વધુ વિગતો આવરી લેવાઈ નથી. તમે તેમને તમારા ઉચ્ચસ્તરીય અભ્યાસમાં વધુ વિગતવાર શીખશો.

વધારા (Increment) અને ઘટાડા (Decrement) સૂચક ઓપરેટરો

C પ્રોગ્રામિંગમાં વધારા-સૂચક (Increment) ઓપરેટરનો ઉપયોગ ચલનું મૂલ્ય એકથી વધારવા માટે થાય છે અને ઘટાડા-સૂચક (Decrement) ઓપરેટરનો ઉપયોગ ચલનું મૂલ્ય એકથી ઘટાડવા માટે થાય છે. આ ઓપરેટરો લૂપોને નિયંત્રિત કરવા, કાઉન્ટર ચલોને મેનેજ કરવા, અને પુનરાવર્તન કામ સંબંધી કોડને સરળ બનાવવા માટે ઉપયોગી છે.

વધારા-સૂચક (++) અને ઘટાડા-સૂચક (--) ઓપરેટરોને યુનરી (unary) ઓપરેટર કહેવામાં આવે છે, કારણ કે તેઓ એક જ ઓપરેન્ડનું મૂલ્ય બદલતા હોય છે.

- **વધારા-સૂચક ઓપરેટર (++) :** આ ઓપરેટર ઓપરેન્ડનું મૂલ્ય એકથી વધારવાનું કામ કરે છે. તેને ચલની આગળ (prefix જેમ કે, ++x) અને ચલની પાછળ (postfix જેમ કે, x++) બંને રીતે વાપરી શકાય છે.
- **ઘટાડા-સૂચક ઓપરેટર (--) :** આ ઓપરેટર ઓપરેન્ડનું મૂલ્ય એકથી ઘટાડે છે. વધારા-સૂચક ઓપરેટરની જેમ જ, તેને ચલની આગળ (prefix જેમ કે, --x) અને ચલની પાછળ (postfix જેમ કે, x--) બંને રીતે વાપરી શકાય છે.

નોંધ : prefix વધારા-સૂચક ઓપરેટર પહેલા ચલની કિંમતમાં 1નો વધારો કરે છે અને પછી ઉપયોગમાં લે છે જ્યારે postfix વધારા-સૂચક ઓપરેટર પહેલા ઉપયોગમાં લે છે અને પછી ચલની કિંમતમાં 1નો વધારો કરે છે. આજ નિયમ ઘટાડા-સૂચક ઓપરેટરને લાગુ પડે છે.

વધારા અને ઘટાડા સૂચક ઓપરેટરોનું ઉદાહરણ

```
/* Program to demonstrate use of the Increment and Decrement
operators.*/
#include <stdio.h>
int main() {
    int a = 5;
    int b = 10;
    /* Using ++ operator */

    a++;    /* same as: a = a + 1  */
    printf("Value of a after increment: %d\n", a);

    /* Using -- operator */

    b--;    /* same as: b = b - 1  */
    printf("Value of b after decrement: %d", b);

    return 0;
}

Result:

Value of a after increment: 6
Value of b after decrement: 9
```

ટર્નરી ઓપરેટર (Ternary Operator)

ટર્નરી ઓપરેટર C પ્રોગ્રામિંગમાં એક અનોખો ઓપરેટર છે, જે ત્રણ ઓપરેન્ડ ધરાવે છે. તે શરતી (conditional) ક્રિયાઓ કરવા માટેની એક સંક્ષિપ્ત રીત છે, જેનો ઉપયોગ ઘણીવાર *if-else* વિધાનના ટુંકારૂપ તરીકે થાય છે.

ટર્નરી ઓપરેટરની વાક્યરચના

```
condition ? expression1 : expression2;
```

ટર્નરી ઓપરેટર સૌપ્રથમ આપેલી શરતનું (condition) મૂલ્યાંકન કરે છે; જો શરત સાચી (true) હોય, તો પછી expression1 નો અમલ કરે છે, નહીં તો expression2 નો અમલ કરે છે. આ ઓપરેટર શરતના મૂલ્યાંકનના આધારે expression1 અથવા expression2 માંથી કોઈ એકનું પરિણામ પાછું આપે છે. તેનો ઉપયોગ સામાન્ય રીતે એક જ લીટીમાં સરળ શરતી અસાઇનમેન્ટ (conditional assignments) અથવા નિર્ણયો માટે થાય છે, જે કોડને વધુ સંક્ષિપ્ત અને વાંચન માટે સરળ બનાવે છે.

