Basis and Practice in Programming

Chapter 5: Operators, expressions, and statements

Prof. Tamer ABUHMED
College of Software

성균관대학교
SUNG KYUN KWAN UNIVERSITY(SKKU)

Lecture Objectives

- Explain the while loop
- Introduce the increment and decrement operators
- Introduce operator procedure and priority
- Introduce the modulus operator
- Explain the type/value casting
- Exercise & Keywords

Loops

- Loop
  - Enables you to repeat actions
  - How to repeat things in C?
    - Maybe like the following example?

```c
// prints the multiplication table of 3
# include <stdio.h>
# define X 3

void main(void)
{

    printf("1 * X = %d,\n", 1 * X);
    printf("2 * X = %d,\n", 2 * X);
    printf("3 * X = %d,\n", 3 * X);
    printf("4 * X = %d,\n", 4 * X);
    printf("5 * X = %d,\n", 5 * X);
    printf("6 * X = %d,\n", 6 * X);
    printf("7 * X = %d,\n", 7 * X);

}
```

This is boooooooooooooring.
But C is not!
So, how does C do it? ☹

Output

```
1 * X = 3,
2 * X = 6,
3 * X = 9,
4 * X = 12,
5 * X = 15,
6 * X = 18,
7 * X = 21,
```

Loops – contd.

- The C loop
  - Using the *while loop*
  - Checks if the argument is correct or not
    - If correct, the code between braces is executed
    - If not correct, the while is terminated (the code is not executed)

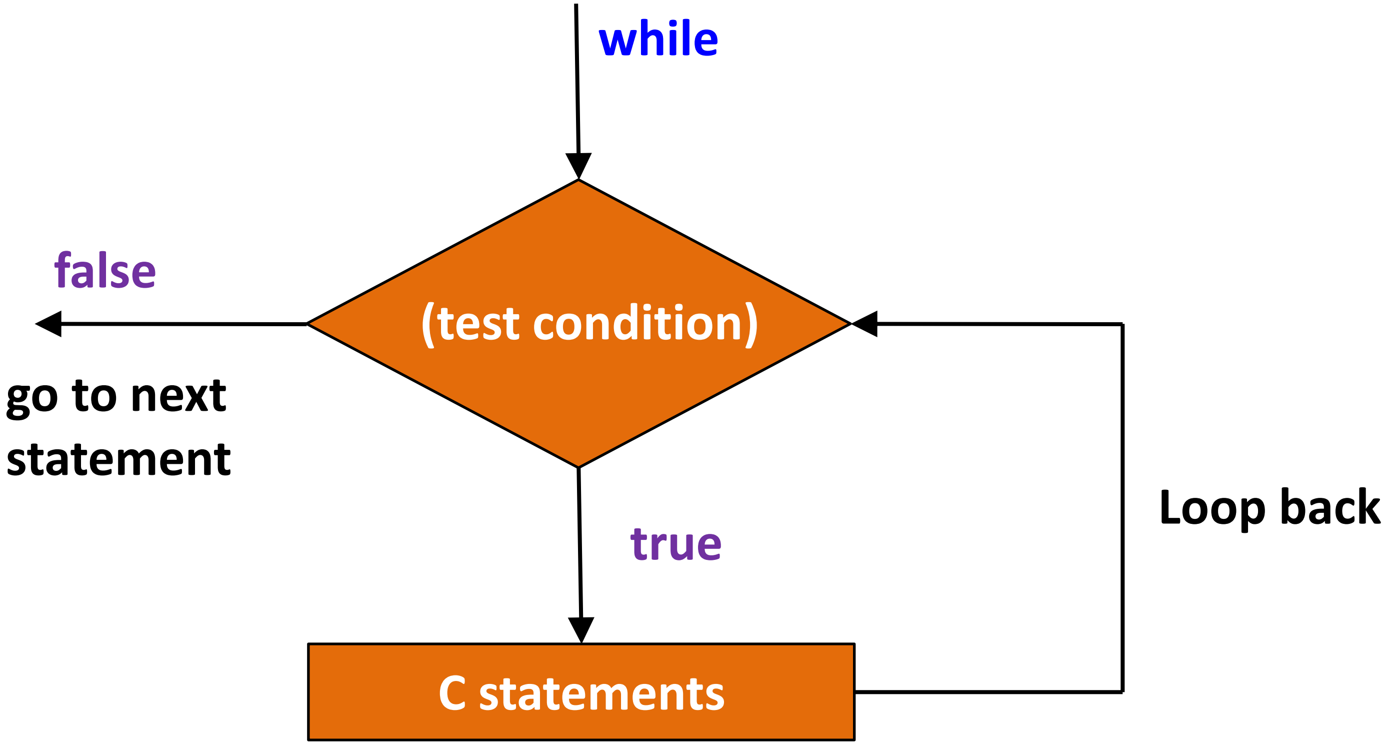
```
// prints the multiplication table of 3
# include <stdio.h>
# define X 3
# define LIMIT 7

void main(void)
{
    int count = 1;

    while(count <= LIMIT)      // loop condition
    {
         printf("%d * X = %d,\n", count, count * X);
         count = count + 1;  // add 1 to count
    }
}
```

