Object oriented programming In C++

Control Structure (1)

Professor 최학남

xncui@vision.inha.ac.kr

Office: high-tech 401

How to compute summation from 1 to 10 ?

- int n1 = 1;
- int n2 = 2;

- int n10 = 10;

- int nsum = n1 + n2 + … + n10;

How to compute summation from 1 to 1000 ?

- int n1 = 1;
- int n2 = 2;
- int n1000 = 1000;
- int nsum = n1 + n2 + … + n1000;

Contents

Control statement

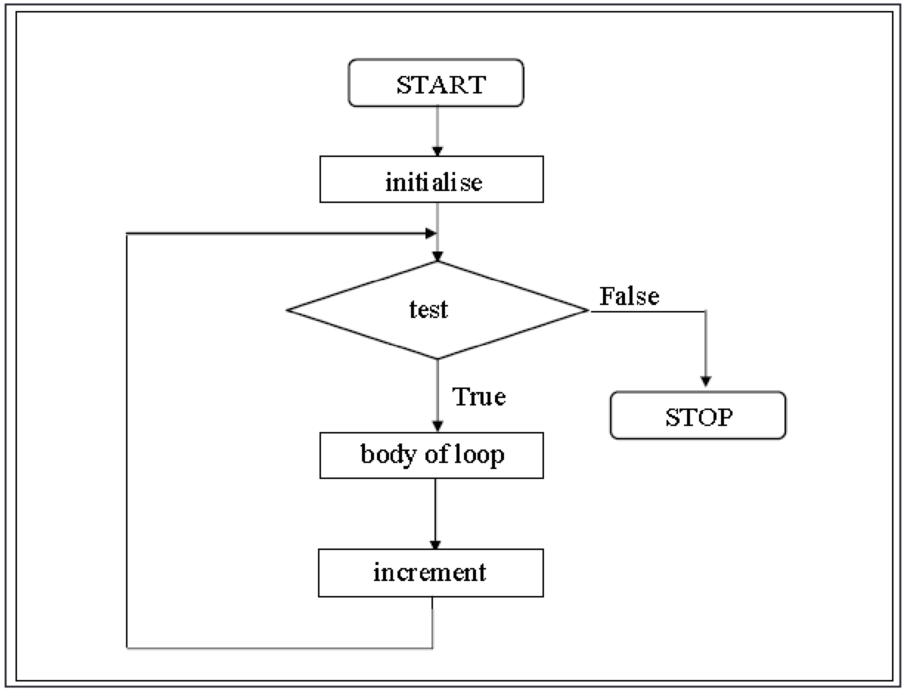
- Loop control statement
  - ✓ *while*
  - ✓ *for*
  - ✓ *do-while*

- Conditional control statement
  - ✓ *if or if-else*
  - ✓ *switch*

while statement

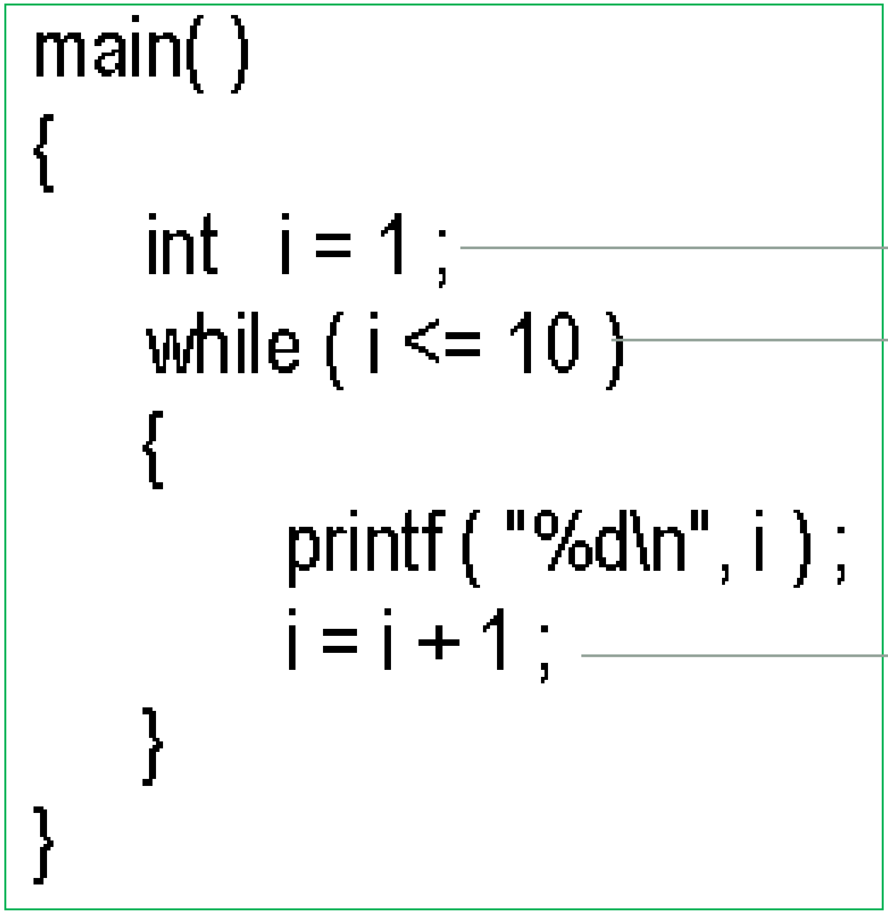
- The general form of *while* is as shown below:

```
initialise loop counter ;
while ( test loop counter using a condition )
{
        do this ;
        and this ;
        increment loop counter ;
}
```



concept of execution of the *while* statement

while statement



```
main( )
{
      int   i = 1 ;
      while ( i <= 10 )
      {
            printf ( "%d\n", i ) ;
            i = i + 1 ;
      }
}
```

Initialize loop counter

Test loop counter using condition

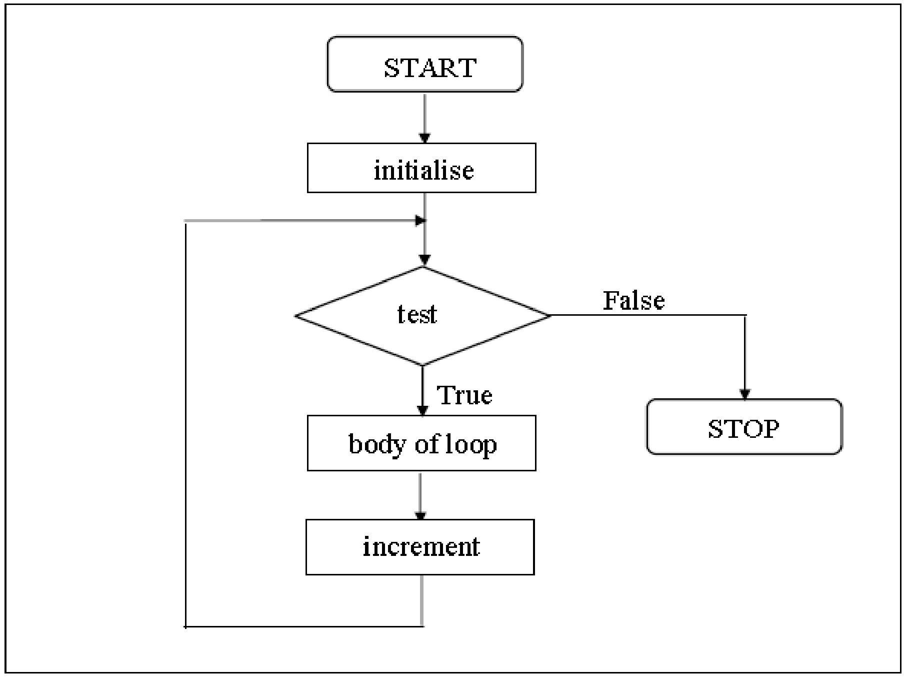
Increment loop counter

for statement

- The general form of *for* is as shown below:

