Practical # 03

C Building Blocks
(Operators and Expression)

OBJECT
To study the different types of arithmetic and logical operators

THEORY
In C, there are various operators, used to form expressions. The data items on which the operators act upon are called operands. Some operators require two operands while other act upon only one operand.

They are classified as:

1. Arithmetic Operators (binary type)
2. Unary Operators
3. Relational and Logical Operators
4. Assignment Operator

Arithmetic Operators
In C++, most programs perform arithmetic calculations. Arithmetic calculations can be performed by using the following arithmetic operators. Table 2.3 summarizes the C++ arithmetic operators. Note the use of various special symbols not used in algebra. The asterisk (*) indicates multiplication and the percent sign (%) is the modulus or remainder operator. The arithmetic operators in Table 2.3 are all binary operators, i.e., operators that take two operands.

Table 2.3
Arithmetic Operators

C++ operation	C++ arithmetic operator	C++ expression
Addition	+	x + y
Subtraction	-	x - y
Multiplication	*	x * y
Division	/	x / y
Modulus	%	x % y

Unary Operators:
In addition to the arithmetic assignment operators, C++ also provides two unary operators that act upon on a single operand to produce a new value, for adding 1 to or subtracting 1 from the value of a numeric variable. These are the unary increment operator, ++, and the unary decrement operator, --, which are summarized in the below table.

Table 2.3
Increment and Decrement Operators

Operators	Operation	Explanation
++a	Pre Increment	Increment ***a*** by 1, then use the new value of a in the expression in which a resides.
a++	Post Increment	Use the current value of ***a*** in the expression in which a resides, then increment a by 1.
--a	Pre Decrement	Decrement ***a*** by 1, then use the new value of ***a*** in the expression in which b resides.
a--	Post Decrement	Use the current value of ***a*** in the expression in which ***a*** resides, then decrement b by 1.

Assignment Operators:

C++ provides several assignment operators for abbreviating assignment expressions. For example, the statement

c = c + 3;

can be abbreviated with the addition assignment operator += as

c += 3;

The += operator adds the value of the expression on the right of the operator to the value of the variable on the left of the operator and stores the result in the variable on the left of the operator.

Thus the assignment `c += 3` adds `3` to `c`. Following table shows the arithmetic assignment operators, sample expressions using these operators and explanations.

Arithmetic assignment operators			
Assignment operator	**Sample expression**	**Explanation**	**Assigns**
Assume: int c = 3, d = 5, e = 4, f = 6, g = 12;			
Subtraction -=	**d -= 4**	**d = d - 4**	**1 to d**
Multiplication *=	**e *= 5**	**e = e * 5**	**20 to e**
Division /=	**f /= 3**	**f = f / 3**	**2 to f**
Remainder %=	**g %= 9**	**g = g % 9**	**3 to g**

Summary of Operator Precedence and Associativity:

Table adds the logical operators to the operator precedence and associativity chart. The operators are shown from top to bottom, in decreasing order of precedence.

Operator Precedence and Associativity.							
Operators						**Associativity**	**Type**
`()`						left to right	parentheses
`++`	`--`					left to right	unary (postfix)
`++`	`--`	`+`	`-`	`!`		right to left	unary (prefix)
`*`	`/`	`%`				left to right	multiplicative
`+`	`-`					left to right	additive
`<<`	`>>`					left to right	insertion/extraction
`<`	`<=`	`>`	`>=`			left to right	relational
`==`	`!=`					left to right	equality
`&&`						left to right	logical AND
`||`						left to right	logical OR
`?:`						right to left	conditional
`=`	`+=`	`-=`	`*=`	`/=`	`%=`	right to left	assignment

Common Programming Errors

- ✓ It is syntax error to place a calculation on the left side of an assignment operator.
- ✓ Forgetting one or both of the double quote surrounding the format control string in a *printf()* or *scanf()* statement.
- ✓ Forgetting to precede a variable name, in scnaf() statement, with an ampersand '&' operator.
- ✓ *An attempt to divide by zero is normally undefined on computer system results in a fatal error.*
- ✓ *A syntax error will occur if the two symbols in any of the operator ==, !=, >= and <= are separated by space.*
- ✓ *Confusing the equality operator == with the assignment operator.*

```
#include<stdio.h>
#include<conio.h>
void main (void)
{
   int a =10;
   int b =5;
   printf("%d + %d =  %d\n", a,b,a+b);
   printf("\n%d - %d =  %d\n", a,b,a-b);
   printf("\n%d * %d =  %d\n", a,b,a*b);
   printf("\n%d / %d = %d\n", a,b,a/b);
   printf("\n%d \%% %d =  %d\n", a,b,a%b);
   printf("\na = 10 and a++ = %d\n",a,a++);
   printf("\nb = 5 and ++b = %d\n",b,++b);
   getch();
}
```

Review Questions/ Exercise:

a) Corrected code: if (c > 7) {
printf("c is greater than 7\n");
}

b) Corrected code: if (c >= 7) {
printf("c is equal to or greater than 7\n");
}

c) Corrected code: printf("Remainder of %d divided by %d is %d\n", x, y, x % y);

d) Corrected code: ans = num1 + num2;

Evaluate the following

1) 9.0/6.0 + 5 / 2

Ans : 9.0/6.0 = 1.5

5 / 2 = 2
1.5 + 2 = 3.5

2) 9 * 3 / 4
Ans: 9 * 3 = 27
27 / 4 = 6

3) 14%7+3%4
Ans: 14 % 7 = 0
3 % 4 = 3
0 + 3 = 3

b) Determine the value assigned to the relevant variable

1) b = 5/4;
Answer: 1.0 because 5/4 is an integer division but assigned to b.

2) a = 5/4;
Answer: 1 because it is an integer division.

3) b = 5/2 + 3.0;
Answer: 5.0 because 5/2 is an integer = 2, 2+3.0 = 5.0.

C) Determine the value of int x after each statement

printf("%d\n", x); x is still 5.
printf("%d\n", ++x); x is incremented to 6.
printf("%d\n", x++); x becomes 7.
printf("%d\n", x); x is now 7.

II.

printf("%d\n", x); — x = 5
printf("%d\n", --x); — x = 4
printf("%d\n", x--); — x = 4 (printed), but x becomes 3
afterward. printf("%d\n", x); — x = 3

3.3) Order of Evaluation and Final Value of x

a) x = 7 + 36 / 2 - 1;
Answer: x = 24

b) x = 2 % 2 + 2 * 2 - 2 / 2;
Answer: x = 3
c) x = (3 * 9 * (3 + (93 / (3))));
Answer: x = 918

3.4)Write a program that asks the user to enter two numbers, obtain the two numbers from the user and print the sum, difference, quotient and remainder of the two.

```
#include <stdio.h>
      int main() {
int num1, num2;
 printf("Enter the first number: ");
 scanf("%d", &num1);
printf("Enter the second number: ");
scanf("%d", &num2);
 printf("Sum: %d\n", num1 + num2);

printf("Difference: %d\n", num1 - num2);
if (num2 != 0) {
        printf("Quotient: %d\n", num1 / num2);
        printf("Remainder: %d\n", num1 % num2);
} else {
        printf("Cannot divide by zero.\n");
        }
return 0; }
```

Name : Yash
Rollno : 24AI60
Date : 9/10/24

SubjectTeacher/LABEngineer :SirOwaisRehmani

remarks: