บทที่ 5
คำสั่งการทำซ้ำและทำงานวนรอบ

วัตถุประสงค์

1. เพื่อให้นักศึกษาทราบรูปแบบคำสั่ง while
2. เพื่อให้นักศึกษาสามารถแก้ปัญหาโดยใช้คำสั่ง while ได้
3. เพื่อให้นักศึกษาทราบรูปแบบคำสั่ง for
4. เพื่อให้นักศึกษาสามารถแก้ปัญหาโดยใช้คำสั่ง for ได้
5. เพื่อให้นักศึกษาทราบรูปแบบคำสั่ง do..while
6. เพื่อให้นักศึกษาสามารถแก้ปัญหาโดยใช้คำสั่ง do..while ได้
7. เพื่อให้นักศึกษาสามารถแก้ปัญหาโดยใช้คำสั่งลูปซ้อนลูปได้

การแก้ปัญหาโปรแกรมนั้นมีโครงสร้างการปฏิบัติการอยู่ 3 รูปแบบได้แก่ sequence , selection และ repetition สำหรับในบทที่ผ่านมาได้ทราบถึงโครงสร้างไป 2 รูปแบบแล้ว สำหรับในบทนี้จะกล่าวถึงโครงสร้างของโปรแกรมที่เป็นการกระทำซ้ำ(Repetition)และการทำงานวนรอบ(Loop) ซึ่งยังคงเกี่ยวข้องกับเงื่อนไขและนิพจน์ตรรกในการปฏิบัติงาน สำหรับในบทนี้เราจะกล่าวถึงคำสั่งในการทำงานวนรอบหรือกระทำซ้ำ 3 คำสั่งด้วยกัน ได้แก่ คำสั่ง while , คำสั่ง for และคำสั่ง do..while

5.1 คำสั่ง While

การแก้ปัญหาโปรแกรมในบางกรณีมีการกระทำงานหรือปฏิบัติงานซ้ำๆกัน โดยนับรอบการกระทำซ้ำได้แน่นอน อาทิเช่นทราบว่าคำนวณหาเกรดเฉลี่ยของนักศึกษาจำนวน 10 คน ต้องทำกิจกรรมต่างๆซ้ำๆกัน 10 รอบ หรือในบางกรณีไม่ทราบว่าทำกี่รอบแต่กระทำไปเรื่อยๆ เช่น การคิดเงินของพนักงานขาย แต่จะเลิกกิจกรรมที่กระทำซ้ำๆนี้เมื่อเลิกงาน เป็นต้น ดังนั้นการกระทำซ้ำจึงขึ้นอยู่กับปัจจัยต่างๆที่แตกต่างกัน ภาษา C++ ได้เตรียมคำสั่ง while ให้โปรแกรมเมอร์เพื่อนำมาใช้งาน มีรูปแบบกระทำซ้ำได้หลายรูปแบบดังนี้

- counter-controlled while loop
- sentinel-controlled while loop
- flag-controlled while loop
- eof-controlled while loop

กรณีที่ 1 : Counter-controlled while loop

เป็นการกระทำกิจกรรมซ้ำๆโดยการทำงานถูกควบคุมด้วยตัวนับ เป็นการแก้ปัญหาที่โปรแกรมเมอร์ทราบแน่นอนว่ามีการทำซ้ำเป็นจำนวนกี่รอบแน่นอน มีรูปแบบทั่วไปดังนี้

```
set loop counter variable to an initial value
While loop control variable < final value
    . . .
    Increase loop control variable by 1
```

while Statement

รูปแบบ:

```
while (loop repetition condition)
    statement ;
```

ตัวอย่าง:

```
// display n asterisks
countStar = 0 ;
while (countStar < n)
{
    cout << "*" ;
    countStar = countStar + 1 ;
}
```

