Introduction to C Programming

—Basics of Programming (5) : Loops (2) —

Waseda University

Today's Topics

- Usage of a Looping statements
    - while, do while
    - break, continue
    - multiloop

Example 1 : The review of the for statement

The sum of 1-100

```
#include <stdio.h>
int main(void){
    int Sum = 0, i;
    printf("Sum=%d¥n",Sum);      /*Output for Sum*/
    for (i=1; i<=100; i=i+1){
       Sum = Sum+i ;                    /*Add i to Sum*/
       printf("+[%d] =%d ¥n",i,Sum); /*Checking the value of Sum*/
    }
    printf("Answer is %d.¥n", Sum);
    return 0;
}
```

while statement

The general form of the while statement

```
while (condition) {
    body-statement
}
```

- The program will repeatedly execute the statement inside the **while** until the *condition* becomes false(0).
- The **while** statement repeatedly executes the *body-statement* in {}.
- The program returns to the *condition*.

Example 2: while statement

The sum of 1-100.

```
#include <stdio.h>
int main(void){
    int Sum = 0, i;
    i=1;                        /* Need the initial value */
    while (i <= 100){           /* Repeat using the  while */
        Sum += i;               /* Need the iteration-statement */
        i++;
    }
    printf("Answer is %d.¥n", Sum);
    return 0;
}
```

- Type this program and run it.
- Let filename be "sum4.c".
- The **while** statement is the **for** statement without *initial-statement* and *iteration-statement*.
  You can also calculate the sum from 1 to 100 in this program.

do-while statement

The general form of the do-while statement

```
do {
     body-statement
} while(condition);
```

- The *body-statement* in {} is first executed.
- The program will repeatedly execute the *body-statement* inside the {} until the *condition* becomes false(0).

Example 3: do-while statement

The sum of 1-100.

```
#include <stdio.h>
int main(void){
    int Sum = 0, i = 1;
    do {
        Sum += i;           /* Add i to Sum */
        i++;
    } while(i <= 100);      /* Don't forget the semicolon */
    printf("Answer is %d.¥n", Sum);
    return 0;
}
```

- Type this program and run it.
- Let filename be “sum5.c”.
- The *body-statement* in {} will always execute at least once.
- If the *condition* in () after the **while** is true, the *body-statement* in {} is repeatedly executed.

Exercise 1

Exercise 1: Write the program to calculate the sum.

Write the program to calculate the sum:

$$S_N := 4 * \sum_{k=0}^{N} \frac{(-1)^k}{2k+1},$$

where the natural number N is a input number.

- Output to six decimal places (e.g. printf(“Sum is %.6f” ,SN)).
- Let filename be “piwhile.c”.
- Example:

  Input positive integer:1000【Enter】
  Sum is 3.14????.

break statement

Programming Example

```
for (i=1; i<=100; i++){
    if(i==5) break;
    printf("%d,",i);
}
```

- Loops and **switch** statement can be exited at any point through the use of a **break** statement.
- If multiloop, the break statement terminates the innermost execution of looping statements.
- This program displays as

  1,2,3,4,

continue statement

Programming Example

```
for (i=1; i<=100; i++){
    if(i==2) continue;
    if(i==5) break;
    printf("%d,",i);
}
```

- **continue** is a flow statement that cause the next execution of a loop to begin.
- This program displays as

  1,3,4,

Exercise 2

Exercise 2: Write the program to calculate the sum.

Let us define a sequence by

$$S_N := 4 * \sum_{k=0}^{N} \frac{(-1)^k}{2k+1}.$$

Write a program to display the minimum of N satisfying

$$|S_N - 3.1415| < error,$$

where $error$ is a input value.

- Let filename be "piwhile2.c".
- Example:

  Input error :0.01【Enter】
  N is 99.

Exercise Hints

- Let's copy "piwhile.c" and edit it.
- You can use infinite loop and **break** statement with **if** statement.

```
while(1){
    body-statement
    if (condition) break;
    body-statement
}
```

- Note that $|S_N - 3.1415| <$ error is equivalent to

$$-\text{error} < S_N - 3.1415 \; and \; S_N - 3.1415 < \text{error}.$$

Exercise 3

Exercise 2: Write the program to display the multiplication table

- Write the program to display the multiplication table.
- Display to the following:

```
1  2  3  4  5  6  7  8  9
2  4  6  8 10 12 14 16 18
3  6  9 12 15 18 21 24 27
4  8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

Exercise Hint

- You can use a double loop with two variables.

```
int i, j;
for (i=1;...;...){
    for (j=1;...;...){
        ....
    }
    ....
}
```

- You can use the **%3d** of **printf** conversions.
  For example, **printf("%3d",i);**
- First, write the program for output of one column. If you can write, write the program for output of two column.

  1 2 3 4 5 6 7 8 9

Summary

- Looping statements
  - while, do while
  - break, continue
  - multiloop