Output

```
1 * X = 3,
2 * X = 6,
3 * X = 9,
4 * X = 12,
5 * X = 15,
6 * X = 18,
7 * X = 21,
```

The *while* Loop

- Structure of a simple while loop



The *while* Loop – contd.

- The C loop
  - Using the *while loop*

```
    while(count <= LIMIT)
    {
        printf("%d * X = %d,\n", count, count * X);
        count = count + 1;
    }
```

First iteration

Variable status

count = 1, LIMIT = 7, X = 3

Output

1 * X = 3,

```
    while(count <= LIMIT)
    {
        printf("%d * X = %d,\n", count, count * X);
        count = count + 1;
    }
```

Second iteration

Variable status

count = 2, LIMIT = 7, X = 3

Output

2 * X = 6,

```
    while(count <= LIMIT)
    {
        printf("%d * X = %d,\n", count, count * X);
        count = count + 1;
    }
```

Third iteration

Variable status

count = 3, LIMIT = 7, X = 3

Output

3 * X = 9,

The *while* Loop – contd.

- The C loop – contd.
  - Using the *while loop*

```
    while(count <= LIMIT)
    {
        printf("%d * X = %d,\n", count, count * X);
        count = count + 1;
    }
```

Fourth iteration

Variable status

count = 4, LIMIT = 7, X = 3

Output

4 * X = 12,

```
    while(count <= LIMIT)
    {
        printf("%d * X = %d,\n", count, count * X);
        count = count + 1;
    }
```

Fifth iteration

Variable status

count = 5, LIMIT = 7, X = 3

Output

5 * X = 15,

```
    while(count <= LIMIT)
    {
        printf("%d * X = %d,\n", count, count * X);
        count = count + 1;
    }
```

Sixth iteration

Variable status

count = 6, LIMIT = 7, X = 3

Output

6 * X = 18,

The *while* Loop – contd.

- The C loop – contd.
  - Using the *while loop*

```
    while(count <= LIMIT)
    {
        printf("%d * X = %d,\n", count, count * X);
        count = count + 1;
    }
```

Seventh iteration

Variable status

count = 7, LIMIT = 7, X = 3

Output

7 * X = 21,

  - At the end of the seventh iteration
    - count is increased → count = 8
  - At the beginning of the eighth iteration
    - count = 8 which is NOT LESS THAN OR EQUAL LIMIT
    - Therefore, the eighth iteration is not executed and the while loop is terminated
    - The control passes to the first statement after the while loop

```
    while(count <= LIMIT) // the loop is terminated
    {
        printf("%d * X = %d,\n", count, count * X);
        count = count + 1;
    }
```

Eighth iteration

Variable status

count = 8, LIMIT = 7, X = 3

Output

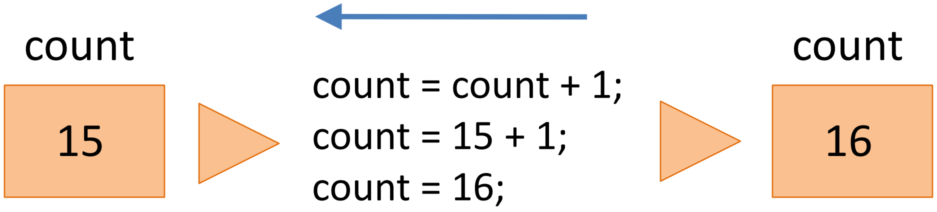
No output

Assignment operation =

- How instructions are performed in C?
  - An example: count = count + 1;

```
# include <stdio.h>

void main(void)
{
    int count = 15;

    count = count + 1;
}
```

  - The values on the right side of the assignment operator are calculated first



Loops (again!) – contd.

