Introduction to Scientific and Engineering Computation (BIL 104E)

Lab 5

Conditional Operators: Measuring Data Sizes

Using the sizeof Operator

```
1:  /* 08L01.c: Using the sizeof operator */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     char   ch = ' ';
7:     int    int_num = 0;
8:     float  flt_num = 0.0f;
9:     double dbl_num = 0.0;
10:
11:    printf("The size of char is: %d-byte\n", sizeof(char));
12:    printf("The size of ch is: %d-byte\n", sizeof ch );
13:    printf("The size of int is: %d-byte\n", sizeof(int));
14:    printf("The size of int_num is: %d-byte\n", sizeof int_num);
15:    printf("The size of float is: %d-byte\n", sizeof(float));
16:    printf("The size of flt_num is: %d-byte\n", sizeof flt_num);
17:    printf("The size of double is: %d-byte\n", sizeof(double));
18:    printf("The size of dbl_num is: %d-byte\n", sizeof dbl_num);
19:    return 0;
20: }
```

Conditional Operators: Measuring Data Sizes

Computer Screen

```
The size of char is: 1-byte
The size of ch is: 1-byte
The size of int is:4-byte
The size of int_num is:4-byte
The size of float is: 4-byte
The size of flt_num is: 4-byte
The size of double is: 8-byte
The size of dbl_num is:8-byte
```

Conditional Operators: AND Operator (&&)

```
#include <stdio.h>

int main( )
{
   int num;

   num = 1;
   printf("%d && %d yields %d\n",(num%2 == 0), (num%3 == 0), (num%2 == 0) && (num%3 == 0));
   num = 2;
   printf("%d && %d yields %d\n",(num%2 == 0), (num%3 == 0), (num%2 == 0) && (num%3 == 0));
   num = 3;
   printf("%d && %d yields %d\n",(num%2 == 0), (num%3 == 0), (num%2 == 0) && (num%3 == 0));
   num = 6;
   printf("%d && %d yields %d\n",(num%2 == 0), (num%3 == 0), (num%2 == 0) && (num%3 == 0));
   getchar();

   return 0;
}
```

```
0 && 0 yields 0
1 && 0 yields 0
0 && 1 yields 0
1 && 1 yields 0
```

Conditional Operators: OR Operator (||)

Using the Logical OR Operator ||

```
1:  /* 08L03.c: Using the logical OR operator */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int   num;
7:
8:     printf("Enter a single digit that can be divided\nby both 2 and 3:\n");
9:     for (num = 1; (num%2 != 0) || (num%3 != 0); )
10:       num = getchar() - '0';
11:    printf("You got such a number: %d\n", num);
12:    return 0;
13: }
```

```
Enter a single digit that can be divided
by both 2 and 3:
2
3
4
5
6
```

Conditional Operators: NEGATION Operator (!)

Using the Logical Negation Operator (!)

```
1:  /* 08L04.c: Using the logical negation operator */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int   num;
7:
8:     num = 7;
9:     printf("Given num = 7\n");
10:    printf("!(num < 7)  yields: %d\n", !(num < 7));
11:    printf("!(num > 7)  yields: %d\n", !(num > 7));
12:    printf("!(num == 7) yields: %d\n", !(num == 7));
13:    return 0;
14: }
```

```
Given num = 7
!(num < 7)  returns: 1
!(num > 7)  returns: 1
!(num == 7) returns: 0
```

Conditional Operators: Bitwise Operators

Using Bitwise Operators

```
1:  /* 08L05.c: Using bitwise operators */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int   x, y, z;
7:
8:     x = 4321;
9:     y = 5678;
10:    printf("Given x = %u, i.e., 0X%04X\n", x, x);
11:    printf("      y = %u, i.e., 0X%04X\n", y, y);
12:    z = x & y;
13:    printf("x & y  returns: %6u, i.e., 0X%04X\n", z, z);
14:    z = x ¦ y;
15:    printf("x ¦ y  returns: %6u, i.e., 0X%04X\n", z, z);
16:    z = x ^ y;
17:    printf("x ^ y  returns: %6u, i.e., 0X%04X\n", z, z);
18:    printf("  ~x   returns: %6u, i.e., 0X%04X\n", ~x, ~x);
19:    return 0;
20: }
```

Conditional Operators: Bitwise Operators

Computer Screen

```
Given x = 4321, i.e., 0X10E1
      y = 5678, i.e., 0X162E
x & y  returns:  4128, i.e., 0X1020
x ¦ y  returns:  5871, i.e., 0X16EF
x ^ y  returns:  1743, i.e., 0X06CF
  ~x   returns: 61214, i.e., 0XEF1E
```