ટર્નરી ઓપરેટરનું ઉદાહરણ

```
/* Program to illustrate use of ternary operator */
#include <stdio.h>
int main() {
    int a = 10;
    int b = 5;
    int max;
```

```
        max = (a > b) ? a : b;
        printf("The maximum of %d and %d is: %d.", a, b, max);
        return 0;
    }
    Result:
    The maximum of 10 and 5 is: 10.
```

આ ઉદાહરણમાં, ટર્નરી ઓપરેટર તપાસે છે કે શું a એ b કરતાં મોટો છે. આ શરત સાચી હોવાથી, max ને a ની કિંમત અસાઈન કરવામાં આવે છે.

યુનરી, બાઈનરી અને ટર્નરી ઓપરેટરો (Unary, Binary and Ternary Operators)

ઓપરેટરને તે જેટલા ઓપરેન્ડ પર પ્રક્રિયા કરે છે તેના આધારે યુનરી, બાઈનરી અને ટર્નરી તરીકે વર્ગીકૃત કરી શકાય છે.

- **યુનરી ઓપરેટર (Unary Operator) :** C ભાષામાં, યુનરી ઓપરેટર એવા છે જે એક જ ઓપરેન્ડ પર કાર્ય કરે છે. તેનો ઉપયોગ સામાન્ય રીતે મૂલ્ય વધારવા (incrementing), ઘટાડવા (decrementing) અથવા નકારાત્મક બનાવવા (negating) જેવી મૂળભૂત કામગીરી માટે થાય છે. વધારા-સૂચક (++), ઘટાડા-સૂચક (--) અને NOT (!) એ યુનરી ઓપરેટરો છે.
- **બાઈનરી ઓપરેટર (Binary Operator) :** બાઈનરી ઓપરેટરને બે ઓપરેન્ડની જરૂર હોય છે. બાઈનરી ઓપરેટરોના ઉદાહરણોમાં ગણિતીય, સંબધસૂચક અને તાર્કિક ઓપરેટરોનો સમાવેશ થાય છે.
- **ટર્નરી ઓપરેટર (Ternary Operator) :** ટર્નરી ઓપરેટર એ C માં એક વિશિષ્ટ ઓપરેટર છે જે ત્રણ ઓપરેન્ડ ધરાવે છે. તે શરતી કામગીરીને અમલમાં મૂકવાની એક સંક્ષિપ્ત રીત પૂરી પાડે છે, જે ઘણીવાર *if-else* સ્ટેટમેન્ટ માટેના ટુંકારૂપ તરીકે કામ કરે છે.

વિશેષ ઓપરેટરો (Special Operators)

ગણિતીય, સંબંધસૂચક, તાર્કિક અને બિટવાઈઝ જેવા સામાન્ય ઓપરેટરો ઉપરાંત, C પ્રોગ્રામિંગમાં કેટલાક વિશેષ ઓપરેટરો (special operators) નો સમાવેશ થાય છે જે અનન્ય કાર્યો કરે છે. જે ઘણીવાર મેમરી મેનેજમેન્ટ, પોઈન્ટર અથવા પદાવલી મૂલ્યાંકન સાથે સંબંધિત હોય છે. ચાલો આપણે આમાંના કેટલાક વિશેષ ઓપરેટરો વિશે જાણીએ. ટેબલ 8.7 વિશેષ ઓપરેટરો અને તેના ઉપયોગોની યાદી આપે છે.