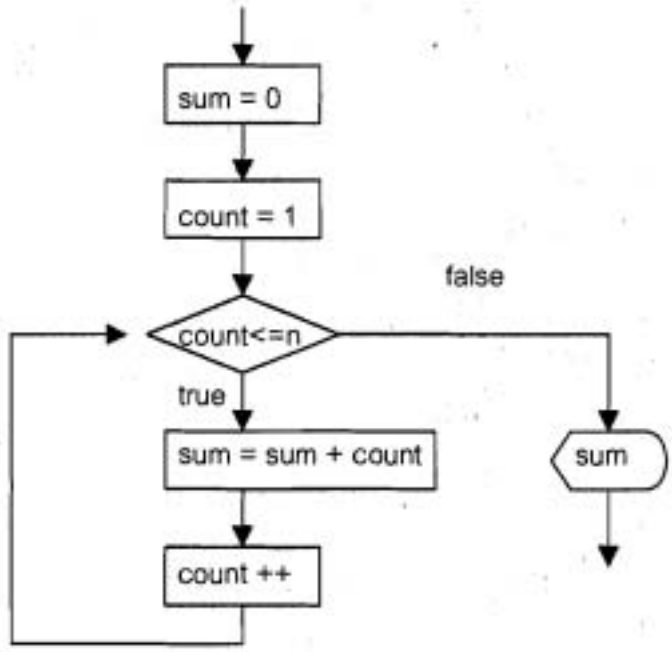
ความหมาย : countStar เป็นตัวนับการทำงาน โดยถ้า n มีค่าเท่ากับ 5 การทำงานจะวนรอบทั้งหมด 5 รอบโดยพิมพ์ "*****" ออกทางจอภาพ และจะหยุดการทำงานเมื่อเงื่อนไขเป็น false

ตัวอย่างที่ 5.1 การหาผลรวมของ 1 + 2 + ...+n

การหาผลรวมของจำนวนตัวเลขที่มีการเพิ่มขึ้นเป็นลำดับครั้งละ 1 นี้เหมาะสมกับการใช้คำสั่ง while ในการแก้ปัญหา แต่เราต้องมีการปรับเปลี่ยนตัวแปรที่ใช้ในการนับรอบให้เหมาะสม ดังนี้

```
sum = 0 ;       count = 1 ;
while (count <= n)
{
    sum = sum + count ;
    count ++ ;
}
cout << sum << endl ;
```

แสดงการทำงานเป็นผังโปรแกรม ดังนี้



การทำงานนั้นจะมีตัวแปร count ทำหน้าที่ในการนับรอบกำหนดให้เริ่มที่ 1 ตัวแปร n เป็นค่าที่ใช้ในการเปรียบเทียบการทำงานซึ่งเก็บค่าสุดท้าย(final value) การกระทำซ้ำจะกระทำเมื่อเปรียบเทียบเงื่อนไขแล้วมีค่าเป็น true ซึ่งจะนำค่าของตัวแปร count มาเก็บสะสมในตัวแปร sum ซึ่งเป็นการเก็บค่าผลรวมของค่าที่เก็บในตัวแปร count ในแต่ละรอบ ดังการทำงาน

count	sum
1	0+1
2	0+1+2
3	0+1+2+3
4	0+1+2+3+4
...	
n	0+1+2+...+n

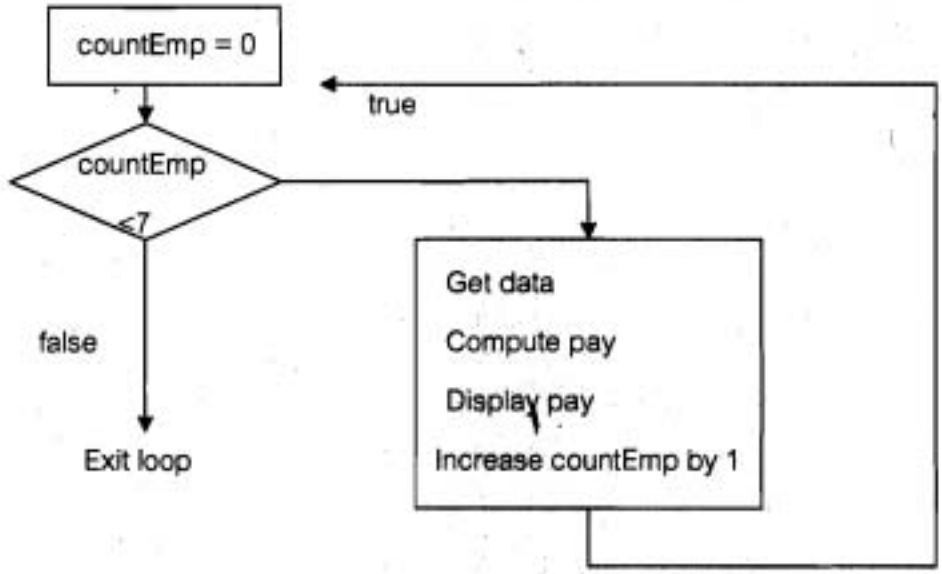
โดยค่าของ count จะเพิ่มขึ้น 1 ในการทำงานแต่ละรอบ และหยุดการทำซ้ำเมื่อค่าของ count >n