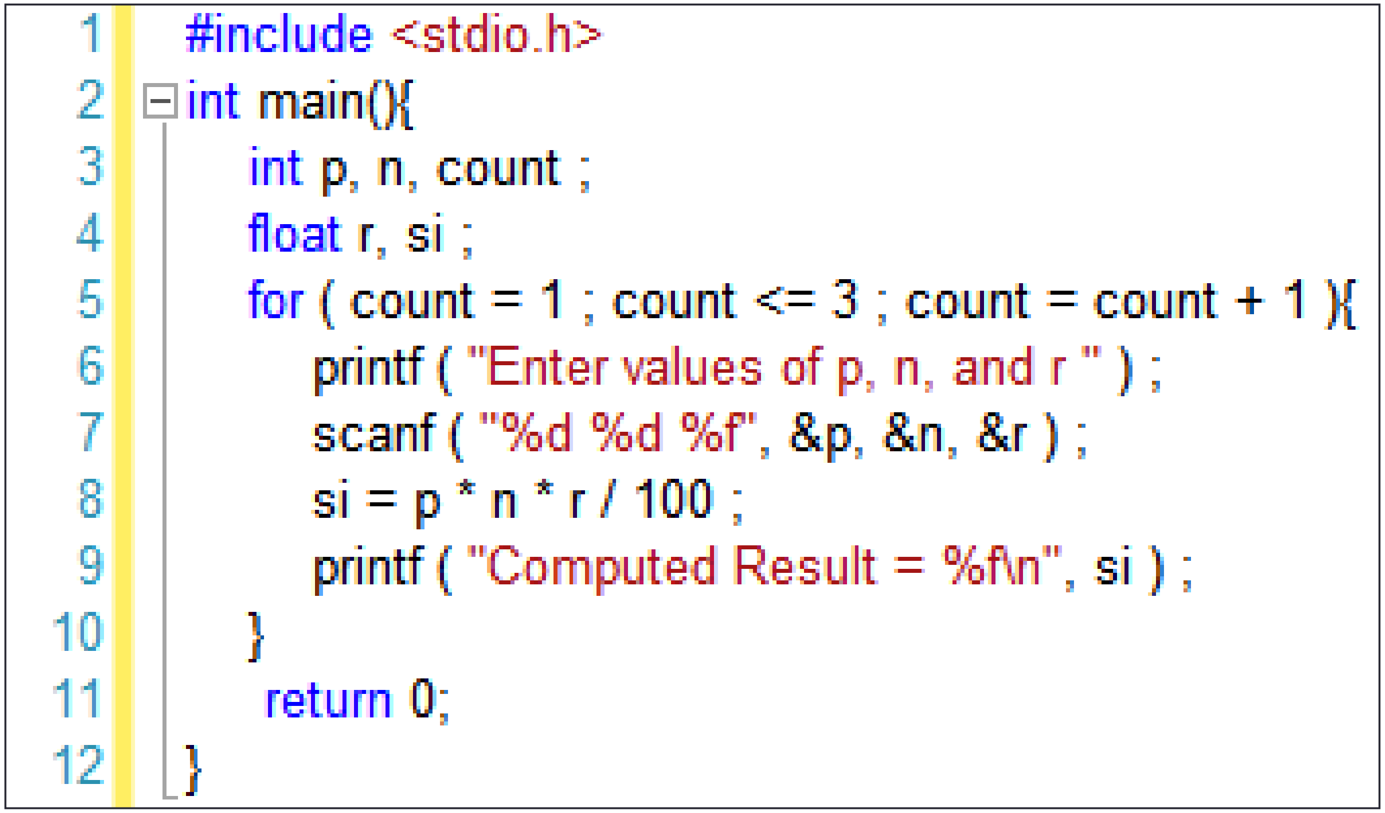
```
for ( initialise  counter ; test  counter ; increment  counter )
{
      do this ;
      and this ;
      and this ;
}
```

```
initialise loop counter ;
while ( test loop counter using a condition )
{
        do this ;
        and this ;
        increment loop counter ;
}
```



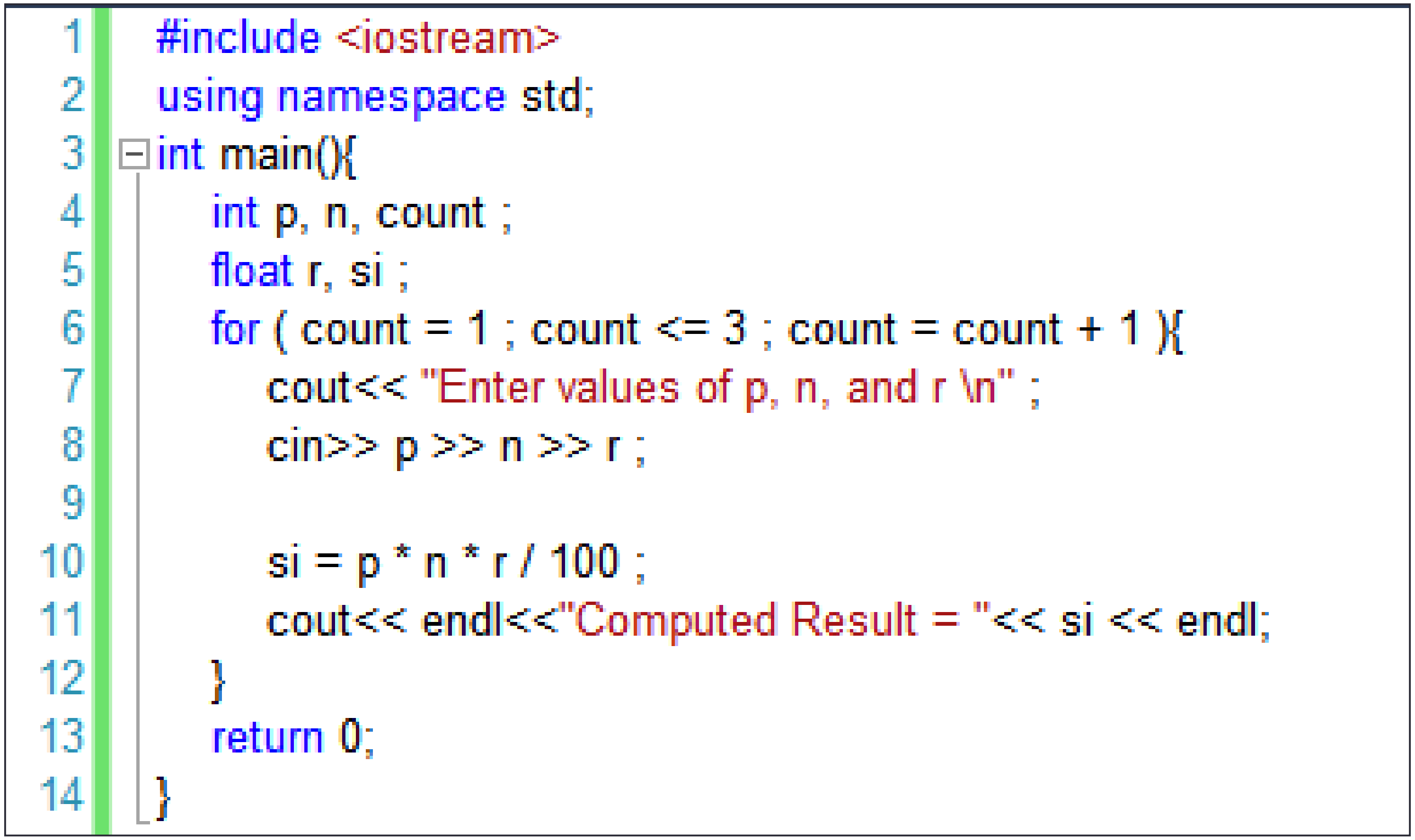
concept of execution of the *for* statement

