Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering

Chapter 5: Repetition Statements

Introduction to Computer Programming
(C language)

Nguyễn Tiến Thịnh, Ph.D.

Email: ntthinh@hcmut.edu.vn

2022 – 2023, Semester 1

Course Content

- C.1. Introduction to Computers and Programming
- C.2. C Program Structure and its Components
- C.3. Variables and Basic Data Types
- C.4. Selection Statements
- **C.5. Repetition Statements**
- C.6. Functions
- C.7. Arrays
- C.8. Pointers
- C.9. File Processing

References

- [1] "*C: How to Program*", 7th Ed. – Paul Deitel and Harvey Deitel, Prentice Hall, 2012.
- [2] "*The C Programming Language*", 2nd Ed. – Brian W. Kernighan and Dennis M. Ritchie, Prentice Hall, 1988
- and others, especially those on the Internet

Content

- Introduction
- while.. Statements
- do..while.. Statements
- for.. Statements
- Nested Repetition Statements
- continue Statements
- break Statements
- Summary

Introduction

- Control statements in C
  - Sequence
    - Assignment
    - Function calling
    - …
  - Selection
    - if
    - if..else..
    - switch..case..
  - Repetition
    - for..
    - while..
    - do..while..

Introduction

- Given a set of n positive numbers, find the smallest one.

(Chapter 1 – Real code in C)

```
void main() {
    double positiveNumber[10] = {2, 1, 3, 10, 8, 3, 4, 5, 9, 12};
    int n = 10;
    double minNumber = positiveNumber[0];
    int iteration = 1;
    while (iteration < n) {
        if (minNumber <= positiveNumber[iteration])
            iteration = iteration + 1;
        else {
            minNumber = positiveNumber[iteration];
            iteration = iteration + 1;
        }
    }
}
```

Control Statements for Repetition

Introduction

- A repetition statement allows you to specify that an action is to be repeated while some condition remains true.
- Example 1: Input validation
  - Ask a user to input a value
  - While his/her input value is invalid, ask him/her to input a value.
- Example 2: Search for any of you who didn't submit homework (i.e. no attachment exists)
- Example 3: Count the number of occurrences of an important keyword in a text

while.. Statements

while (<Condition>**)** <Statement>;

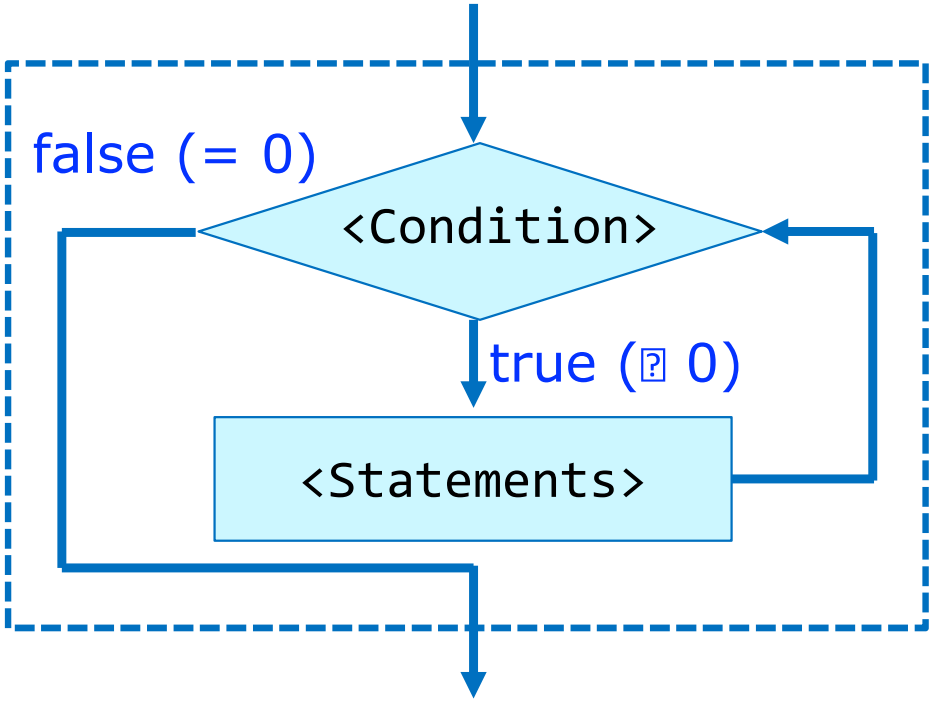
while (<Condition>**)** {

<Statement1>;

…

<Statementk>;

}

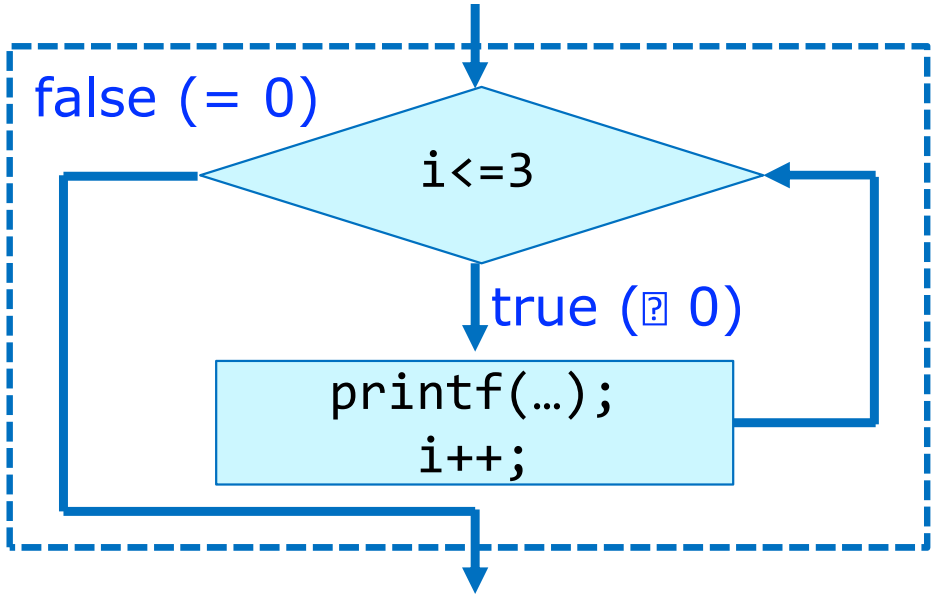


- 1. <Condition> is evaluated.
- 2. If <Condition> is true (0), <Statements> are performed and then go to step 3. Otherwise, i.e. if <Condition> is false (=0), go to step 4.
- 3. Go back to step 1.
- 4. End the repetition statement and move forward.

