C Programming Introduction

Week 5:Expressions

Topic of this week

- Expressions
  - Class Lecture Review
    - mathematic operators
    - boolean operators
    - conditional expressions
  - Programming Exercises

Expression and Operations

- Arithmetic Operators
  - Addition **+**
  - Subtraction **-**
  - Multiplication *****
  - Division **/**
  - Modulation **%**
- Example
  - `fag = x % y;`
  - `c = a - (a/b)*b;`
  - `sum = var1 + var2 + var3;`

Expression and Operations

- Operator precedence
  - Some arithmetic operators act before others (i.e., multiplication before addition)
    - Use parenthesis when needed
  - Example: Find the average of three variables **`a`**, **`b`** and **`c`**
    - Do not use: **`a + b + c / 3`**
    - Use: **`(a + b + c ) / 3`**

Expression and Operations

- Rules of operator precedence:

Operator(s)	Operation(s)	Order of evaluation (precedence)
`()`	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
`*`, `/`, or `%`	Multiplication Division Modulus	Evaluated second. If there are several, they re evaluated left to right.
`+` or `-`	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

Decision Making: Equality and Relational Operators

- Executable statements
  - Perform actions (calculations, input/output of data)
  - Perform decisions
    - May want to print "pass" or "fail" given the value of a test grade
- `if` control structure
  - Simple version in this section, more detail later
  - If a condition is true, then the body of the `if` statement executed
    - `0` is false, non-zero is true
  - Control always resumes after the `if` structure
- Keywords
  - Special words reserved for C
  - Cannot be used as identifiers or variable names

Decision Making: Equality and Relational Operators

- Relational Operators
  - Less than **<** a < 5
  - Less than or equal **<=** a <= b
  - More than **>** a > b + c
  - More than or equal **>=** a >= b + 5
  - Equal **==** a == -6
  - Not equal **!=** a != 0

Decision Making: Equality and Relational Operators

Keywords			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Example 1

- Using if statements, relational

```
2  #include <stdio.h>
3
4  int main()
5  {
6     int num1, num2;
7
8     printf( "Enter two integers, and I will tell you\n" );
9     printf( "the relationships they satisfy: " );
10    scanf( "%d%d", &num1, &num2  );   /* read two integers */
11
12    if ( num1 == num2 )
13       printf( "%d is equal to %d\n", num1, num2 );
14
15    if ( num1 != num2 )
16       printf( "%d is not equal to %d\n", num1, num2 );
17
18    if ( num1 < num2 )
19       printf( "%d is less than %d\n", num1, num2 );
20
21    if ( num1 > num2 )
22       printf( "%d is greater than %d\n", num1, num2 );
23
24    if ( num1 <= num2 )
25       printf( "%d is less than or equal to %d\n",
26               num1, num2 );
```

```
27
28    if ( num1 >= num2 )
29       printf( "%d is greater than or equal to %d\n",
30               num1, num2 );
31
32    return 0;    /* indicate program ended successfully */
33 }
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

Expression and Operations

- Logical Operators
  - AND **&&** (a > 0) && (b > 0)
  - OR **||** (a <= 0) || (b <= 0)
  - Negation **!** !(a && c)

Expression and Operations

- Bitwise Operators
  - Bitwise AND **&**
  - Bitwise OR (Inclusive OR) **|**
  - Bitwise XOR (Exclusive OR) **^**
  - Left shift **<<**
  - Right shift **>>**
  - One's complement **~**
- Example
  - x = 01001011 y = 00101100 ~x = 10110100
  - x & y = 00001000 x | y = 01101111
  - x ^ y = 01100111 x << 2 = 00101100

Expression and Operations

- Assignment Operators and Expressions
  - **op** is + - * / % << >> & ^ |
  - If **expr1** and **expr2** are expressions, then

```
expr1 op= expr2
```

  - Equivalent to

```
expr1 = (expr1) op (expr2)
```

- Example
  - `x += 1;`
  - `x = x + 1;`

} Equivalent

Expression and Operations

- Conditional Expressions

```
expr1 ? expr2 : expr3
```

  - If **expr1** is *true* do **expr2**
  - If **expr1** is *false* do **expr3**
- Example
  - ```
    a = 5;
    b = 10;
    min = a < b ? a : b;
    ```
    ```

Expression and Operations

- Increment and Decrement Operators
  - Pre-increment operation **++variable**
  - Post-increment operation **variable++**
  - Pre-decrement operation **--variable**
  - Post-decrement operation **variable--**
- Example
  - ```
    x = 4;
    y = x++ + 5;        // x = 5, y = 9
    ```
  - ```
    x = 4;
    y = ++x + 5;        // x = 5, y = 10
    ```

Expression and Operations

- Type Cast Operator (Casting)

```
(type-specifier) expression;
```

- Example
  - `(double) date;`
  - `fload var1 = 2.7;`
    `int var2 = (int) var1;` `//var2 = 7`
  - `(char) x;`
  - `(int) d1 + d2;`

Exercise 5.1

- Write a program that converts distances from kilometers to miles.
- Ask user to input the kilometers value then output to screen the miles value.