for statement



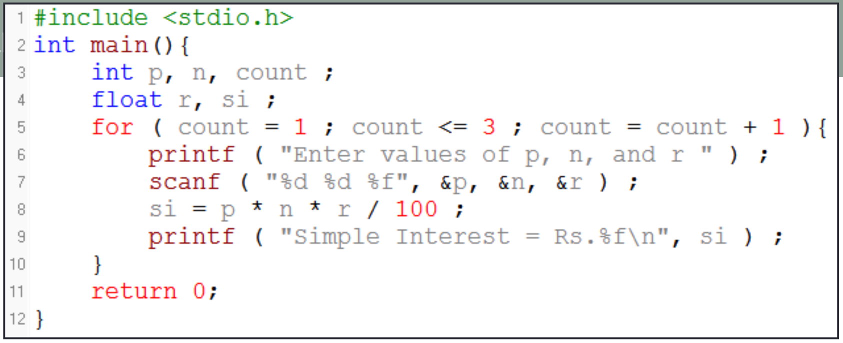
```
#include <stdio.h>
int main(){
    int p, n, count ;
    float r, si ;
    for ( count = 1 ; count <= 3 ; count = count + 1 ){
        printf ( "Enter values of p, n, and r " ) ;
        scanf ( "%d %d %f", &p, &n, &r ) ;
        si = p * n * r / 100 ;
        printf ( "Computed Result = %f\n", si ) ;
    }
     return 0;
}
```

for statement



```cpp
#include <iostream>
using namespace std;
int main(){
    int p, n, count ;
    float r, si ;
    for ( count = 1 ; count <= 3 ; count = count + 1 ){
        cout<< "Enter values of p, n, and r \n" ;
        cin>> p >> n >> r ;

        si = p * n * r / 100 ;
        cout<< endl<<"Computed Result = "<< si << endl;
    }
    return 0;
}
```

for statement

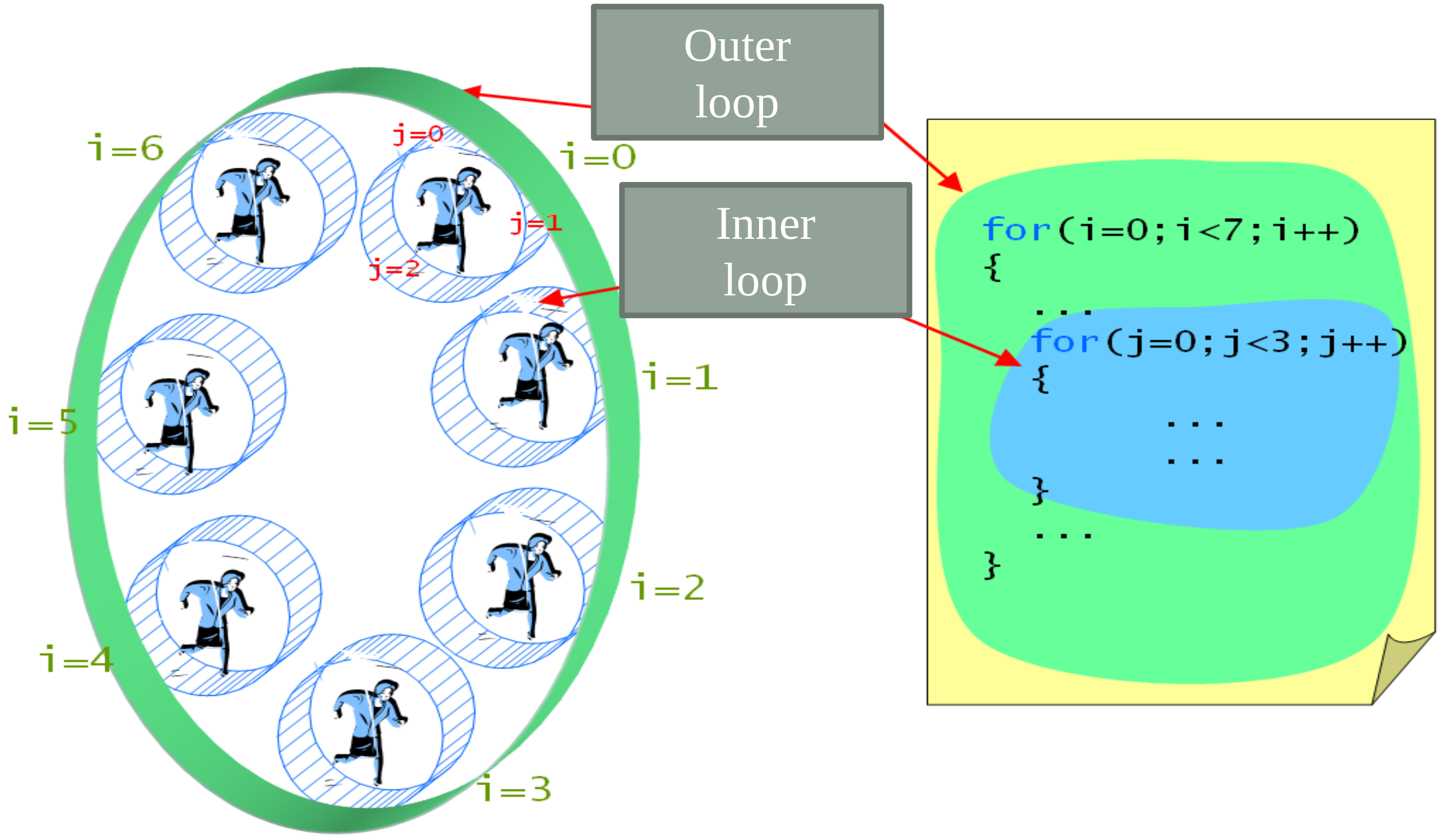


```c
#include <stdio.h>
int main(){
    int p, n, count ;
    float r, si ;
    for ( count = 1 ; count <= 3 ; count = count + 1 ){
        printf ( "Enter values of p, n, and r " ) ;
        scanf ( "%d %d %f", &p, &n, &r ) ;
        si = p * n * r / 100 ;
        printf ( "Simple Interest = Rs.%f\n", si ) ;
    }
    return 0;
}
```

- Let us now examine how the **for** statement gets executed:
  - ✓ When the **for** statement is executed for the first time, the value of **count** is set to an initial value 1.
  - ✓ Now the condition **count <= 3** is tested. Since **count** is 1 the condition is satisfied and the body of the loop is executed for the first time.
  - ✓ Upon reaching the closing brace of **for**, control is sent back to the **for** statement, where the value of **count** gets incremented by 1.
  - ✓ Again the test is performed to check whether the new value of **count** exceeds 3.
  - ✓ If the value of **count** is still within the range 1 to 3, the statements within the braces of **for** are executed again.
  - ✓ The body of the **for** loop continues to get executed till **count** doesn’t exceed the final value 3.
  - ✓ When **count** reaches the value 4 the control exits from the loop and is transferred to the statement (if any) immediately after the body of **for**.