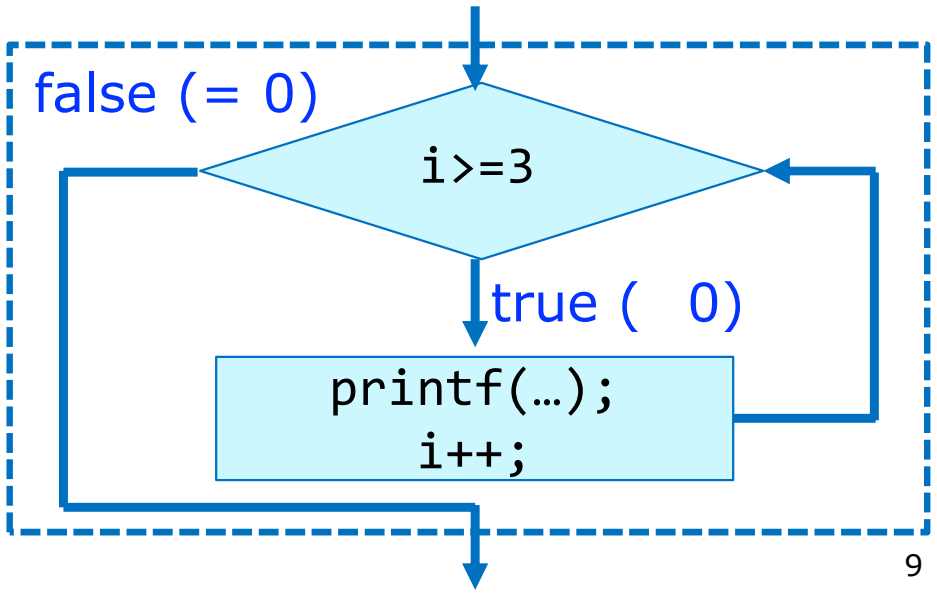
while.. Statements

```
int i=1;

while (i<=3) {

    printf("1.while: i = %d\n\n", i);

    i++;
}
```

```
1.while: i = 1

1.while: i = 2

1.while: i = 3
```




```
i=1;
while (i>=3) {

    printf("1.2.while: i = %d\n\n", i);

    i++;
}
```



while.. Statements

Write a program to receive a natural number from a user. If an invalid value is entered, ask the user to input again until a valid one is obtained.

```
Enter a natural number: N = a string

Enter a natural number: N = A

Enter a natural number: N = -23

Enter a natural number: N = 12

A valid natural number that has been input is: N = 12.
```

```c
#include <stdio.h>

int main() {

    int N = -1;

    while (N<0) {

        printf("\n\nEnter a natural number: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }

    printf("\n\nA valid natural number that has been input is: N = %d.\n", N);

    return 0;
}
```

while.. Statements

Given N, the number of the first natural numbers greater than 0, calculate and print the total *sum of these N* first natural numbers.

```c
#include <stdio.h>

void main() {

    int N = 0; //the number of the first natural numbers
    int Sum = 0; //the total sum of n first natural numbers = 1 + 2 + .. + (n-1) + n = sum_(n-1) + n

    while (N<=0) {

        printf("\n\nEnter a natural number: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }

    int i=1;

    while (i<=N) {

        Sum += i;

        i++;
    }

    //check if your while statement is correct
    if (Sum == N*(N+1)/2) printf("\n\nSum of %d first natural numbers = %d is correct!", N, Sum);
}
```

```
Enter a natural number: N = -2

Enter a natural number: N = 0.3

Enter a natural number: N = 5

Sum of 5 first natural numbers = 15 is correct!
```

while.. Statements

Given N, a natural number greater than 0, calculate and print the *factorial of N*: N! = 1*2*..*N = (N-1)!*N

```
Enter a natural number: N = 5

The factorial of 5 = 120.
```

```c
#include <stdio.h>

void main() {

    int N = 0; //the number of the first natural numbers
    int Fact = 1; //the factorial of N = 1 * 2 * .. * (N-1) * N = Fact_(N-1) * N

    while (N<=0) {

        printf("\n\nEnter a natural number: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }

    int i=1;

    while (i<=N) {

        Fact *= i;

        i++;
    }

    printf("\n\nThe factorial of %d = %d.\n", N, Fact);
}
```

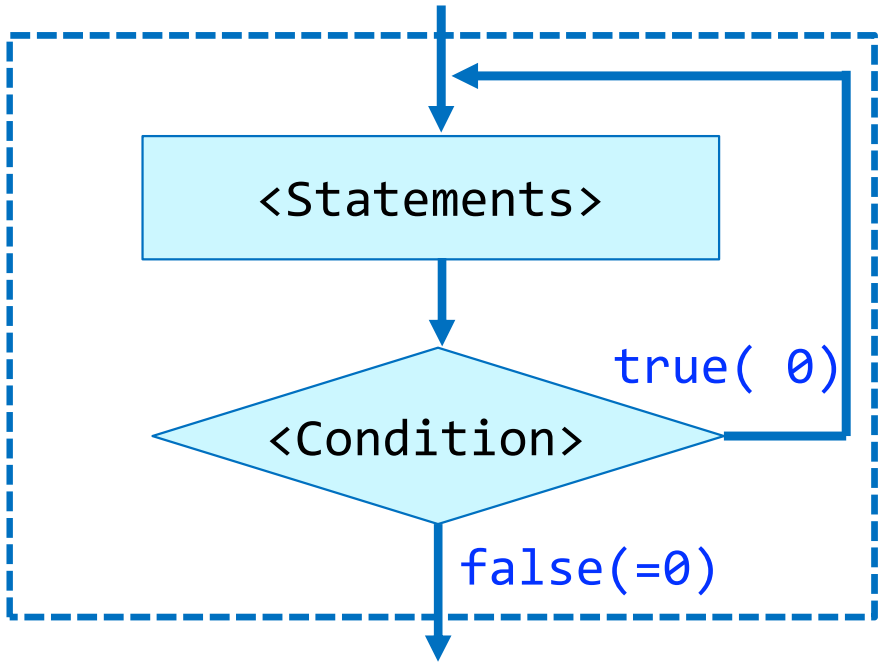
Given a natural number N greater than 0, list a *series of Fibonacci numbers* smaller than N.

```c
#include <stdio.h>

void main() {

    int N = 0, n_1F, n_2F;

    while (N<=0) {
        printf("\n\nEnter a positive integer number: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }

    printf("\n\nA series of Fibonacci numbers smaller than %d is listed as follows.\n\n", N);

    n_1F = 1; //F(0)
    n_2F = 1; //F(1)

    if (N==2) printf("%d \t %d\n", n_1F, n_2F); //F(0)  F(1)
    else {
        printf("%d \t %d \t", n_1F, n_2F); //F(0) F(1)

        int temporary = n_1F + n_2F;
        while (temporary < N) {
            printf("%d \t", temporary); //F(n), n>1 and F(n)<N
            n_1F = n_2F;
            n_2F = temporary;
            temporary = n_1F + n_2F;
        }
    }
}
```

```
Enter a positive integer number: N = 21

A series of Fibonacci numbers smaller than 21 is listed as follows.

1       1       2       3       5       8       13
```

do..while.. Statements

```
do {
        <Statement1>;
        …
        <Statementk>;
}
while (<Condition>);
```