ઓપરેટર	નામ	વર્ણન	સાદું ઉદાહરણ
sizeof	Sizeof operator	ડેટા ટાઈપ અથવા ચલની સાઈઝ આપે છે (બાઇટમાં)	sizeof(int) 4 આપે છે
&	Address-of operator	ચલનું મેમરી એડ્રેસ પાછું આપે છે	&a (ચલ a નું મેમરી એડ્રેસ આપે છે)
*	Value-at operator (Pointer dereference)	પોઈન્ટર દ્વારા નિર્દેશિત મેમરી સ્થાન પર સંગ્રહિત મૂલ્યને ઍક્સેસ કરે છે	*ptr (પોઈન્ટર ptr દ્વારા નિર્દેશિત એડ્રેસ પરની કિંમત આપે છે)
. (dot)	Structure member access	સ્ટ્રક્ચર પ્રકારના ચલનો ઉપયોગ કરીને સ્ટ્રક્ચરના સભ્યને ઍક્સેસ કરે છે.	s.age (સ્ટ્રક્ચર s ના age સભ્યની કિંમત)
->	Structure pointer member access	સ્ટ્રક્ચર તરફ નિર્દેશ કરતા પોઈન્ટરનો ઉપયોગ કરીને સ્ટ્રક્ચરના સભ્યને ઍક્સેસ કરે છે.	ptr->age (જો ptr સ્ટ્રક્ચરનું પોઈન્ટર હોય)

ટેબલ 8.7 : વિશેષ ઓપરેટરો

sizeof ઓપરેટર (sizeof Operator)

sizeof ઓપરેટર એ લોકપ્રિય યુનરી ઓપરેટરમાંનો એક છે જે આપેલા ચલની સાઈઝ બાઇટમાં આપે છે. તેનો મુખ્યત્વે ઉપયોગ ડાયનેમિક મેમરી એલોકેશન (dynamic memory allocation) દરમિયાન અને વિવિધ સિસ્ટમ પર પોર્ટેબિલિટી (portability) સુનિશ્ચિત કરવા માટે થાય છે, જ્યાં ડેટા ટાઈપની સાઈઝ અલગ-અલગ હોઈ શકે.

sizeof ઓપરેટરનું ઉદાહરણ

```
/* Program to illustrate use of sizeof operator */

#include <stdio.h>
int main() {
    int num;
    char ch;

    /* To print size of num variable */
    printf("Size of num (int): %d bytes\n", sizeof(num));

    /* To print size of ch variable */
    printf("Size of ch (char): %d byte\n", sizeof(ch));

    return 0;
}

Result:
Size of num (int): 4 bytes
Size of ch (char): 1 byte
```

આ પ્રોગ્રામ num અને ch ચલો દ્વારા મેમરીમાં રોકાયેલ સાઈઝ પ્રિન્ટ કરશે. તમે તમારા આગામી ધોરણોમાં વિશેષ ઓપરેટરો વિશે વધુ ઊંડાણપૂર્વક અભ્યાસ કરશો.

નોંધ : પૂર્ણાંક ચલની ચોક્કસ સાઈઝ સિસ્ટમ આર્કિટેક્ચર અને કમ્પાઈલરના આધારે બદલાઈ શકે છે. ઉદાહરણ તરીકે, 64-બીટ સિસ્ટમ પર *int* ની સાઈઝ 4 બાઈટ અને *char* ની સાઈઝ 1 બાઈટ હોય છે.

પદાવલી (expression) શું છે?

પ્રોગ્રામિંગમાં, પદાવલી યુઝરને ગણતરીઓ કરવા, નિર્ણયો લેવા અને પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવા માટે જરૂરી સવલત પૂરી પાડે છે.

પદાવલી એ ચલ, અચલ અને ઓપરેટરોનું એક સંયોજન છે જે કોઈ મૂલ્ય ઉત્પન્ન કરે છે.

પદાવલી એક જ ચલ અથવા અચલની બનેલી હોઈ શકે છે, અથવા બહુવિધ ચલો, અચલો અને ઓપરેટરોના સંયોજનથી પણ બનેલી હોઈ શકે છે.

ઉદાહરણ

```
int a = 10, b = 5;
int result = a + b;     /* a + b is an expression */
```

આ ઉદાહરણમાં, a+b એ એક પદાવલી, a અને b ચલો અને + એ ઓપરેટર છે. આ પદાવલી a અને b માં સંગ્રહિત મૂલ્યોનો સરવાળો કરી તેને result ચલમાં સંગ્રહ કરે છે.

તે જ રીતે, પદાવલીઓ નિર્ણયો લેવા માટે તાર્કિક ઓપરેટરોનો સમાવેશ કરી શકે છે, જેમ કે a > b માં જો a એ b કરતા મોટો હોય તો 'સાચું' (true) તરીકે મૂલ્યાંકન કરે છે.