Nesting *for* loops

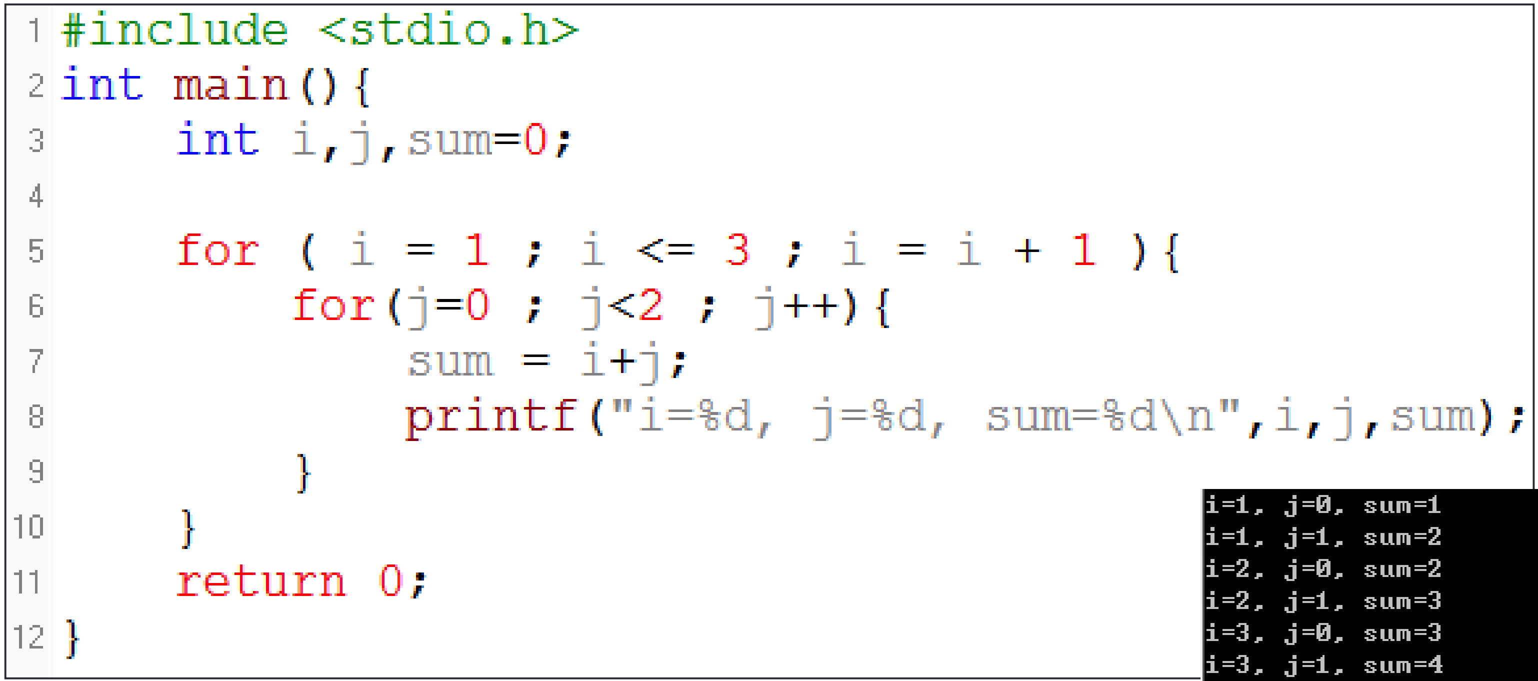
- Concept of the nesting *for* loop



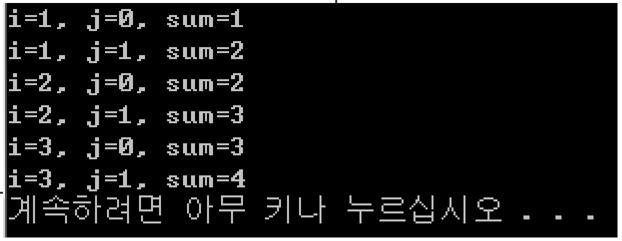
```
for(i=0;i<7;i++)
{
    ...
    for(j=0;j<3;j++)
    {
            ...
            ...
    }
    ...
}
```

Nesting *for* loops

- The way ***if*** statements can be nested, similarly ***while*** and ***for*** can also be nested.



```c
#include <stdio.h>
int main(){
    int i,j,sum=0;

    for ( i = 1 ; i <= 3 ; i = i + 1 ){
        for(j=0 ; j<2 ; j++){
            sum = i+j;
            printf("i=%d, j=%d, sum=%d\n",i,j,sum);
        }
    }
    return 0;
}
```



```
i=1, j=0, sum=1
i=1, j=1, sum=2
i=2, j=0, sum=2
i=2, j=1, sum=3
i=3, j=0, sum=3
i=3, j=1, sum=4
계속하려면 아무 키나 누르십시오 . . .
```

break statement

- We often come across situations where we want to jump out of a loop instantly, without waiting to get back to the conditional test.
- The keyword ***break*** allows us to do this.
  - ✓ When **break** is encountered inside any loop, control automatically passes to the first statement after the loop.
  - ✓ A **break** is usually associated with an **if**.

break statement

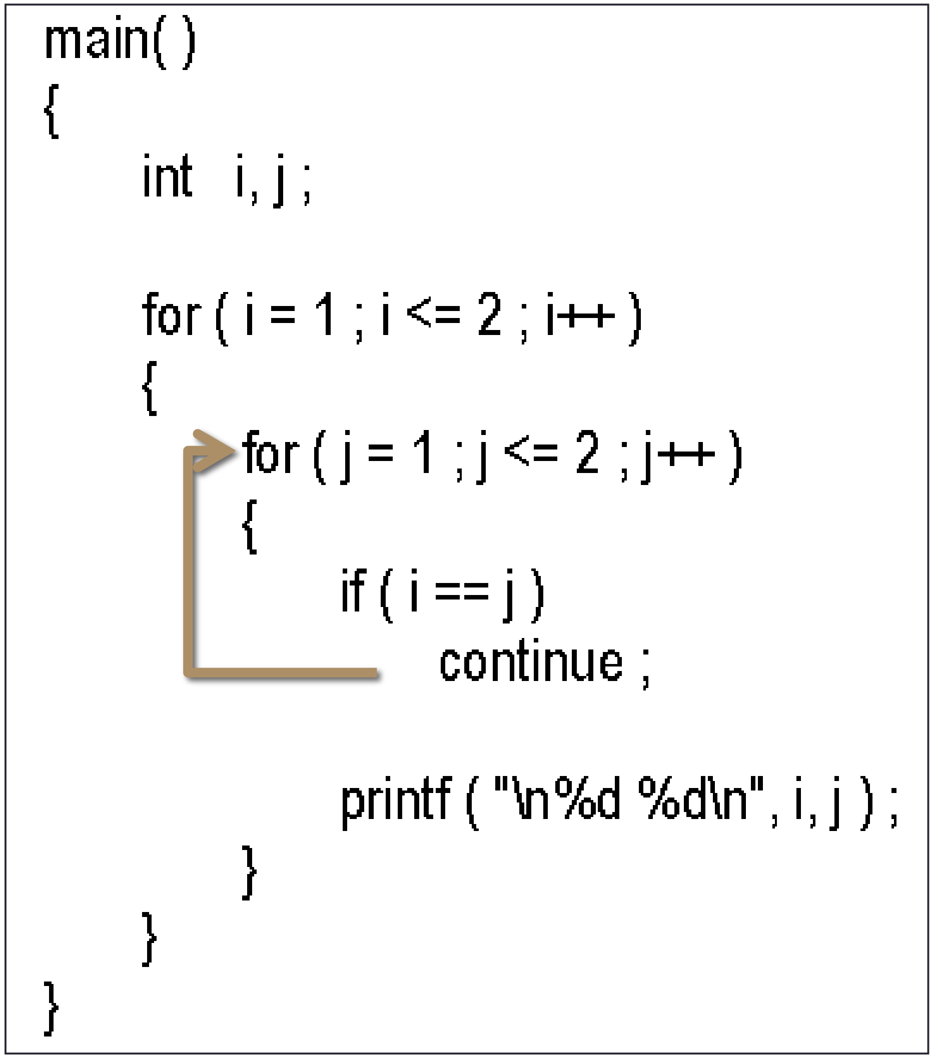
```
main( )
{
    int  i = 1 , j = 1 ;

    while ( i++ <= 100 )
    {
        while ( j++ <= 200 )
        {
            if ( j == 150 )
                break ;
            else
                printf ( "%d %d\n", i, j ) ;
        }

    }
}
```

```
main( )
{
    int   num, i ;

    printf ( "Enter a number " ) ;
    scanf ( "%d", &num ) ;

    i = 2 ;
    while ( i <= num - 1 )
    {
        if ( num % i == 0 )
        {
            printf ( "Not a prime number" ) ;
            break ;
        }
        i++ ;
    }

    if ( i == num )
        printf ( "Prime number" ) ;
}
```