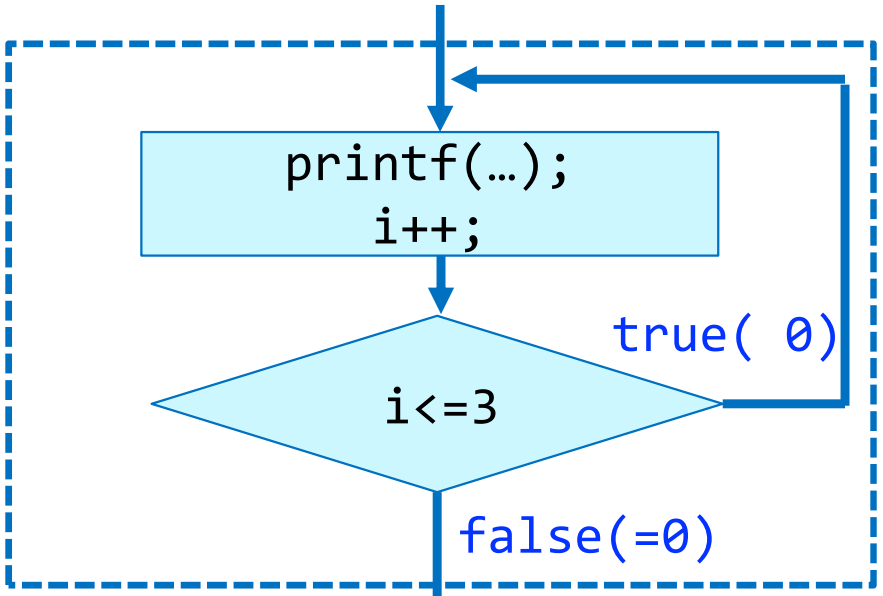
- 1. <Statements> are performed.
- 2. <Condition> is evaluated.
- 3. If <Condition> is true (≠0), go to step 1. Otherwise, i.e. if <Condition> is false (=0), go to step 4.
- 4. End the repetition statement and move forward.

<Statements> are always performed at least once!

do..while.. Statements

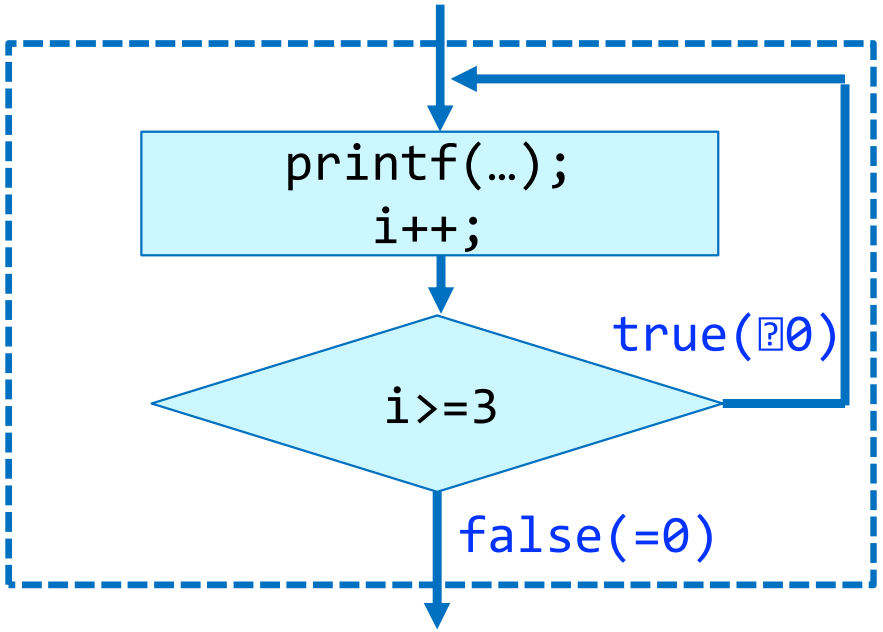
```
i=1;
do {
    printf("2.do..while: i = %d\n\n", i);

    i++;
}
while (i<=3);
```

```
2.do..while: i = 1

2.do..while: i = 2

2.do..while: i = 3
```




```
i=1;
do {
    printf("2.2.do..while: i = %d\n\n", i);

    i++;
}
while (i>=3);
```

```
2.2.do..while: i = 1
```



do..while.. Statements

Write a program to receive a natural number from a user. If an invalid value is entered, ask the user to input again until a valid one is obtained.

```
Enter a natural number: N = a string, not a number

Enter a natural number: N = A

Enter a natural number: N = -1234

Enter a natural number: N = 12

A valid natural number that has been input is: N = 12.
```

```c
#include <stdio.h>

void main() {

    int N = -1;

    do {

        printf("\n\nEnter a natural number: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }
    while (N<0);

    printf("\n\nA valid natural number that has been input is: N = %d.\n", N);
}
```

do..while.. Statements

Given N, the number of the first natural numbers greater than 0, calculate and print the total *sum of these N* first natural numbers.

```c
#include <stdio.h>

void main() {

    int N = 0; //the number of the first natural numbers
    int Sum = 0; //the total sum of n first natural numbers = 1 + 2 + .. + (n-1) + n = sum_(n-1) + n

    while (N<=0) {

        printf("\n\nEnter a natural number: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }

    int i=1;

    do {
        Sum += i;

        i++;
    }
    while (i<=N);

    //check if your do..while statement is correct
    if (Sum == N*(N+1)/2) printf("\n\nSum of %d first natural numbers = %d is correct!", N, Sum);
}
```

```
Enter a natural number: N = 5

Sum of 5 first natural numbers = 15 is correct!
```

do..while.. Statements

Given N, a natural number greater than 0, calculate and print the *factorial of N*: N! = 1*2*..*N = (N-1)!*N

```
Enter a natural number: N = 5

The factorial of 5 = 120.
```