Conditional Operators: Logical -Bitwise Operators

```
#include<stdio.h>

int main()
{
   int a, b, result;

   a=1;
   b=10;
   result = a && b;
   printf(" The result of a && b is %d\n", result);
   result = a & b;
   printf(" The result of a & b is %d\n", result);
   getchar();
   return 0;
}
```

```
The result of a && b is 1
The result of a & b is  0
```

Conditional Operators: Shift Operators

```
#include<stdio.h>
main()
{
    int num, i, power_of, temp, right_shift, divider, digit1, digit2, digit3, digit4;

    printf("please write decimal number?\n");
    scanf("%d", &num);
    printf("please write decimal number for right shifting?\n");
    scanf("%d", &right_shift);

    digit1 = num % 2;
    divider = num / 2;
    digit2 = divider % 2;
    divider /= 2;
    digit3 =divider % 2;
    divider /= 2;
    digit4 = divider % 2;
    printf("Binary representation of %d is %d %d %d %d \n",num, digit4, digit3, digit2,digit1);
    temp = num;
```

Conditional Operators: Shift Operators

```
num = num >> right_shift;
    digit1 = num % 2;
    divider = num / 2;
    digit2 = divider % 2;
    divider /= 2;
    digit3 =divider % 2;
    divider /= 2;
    digit4 = divider % 2;
    printf("Binary representation of %d right shifted number %d is %d %d %d %d
\n",right_shift, num, digit4, digit3, digit2,digit1);
    power_of = 1;
    for(i=1; i <= right_shift;i++)
          power_of *= 2;
    printf( "%d / %d gives same result as right shifting %d\n", temp, right_shift,
temp/power_of);
    getchar();
    getchar();
    return 0;
}
```

```
please write decimal number?
15
please write decimal number for right shifting?
2
Binary representation of 15 is 1 1 1 1
Binary representation of 2 right shifted number 3 is 0 0 1 1
15 / 2 gives same result as right shifting 3
```

Conditional Operators: x?y:z operation

Using the Conditional Operator

```
1:  /* 08L07.c: Using the ?: operator */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int   x;
7:
8:     x = sizeof(int);
9:     printf("%s\n",
10:       (x == 2)
11:       ? "The int data type has 2 bytes."
12:       : "int doesn't have 2 bytes.");
13:    printf("The maximum value of int is: %d\n",
14:       (x != 2) ? ~(1 << x * 8 - 1) : ~(1 << 15) );
15:    return 0;
16: }
```

```
int doesn't have 2 bytes.
The maximum value of int is: 2147483647
```

Data Modifiers : Changing Data Sizes

Modifying Data with short and long

```
1:  /* 09L02.c: Using short and long modifiers */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:      printf("The size of short int is: %d.\n",
7:          sizeof(short int));
8:      printf("The size of long int is: %d.\n",
9:          sizeof(long int));
10:     printf("The size of float is: %d.\n",
11:         sizeof(float));
12:     printf("The size of double is: %d.\n",
13:         sizeof(double));
14:     printf("The size of long double is: %d.\n",
15:         sizeof(long double));
16:     return 0;
17: }
```

```
The size of short int is: 2.
The size of long int is: 4.
The size of float is: 4.
The size of double is: 8.
The size of long double is: 10.
```

Mathematical Functions in C: sin(), cos(), tan()

Calculating Trigonometric Values with sin(), cos(), and tan()

```
1:  /* 09L04.c: Using sin(), cos(), and tan() functions */
2:  #include <stdio.h>
3:  #include <math.h>
4:
5:  main()
6:  {
7:     double x;
8:
9:     x = 45.0;                    /* 45 degree */
10:    x *= 3.141593 / 180.0;       /* convert to radians */
11:    printf("The sine of 45 is:    %f.\n", sin);
12:    printf("The cosine of 45 is:  %f.\n", cos);
13:    printf("The tangent of 45 is: %f.\n", tan);
14:    return 0;
15: }
```

```
The sine of 45 is:    0.707107.
The cosine of 45 is:  0.707107.
The tangent of 45 is: 1.000000.
```

Mathematical Functions in C: pow(), sqrt()

Applying the pow() and sqrt() Functions

```
1:  /* 09L05.c: Using pow() and sqrt() functions */
2:  #include <stdio.h>
3:  #include <math.h>
4:
5:  main()
6:  {
7:     double x, y, z;
8:
9:     x = 64.0;
10:    y = 3.0;
11:    z = 0.5;
12:    printf("pow(64.0, 3.0) returns: %7.0f\n", pow(x, y));
13:    printf("sqrt(64.0) returns:     %2.0f\n", sqrt);
14:    printf("pow(64.0, 0.5) returns: %2.0f\n", pow(x, z));
15:    return 0;
16: }
```

```
pow(64.0, 3.0) returns: 262144
sqrt(64.0) returns:     8
pow(64.0, 0.5) returns: 8
```