- Revisiting the while loop
  - Enter a sequence of non-negative integers and find their number and sum

```c
#include <stdio.h>

void main(void)
{
    int sum = 0, ii = 0, num;
    int status = 1;

    printf("Enter a non-negative number: ");
    status = scanf("%d", &num);

    while( num >= 0 && status == 1)
    {
        sum = sum + num;
        printf("Enter a non-negative number: ");
        status = scanf("%d", &num);
        ii++;
    }

    printf("\nYou entered %d non-negative numbers,\n", ii);
    printf("their sum is %d.\n", sum);

}
```

Output 1

```
Enter a non-negative number: 5
Enter a non-negative number: 1
Enter a non-negative number: 10
Enter a non-negative number: -5

You entered 3 non-negative numbers,
Their sum is 16.
```

Output 2

```
Enter a non-negative number: 7
Enter a non-negative number: 3
Enter a non-negative number: 105
Enter a non-negative number: 25
Enter a non-negative number: A

You entered 4 non-negative numbers,
Their sum is 140.
```

do while Loops

- Revisiting the while loop
  - Enter a sequence of non-negative integers and find their number and sum

```c
#include <stdio.h>

void main(void)
{
    int sum = 0, ii = 0, num;
    int status = 1;

    printf("Enter a non-negative number: ");
    status = scanf("%d", &num);

    do
    {
        sum = sum + num;
        printf("Enter a non-negative number: ");
        status = scanf("%d", &num);
        ii++;
    } while( num >= 0 && status == 1);

    printf("\nYou entered %d non-negative numbers,\n", ii);
    printf("their sum is %d.\n", sum);

}
```

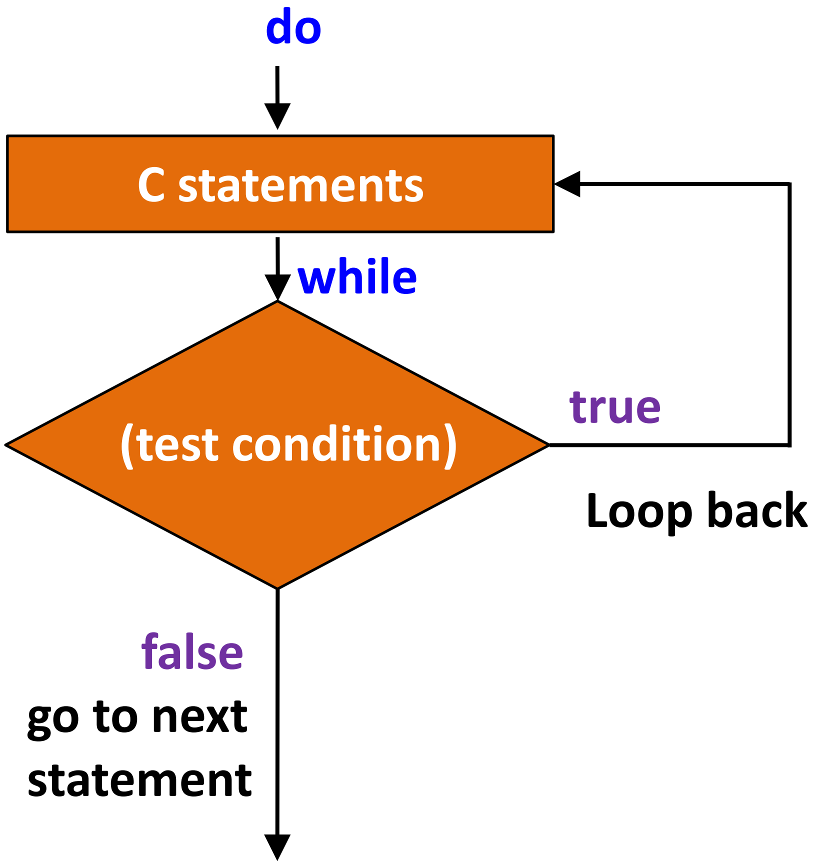
Output 1

```
Enter a non-negative number: 5
Enter a non-negative number: 1
Enter a non-negative number: 10
Enter a non-negative number: -5

You entered 3 non-negative numbers,
Their sum is 16.
```