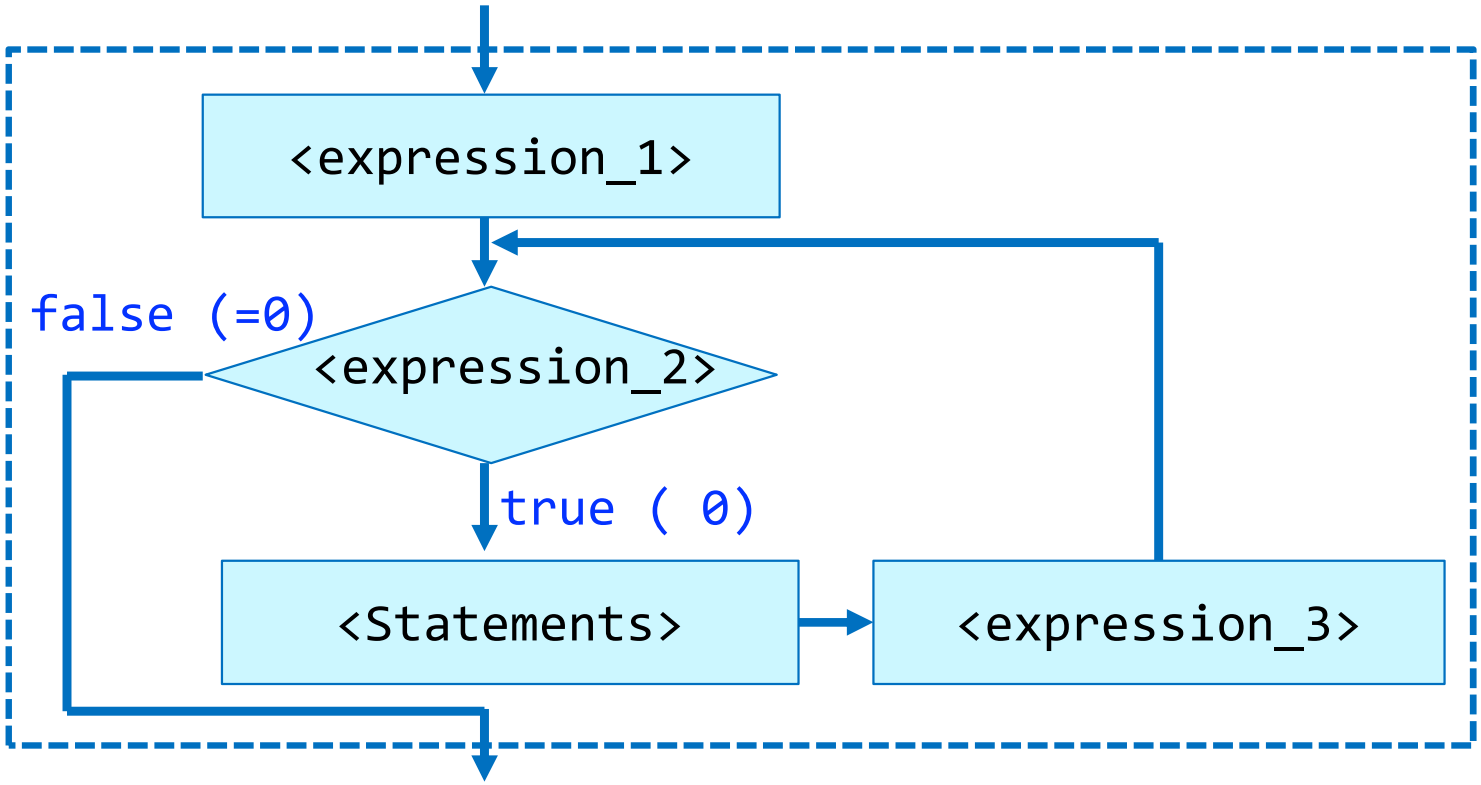
```c
#include <stdio.h>

void main() {

    int N = 0; //the number of the first natural numbers
    int Fact = 1; //the factorial of N = 1 * 2 * .. * (N-1) * N = Fact_(N-1) * N

    while (N<=0) {

        printf("\n\nEnter a natural number: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }

    int i=1;

    do {

        Fact *= i;

        i++;
    }
    while (i<=N);

    printf("\n\nThe factorial of %d = %d.\n", N, Fact);
}
```

Given a sequence of N numbers input sequentially by a user, find the minimum one.

```c
#include <stdio.h>
#include <float.h>

void main() {

    int N = 0; //the number of the input numbers in c

    float currentNumber; //the current number that hc

    float minNumber = FLT_MAX;  //the minimum number

    while (N<=0) {

        printf("\n\nEnter a natural number: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }

    int count = 1; //the number of the input numbers that have been processed so far

    do {
        printf("\n\nEnter the %d-th number: ", count);
        scanf("%f", &currentNumber);

        fflush(stdin);

        if (minNumber > currentNumber) {
            minNumber = currentNumber;
            printf("\n\nThe minimum number that has been input so far is: %f", minNumber);
        }

        count++;
    }
    while (count<=N);

    printf("\n\nThe minimum number that has been input so far is finalized as: %f", minNumber);
}
```

```
Enter a natural number: N = 3

Enter the 1-th number: 1.3

The minimum number that has been input so far is: 1.300000
Enter the 2-th number: 0.92

The minimum number that has been input so far is: 0.920000
Enter the 3-th number: .8

The minimum number that has been input so far is: 0.800000
The minimum number that has been input so far is finalized as: 0.800000
```

for.. Statements



for (<expression_1>**;** <expression_2>**;** <expression_3>**)** <Statement>**;**

