**ddddddddddddddddddddddddddddddddddddddd**

Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Roll.No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Semester \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Operators in C Language

|  |  |  |
| --- | --- | --- |
| **Activities** | **Remarks** | **Signature** |
|  |  |  |

**Operators in C language**

**LAB-3**

**Objectives:**

Familiarization with operators.

To understand the concept of using operators in C Language.

To understand the concept of performing actions using operators To manipulate data.

Utilization of input and output function

**Operator**

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C language is rich in built-in operators and provides the following types of operators:

1. Arithmetic Operators
2. Arithmetic Assignment Operators 3- Increment/Decrement operators
3. Relational Operators
4. Logical Operators
5. Bitwise Operators

**1- Arithmetic Operators**

Following table shows all the arithmetic operators supported by C language.

**Assume variable A holds 10 and variable B holds 20 then:**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two operands | A + B will give 30 |
| - | Subtracts second operand from the first | A - B will give -10 |
| \* | Multiplies both operands | A \* B will give 200 |
| / | Divides numerator by de-numerator | B / A will give 2 |
| % | Modulus Operator (Remainder Operator) and remainder of after an integer division | B % A will give 0 |

**Example 1:**

|  |
| --- |
| #include<stdio.h> #include<conio.h>  void main(void)  {  int a =21;  int b =10; int c ;  c = a + b;  printf("Line 1 - Value of c is %d\n", c ); c = a - b;  printf("Line 2 - Value of c is %d\n", c ); c = a \* b;  printf("Line 3 - Value of c is %d\n", c ); c = a / b;  printf("Line 4 - Value of c is %d\n", c ); c = a % b;  printf("Line 5 - Value of c is %d\n", c ); getch();  } |

**Output**

|  |
| --- |
| Line1-Value of c is31 Line2-Value of c is11 Line3-Value of c is210 Line4-Value of c is2  Line5-Value of c is1 |

**Do it yourself - 1:**

f and g are floating point variables whose values are 18.5 and 2.0 respectively. Compute the following arithmetic expressions and note output:

1. f+g
2. f-g
3. f\*g
4. f/g

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Do it yourself - 2:**  char a = ‘F’; char b = ‘S’;  What will the output of:   1. a 2. a + b   3- a + b + 9  4- a + b + ‘9’  And why?  **2- Arithmetic Assignment Operators**  Following table shows all the arithmetic operators supported by C language.  **Assume variable A holds 10 and variable B holds 20 then:** | | | | |
|  | **Operator** | **Description** | **Example** |  |
| += | Adds two operands | A += B will give 30 and result stored in A |
| -= | Subtracts second operand from the first | A -= B will give -10 and result stored in A |
| \*= | Multiplies both operands | A \*= B will give 200 and result stored in A |
| /= | Divides numerator by de-numerator | B / =A will give 2 and result stored in B |
| %= | Modulus Operator (Remainder Operator) and remainder of after an integer division | B % =A will give 0 and result stored in B |
| = | Assignment operator that move value |  |
|  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  | from right to left. |  |

**Example 2:**

|  |
| --- |
| #include<stdio.h> #include<conio.h>  void main(void)  {  int a =21;  int b =10;  a += b;  printf("Line 1 - Value of c is %d\n", a); a -= b;  printf("Line 2 - Value of c is %d\n", a); a \*= b;  printf("Line 3 - Value of c is %d\n", a); a /= b;  printf("Line 4 - Value of c is %d\n", a); a %= b;  printf("Line 5 - Value of c is %d\n", a); getch();  } |

**Do it yourself - 3:**

void main(void)

{

int total = 0; int count = 10;

printf(“ Total = %d \n ” , total); total += count;

printf(“ Total = %d \n ” , total); total += count;

printf(“ Total = %d \n ” , total);

}

**Output:**

* 1. **Increment/Decrement operators**

The operator that is used to increment 1 in the value OR decrement 1 in the value. It is a unary operator.