ตัวอย่างที่ 5.2 ส่วนของคำสั่งในการคิดเงินเดือนของพนักงาน 7 คน

ในที่นี้เราทราบแน่นอนว่ามีการคิดเงินเดือนซึ่งมีการกระทำซ้ำๆกัน 7 ครั้งด้วยกัน ดังนั้นเราสามารถใช้คำสั่ง while ในการแก้ปัญหาได้โดยเริ่มจากการ

- initialized เป็นการกำหนดตัวแปรที่ใช้ในการนับรอบการทำงาน ในที่นี้ให้เป็นตัวแปร countEmp เริ่มต้นให้มีค่าเท่ากับ 0
- tested เป็นเงื่อนไขทางตรรกที่มีผลให้มีค่าเป็น true และ false โดยนำตัวแปร countEmp เปรียบเทียบกับค่าสุดท้ายที่กระทำงานในที่นี้คือ 7
- updated เป็นการเพิ่มตัวนับให้เพิ่มขึ้นจากเดิม 1 เพื่อวนรอบไปทำงานในรอบต่อไป

ซึ่งแสดงเป็นผังโปรแกรมได้ดังนี้



เราสามารถเขียนเป็นคำสั่งภาษา C++ ได้ดังนี้

```
countEmp = 0;                          // no employees processed yet
while (countEmp < 7)                   // test the count of employees
{
 cout << "Hours: " ;
 cin >> hours ;
```

```
    cout << "Rate : " ;
    cin >> rate;
    pay = hours * rate ;
    cout << "Weekly pay is " << pay << endl;
    countEmp ++ ;                          // increment count of employees
   }
   cout << "All employees processed" << endl ;
```

เราสามารถนำมาเขียนเป็นโปรแกรมภาษา C++ ที่สมบูรณ์โดยกำหนดตัวแปร totalPay เพื่อเก็บผลรวมของเงินเดือนของพนักงานทั้งหมด

```
// File:computePay.cpp
//Computes the payroll for a company
#include <iostream>
using namespace std;
int main()
{
   int numberEmp;    // input – number of employees
   int countEmp;     // counter – current employee number
   float hours;      // input – hours  worked
   float rate;       // input – hourly rate
   float pay;        // output – weekly pay
   float totalPay;   // output – company payroll

   // Get number of employees from user
   cout << "Enter number of employees: " ;
   cin >> numberEmp ;
```

```
    // Process payroll for all employees
    totalPay = 0.0 ;
    countEmp = 0 ;
    while (countEmp < numberEmp)
    {
        cout << "Hours: " ;
        cin >> hours ;
        cout << "Rate : ";
        cin >> rate;
        pay = hoirs * rate ;
        cout << "Pay is" << pay << endl << endl ;
        totalPay = totalPay + pay ;                    // add next pay
        countEmp = countEmp + 1;
    }
    cout << "Total  payroll  is " << totalPay << endl ;
    cout  << "All  employees  processed. " << endl ;

    return 0 ;
}
```

ทดสอบ

```
Enter number of employees : 3
Hours:  50
Rate :  5.25
Pay is  262.5
Hours:  6
Rate :  5
```

```
Pay is  30
Hours:  15
Rate:   7
Pay is  105
Total payroll is  397.5
All employees processed.
```

การทำงานผู้ใช้ป้อน 3 เป็นการบังคับให้โปรแกรมมีการปฏิบัติการซ้ำๆกันเป็นจำนวน 3 รอบ ในแต่ละรอบจะให้ผู้ใช้ป้อนจำนวนชั่วโมงทำงาน(hours)และอัตราค่าจ้างต่อชั่วโมง(rate)ทาง แป้นพิมพ์นำไปคำนวณหาค่าเงินเดือนของพนักงาน(pay) และนำเงินเดือนบวกสะสมในตัวแปร totalPay ตัวนับ countEmp มีค่าเริ่มต้นก่อนการทำงานวนรอบเท่ากับ 0 และเมื่อทำงานในแต่ละรอบค่าจะเพิ่มขึ้นรอบละ 1 เพิ่มเป็น 1 , 2 จนกระทั่งมีค่าเท่ากับ 3 จะหยุดการทำซ้ำและออกจากคำสั่ง while พิมพ์เงินเดือนทั้งหมดของพนักงานออกทางจอภาพ

