Operators

- C contains large no. of operators fall into different categories.

- **ARITHMETIC OPERATORS**
- **UNARY OPERATORS**
- **RELATIONAL AND LOGICAL OPERATORS**
- **ASSIGNMENT OPERATORS**
- **THE CONDITIONAL OPERATORS**.

Arithmetic Operators

Operation	Operator	Example	Value of Sum before	Value of sum after
Multiply	*	`sum = sum * 2;`	4	8
Divide	/	`sum = sum / 2;`	4	2
Addition	+	`sum = sum + 2;`	4	6
Subtraction	-	`sum = sum -2;`	4	2
Increment	++	`++sum;`	4	5
Decrement	--	`--sum;`	4	3
Modulus	%	`sum = sum % 3;`	4	1

Programming Language C

Types, Operators and Expressions

Relational and Logical Operators

Click to add text

Relation Operators

Operator	Meaning
==	equal to
!=	not equal
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

Logical Operators

Operator	Meaning
&&	and
||	or
!	not

Operand 1	Operand 2	op1 || op2	op1 && op2	! op1
0	0	0	0	1
0	non-zero	1	0	1
non-zero	0	1	0	0
non-zero	non-zero	1	1	0

Example: Squeeze

```c
#include <stdio.h>

void squeeze(char[], int);

main()
{
    char str[]="This is a string";

    squeeze(str, 'i');
    printf("%s\n", str);
}

/* squeeze: delete all c from s */
void squeeze(char s[], int c)
{
    int i, j;
    for (i = j = 0; s[i] != '\0'; i++)
        if (s[i] != c) s[j++] = s[i];
    s[j] = '\0';
}
```

Exercises

8. Write an alternative version of `squeeze(s1,s2)` that deletes each character in `s1` that matches any character in the string `s2`.

   Write the function `any(s1,s2)`, which returns the first location in a string `s1` where any character from the string `s2` occurs, or `-1` if `s1` contains no characters from `s2`. (The standard library function strpbrk does the same job but returns a pointer to the location.)

Bitwise Operators

Operation	Operator	Comment	Value of Sum before	Value of sum after
AND	&	`sum = sum & 2;`	4	0
OR	|	`sum = sum | 2;`	4	6
Exclusive OR	^	`sum = sum ^ 2;`	4	6
1's Complement	~	`sum = ~sum;`	4	-5
Left Shift	<<	`sum = sum << 2;`	4	16
Right Shift	>>	`sum = sum >> 2;`	4	1

```
 1=00000000 00000000 00000000 00000001
 2=00000000 00000000 00000000 00000010
 4=00000000 00000000 00000000 00000100
 8=00000000 00000000 00000000 00001000
16=00000000 00000000 00000000 00010000
```

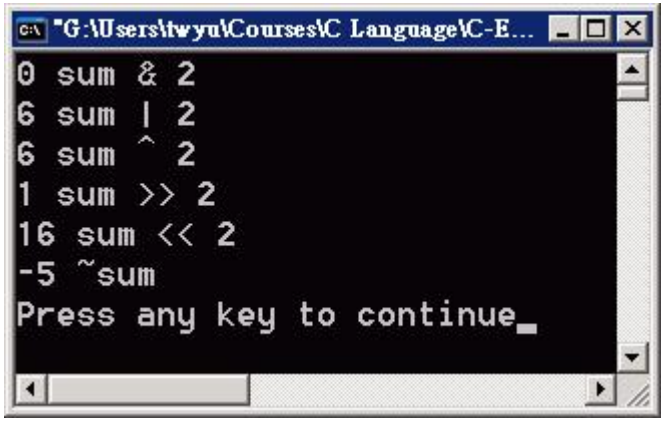
```
 5=00000000 00000000 00000000 00000101
   11111111 11111111 11111111 11111010
-5=11111111 11111111 11111111 11111011
```

Example: Bitwise Operators

```c
#include <stdio.h>

main()
{
    unsigned sum = 4;

    printf("%d sum & 2\n", sum & 2);
    printf("%d sum | 2\n", sum | 2);
    printf("%d sum ^ 2\n", sum ^ 2);
    printf("%d sum >> 2\n", sum >> 2);
    printf("%d sum << 2\n", sum << 2);
    printf("%d ~sum\n", ~sum);
}
```




```
 1=00000000 00000000 00000000 00000001
 2=00000000 00000000 00000000 00000010
 4=00000000 00000000 00000000 00000100
 8=00000000 00000000 00000000 00001000
16=00000000 00000000 00000000 00010000
```

```
 5=00000000 00000000 00000000 00000101
   11111111 11111111 11111111 11111010
-5=11111111 11111111 11111111 11111011
```

Assignment Operators

```
i = i + 2;
```

```
i += 2;
```

```
x = x * (y + 1);
```

```
x *= y + 1;
```

exp_1 **=** exp_1 **op** (*exp*$_2$)

exp_1 **op=** exp_2

Assignment Operators

Operator	Operation Performed
=	Simple assignment
*=	Multiplication assignment
/=	Division assignment
%=	Remainder assignment
+=	Addition assignment
-=	Subtraction assignment
<<=	Left-shift assignment
>>=	Right-shift assignment
&=	Bitwise-AND assignment
^=	Bitwise-exclusive-OR assignment
|=	Bitwise-inclusive-OR assignment