**Assume variable A holds 10 and variable B holds 20 then:**

|  |  |  |
| --- | --- | --- |
| ++ | Increments operator increases integer value by one | A++ will give 11 |
| -- | Decrements operator decreases integer value by one | A-- will give 9 |

**Example 3:**

#include<stdio.h>

**Output**

|  |
| --- |
| Line6-Value of c is21  Line7-Value of c is22 |

**Task: Discuss the output of the above program.**

**Do it yourself - 4:**

void main(void)

{

int num=0;

printf(“Number = %d \n” , num); printf(“Number = %d \n” , num++); printf(“Number = %d \n” , num);

}

**Output:**

**+**

**Do it yourself - 5:**

void main(void)

{

int num=0;

printf(“Number = %d \n” , num); printf(“Number = %d \n” , ++num); printf(“Number = %d \n” , num);

}

**Output:**

* 1. **Relational Operators**

Relational Operators are binary operators, which are used to compare two numeric operands. The operands can be variables, constants or expressions that ultimately get evaluated to a numeric value. Since characters are represented as integers using the ASCII code, therefore operands to be compared can be two characters as well. Following table shows all the relational operators supported by C language.

**Assume variable A holds 10 and variable B holds 20, then:**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | Checks if the values of two operands are equal or not, if yes then condition becomes true. | (A == B) is not true. |
| != | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is  true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. |

|  |  |  |
| --- | --- | --- |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is  true. |

**Example 4:**

#include<stdio.h> #include<conio.h> void main(void)

{

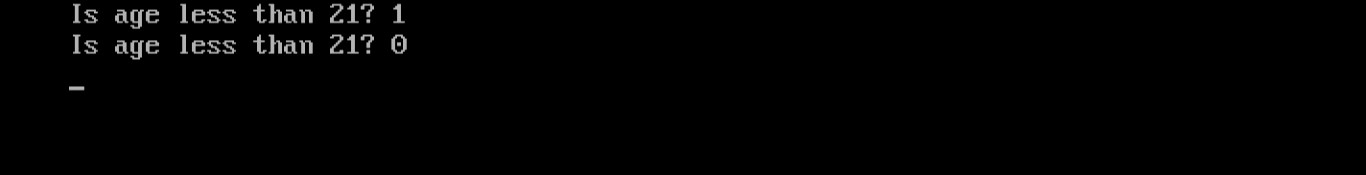
int age; age=15; clrscr();

printf("Is age less than 21? %d \n",age<21); age=30;

printf("Is age less than 21? %d \n", age<21); getch();

}

**Output:**



**Do it yourself - 6:**

#include<stdio.h> #include<conio.h> void main(void)

{ int speed; speed=75; clrscr();

printf("Is speed equal to 55? %d \n", speed==55); speed=55;

printf(“Is speed equal to 55? %d \n", speed==55); getch(); }

**Output:**

**Do it yourself – 7:**

What will this statement print printf(“ Answer is %d” , 2+1 < 4); ?

**Do it yourself – 8:**

What will this statement print printf(“ Answer is %d” , 1<2 + 4); ?

**Example – 5:**

Suppose **i** is an integer variable with value 7 and **f** is a floating point variable with value 5.5 and c is a character variable that represents the character ‘w’.

Then:

|  |  |  |
| --- | --- | --- |
| **Expression** | **Interpretation** | **Value** |
| **f>5** | true | 1 |
| **(i + f) < = 10** | false | 0 |
| **c = = 119** | true | 1 |
| **c ! = ‘p’** | true | 1 |
| **c > = 10 \* ( i \* f )** | false | 0 |

* 1. **Logical Operators**

A logical operator combines the result of one or more expressions and the resultant expression is called the logical expression. After testing the conditions, they return either TRUE (1) or FALSE (0) as net result. Following table shows all the logical operators supported by C language.