ตัวอย่างที่ 5.3 โปรแกรมการหาค่าเฉลี่ย

```
// Program : AVG1
#include <iostream>
using namespace std;
int main()
{
    int  limit ;      // variable to store the number of items int the list .
    int  number ; // variable to store the number
    int  sum;       // variable to store the sum
    int  counter ; // loop control variable
    cout << " Enter data for processing " << endl ;
    cin >> limit ;
    sum = 0 ;
```

```
    counter = 0 ;
    while (counter < limit)
    {
        cin >> number ;
        sum = sum + number ;
        counter++ ;
    }
    cout << " The sum of the " << limit << "numbers = " << sum << endl ;
    if (counter != 0)
        cout << "The average = " << sum / counter << endl ;
    else
        cout << "No input." << endl ;
    return 0;
}
```

ทดสอบ

```
Enter data for processing
12 8 9 2 3 90 38 56 8 23 89 7 2 8 3 8
The  sum of the 12 number = 335
The average = 7
```

การทำงานของโปรแกรมจะให้ผู้ใช้ป้อนจำนวนของตัวเลขทั้งหมดเก็บในตัวแปร limit ซึ่งเป็นค่าสุดท้ายของตัวนับการทำงานซ้ำ สำหรับการปฏิบัติงานคำสั่งซ้ำๆกันนั้นจะให้ผู้ใช้ป้อนตัวเลขจำนวนเต็มใดๆในการทำงานแต่ละรอบเพื่อนำไปบวกสะสมในตัวแปร sum โดยตัวนับ counter จะเพิ่มขึ้น 1 ในการทำงานแต่ละรอบ เมื่อตัวแปร counter มีค่าเท่ากับ limit จะหยุดการทำซ้ำและพิมพ์ผลรวมทั้งหมดออกทางจอภาพ ต่อจากนั้นจะตรวจสอบค่าของ counter ในกรณีที่มีค่า

ไม่เท่ากับ 0 จะพิมพ์ค่าเฉลี่ยของกลุ่มข้อมูลออกทางจอภาพ แต่ถ้ามีค่าเท่ากับ 0 จะแสดงข้อความ " No input " ออกทางจอภาพ

กรณีที่ 2 : sentinel-controlled while loop

ในกรณีที่การกระทำซ้ำนั้นไม่ทราบจำนวนรอบแน่นอน เราไม่สามารถใช้วิธีการในการแก้ปัญหาในลักษณะของกรณีที่ 1 ได้เพราะไม่ทราบว่าจุดสิ้นสุดอยู่ที่ใด การแก้ปัญหาในลักษณะนี้นั้นโปรแกรมเมอร์สามารถกำหนดค่าพิเศษ(special value) โดยป้อนเป็นลำดับสุดท้ายเพื่อหยุดการทำงานซ้ำได้ เราเรียกค่านี้ว่า sentinel ดังนั้นรูปแบบการทำงานจะมีการปรับเปลี่ยนไปเป็นดังนี้

```
cin >> variable ;                // initialize the loop control variable
while (variable != sentinel)     // test the loop control variable
{
        . . .
        cin >> variable ;        // re-initialize the loop control variable
}
```

ตัวอย่างที่ 5.4 โปรแกรมการหาค่าเฉลี่ย

```
// Program : AVG2
#include <iostream>
using namespace std ;
const int SENTINEL = -999 ;
int main()
{
        int number ;    // variable to store the number
        int sum = 0;    // variable to store the sum
        int count = 0;  // variable to store the total

        // numbers read
        cout << "Enter numbers ending with " << sentinel <<endl ;
```

```
    cin >> number ;

    while ( number != SENTINEL)
    {
        sum = sum + number ;
        count ++ ;
        cin >> number ;
    }
    cout << "The sum of the " << count << "numbers is " << sum << endl;
    if (count != 0 )
        cout << "The average is " << sum/count << endl ;
    else
        cout << "No input" << endl ;
    return 0;
}
```

ทดสอบ

```
Enter numbers ending with -999
34  23  9  45  78  0  77  8  3  5  -999
The sum of the 10 numbers is 282
The average is 28
```

การแก้ปัญหาในกรณีที่ไม่ทราบว่ามีข้อมูลทั้งหมดกี่ตัวนั้น เราเริ่มจากกำหนดค่าตัวนับ (count) ให้เริ่มต้นเป็น 0 และผลรวม(sum) ให้มีค่าเป็น 0 ด้วย ต่อจากนั้นให้โปรแกรมรับข้อมูลจากผู้ใช้ทีละค่า โดยตรวจสอบกับข้อมูลพิเศษที่กำหนดให้เป็นตัวจบการทำซ้ำในที่นี้คือ SENTINEL ซึ่งมีค่าเท่ากับ –999 จากข้อมูลทดสอบผู้ใช้ป้อนค่า 34 ซึ่งไม่ใช่ข้อมูลพิเศษ มีผลให้เงื่อนไขเป็น trueจะกระทำงานตามที่กำหนดโดยนำค่า 34 ไปรวมกับข้อมูลเดิมที่เก็บในตัวแปร SUM และนำ ผลรวมใหม่เก็บแทนที่ค่า SUM เดิม ต่อจากนั้นเพิ่มค่าตัวนับ(count) ขึ้น 1 เป็น