for (<expression_1>**;** <expression_2>**;** <expression_3>**)** {

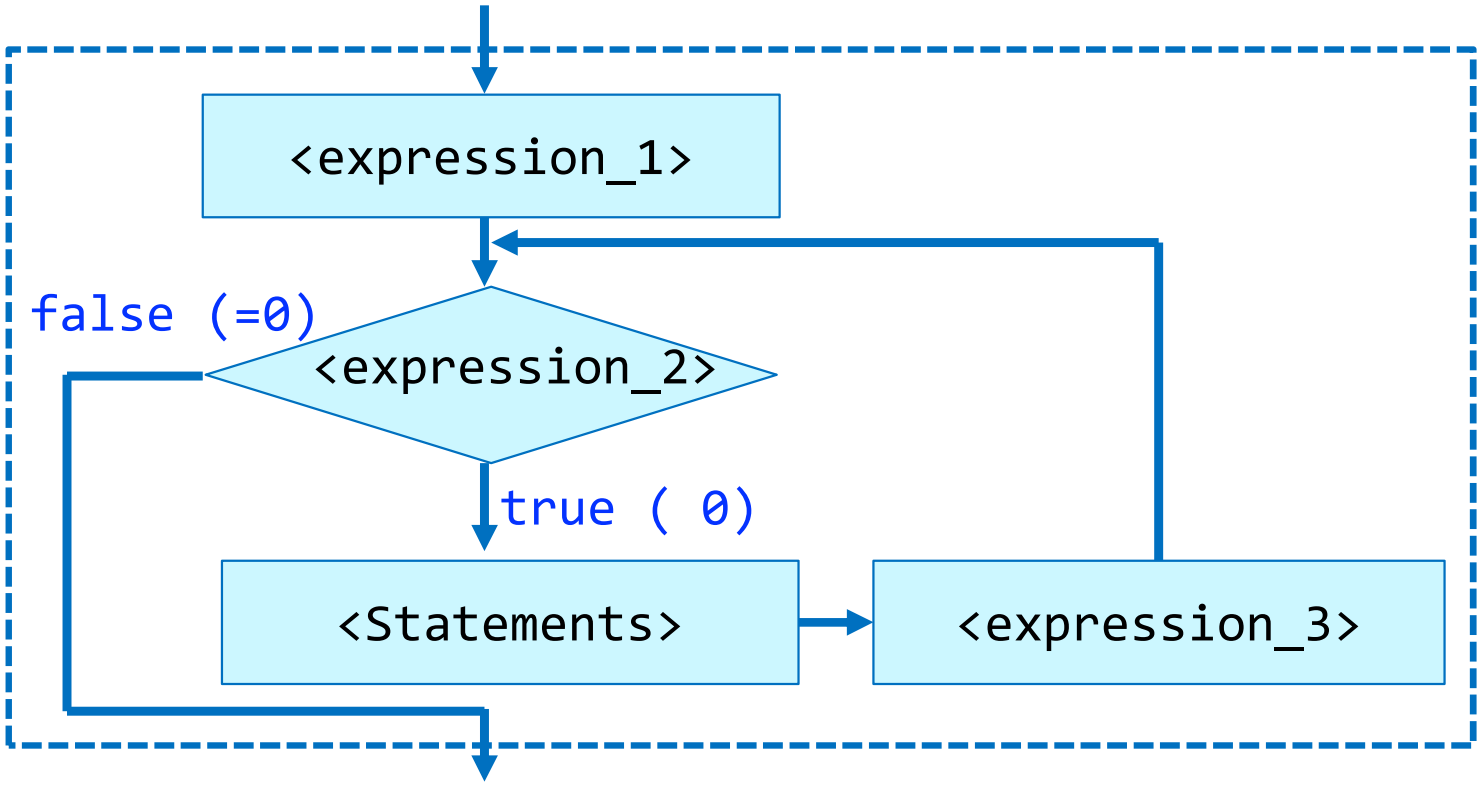
<Statement1>;

…

<Statementk>;

}

for.. Statements



1. <expression_1> is evaluated.
2. <expression_2> is evaluated.
3. If <expression_2> is true (≠0), <Statements> are performed, <expression_3> is evaluated, and then go to step 2. Otherwise, i.e. if <expression_2> is false (=0), go to step 4.
4. End the repetition statement and move forward.

for.. Statements

for (<expression_1>**;** <expression_2>**;** <expression_3>**)** <Statement>**;**

is regarded as:

```
<expression_1>;
while (<expression_2>) {
        <Statement>;
        <expression_3>;
}
```

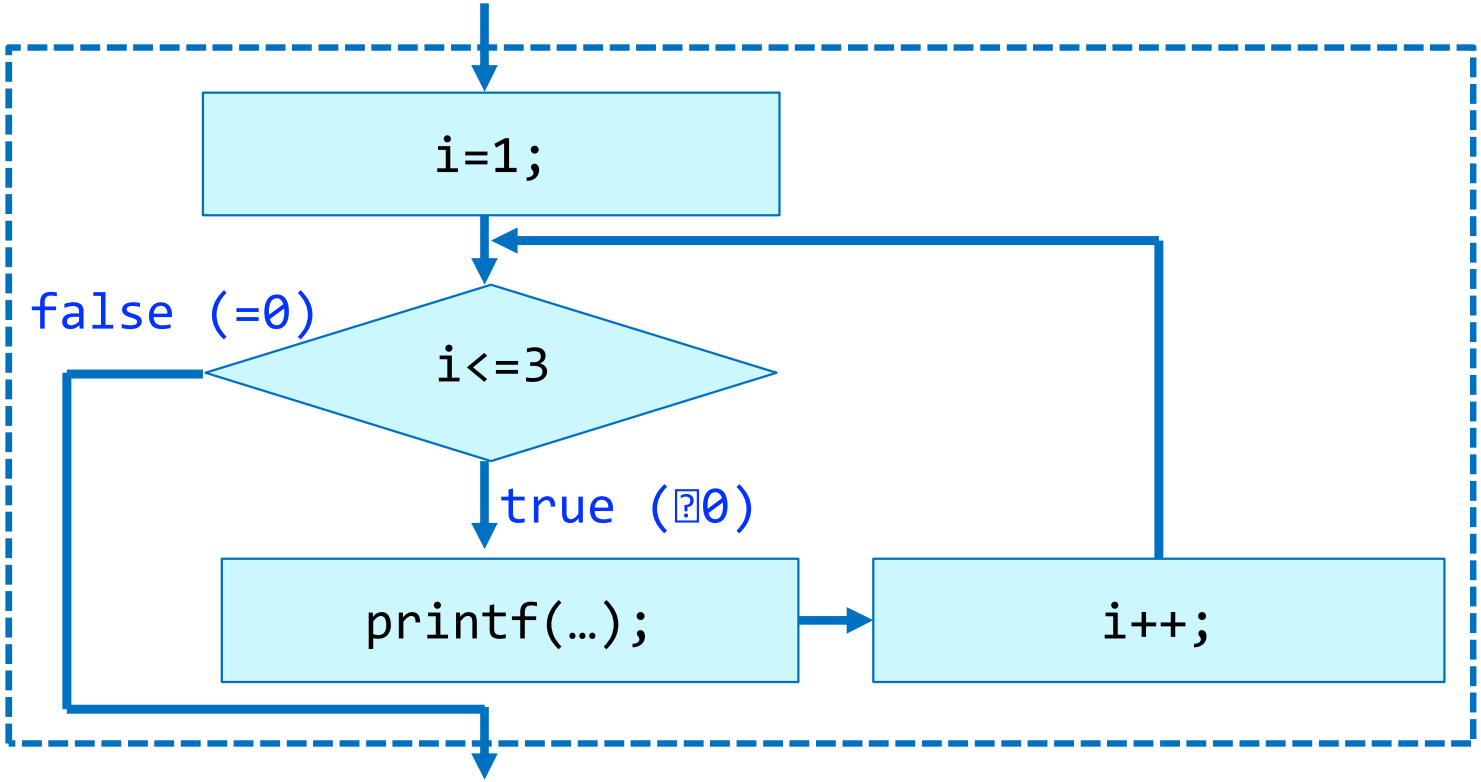
for (<expression_1>**;** <expression_2>**;** <expression_3>**)** {

```
    <Statement1>;
    …
    <Statementk>;
}
```