પદાવલી માત્ર ગણિતીય ક્રિયાઓ પૂરતી મર્યાદિત નથી, તેમાં સંબંધસૂચક, તાર્કિક અને બિટવાઈઝ ક્રિયાઓ પણ સામેલ હોઈ શકે છે. ઉદાહરણ તરીકે, (a > b) && (b < c) જેવી પદાવલી મૂલ્યોની તુલના કરવા માટે સંબંધસૂચક ઓપરેટરનો અને પરિણામોને જોડવા માટે તાર્કિક ઓપરેટરનો ઉપયોગ કરે છે, જે આખરે બુલિયન (boolean) મૂલ્ય આપે છે.

પદાવલી પ્રોગ્રામરને ગણતરીઓ કરવા, નિર્ણયો લેવા અને પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવા સક્ષમ બનાવે છે, જેનાથી સોફ્ટવેર ડેવલપમેન્ટનો પાયો નંખાય છે.

પદાવલીઓમાં ઓપરેટરનો ઉપયોગ

ટેબલ 8.8 પદાવલીઓમાં સામાન્યપણે વપરાતા ઓપરેટરનો ઉપયોગ દર્શાવે છે.

ઓપરેટરનો પ્રકાર	પદાવલીનું ઉદાહરણ	વર્ણન
Arithmatic	a + b	બે સંખ્યાનો સરવાળો
Relational	a > b	ચકાસો કે a એ b કરતા મોટો છે
Logical	a > 0 && b > 0	ચકાસો કે a એ b બંને ધન છે
Assignment	c = a + b	a + b નું પરિણામ ચલ c ને આપે છે
Increment/Decrement	a++, --b	મૂલ્યને 1 થી વધારે કે ઘટાડે
Bitwise	a & b	a અને b પર બીટવાઈઝ AND કરવું

ટેબલ 8.8 : પદાવલીઓમાં સામાન્યપણે ઉપયોગ થતાં ઓપરેટર

પદાવલીઓમાં ઓપરેટરનો ઉપયોગ C પ્રોગ્રામને નિર્ણયો લેવા, ગણતરીઓ કરવા અને પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવાની સગવડતા આપે છે. નિર્ણય સંરચનાઓ (Decision Structures) અને નિયંત્રણ સંરચનાઓ (Control Structures) સંબંધિત વધારે આગામી પ્રકરણમાં સમજીશું.

સારાંશ

આ પ્રકરણમાં, આપણે C પ્રોગ્રામિંગમાં ઓપરેટરો અને પદાવલીઓના મૂળભૂત ખ્યાલો વિશે જાણકારી મેળવી છે. તેઓ પ્રોગ્રામમાં થતી તમામ ગણતરીઓ અને તાર્કિક ક્રિયાઓ માટેના આધારસ્તંભ છે. આપણે ઓપરેટરોના વિવિધ પ્રકારો વિશે જાણકારી મેળવી છે, જેમાં ગાણિતિક ગણતરીઓ માટે ગણિતીય, તુલનાઓ માટે સંબંધસૂચક, શરતોને જોડવા માટે તાર્કિક, નીચલા-સ્તરના ડેટા હેરફેર માટે બિટવાઈઝ (bitwise) અને અનન્ય વધારો/ઘટાડો તથા sizeof અને & જેવા વિશેષ ઓપરેટરોનો સમાવેશ થાય છે. ઓપરેટર પ્રાથમિકતા અને સંબંધતાને સમજવું એ પદાવલીનું સચોટ અર્થઘટન કરવા અને તેનું નિર્માણ કરવા માટે મહત્વપૂર્ણ છે. પદાવલીમાં ઓપરેટર અને ઓપરેન્ડનું સંયોજન હોય છે જે આખરે એક જ મૂલ્યમાં મૂલ્યાંકન કરે છે. આ ખ્યાલોમાં નિપુણતા મેળવવી ખૂબ જ મહત્ત્વપૂર્ણ છે, કારણ કે તે પ્રોગ્રામરોને કાર્યક્ષમ, સચોટ અને મજબૂત C પ્રોગ્રામ બનાવવા માટે સક્ષમ બનાવે છે.

સ્વાધ્યાય

1. પ્રોગ્રામિંગમાં પદાવલીનો અર્થ શું છે?
2. C પ્રોગ્રામિંગ ભાષામાં ઓપરેટરોનો ઉપયોગ જણાવો.
3. C પ્રોગ્રામિંગના તાર્કિક ઓપરેટરોની યાદી બનાવો.
4. કયો ઓપરેટર ચલની કિંમતમાં 1 નો વધારો કરવા માટે ઉપયોગમાં લેવાય છે?