continue statement

- In some programming situations we want to take the control to the beginning of the loop, bypassing the statements inside the loop, which have not yet been executed.
  - ✓ The keyword **continue** allows us to do this.
- When **continue** is encountered inside any loop, control automatically passes to the beginning of the loop.

continue statement



```
main( )
{
        int   i, j ;

        for ( i = 1 ; i <= 2 ; i++ )
        {
                for ( j = 1 ; j <= 2 ; j++ )
                {
                        if ( i == j )
                                continue ;

                        printf ( "\n%d %d\n", i, j ) ;
                }
        }
}
```

do-while statement

- The **do-while** loop looks like this:

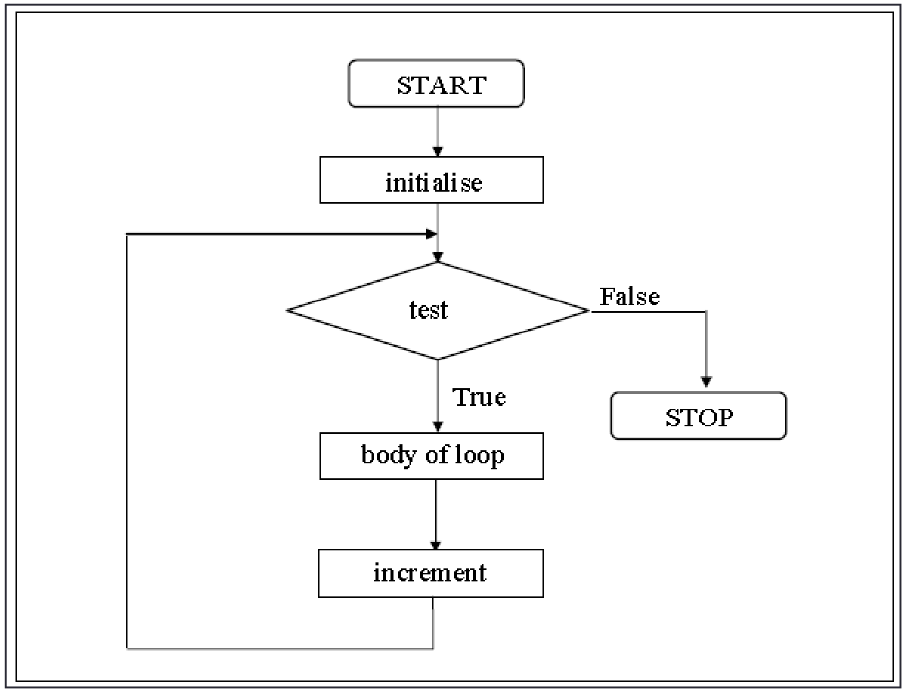
```
do
{
        this ;
        and this ;
        and this ;
        and this ;
} while ( this condition is true ) ;
```

do-while statement

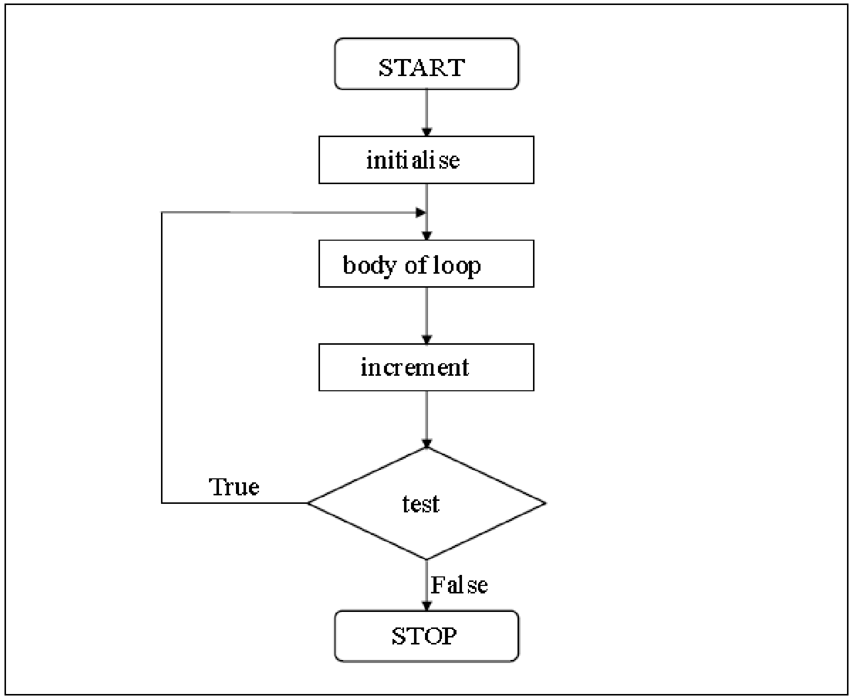
- There is a minor difference between the working of **while** and **do-while** loops.
  - ✓ This difference is the place where the condition is tested.
  - ✓ The **while** tests the condition before executing any of the statements within the **while** loop.
  - ✓ As against this, the **do-while** tests the condition after having executed the statements within the loop.