is regarded as:

```
<expression_1>;
while (<expression_2>) {
        <Statement1>;
        …
        <Statementk>;
        <expression_3>;
}
```

for.. Statements




```
for(i=1; i<=3; i++) printf("3.for: i = %d\n\n", i);
```

```
3.for: i = 1

3.for: i = 2

3.for: i = 3
```

```
for(i=1; i>=3; i++) printf("3.2.for: i = %d\n\n", i);
```

No printing result exists due to a false value of the expression “i>=3”.

for.. Statements

Given N, the number of the first natural numbers greater than 0, calculate and print the total *sum of these N* first natural numbers.

```
#include <stdio.h>

void main() {

    int N = 0; //the number of the first natural numbers
    int Sum = 0; //the total sum of n first natural numbers = 1 + 2 + .. + (n-1) + n = sum_(n-1) + n

    while (N<=0) {

        printf("\n\nEnter a natural number: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }

    int i;

    for (i=1; i<=N; i++) Sum += i;

    //check if your do..while statement is correct
    if (Sum == N*(N+1)/2) printf("\n\nSum of %d first natural numbers = %d is correct!", N, Sum);
}
```

```
Enter a natural number: N = 5

Sum of 5 first natural numbers = 15 is correct!
```

for.. Statements

Given N, a natural number greater than 0, calculate and print the *factorial of N*: N! = 1*2*..*N = (N-1)!*N

```
Enter a natural number: N = 5

The factorial of 5 = 120.
```

```c
#include <stdio.h>

void main() {

    int N = 0; //the number of the first natural numbers
    int Fact = 1; //the factorial of N = 1 * 2 * .. * (N-1) * N = Fact_(N-1) * N

    while (N<=0) {

        printf("\n\nEnter a natural number: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }

    int i;

    for (i=1; i<=N; i++) Fact *= i;

    printf("\n\nThe factorial of %d = %d.\n", N, Fact);
}
```

for.. Statements

Given a natural number N greater than 0, list all the *squared numbers* smaller than the given number.

```c
//Chapter 5 - while.. and for.. statements
//Squared numbers smaller than N which is input by a user

#include <stdio.h>

void main() {

    int N = 0, i;

    do {
        printf("\n\nEnter a positive integer number: N = ");
        scanf("%d", &N);
        fflush(stdin);
    }
    while (N<=0);

    printf("\n\nAll the squared numbers smaller than %d are printed as follows.\n\n", N);

    for (i=1; i*i<N; i++) printf("%d \t", i*i);

}
```

```
Enter a positive integer number: N = 27

All the squared numbers smaller than 27 are printed as follows.

1       4       9       16      25
```

for.. Statements

```c
//Chapter 5 - while.. and for.. statements
//Squared numbers smaller than N which is input by a user

#include <stdio.h>

void main() {

    int N = 0, i;

    do {
        printf("\n\nEnter a positive integer number: N = ");
        scanf("%d", &N);
        fflush(stdin);
    }
    while (N<=0);

    printf("\n\nAll the squared numbers smaller than %d are printed as follows.\n\n", N);

    i = 1;

    while (i*i<N) {

        printf("%d \t", i*i);

        i++;
    }
}
```

```c
for (i=1; i*i<N; i++) printf("%d \t", i*i);
```

```
Enter a positive integer number: N = 27

All the squared numbers smaller than 27 are printed as follows.

1       4       9       16      25
```

Nested Repetition Statements

```
while (<expression_1>) {
        …
        do {
                …
        }
        while (<expression_2>);
        …
        for(<expression_3>;<expression_4>;<expression_5>) {
                …
        }
        …
}
```

```
for (…; …; …) {
        …
        for (…; …; …) {
                …
        }
        …
        while (…) {
                …
        }
        …
}
…
```

A repetition statement can contain other repetition statements in many various combination ways.

Nested Repetition Statements

Given a size of a window (N is an odd number and N>=5),
print a star of stars: diagonal and center lines

```
//given a size of a window (odd number and >=5), print a star of stars: diagonal and center lines

#include <stdio.h>

void main() {

    int N; //size of a window

    do {
        printf("\n\nEnter a positive integer number: N = ");
        scanf("%d", &N);
        fflush(stdin);
    }
    while (N<5 || N%2 == 0);

    int i; //row index
    int j; //column index

    for (i=1; i<=N; i++) {
        for (j=1; j<=N; j++) {
            if (i == j || j == N-i+1 || i == N/2+1 || j == N/2+1) printf("*");
            else printf(" ");
        }
        printf("\n");
    }
}
```

```
Enter a positive integer number: N = 7

Your input number N = 7 is valid!

*   *   *
 * * *
  ***
*******
  ***
 * * *
*   *   *
```

*		*		*
	*	*	*	
*	*	*	*	*
	*	*	*	
*		*		*

Nested Repetition Statements

Given a natural number N greater than 0, print a triangle full of stars. N is the height of the star triangle.

```c
#include <stdio.h>

void main() {

    int N; //height of a star triangle

    do {
        printf("\n\nEnter a natural number greater than 0: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }
    while (N<=0);

    int i; //row index

    for (i=1; i<=N; i++) {

        int j; //column index

        for (j=1; j<=N-i; j++) printf(" ");

        for (j=1; j<=2*i-1; j++) printf("*");

        printf("\n");
    }
}
```