5. (=) અને (==) વચ્ચે શું ફરક છે?

6. અસાઇનમેન્ટ ઓપરેટરનો ઉપયોગ ઉદાહરણ સાથે સમજાવો.

7. એડ્રેસ-ઓફ ઓપરેટર (&) નો ઉપયોગ શું છે?

8. તાર્કિક AND ઓપરેટર (&&) નો ઉપયોગ યોગ્ય ઉદાહરણ સાથે સમજાવો.

9. કયા ઓપરેટરને વધુ પ્રાથમિકતા છે - Division (/) કે Addition (+) ?

10. C પ્રોગ્રામિંગમાં 5 + 2 * 3 + 5 પદાવલીનો આઉટપુટ શું આવશે?

11. **સાચું કે ખોટું જણાવો.**

(1) બાઇનરી ઓપરેટરને બે ઓપરેન્ડની જરૂર હોય છે.

(2) વધારા-સૂચક ઓપરેટર એ યુનરી ઓપરેટરનું ઉદાહરણ છે.

(3) ગુણાકાર (*) ઓપરેટરની પ્રાથમિકતા (precedence) સરવાળા (+) ઓપરેટર કરતાં વધારે છે.

(4) && ને C પ્રોગ્રામિંગમાં મોડ્યુલસ (modulus) ઓપરેટર તરીકે ઓળખવામાં આવે છે.

(5) || પ્રતીકનો ઉપયોગ તાર્કિક OR ઓપરેટરને રજૂ કરવા માટે થાય છે.

12. **ખાલી જગ્યાઓ પૂરો.**

(1) ટર્નરી ઓપરેટર એ C માં એક અનોખો ઓપરેટર છે જે _____ ઓપરેન્ડ લે છે.

(2) મોડ્યુલસ (Modulus) ઓપરેટરનો ઉપયોગ માત્ર _____ પ્રકારના ઓપરેન્ડ સાથે જ થઈ શકે છે.

(3) _____ ઓપરેટર એક લોકપ્રિય યુનરી ઓપરેટર છે જે આપેલા ચલની બાઇટમાં સાઈઝ પરત કરે છે.

(4) _____ પ્રતીકનો ઉપયોગ લોજિકલ NOT ઓપરેટરને રજૂ કરવા માટે થાય છે.

(5) ઈંક્રીમેન્ટ (++), ડિક્રીમેન્ટ (--) અને NOT (!) એ _____ ઓપરેટરો (યુનરી / બાઇનરી / ટર્નરી) છે.

13. **બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.**

(1) નીચેનામાંથી કયો C માં ગણિતીય ઓપરેટર છે?

(a) && (b) || (c) + (d) !

(2) પદાવલી num1 == num2 શું ચકાસે છે?

(a) num2, num1 ને અસાઈન કરે છે (b) num1 અને num2 ની તુલના કરે છે

(c) num1 ને 1 થી વધારે છે (d) num1 અને num2 નો ગુણાકાર કરે છે

(3) નીચેના કોડનો અમલ કર્યા બાદ number નું મૂલ્ય શું હશે?

```
int number = 5;
number *= 5;
```

(a) 10 (b) 15 (c) 20 (d) 25

(4) નીચનામાંથી કયો C માં તાર્કિક AND ઓપરેટર છે?

(a) & (b) && (c) | (d) ||

(5) નીચેના C પ્રોગ્રામનો આઉટપુટ શું હશે?

```
#include <stdio.h>
int main() {
     int a = 5, b = 2;
     printf("%d", a % b);
     return 0;
}
```

(a) 1 (b) 2 (c) 5 (d) 10

(6) a + b * c પદાવલીના અમલ દરમ્યાન પહેલા કયું ઓપરેશન થશે?

(a) a + b (b) b * c (c) (a+b) અને (b *c) (d) a * c

(7) નીચેનામાંથી કયો ઓપરેટર મૂલ્ય 1 થી ઘટાડવા માટે વપરાશે?