Output 2

```
Enter a non-negative number: 7
Enter a non-negative number: 3
Enter a non-negative number: 105
Enter a non-negative number: 25
Enter a non-negative number: A

You entered 4 non-negative numbers,
Their sum is 140.
```

The `do while` loop



Loops (again!) – contd.

- Revisiting the while loop – analysis
  - status = scanf("%d", &num);
    - If num is entered as an integer, then status = 1
    - If num is entered as a non-integer value (character , symbol, or float), then status = 0
  - status = scanf("%f", &gpa);
    - If gpa is entered as a number (float or integer), status = 1
    - If gpa is entered as not numerical (i.e., character), status = 0
  - while(num >= 0)
    - If num >= 0, then the condition is true and the loop is executed
    - If num < 0, then the condition is false and the loop is not executed (control moves to the first statement after the loop)
  - while(num >=0 && status == 1)
    - If (num >= 0) **and** (status = 1), then the condition is true and the loop is executed
    - If (num < 0) **or** (status != 1), then the condition is false and the loop is not executed

Loops (again!) – contd.

- Logical and arithmetic operators

Logical operators

Operators	meaning
<, >	Less than, greater than
<=, >=	Less than or equal, greater than or equal
==	Equal
!=	Not equal

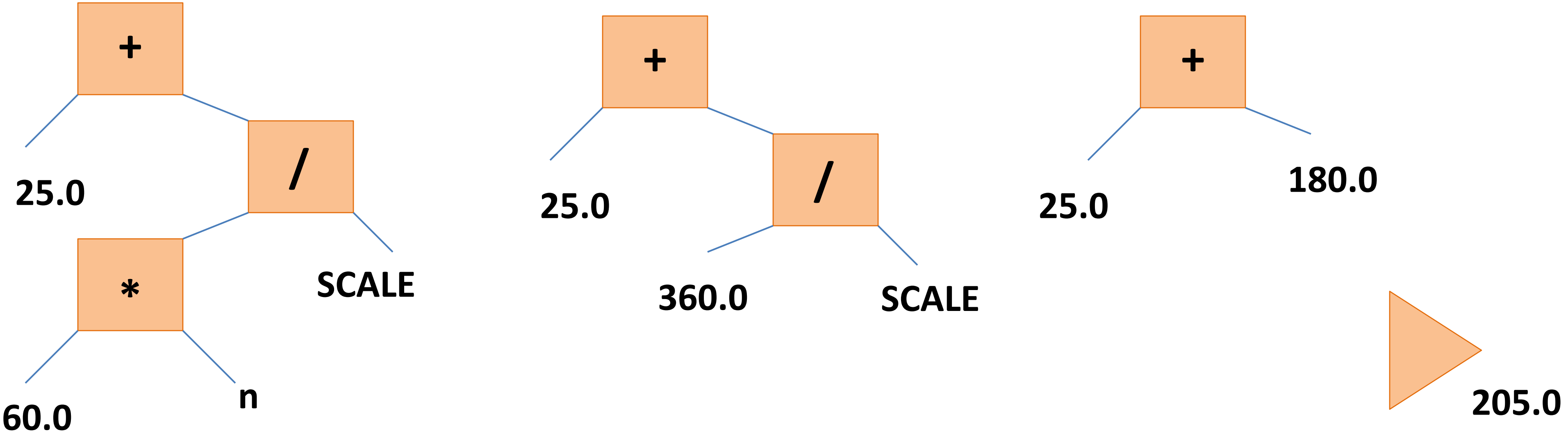
Arithmetic operators

Operators	meaning
=	Equal (assignment)
+, -, /, *	Addition, subtraction, multiplication, division
%	Modulus
+=, -=, *=, /=, %=	x += y → x = x + y