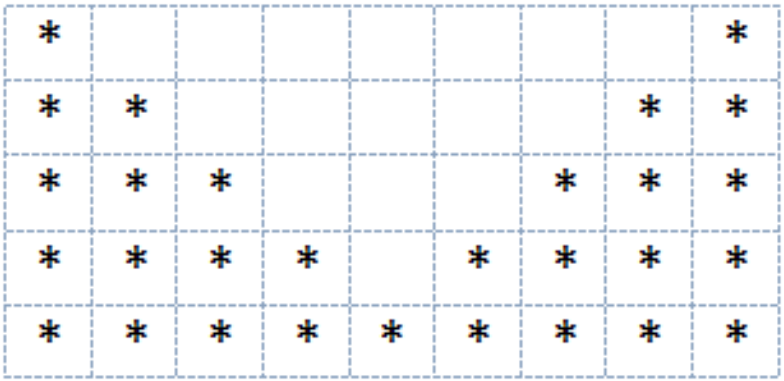
```
Enter a natural number greater than 0: N = 4
   *
  ***
 *****
*******
```

			*			
		*	*	*		
	*	*	*	*	*	
*	*	*	*	*	*	*

N = 4

Given a natural number greater than 0, print two squared isosceles triangles of stars.

```c
//Chapter 5 - repetition statements
//Two opposite triangles

#include <stdio.h>

void main() {
    int N; //height

    do {
        printf("\n\nEnter a natural number greater than 0: N = ");
        scanf("%d", &N);

        fflush(stdin);
    }
    while (N<=0);

    int i; //row index
    for (i=1; i<=N; i++) {

        int j; //column index for the 1st triangle
        for (j=1; j<=i; j++) printf("*");

        int k; //column index for the 2nd triangle
        for (k=1; k<=2*N-2*i-1; k++) printf(" ");
        for (k=1; k<=i; k++)
            if (k<N) printf("*");

        printf("\n");
    }
}
```



N=5

```
Enter a natural number greater than 0: N = 5
*         *
**       **
***     ***
****   ****
*********
```

continue Statements

- the ***continue*** statement for skipping the remainder of the body of a repetition statement and proceeding with the next iteration of the loop
  - while.. statements
  - do.. while.. statements
  - for.. statements

continue Statements

```c
int i=1;

while (i<=3) {

    if (i==2) {
        i++;
        continue;
    }

    printf("1.while: i = %d\n\n", i);

    i++;
}

i=1;
do {
    if (i==2) {
        i++;
        continue;
    }

    printf("2.do..while: i = %d\n\n", i);

    i++;
}
while (i<=3);

for(i=1; i<=3; i++) {

    if (i==2) continue;

    printf("3.for: i = %d\n\n", i);
}
```

```
1.while: i = 1

1.while: i = 3

2.do..while: i = 1

2.do..while: i = 3

3.for: i = 1

3.for: i = 3
```

The second loop has been skipped!

break Statements

- the ***break*** statement for exiting immediately from certain control statements
  - switch..case statements
  - while.. statements
  - do.. while.. statements
  - for.. statements

break Statements

```
int i=1;

while (i<=3) {

    if (i==2) {
        i++;
        break;
    }

    printf("1.while: i = %d\n\n", i);

    i++;
}

i=1;
do {
    if (i==2) {
        i++;
        break;
    }

    printf("2.do..while: i = %d\n\n", i);

    i++;
}
while (i<=3);

for(i=1; i<=3; i++) {

    if (i==2) break;

    printf("3.for: i = %d\n\n", i);
}
```

```
1.while: i = 1

2.do..while: i = 1

3.for: i = 1
```

All the loops from the second loop have been skipped!

A corresponding repetition statement is then ended.

Infinite Loops

```
while (1) <Statement>;

while (1) {
        <Statement1>;
        …
        <Statementk>;
}
```

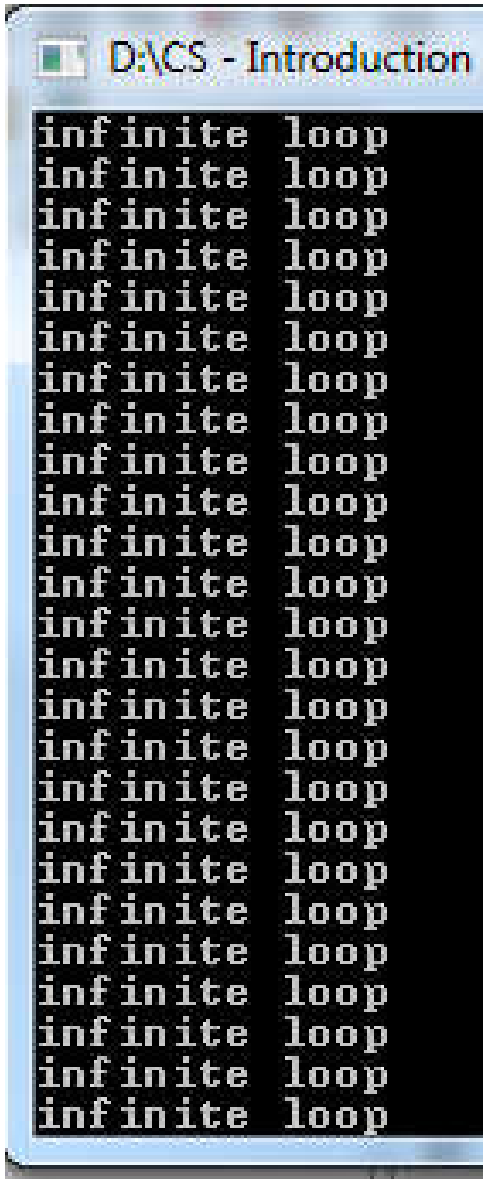
```
for (; ;) <Statement>;

for (; ;) {
        <Statement1>;
        …
        <Statementk>;
}
```

```
do {
        <Statement1>;
        …
        <Statementk>;
}
while (1);
```

```
for (<expression_1>; 1;<expression_3>)
        <Statement>;

for (<expression_1>; 1;<expression_3>)
{       <Statement1>;
        …
        <Statementk>;
}
```

Infinite Loops

```
i=1;
while (i>=0) {
    printf("infinite loop\n");
    i++;
}

i=1;
while (i=1) {
    printf("infinite loop\n");
    i++;
}

i=1;
while (1) {
    printf("infinite loop\n");
    i++;
}

for (i=1; 1; i++) printf("infinite loop\n");

for (;;) printf("infinite loop\n");
```



Put them all together

```c
//approximation for e

#include <stdio.h>
#include <math.h>

#define e 2.71828

void main() {

    int N = -1, NFact = 1;
    float e_approx = 1;

    printf("\n\nEnter a natural number: N = ");
    scanf("%d", &N);

    if (N != 0) {

        short count = 1;

        while (count<=N) {

            NFact *= count;

            e_approx += 1.0/NFact;

            count++;
        }
    }

    printf("\ne = %f\n\n", e);

    printf("Approximation of e = %f\n\n", e_approx);

    printf("Difference = %f - %f = %f\n\n", e, e_approx, e - e_approx);
}
```

Write a program to compute an approximation of **e** with a positive number N input by a user. Print the approximated value and its difference from a commonly used value which is 2.71828. It is given that: 0! = 1! = 1.