do-while statement

- Comparison between the *while* and *do-while* basic concept



concept of execution of the *while* statement



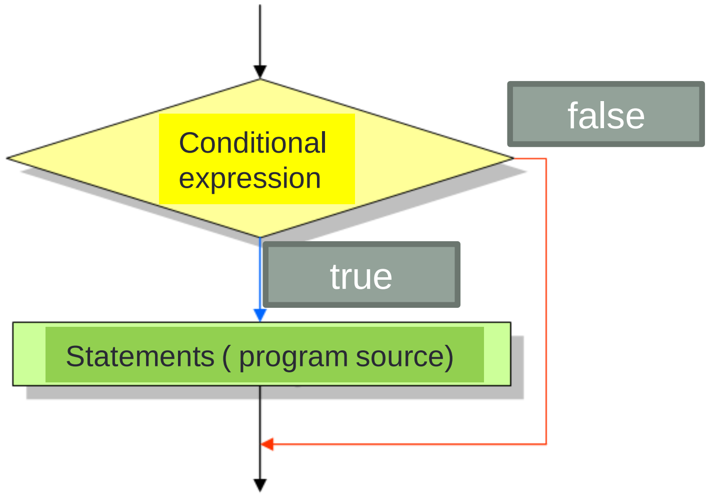
concept of execution of the *do-while* statement

if statement

```
if( conditional expression )
{
      do this;
}
```

The condition is true

Execute the statements



if statement

- if (x > y)   // if x is greater than y
- if (x < y)   // if x is smaller than y
- if (x >= y)   // if x is greater than or equal to y
- if (x <= y)   // if x is smaller than or equal to y
- if (x == y)   // if x is equal to y
- if (x != y)   // if x is not equal to y
- if ((x > y) && (x > z)) // if x> y AND x > z
- if ((x > y) || (x > z))   // if x>y OR x > z

if statement

```
if( sales > 2000 )
     bonus = 200;

if( score >= 60 )
     printf("passed.\n");

if( height >= 130 && age >= 10 )
     printf("passed \n");
```

```
int temperature = -10;
if ( temperature < 0 )
  printf(" Minus degree \n");          // when the condition is true
printf("the temperature is %d degree.\n", temperature); // always
```

if statement

- Compound statement

```
if( score >= 60 )
 {
    printf("passed \n");
    printf("can get the scholarship\n");
 }
```

if st

```c
// if 문을 사용하여 음수와 양수를 구별하는 프로그램
#include <stdio.h>

int main(void)
{
        int number;

        printf("insert the integer value:");
        scanf("%d", &number);

        if( number > 0 )
                printf("the value is positive \n");

        if( number == 0 )
                printf("the value is zero\n");

        if( number < 0 )
                printf("the value is negative\n");

        return 0;
}
```

If the value is 25

```
25
the value is positive
```

if-else statement

```
if(conditional expression )
      do this 1;
else
      do this 2;
```

if-else statement

```
// 윤년 판단 프로그램
#include <stdio.h>

int main(void)
{
	int year;

	printf("insert the year: ");
	scanf("%d", &year);

	if((year % 4 == 0 && year % 100 != 0) || year % 400 == 0)
		printf("%d year is a leap year.\n", year);
	else
		printf("%d year is a normal year.\n", year);

	return 0;
}
```

```
Insert the year: 2005 .
2005 year is a normal year.
```

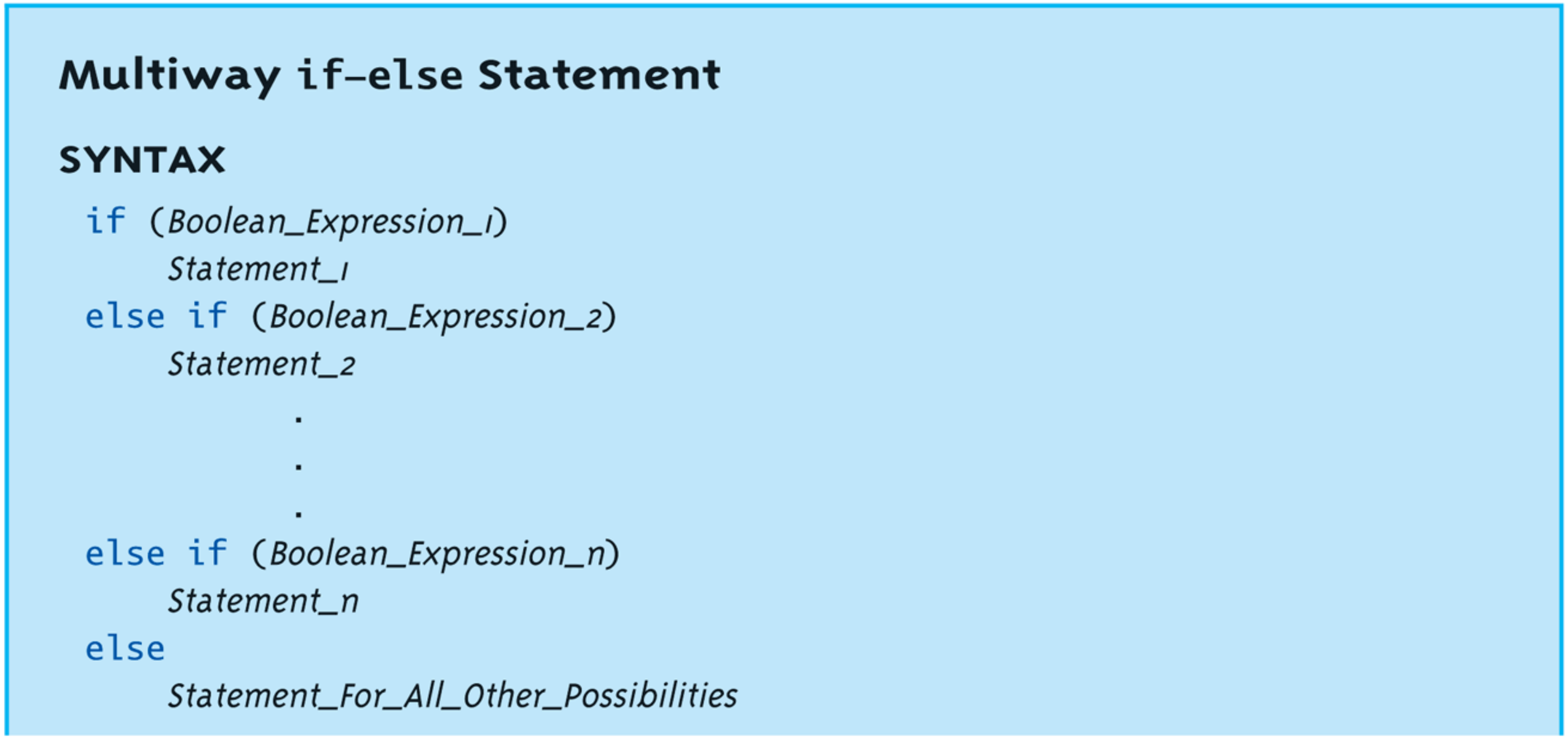
Nested *if*

```
if( conditional expression 1 )
      if(conditional expression 2 )
              do this;
```

```
if( score > 80 )
              if( score > 90 )
                            printf(" your score is A.\n");
```

```
if( score > 80 )
              if( score > 90 )
                            printf("your score is A.\n");
              else
                            printf("your score is B.\n");
```