เป็นอันว่าผู้ใช้ป้อน ข้อมูลไปเก็บไว้ 1 ค่าเรียบร้อยแล้ว ต่อจากนั้นโปรแกรมจะให้ผู้ใช้ป้อนค่าใหม่ตัวต่อไป เงื่อนไขของการทำซ้ำจะเปลี่ยนแปลงไปเพราะค่าที่ผู้ใช้ป้อนใหม่จะเก็บแทนที่ ค่าของ number เดิมในที่นี้ตัวสองคือ 23 ซึ่งการทำงานจะตรวจสอบเงื่อนไขและกระทำซ้ำเมื่อเงื่อนไขเป็น true จะกระทำคำสั่งเป็นลำดับ จนกระทั่งผู้ใช้ป้อน –999 ซึ่งมีผลให้เงื่อนไขเป็น false จะหยุดการทำซ้ำและพิมพ์ค่า ผลรวมสะสมทั้งหมดออกมาทางจอภาพ ต่อจากนั้นจะตรวจสอบจำนวนของข้อมูลในกรณีที่ไม่มีข้อมูลหรือ count มีค่าเท่ากับ 0 จะแสดงข้อความ "No input" ออกมาทางจอภาพ แต่ถ้ามีข้อมูล จะแสดงค่าเฉลี่ยของข้อมูลกลุ่มนี้ออกมาทางจอภาพ

ตัวอย่างที่ 5.5 Telephone DigIts

ตัวอย่างนี้เป็นการใช้คำสั่ง while ในการรับตัวอักษรภาษาอังกฤษ(A..Z) โดยให้โปรแกรมนั้นแปลงตัวอักษรนั้นให้เป็นตัวเลขที่สอดคล้องกัน อาทิเช่นปุ่มของโทรศัพท์มือถือที่ อักษร 'A' ,'B' และ 'C' จะอยู่ที่ปุ่ม 2 อักษร 'D' ,'E' และ 'F' อยู่ที่ปุ่ม 3 โดยการทำงานจะเปลี่ยนตัวอักษรทุกตัวที่ผู้ใช้ป้อนให้เป็นตัวเลข และหยุดการทำงานเมื่อผู้ใช้กดปุ่ม '#'

```
//*****************************************************************************************
// Program: Telephone Digits
// This is an example of a sentinel-controlled loop. This  program converts uppercase
// letters to their corresponding  telephone digits.
//*****************************************************************************************
#include ,iostream>
using namespace std;
int main()
{
   char  letter ;
   cout << "program to convert uppercase letters to their corresponding "
        << "telephone digits. " << endl ;
   cout << "Enter a letter: " ;
   cin   >> letter ;
```

```
cout << endl ;
while (letter != '#')
{
    cout << "The letter you entered is: " << letter << endl ;
    cout << "The corresponding telephone " << "digit is : " ;
    if (letter >= 'A' && letter <= 'Z')
       switch (letter)
       {
       case 'A' : case 'B' : case 'C' : cout << "2" << endl ;
                                        break ;
       case 'D' : case 'E' : case 'F' : cout << "3" << endl ;
                                        break ;
       case 'G' : case 'H' : case 'I' : cout << "4" << endl ;
                                        break ;
       case 'J' : case 'K' : case 'L' : cout << "5" << endl ;
                                        break ;
       case 'M' : case 'N' : case 'O' : cout << "6" << endl ;
                                        break ;
       case 'P' : case 'Q' :
       case 'R' : case 'S' :            cout << "7" << endl ;
                                        break ;
       case 'T' : case 'U' : case 'V' : cout << "8" << endl ;
                                        break ;
       case 'W' : case 'X' :
       case 'Y'  : case 'Z' :           cout << "9" << endl ;
                                        break ;
       }
```

```
        else
            cout << "Invalid input ." <, endl ;
        cout << "\nEnter  another uppercase letter to find its corresponding "
             << "telephone digit . " << endl ;
        cout << "To stop the program enter #." <, endl ;
        cout << "Enter a letter: " ;
        cin >> letter ;
        cout << endl ;
        } // end while
        return 0 ;
}
```

ทดสอบ

Program to convert uppercase letters to their corresponding telephone digits.

To stop the program enter #.

