Introduction to Scientific and Engineering Computation (BIL 104E)

Lab 4

Using Arithmetic Assignment Operators

Using Arithmetic Assignment Operators

```
1:  /* 06L01.c: Using arithemtic assignment operators */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int x, y, z;
7:
8:     x = 1;   /* initialize x */
9:     y = 3;   /* initialize y */
10:    z = 10;  /* initialize z */
11:    printf("Given x = %d, y = %d, and z = %d,\n", x, y, z);
12:
13:    x = x + y;
14:    printf("x = x + y  assigns %d to x;\n", x);
15:
```

Using Arithmetic Assignment Operators

```
16:     x = 1;  /* reset x */
17:     x += y;
18:     printf("x += y  assigns %d to x;\n", x);
19:
20:     x = 1;  /* reset x */
21:     z = z * x + y;
22:     printf("z = z * x + y  assigns %d to z;\n", z);
23:
24:     z = 10;  /* reset z */
25:     z = z * (x + y);
26:     printf("z = z * (x + y) assigns %d to z;\n", z);
27:
28:     z = 10;  /* reset z */
29:     z *= x + y;
30:     printf("z *= x + y assigns %d to z.\n", z);
31:
32:     return 0;
33: }
```

Using Arithmetic Assignment Operators

Computer Screen

```
Given x = 1, y = 3, and z = 10,
x = x + y  assigns 4 to x;
x += y  assigns 4 to x;
z = z * x + y  assigns 13 to z;
z = z * (x + y) assigns 40 to z;
z *= x + y assigns 40 to z.
```

Using Pre- or Post-Increment and Decrement Operators

Using Pre- or Post-Increment and Decrement Operators

```
1:  /* 06L02.c: pre- or post-increment(decrement) operators */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int w, x, y, z, result;
7:
8:     w = x = y = z = 1;    /* initialize x and y */
9:     printf("Given w = %d, x = %d, y = %d, and z = %d,\n", w, x, y, z);
10:
11:    result = ++w;
12:    printf("++w evaluates to %d and w is now %d\n", result, w);
13:    result = x++;
14:    printf("x++ evaluates to %d and x is now %d\n", result, x);
15:    result = --y;
16:    printf("--y evaluates to %d and y is now %d\n", result, y);
17:    result = z--;
18:    printf("z-- evaluates to %d and z is now %d\n", result, z);
19:    return 0;
20: }
```

Using Pre- or Post-Increment and Decrement Operators

Computer Screen

```
Given w = 1, x = 1, y = 1, and z = 1,
++w evaluates to 2 and w is now 2
x++ evaluates to 1 and x is now 2
--y evaluates to 0 and y is now 0
z-- evaluates to 1 and z is now 0
```

Results Produced by Relational Expressions

Results Produced by Relational Expressions

```
1:  /* 06L03.c: Using relational operators */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int x, y;
7:     double z;
8:
9:     x = 7;
10:    y = 25;
11:    z = 24.46;
12:    printf("Given x = %d, y = %d, and z = %.2f,\n", x, y, z);
13:    printf("x >= y  produces: %d\n", x >= y);
14:    printf("x == y  produces: %d\n", x == y);
15:    printf("x < z   produces: %d\n", x < z);
16:    printf("y > z   produces: %d\n", y > z);
17:    printf("x != y - 18  produces: %d\n", x != y - 18);
18:    printf("x + y != z   produces: %d\n", x + y != z);
19:    return 0;
20: }
```

Results Produced by Relational Expressions

Computer Screen

```
Given x = 7, y = 25, and z = 24.46,
x >= y   produces: 0
x == y   produces: 0
x < z    produces: 1
y > z    produces: 1
x != y - 18   produces: 0
x + y != z    produces: 1
```

Playing with the Cast Operator

Playing with the Cast Operator

```
1:  /* 06L04.c: Using the cast operator */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int x, y;
7:
8:     x = 7;
9:     y = 5;
10:    printf("Given x = %d, y = %d\n", x, y);
11:    printf("x / y produces: %d\n",  x / y);
12:    printf("(float)x / y produces: %f\n",  (float)x / y);
13:    return 0;
14: }
```

```
Given x = 7, y = 5
x / y produces: 1
(float)x / y produces: 1.400000
```