Multiway *If* statement

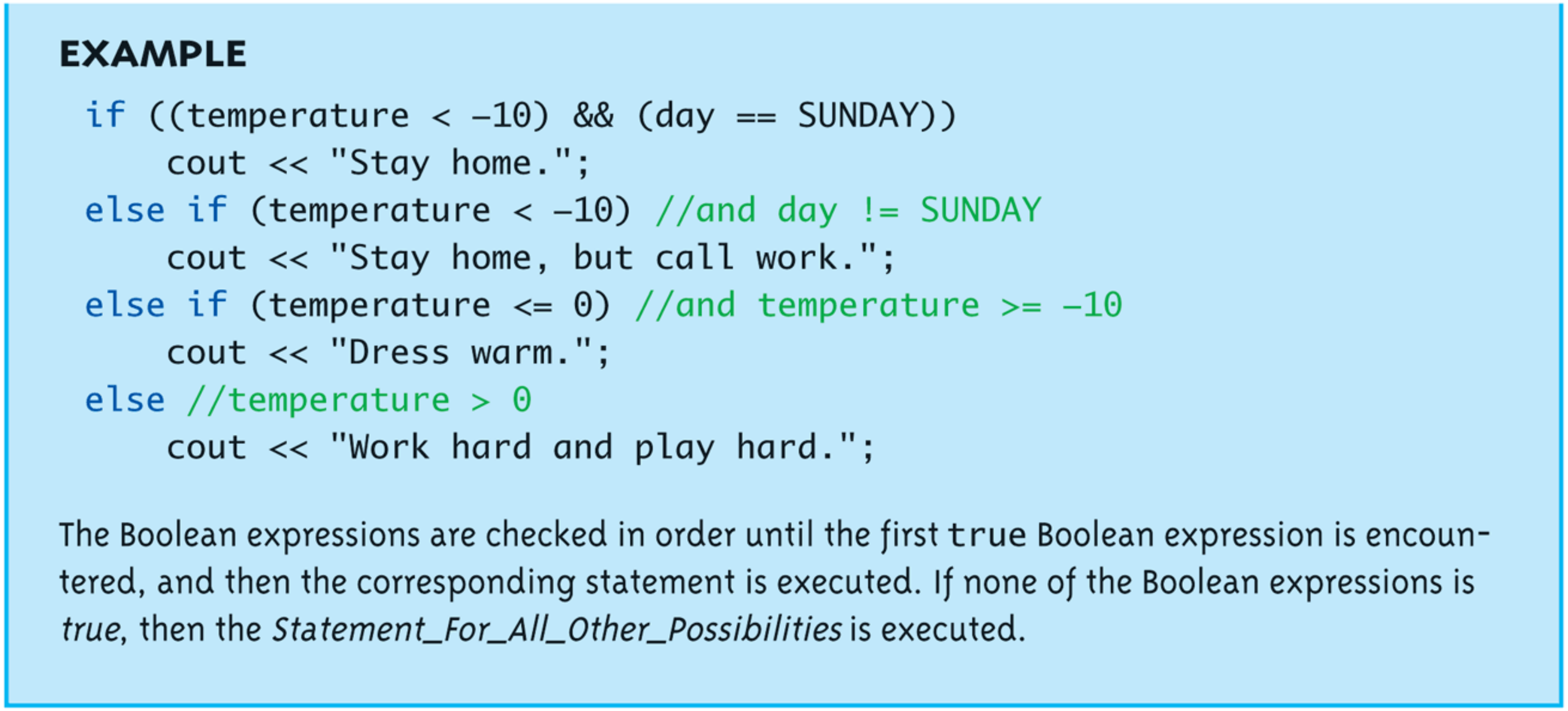


Multiway if-else Statement

SYNTAX

```
if (Boolean_Expression_1)
     Statement_1
else if (Boolean_Expression_2)
     Statement_2
            .
            .
            .
else if (Boolean_Expression_n)
     Statement_n
else
     Statement_For_All_Other_Possibilities
```

Multiway *If* statement



EXAMPLE

```
if ((temperature < -10) && (day == SUNDAY))
    cout << "Stay home.";
else if (temperature < -10) //and day != SUNDAY
    cout << "Stay home, but call work.";
else if (temperature <= 0) //and temperature >= -10
    cout << "Dress warm.";
else //temperature > 0
    cout << "Work hard and play hard.";
```

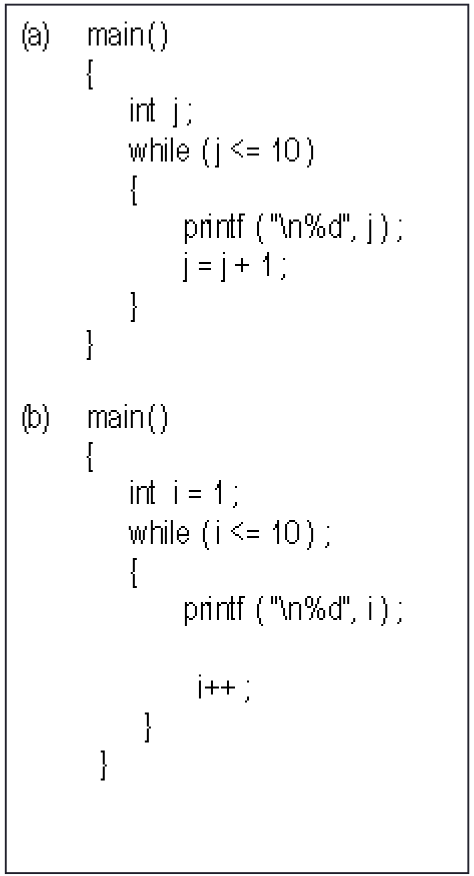
The Boolean expressions are checked in order until the first `true` Boolean expression is encountered, and then the corresponding statement is executed. If none of the Boolean expressions is *true*, then the *Statement_For_All_Other_Possibilities* is executed.

Summary

- The three type of loops available in C or C++ are **for**, **while**, and **do-while**.
- A **break** statement takes the execution control out of the loop.
- A **continue** statement skips the execution of the statements after it and takes the control to the beginning of the loop.
- A **do-while** loop is used to ensure that the statements within the loop are executed at least once.
- The ++ operator increments the operand by 1, whereas, the -- operator decrements it by 1.
- The operators +=, -=, *=, /=, %= are compound assignment operators. They modify the value of the operand to the left of them.
  - ✓ +=: a+=b → a = a+b;

Exercise

- *while* Loop
  - ✓What would be the output of the following programs:



```
(a)   main()
      {
          int  j ;
          while ( j <= 10 )
          {
               printf ( "\n%d", j ) ;
               j = j + 1 ;
          }
      }

(b)   main()
      {
          int  i = 1 ;
          while ( i <= 10 ) ;
          {
               printf ( "\n%d", i ) ;

                 i++ ;
            }
       }
```

Exercise

```
(c)   main( )
      {
          int  j ;
          while ( j <= 10 )
          {
              printf ( "\n%d", j ) ;
              j = j + 1 ;
          }
      }

(d)   main( )
      {
          int  x = 1 ;
          while ( x == 1 )
          {
              x = x - 1 ;
              printf ( "\n%d", x ) ;
          }
      }

(e)   main( )
      {
          int  x = 1 ;
          while ( x == 1 )
              x = x - 1 ;
          printf ( "\n%d", x ) ;
      }
```

```
(f)   main( )
      {
          char  x ;

          while ( x = 0 ; x <= 255 ; x++ )
              printf ( "\nAscii value %d Character %c", x, x ) ;
      }

(g)   main( )
      {
          int  x = 4, y, z ;
          y = --x ;
          z = x-- ;
          printf ( "\n%d %d %d", x, y, z ) ;
      }

(h)   main( )
      {
          int  x = 4, y = 3, z ;

          z = x-- -y ;
          printf ( "\n%d %d %d", x, y, z ) ;
      }
```