(a) – – (b) ++ (c) – (d) *

(8) નીચેના C પ્રોગ્રામનો આઉટપુટ શું આવશે?

```
#include <stdio.h>
int main() {
    printf("%d", !(1));
    return 0;
}
```

(a) 0 (b) 1 (c) One (d) Not output

(9) C પ્રોગ્રામિંગમાં મોડ્યુલસ (%) ઓપરેટરનો પ્રાથમિક હેતુ શો છે?

(a) શેષ શોધવા (b) ગુણાકાર કરવા

(c) તાર્કિક AND મૂલ્ય શોધવા (d) ભાગફળ શોધવા

(10) NUM += 5 પદાવલી પ્રોગ્રામમાં શું કરશે?

(a) NUM ને 5 અસાઈન કરશે (b) NUM માં 5 ઉમેરશે

(c) NUM ને 5 ગુણશે (d) NUM માં કોઈ ફેરફાર નહી કરે

પ્રાયોગિક સ્વાધ્યાય

1. એવો C પ્રોગ્રામ લખો જે યુઝરને યોગ્ય ઈનપુટ ફંક્શનનો ઉપયોગ કરીને બે પૂર્ણાંક મૂલ્યો દાખલ કરવા કહે. નીચે જણાવેલ કાર્યોનું પરિણામ દર્શાવો
   - બે સંખ્યાઓનો સરવાળો
   - તફાવત (પ્રથમ સંખ્યા – બીજી સંખ્યા)
   - બંને સંખ્યાઓનો ગુણાકાર
   - પ્રથમ સંખ્યાને બીજી સંખ્યાથી ભાગાકાર કરતાં મળતું ભાગફળ - (integer division નું પરિણામ)
   - પ્રથમ સંખ્યાને બીજી સંખ્યાથી ભાગાકાર કરતાં મળતી શેષ - (division પછીનો remainder અથવા modulus)
2. એવો C પ્રોગ્રામ લખો જે વધારા-સૂચક (Increment) ઓપરેટરના ઉપયોગનું પ્રદર્શન કરે. યુઝર પાસેથી NUM નામના ચલનું મૂલ્ય વાંચો. પ્રોગ્રામમાં ++NUM અને NUM++ વચ્ચેનો તફાવત તેનું મૂલ્ય પ્રિન્ટ કરીને બતાવો.

3. એવો C પ્રોગ્રામ લખો જે સંબંધસૂચક ઓપરેટરના ઉપયોગનું પ્રદર્શન કરે. તમારા પ્રોગ્રામમાં ==, !=, <, >, <=, >= ઓપરેટરો નો ઉપયોગ કરો. બે સંખ્યાઓ વાંચો, તેમની તુલના કરો અને પરિણામ મુજબ યોગ્ય સંદેશો પ્રિન્ટ કરો.

4. એવો C પ્રોગ્રામ લખો જે ટર્નરી ઓપરેટરનો ઉપયોગ કરીને બે સંખ્યાઓમાંથી મહત્તમ સંખ્યા શોધે. **સૂચન :** બે સંખ્યાઓની તુલના કરવા અને કઈ મોટી છે તે નક્કી કરવા માટે ટર્નરી ઓપરેટર (?:) નો ઉપયોગ કરો.

5. નીચેની પદાવલીનું મૂલ્યાંકન કરવા માટેનો C પ્રોગ્રામ લખો.

```
a + b * c / d - e
```

જરૂરી કિંમતો યુઝર પાસેથી વાંચો.

6. નીચેનો C પ્રોગ્રામ ગણિતીય ઓપરેટરોનો ઉપયોગ દર્શાવે છે. પરંતુ, તેમાં વાક્યરચનાને લગતી તેમજ તાર્કિક ભૂલો છે. આ બધી ભૂલો ઓળખી તેને દૂર કરી પ્રોગ્રામને ફરીથી લખો જેથી એ સાચો આઉટપુટ પ્રિન્ટ કરે.

```
/* Program to illustrate use of Arithmatic Operators */
#include <stdio.h>
int main() {
    int a=20, b=5;

    printf("Result of Arithmatic Operations on a and b:\n");

    printf("Addition       : a * b = %d \n", a + b);
    printf("Subtraction    : a - b = %d \n , a - b)
    print("Multiplication  : a + b = %d \n , a * b);
    print("Division        : a / b = %d \n , a / b)
    printf("Modulus        : a % b = %d \n", a % b);

    return 0;
}
```