Enter a letter : A

The letter you entered is : A

The corresponding telephone digit is : 2

Enter another uppercase letter to find its correponding telephone digit.

To stop the program enter #.

Enter a letter : D

The letter you entered is : D

The corresponding telephone digit is : 3

Enter another uppercase letter to find its correponding telephone digit.

To stop the program enter #.

Enter a letter : #

การทำงานของโปรแกรมนั้นจะนำตัวอักษรภาษาอังกฤษตัวใหญ่ที่ผู้ใช้ป้อนไปแปลงให้เป็นตัวเลขที่สอดคล้อง โดยการตรวจสอบนั้นเป็นทางเลือกหลายกรณีด้วยกัน การทำงานจะเริ่มจากให้ผู้ใช้ป้อนตัวอักษรที่ต้องการ 1 ตัวทางแป้นพิมพ์ โปรแกรมจะตรวจสอบว่ามีค่าเท่ากับ '#' หรือไม่ในกรณีที่เป็น true จะหยุดการทำงาน แต่ถ้าผู้ใช้ป้อนตัวอักษรใดๆที่ไม่ใช่ภาษาอังกฤษตัวใหญ่โปรแกรมจะแสดงข้อความว่า "Invalid input" มิเช่นนั้นจะแสดงตัวเลขที่สอดคล้องกับตัวอักษรภาษาอังกฤษนั้นๆออกทางจอภาพ

กรณีที่ 3 : flag-controlled while loop

เป็นการแก้ปัญหาโปรแกรมโดยนำตัวแปรชนิด bool มาใช้สำหรับควบคุมการทำงานซ้ำ โดยกำหนดตัวแปร found ซึ่งเป็นข้อมูลชนิด bool ขึ้นมาเพื่อตรวจสอบการทำงาน
โดยมีรูปแบบการทำงานดังนี้

```
found = false ;              // initialize the loop control variable
while (!found)               // test the loop control variable
{
      ...
      if (expression)
        found = true ;       // re-initialize the loop control variable
      ...
}
```

การแก้ปัญหาโปรแกรมในลักษณะนี้จะเหมาะสำหรับการสืบค้นข้อมูลที่ต้องการ กล่าวคือจะกำหนดตัวแปร found ให้มีค่าเท่ากับ false เมื่อค้นหาข้อมูลยังไม่พบหรือเริ่มต้นค้นหา การทำงานโปรแกรมจะนำค่าที่ต้องการค้นหาเปรียบเทียบกับข้อมูลที่มีอยู่ทีละค่าโดยจะทำงานวนรอบเปรียบเทียบไปเรื่อยๆ เมื่อใดที่ค้นหายังไม่พบจะทำการค้นหาวนรอบกระทำซ้ำอยู่อย่างนั้น แต่เมื่อใดที่ค้นหาข้อมูลพบ จะหยุดการค้นหา ซึ่งจะกล่าวโดยละเอียดในเรื่องของข้อมูลชนิดแถว

ตัวอย่างที่ 5.6 ฟังก์ชันในการค้นหาอักขระที่เป็นตัวเลข

ฟังก์ชันนี้เป็นการตรวจสอบข้อความที่ผู้ใช้ป้อนโดยจะส่งผ่านค่าตัวอักษรที่เป็นตัวแรกในข้อความนั้นส่งกลับไปยังจุดเรียกใช้

```
// Returns the first digit character read
// Pre: The user has entered a line of data
char getDigit()
{
    char  nextChar ;          // user input – next data character
    bool  digitRead ;         // status flag – set true
                              // when digit character is read

    digitRead = false;        // no digit character read yet
    while  (!digitRead)
    {
        cin >> nextChar ;
        digitRead = (('0' <= nextChar) && (nextChar <= '9'));
    }

    return nextChar ;
}
```

การเรียกใช้ ch = getChar ( ) ;

โดยการทำงานวนรอบของฟังก์ชันนั้นจะถูกควบคุมด้วยตัวแปร digitRead ซึ่งเป็น flag ที่ตรวจสอบว่าตัวอักขระที่อ่านนั้นเป็นตัวเลขหรือไม่ ในกรณีที่เป็นตัวเลขจะส่งผลให้ตัวแปรนี้มีค่าเป็น true จะหยุดการกระทำซ้ำ และออกจากการทำงานวนรอบ เพื่อส่งค่าตัวอักษรที่เป็นตัวเลขนี้กลับมายังจุดเรียกใช้