Natural number **e** is approximated:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} \approx 2.71828$$

```
Enter a natural number: N = 10

e = 2.718280

Approximation of e = 2.718282

Difference = 2.718280 - 2.718282 = -0.000002
```

Put them all together

```c
//approximation for e^x

#include <stdio.h>
#include <math.h>

#define e 2.71828

void main() {

    int N = -1, NFact = 1;
    float x = 0, xPower = 1, e_x = 1;

    printf("\n\nEnter a natural number: N = ");
    scanf("%d", &N);

    printf("\nEnter a floating-point number: x = ");
    scanf("%f", &x);

    if (N != 0 && x != 0) {

        short count = 1;

        while (count<=N) {

            xPower *= x;

            NFact *= count;

            e_x += xPower/NFact;

            count++;
        }
    }

    printf("\ne^x = %f\n\n", pow(e, x));

    printf("Approximation of e^x = %f\n\n", e_x);

    printf("Difference = %f - %f = %f\n\n", pow(e, x), e_x, pow(e, x) - e_x);
}
```

Write a program to compute an approximation of the power *x* of *e* where there is no use of a true value of *e*. Accuracy of an approximation is dependent on how large *n* is.

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

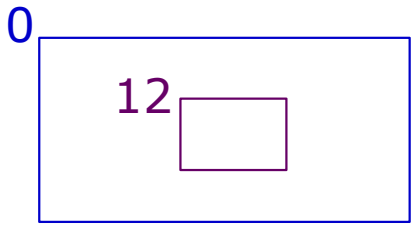
```
Enter a natural number: N = 6
Enter a floating-point number: x = 1.5
e^x = 4.481685
Approximation of e^x = 4.477539
Difference = 4.481685 - 4.477539 = 0.004145
```

Put them all together

- Given problem: print the first NxN natural numbers greater than 0 in a spiral-shaped matrix with a given N.

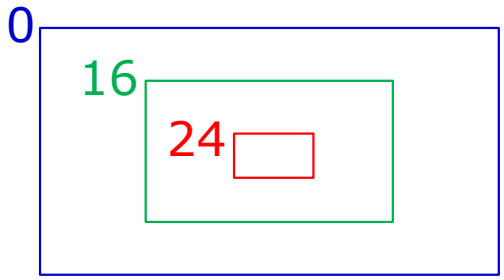
Given N = 4, a printed spiral-shaped matrix is as follows:

```
 1   2   3   4
12  13  14   5
11  16  15   6
10   9   8   7
```



Given N = 5, a printed spiral-shaped matrix is as follows:

```
 1   2   3   4   5
16  17  18  19   6
15  24  25  20   7
14  23  22  21   8
13  12  11  10   9
```



```c
#include <stdio.h>

void main() {
    int i, j; //the row and column indices, respectively
    int N=0; //the size of the matrix

    while (N<=0) {
        printf("\nEnter a size of the spiral matrix: N = ");
        scanf("%d", &N);

        fflush(stdin); //flush the buffer of the standard input stream
    }

    //scan the matrix to fill a value in each cell
    for(i=1; i<=N; i++) {
        for (j=1; j<=N; j++) {
            int newN;             //the size of the matrix corresponding to the circle k
            int k;                //the circle k
            int previousSum = 0; //the maximum number of the previous circle (k-1)
            int aValue;           //the value of the current number in the current cell

            //identify the circle k of the current number
            for (k=1; k<=(N%2?N/2+1:N/2); k++)
                if (i==k || i==N-k+1 || j==k || j==N-k+1) {
                    newN = N-k*2+2;
                    break;
                }

            //calculate previousSum
            int k1; //circles
            for (k1=2; k1<=k; k1++) previousSum += 4*(N - 2*(k1-2)) - 4;

            //calculate the value
            //top
            if (i==k) aValue = previousSum + j - k + 1;
            //bottom
            else if (i==N-k+1) aValue = previousSum + newN + newN - 2 + newN - (j - k);
            //left
            else if (j==k) aValue = previousSum + newN + newN - 2 + newN + newN - (i-k) - 1;
            //right
            else if (j==N-k+1) aValue = previousSum + newN + i - k;

            //print the value
            printf("%3d ", aValue);
        }
        printf("\n");
    }
}
```

```
  1   2   3   4   5   6   7   8   9  10
 36  37  38  39  40  41  42  43  44  11
 35  64  65  66  67  68  69  70  45  12
 34  63  84  85  86  87  88  71  46  13
 33  62  83  96  97  98  89  72  47  14
 32  61  82  95 100  99  90  73  48  15
 31  60  81  94  93  92  91  74  49  16
 30  59  80  79  78  77  76  75  50  17
 29  58  57  56  55  54  53  52  51  18
 28  27  26  25  24  23  22  21  20  19
```

N=10

```
  1   2   3   4   5   6   7   8   9
 32  33  34  35  36  37  38  39  10
 31  56  57  58  59  60  61  40  11
 30  55  72  73  74  75  62  41  12
 29  54  71  80  81  76  63  42  13
 28  53  70  79  78  77  64  43  14
 27  52  69  68  67  66  65  44  15
 26  51  50  49  48  47  46  45  16
 25  24  23  22  21  20  19  18  17
```

N=9

How to start the spiral at any given position (i,j)?

Summary

- Three control statement types of repetition
  - **while**..
  - **do**.. **while**..
  - **for**...

  → Repeat actions in connection with conditions
- **break** and **continue** statements
  - Important and helpful for controlling the loops in repetition
- Infinite loops
  - Check the conditions for repetition carefully

Chapter 5: Repetition Statements