Solution

```
#include <stdio.h>
/* printf, scanf definitions*/

int main(void)
{
    double          miles,    /* distance in miles */
    kms;            /* equivalent distance in kilometers */

    /* Get the distance in kilometers. */
    printf("Enter the distance in kilometers > ");
    scanf("%lf", & kms);

    /* Convert the distance to miles. */
     miles = 1000 * kms;

    /* Display the distance in miles. */
    printf("That equals %f miles.\n", miles);
    return (0);
}
```

Exercise 5.2

- Run the exercise5_2.c program below to illustrate the operation of Logical operators and relational operators.
- Replace **b - a == b – c** by **a = b-c** and then explane the result.

exercise5_2.c

```
#include <stdio.h>

main()
{
  int a = 5, b = 6, c = 7;
   puts("int a = 5, b = 6, c = 7;\n");

  printf("The value of  a > b  is        \t%i\n\n", a > b);
  printf("The value of  b < c  is        \t%i\n\n", b < c);

  printf("The value of  a + b >= c  is    \t%i\n\n", a + b >= c);
  printf("The value of  a - b <= b-c is\t%i\n\n", a - b <=b- c);

  printf("The value of  b - a == b - c  is\t%i\n\n",b- a==b- c);
  printf("The value of  a * b != c * c  is\t%i\n\n",a * b<c * c);
}
```

Exercise 5.3

- Type and compile the exercise5_3.c below, the program illustrates the operation of conditional expressions.
- Alter the program by eliminating the abs and max variables.

exercise5_3.c

```
#include <stdio.h>
main()
{
  int n, m, abs, max;

  printf("Enter a positive or negative integer: ");
  scanf("%i", &n);

  printf("\nYou entered %i.\n", n);
  abs = n < 0 ? -n : n;
  printf("Its absolute value is %i.\n", abs);

  printf("\nEnter two integers (e.g. 1 2): ");
  scanf("%i %i", &n, &m);

  printf("\nYou entered %i and %i.\n", n, m);
  max = n > m ? n : m;
  printf("%i is the larger value.\n", max);
}
```

Exercise 5.4

- This example illustrates the **integer overflow** that occurs when an arithmetic operation attempts to create a numeric **value** that is larger than can be represented.
- Type and compile the program to see the result.

exercise5_4.c

```
#include <stdio.h>
#include <limits.h>

void main(void)
{
  unsigned int x = UINT_MAX - 1;
  signed int y = INT_MAX - 1;

  printf("x is an unsigned int, occupying %i bytes.\n\n", sizeof(x));

  printf("The initial value of x is %u\n", x);
  x++;
  printf("Add 1; the new value of x is %u\n", x);
  x++;
```

exercise5_4.c

```
printf("Add 1; the new value of x is %u\n", x);
  x++;
  printf("Add 1; the new value of x is %u\n", x);

  printf("\ny is a signed int, occupying %i bytes.\n\n", sizeof(y));

  printf("The initial value of y is %i\n", y);
  y++;
  printf("Add 1; the new value of y is %i\n", y);
  y++;
  printf("Add 1; the new value of y is %i\n", y);
  y++;
  printf("Add 1; the new value of y is %i\n", y);

  return;
}
```

Exercise 5.5

- Write a program that requires user to input two double values stored in two variables x,y.
- Use **if** control structure to examine all the relation between x and y.

Solution

```
#include <stdio.h>

main()
{
   double num1, num2;

   printf( "Enter two doubles, and I will tell you\n" );
   printf( "the relationships they satisfy: " );
   scanf( "%f%f", &num1, &num2  ); /* read two integers */

   if ( num1 == num2 )
      printf(  "%f is equal to %f\n", num1, num2 );

   if ( num1 != num2 )
      printf(  " %f is not equal to %f\n ", num1, num2 );
```

Solution

```
if ( num1 < num2 )
      printf( "%f is less than %f\n", num1, num2 );

   if ( num1 > num2 )
      printf( "%f is greater than %f\n", num1, num2 );

   if ( num1 <= num2 )
      printf( "%f is less than or equal to %f\n",
              num1, num2 );

   if ( num1 >= num2 )
      printf( "%f is greater than or equal to %f\n",
              num1, num2 );

   return 0;   /* indicate program ended successfully */
}
```

Exercise 5.6

- A group of n students is to be divided into 7 classes, as evenly as possible. (No class size should differ by more than 1 student when compared to all other class sizes.) Write a C expression for:
  - the number of students in the smallest class
  - the number of students in the largest class
  - the average number of students per class;
  - the number of classes of above average size;
  - the number of classes of at most average size;
  - the number of students in classes of larger than average size;
  - the number of classes of exactly average size.

Solution

- 7(a) n / 7
- 7(b) (n + 6) / 7 , NOT n / 7 + 1.
  - Both expressions give the same result, except when n is divisible by 7. In this case, n/7+1 gives the wrong answer.
- 7(c) n / 7.0
- 7(d) n % 7
- 7(e) 7 - n % 7 (would be trickier if asked for *below* average ...)
- 7(f) (n % 7) * ((n + 6) / 7)
- 7(g) (n % 7 == 0) * 7 or ! (n % 7) * 7