กรณีที่ 4 : eof-controlled while loop

ในกรณีที่ข้อมูลมีการเพิ่มหรือลบบ่อยๆ นั้นการอ่านข้อมูลและใช้ค่า sentinel เป็นตัวจบการทำงานนั้นไม่ดีเท่าที่ควร เพราะอาจทำให้เกิดความผิดพลาดเกิดขึ้นในกรณีที่มีโปรแกรมเมอร์หลายคนหรือผู้ใช้งานหลายคนถ้าไม่ทราบว่าจะจบการทำงานได้อย่างไร หรือกำหนด sentinel ให้มีค่าแตกต่างกันหรือเกิดหลงลืมทำให้เกิดความสับสนและใช้งานโปรแกรมได้ไม่มีประสิทธิภาพ ดังนั้นการแก้ไขโดยใช้ EOF (End of File) เข้ามาช่วยแก้ปัญหา

การป้อนข้อมูลของผู้ใช้นั้นราถือว่าเป็น ตัวแปรสายข้อมูลเข้า(input stream variable) ในที่นี้ข้อมูลต่างๆที่ผู้ใช้ป้อนจะถูกเก็บในตัวแปรในที่นี้คือ cin นั่นเองเป็นลำดับ ตัวกระทำ extraction (">>") จะอ่านข้อมูลและนำมาเก็บในตัวแปรที่กำหนดทีละค่า ซึ่งตัวแปร cin จะมีการส่งค่ากลับหลังจากการอ่านข้อมูลสิ้นสุดลง โดยมีการส่งผ่านค่ากลับไม่ true ก็ false ดังนี้

1. ถ้าโปรแกรมอ่านข้อมูลจนกระทั่งไม่มีข้อมูลให้อ่านอีกแล้ว (end of input data) ตัวแปรสายข้อมูลเข้าจะส่งผ่านค่า false หรือ cin มีค่าเท่ากับ false นั่นเอง
2. ถ้าโปรแกรมอ่านข้อมูลแต่มีการกำหนดข้อมูลผิดประเภท เช่น นำข้อมูลชนิด char มาเก็บในตัวแปร int จะเป็นสถานะหรือเหตุการณ์ที่ผิดพลาด คอมพิวเตอร์จะไม่ส่งข่าวสารความผิดพลาดแต่อย่างใด ยังคงปฏิบัติงานต่อไปโดยเพิกเฉยต่อข้อมูลนั้น และกำหนดให้ตัวแปรมีค่าเท่ากับ null และค่าของ cin มีค่าเท่ากับ false
3. แต่ถ้าการอ่านข้อมูลถูกต้อง ไม่เกิดเหตุการณ์ทั้งในกรณีที่ 1 และกรณีที่ 2 ข้างต้นค่าของ cin มีค่าเท่ากับ true

เราสามารถนำตัวแปรสายข้อมูลเข้า หรือ cin มาใช้ในการแก้ปัญหาได้ โดยมีรูปแบบการทำงานดังนี้

```
cin >> variable ;          // initialize the loop control variable
while (cin)                // test the loop control variable
{
    ...
    cin >> variable ;      // re-initialize the loop control variable
}
```

ตัวอย่างที่ 5.7 FIBONACCI NUMBER

การหาค่าจำนวน fibonacci นั้นมีลำดับของจำนวนดังนี้

1, 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 , ...

ถ้าให้จำนวน 2 ตัวแรกของลำดับมีค่าเท่ากับ a1 และ a2 ค่าของลำดับต่อมาจะมีค่าเท่ากับ

$$a_3 = a_1 + a_2$$

ดังนั้น ถ้าให้ a n โดย n >= 3 แล้ว

$$a_n = a_{n-1} + a_{n-2}$$

$$a_3 = a_2 + a_1$$

$$= 1 + 1$$

$$= 2$$

$$a_4 = a_3 + a_2$$

$$= 2 + 1$$

$$= 3$$

```
// Program: nth Fibonacci number
#include <iostream>
using namespace std;
int main()
{
    //Declare variable
    int  previous1;
    int  previous2;
    int  current ;
    int  counter;
    int  nthFibonacci;

    cout << "Enter the first two Fibonacci  numbers : " ;
```

```
    cin >> previous1 >>previous2 ;
    cout << endl ;
    cout << "The firat two Fibonacci numbers are " << previous1 << "and"
         << previous2 << endl ;
    cout << "Enter the position of the desired Fibonacci number : " ;
    cin >> nthFibonacci ;
    cout << endl ;

    if (nthFibonacci == 1)
        current = previous1 ;
    else if (nthFibonacci == 2)
        current = previous2 ;
    else
    {
       counter = 3 ;
       while (counter <= nthFibonacci)
       {
           current = previous2 + previous1 !
           previous1 = previous2 ;
           previous2  = current ;
           counter ++ ;
       } // end while
    } // end else
    cout << "The Fibonacci number at position " << nthFibonacci << "is"
         << current << endl ;
    return 0 ;
} // end main
```