Operator Procedure

- Example

```
# include <stdio.h>

void main(void)
{
    double SCALE = 2, n = 6, butter = 0;

    butter = 25.0 + 60.0 * n / SCALE;
}
```



Exercise 1

- What does the following program do?

```
# include <stdio.h>

void main(void)
{
    int ii = 0;
    int out = 0;

   do {
        out = out + (ii * ii);
        ii = ii + 1;
    } while(ii <= 5);


    printf("The result is: %d\n", out);

}
```

Order of Evaluation

Operators in order of decreasing precedence	
Operators	**Associativity**
( )	Left to right
+ - (unary; numbers)	Right to left
* / %	Left to right
+ - (binary; 0 and 1)	Left to right
< <= >= >	Right to left
== !=	Left to right
&&	Left to right
||	Left to right
=	Right to left

Modulus Operator

- Modulus
  - It is the remainder of division
  - **Example:** 15 % 4 = 3 (reads, 15 modulo 4 equals 3)

```
% prints 1 if count is odd, prints 0 if count is even
# include <stdio.h>

void main(void)
{
    int count = 0;

    while(count <= 10)
    {
         printf("%d\n", count % 2);
          count++;    // count = count + 1;
    }
}
```

Output

```
0
1
0
1
0
1
0
1
0
1
0
```

- count++;
  - Is equivalent to “count = count + 1”;
- count--;
  - Is equivalent to “count = count – 1”;

Exercise 2

```c
#include <stdio.h>
int main (void)
{
    int number, right_digit;
    printf ("Enter your number.\n");
    scanf ("%i", &number);
    while ( number != 0 )
    {
        right_digit = number % 10;
        printf ("%i", right_digit);
        number = number / 10;
    }
    printf ("\n");
    return 0;
}
```

More on Increment and Decrement Operators: + -

- Examples (let us do it!)
  - count++
    - Add 1 to the value of count after performing the operation
  - ++count
    - Add 1 to the value of count before performing the operation

```
# include <stdio.h>
void main(void)
{
    int count, factor = 3, scale = 2;

    count = factor * scale;
    printf("Line 1: %d and %d\n", count, scale);

    count = factor * ++scale;
    printf("Line 2: %d and %d\n", count, scale);

    count = factor * --scale;
    printf("Line 3: %d and %d\n", count, scale);

    count = factor * scale++;
    printf("Line 4: %d and %d\n", count, scale);

}
```

Output

```
Line 1: 6 and 2
Line 2: 9 and 3
Line 3: 6 and 2
Line 4: 6 and 3
```

The Cast Operator

- Type conversion
  - In C, can we covert the type of data? Yes

```c
# include <stdio.h>

void main(void)
{
    double count = 0;

    count = 1.5 + 2.2;
    printf("Line 1: %.2f\n", count);

    count = (int) 1.5 + (int) 2.2;
    printf("Line 1: %.2f\n", count);

    count = (int) 1.5 + 2.2;
    printf("Line 1: %.2f\n", count);

    printf("Line 3: %d\n", (int) count);

}
```

Output

```
Line 1: 3.70
Line 2: 3.00
Line 3: 3.20
Line 4: 3
```

Exercise 3

This program should ask the user to enter two numbers through the keyboard. Then the program finds the value of one number raised to the power of another. $result = base^{power}$

```
1   #include <stdio.h>
2   int main(void)
3   {
4     int base;
5     int power
6    const int result = 1;
7    printf("Enter the base number \n");
8    scanf("%i", base);
9    printf("Enter the power \n");
10   scanf("%i", power);
11   int count = 1;
12   while (count <= power) {
13     result *= base;
14     ++count;
15  }
16   printf("Result: %hh" , result);
17   return 0;
18  }
```

1- Find the error lines and error details

2- Fix the program and make the program works correctly

Suggested Output

```
Enter the base number
2
Enter the power
5
Result: 32
```

Lecture Keywords

- Keywords
  - While loop
  - do While loop
  - Increment decrement
  - ++, --
  - Type conversion (type casting)

Lecture Summary

- Explain the while loop
- Introduce the increment and decrement operators
- Introduce operator procedure and priority
- Introduce the modulus operator
- Exercise activities & Keywords