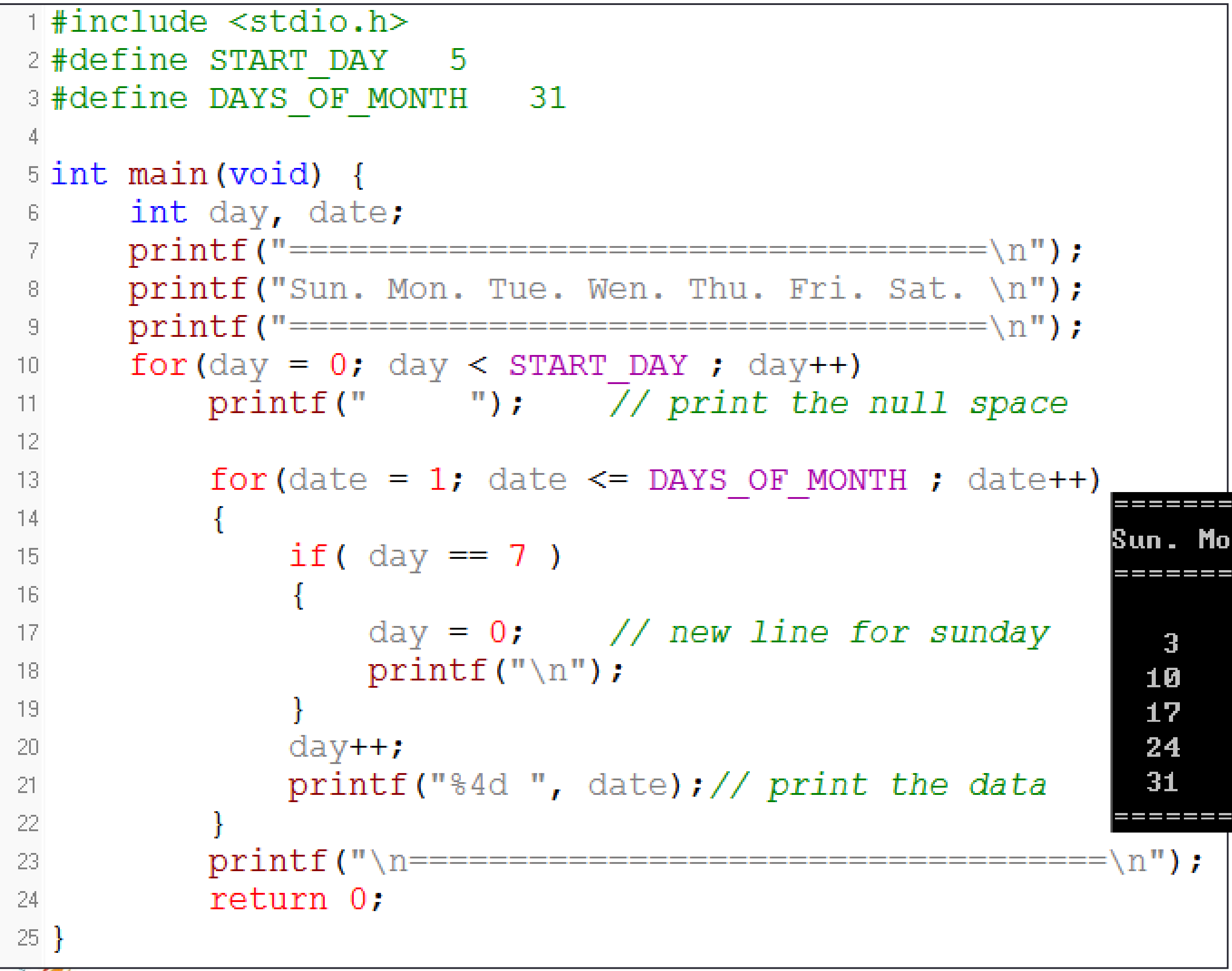
```
(i)   main( )
      {
          while ( 'a' < 'b' )
              printf ( "\nmalyalam is a palindrome" ) ;
      }

(j)   main( )
      {
          int  i = 10 ;
          while ( i = 20 )
              printf ( "\nA computer buff!" ) ;
      }

(k)   main( )
      {
          int  i ;
          while ( i = 10 )
          {

              printf ( "\n%d", i ) ;
              i = i + 1 ;
          }
       }
```

Exercise

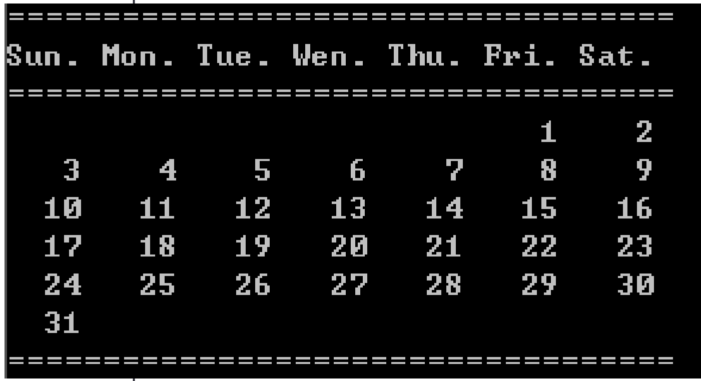
```
(l)     main( )
        {
            float  x = 1.1 ;
            while ( x == 1.1 )
            {
                printf ( "\n%f", x ) ;
                x = x – 0.1 ;
            }
        }

(m)     main( )
        {
            while ( '1' < '2' )
                printf ( "\nIn while loop" ) ;
        }

(n)     main( )
        {
            char  x ;
            for ( x = 0 ; x <= 255 ; x++ )
                printf ( "\nAscii value %d Character %c", x, x ) ;
        }
```

```
    (o)     main( )
            {
                int  x = 4, y = 0, z ;
                while ( x >= 0 )
                {
                    x-- ;
                    y++ ;
                    if ( x == y )
                        continue ;
                    else
                        printf ( "\n%d %d", x, y ) ;
                }
            }

(p)     main( )
        {
            int  x = 4, y = 0, z ;
            while ( x >= 0 )
            {
                if ( x == y )
                    break ;
                else
                    printf ( "\n%d %d", x, y ) ;
                x-- ;
                y++ ;
            }
        }
```

Exercise



```c
#include <stdio.h>
#define START_DAY    5
#define DAYS_OF_MONTH    31

int main(void) {
    int day, date;
    printf("===================================\n");
    printf("Sun. Mon. Tue. Wen. Thu. Fri. Sat. \n");
    printf("===================================\n");
    for(day = 0; day < START_DAY ; day++)
        printf("     ");    // print the null space

        for(date = 1; date <= DAYS_OF_MONTH ; date++)
        {
            if( day == 7 )
            {
                day = 0;    // new line for sunday
                printf("\n");
            }
            day++;
            printf("%4d ", date);// print the data
        }
        printf("\n===================================\n");
        return 0;
}
```



```
====================================
Sun. Mon. Tue. Wen. Thu. Fri. Sat.
====================================
                           1    2
   3    4    5    6    7    8    9
  10   11   12   13   14   15   16
  17   18   19   20   21   22   23
  24   25   26   27   28   29   30
  31
====================================
```

HW #3

1. Write a program to calculate the summation of nature number from 1 to 100 using each of the following statements.
   - ✓ *while*
   - ✓ *do-while*
   - ✓ *for*

HW #3

2. Write a program to produce the following output(Using loop statement):

```
A B C D E F G F E D C B A
A B C D E F   F E D C B A
A B C D E       E D C B A
A B C D           D C B A
A B C               C B A
A B                   B A
A                       A
```

HW #3

3. Write a program to calculate the following equations:

- ✓ $$S_1 = \sum_{i=1}^{30} (i^2 + 1)$$
- ✓ $$S_2 = \sum_{i=10}^{30} \sum_{j=0}^{5} (i \times j)$$