**Assume variable A holds 1 and variable B holds 0, then:**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && | Called Logical AND operator. If both the operands are non-zero, then condition becomes true. | (A && B) is false. |
| || | Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true | (A || B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false. | !(A && B) is true. |

**Example - 6:**

Assume f=5.5 , i=7, c = ‘w’

|  |  |  |
| --- | --- | --- |
| **Expression** | **Interpretation** | **Value** |
| **f > 5** | true | 1 |
| **! ( f > 5 )** | false | 0 |
| **( i >= 6 ) && (c == ‘w’)** | true | 1 |
| **(i >= 6) || (c = = 119)** | true | 1 |
| **(f < 11) && ( i > 100)** | false | 0 |
| **(c ! = ‘p’ ) || ( (i+f) <= 10)** | true | 1 |

**Do it yourself - 9:**

Assume:

i=0, j=1, k= -1 f=2.5, g=0.0

|  |
| --- |
| **Expression Interpretation Value** |
| **i && j** |
| **i< j && k<j** |
| **j+k || !i** |
| **f\*5 && 5 || j/k** |
| **i<= 10 && f>= 1 && j** |
| **!f || !k || j +k** |
| **f\*g < i + j ||k** |

* 1. **Bitwise Operators**

Bitwise operator works on bits and perform bit-by-bit operation.

**Assume if A = 60; and B = 13; now in binary format they will be as follows: A = 0011 1100**

**B = 0000 1101**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & | Binary AND Operator copies a bit to the result if it exists in both operands. | (A & B) will give 12, which is  0000 1100 |
| | | Binary OR Operator copies a bit if it exists in either operand. | (A | B) will give 61, which is  0011 1101 |
| ^ | Binary XOR Operator copies the bit if it is set in one operand but not both. | (A ^ B) will give 49, which is  0011 0001 |
| ~ | Binary Ones Complement Operator is unary and has the effect of 'flipping' bits. | (~A ) will give - 61, which is  1100 0011 in 2's  complement form. |
| << | Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand. | A << 2 will give 240 which is  1111 0000 |
| >> | Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand. | A >> 2 will give 15 which is  0000 1111 |

**Assignment Tasks**

**Task 01:** Create a program that will take subject marks as input and show individual percentage, total marks obtain and total percentage.

|  |  |  |  |
| --- | --- | --- | --- |
| Subject | Marks Obtain | Total Mark | Percentage |
| English | 35 | 100 | 35% |
| Urdu | 40 | 100 | 40% |
| Islamiat | 60 | 100 | 60% |
| Total Marks | 135 | 300 | 45% |

**Task 05:** If a five-digit number is input through the keyboard, write a program to calculate the sum of its digits. (Hint: Use the modulus operator ‘%’)

**Task 06:** If a five-digit number is input through the keyboard, write a program to calculate the sum of its digits. (Hint: Use the modulus operator ‘%’)

**Task 07:** In a town, the percentage of men is 52. The percentage of total literacy is 48. If total percentage of literate men is 35 of the total population, write a program to find the total number of illiterate men and women if the population of the town is 80,000.

**Task 08:**

## Review Questions:

1. Which of the following are invalid variable names and why?

BASICSALARY

\_basic basic-hra #MEAN

group.

422 population

in 2006 over time

mindovermatter FLOAT

hELLO

queue. team’svictory Plot # 3 2015\_DDay

1. Point out the errors, if any, in the following C statements: (a) int = 314.562 \* 150 ;
   1. name = ‘Ajay’ ;
   2. varchar = ‘3’ ;
   3. 3.14 \* r \* r \* h = vol\_of\_cyl ;

(e) k = ( a \* b ) ( c + ( 2.5a + b ) ( d + e ) ;

(f) m\_inst = rate of interest \* amount in rs ;

1. Evaluate the following expressions and show their hierarchy.
2. g = big / 2 + big \* 4 / big - big + abc / 3 ; (abc = 2.5, big = 2, assume g to be a float)
3. on = ink \* act / 2 + 3 / 2 \* act + 2 + tig ; (ink = 4, act = 1, tig = 3.2, assume on to be an int)
4. s = qui \* add / 4 - 6 / 2 + 2 / 3 \* 6 / god ; (qui = 4, add = 2, god = 2, assume s to be an int) (d) s = 1 / 3 \* a / 4 - 6 / 2 + 2 / 3 \* 6 / g ; (a = 4, g = 3, assume s to be an int)
5. C programs are converted into machine language with the help of? 5- A character variable can at a time store how many characters?
6. The maximum value that an integer constant can have?
7. What value does a variable contain if it is declared but not initialized?
8. What out is generated when we are using relational or logical operators?
9. Which operator requires all the given conditions to be true for the execution? 10- What is an opcode?