Using a while Loop

Using a while Loop

```
1:  /* 07L01.c: Using a while loop */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int c;
7:
8:     c = ' ';
9:     printf("Enter a character:\n(enter x to exit)\n");
10:    while (c != 'x') {
11:       c = getc(stdin);
12:       putchar(c);
13:    }
14:    printf("\nOut of the while loop. Bye!\n");
15:    return 0;
16: }
```

Using a while Loop

Computer Screen

```
Enter a character:
(enter x to exit)
H
H
i
i
x
x
Out of the while loop. Bye!
```

Using a do-while Loop

Using a do-while Loop

```
1:  /* 07L02.c: Using a do-while loop */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int i;
7:
8:     i = 65;
9:     do {
10:       printf("The numeric value of %c is %d.\n", i, i);
11:       i++;
12:    } while (i<72);
13:    return 0;
14: }
```

```
The numeric value of A is 65.
The numeric value of B is 66.
The numeric value of C is 67.
The numeric value of D is 68.
The numeric value of E is 69.
The numeric value of F is 70.
The numeric value of G is 71.
```

Loops: Ling Under the for Statement

Converting 0 through 15 to Hex Numbers

```
1:  /* 07L03.c: Converting 0 through 15 to hex numbers */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int i;
7:
8:     printf("Hex(uppercase)   Hex(lowercase)   Decimal\n");
9:     for (i=0; i<16; i++){
10:       printf("%X                %x                %d\n", i, i, i);
11:    }
12:    return 0;
13: }
```

Hex(uppercase)	Hex(lowercase)	Decimal
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
A	a	10
B	b	11
C	c	12
D	d	13
E	e	14
F	f	15

Adding Multiple Expressions to the for Statement

Adding Multiple Expressions to the for Statement

```
1:  /* 07L04.c: Multiple expressions */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int i, j;
7:
8:     for (i=0, j=8; i<8; i++, j--)
9:        printf("%d  +  %d  =  %d\n", i, j, i+j);
10:    return 0;
11: }
```

```
0  +  8  =  8
1  +  7  =  8
2  +  6  =  8
3  +  5  =  8
4  +  4  =  8
5  +  3  =  8
6  +  2  =  8
7  +  1  =  8
```

Another Example of Using Multiple Expressions in the for Statement

Another Example of Using Multiple Expressions in the for Statement

```
1:  /* 07L05.c: Another example of multiple expressions */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int i, j;
7:
8:     for (i=0, j=1; i<8; i++, j++)
9:        printf("%d  -  %d  =  %d\n", j, i, j - i);
10:    return 0;
11: }
```

```
1  -  0  =  1
2  -  1  =  1
3  -  2  =  1
4  -  3  =  1
5  -  4  =  1
6  -  5  =  1
7  -  6  =  1
8  -  7  =  1
```

Using Nested Loops

Using Nested Loops

```
1:  /* 07L06.c: Demonstrating nested loops */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int i, j;
7:
8:     for (i=1; i<=3; i++) {   /* outer loop */
9:        printf("The start of iteration %d of the outer loop.\n", i);
10:       for (j=1; j<=4; j++)  /* inner loop */
11:          printf("    Iteration %d of the inner loop.\n", j);
12:       printf("The end of iteration %d of the outer loop.\n", i);
13:    }
14:    return 0;
15: }
```

Loops: Using Nested Loops

Computer Screen

```
The start of iteration 1 of the outer loop.
    Iteration 1 of the inner loop.
    Iteration 2 of the inner loop.
    Iteration 3 of the inner loop.
    Iteration 4 of the inner loop.
The end of iteration 1 of the outer loop.
The start of iteration 2 of the outer loop.
    Iteration 1 of the inner loop.
    Iteration 2 of the inner loop.
    Iteration 3 of the inner loop.
    Iteration 4 of the inner loop.
The end of iteration 2 of the outer loop.
The start of iteration 3 of the outer loop.
    Iteration 1 of the inner loop.
    Iteration 2 of the inner loop.
    Iteration 3 of the inner loop.
    Iteration 4 of the inner loop.
The end of iteration 3 of the outer loop.
```