ทดสอบ1

Enter the first two Fibonacci numbers : 12 16

The first two Fibonacci numbers are 12 and 16

Enter the position of the desired Fibonacci number : 10

The Fibonacci number at position 10 is 796

ทดสอบ2

Enter the first two Fibonacci numbers : 1 1

The first two Fibonacci numbers are 1 and 1

Enter the position of the desired Fibonacci number : 15

The Fibonacci number at position 15 is 610

ทดสอบ3

Enter the first two Fibonacci numbers : 20 25

The first two Fibonacci numbers are 20 and 25

Enter the position of the desired Fibonacci number : 10

The Fibonacci number at position 10 is 1270

การแก้ปัญหาโปรแกรมนั้นเราจะป้อนค่าของจำนวนที่ 1,ค่าของจำนวนที่ 2 และลำดับที่ที่ต้องการหาค่า(nthFibonacci) ทางแป้นพิมพ์มีการใช้คำสั่ง if ในการกำหนดทางเลือกในการทำงานและใช้คำสั่ง while เพื่อวนรอบกระทำงานซ้ำโดยใช้ตัวแปร count เป็นตัวเปรียบเทียบกับลำดับที่ต้องการ เพื่อหาค่าของข้อมูล

5.2 คำสั่ง for

ภาษา C++ ได้เตรียมคำสั่ง for ไว้สำหรับโปรแกรมเมอร์ในการแก้ปัญหาที่ทราบจำนวนการทำงานวนรอบแน่นอน โดยผนวกการ initialization , testing และ updating ไว้ด้วยกัน มีรูปแบบดังนี้

for Statement
รูปแบบ: for (initialization expression ; loop repetition condition ; update expression) statement ;

การทำงานของคำสั่ง for นี้เหมือนกับการทำงานของ while ในกรณีของ counter-controlled while loop เราจึงเรียกการทำงานของคำสั่ง for นี้ว่า counter หรือ indexed for loop

ตัวอย่างที่ 5.8 การพิมพ์ '*' จำนวน n ครั้ง

```
// Display N asterisks.
for (countStar = 0 ;  countStar < n ;  countStar ++)
    cout << '*' ;
```

การทำงานนั้น อันดับแรกจะมีการกำหนดให้ countStar มีค่าเริ่มต้นเท่ากับ 0 โดยการทำงานนี้จะกระทำเพียงครั้งแรกครั้งเดียว ต่อจากนั้นจะนำค่า countStar เปรียบเทียบกับค่า n ที่เก็บค่าสุดท้ายในการทำงาน ถ้าผลของการทำงานมีค่าเป็น true จะกระทำการพิมพ์ '*' ต่อจากนั้นจะเพิ่มค่า countStar ขึ้น 1 และนำค่าตัวนับที่เปลี่ยนแปลงเปรียบเทียบกับค่า n ถ้าเป็น true จะกระทำซ้ำในคำสั่งที่กำหนด และเพิ่มค่าขึ้นอีก 1 ทุกครั้งในการทำงานแต่ละรอบ จนกระทั่งค่าของตัวนับมีค่าเท่ากับค่า n จะหยุดการทำซ้ำ เพื่อไปทำคำสั่งต่อไปทันที

ตัวอย่างที่ 5.9 ฟังก์ชันหาค่าแฟคตอเรียล

```
// Computes factorial (n!)
// Pre: n is greater than or equal to zero
int factorial(int n)
{
    int product ;           // accumulator for product computation
    product = 1;
    // Computes the product n x (n-1) x (n-2) x ... x 2
    for (int i = n ; i > 1 ; i - - )
        product = product * i ;
    // Returns  function result
    return product ;
}
```

การเรียกใช้ fac = factorial(5) ;

การทำงานจะมีการส่งผ่านค่า 5 ไปให้กับ formal parameter n มีผลให้ภายในฟังก์ชันมีการควบคุมให้มีการทำงานซ้ำ 4 รอบ โดยกำหนดให้ตัวนับรอบมีค่าเริ่มต้นเท่ากับ 5 และลดลงรอบละ1 มีค่าเป็น 4 ,3 , 2 และเมื่อมีค่าเท่ากับ 1 จะหยุดการกระทำซ้ำ โดยการทำงานแต่ละรอบจะทำการนำค่าของตัวนับรอบไปคูณสะสม ดังนี้

i	product
5	1 * 5
4	1 * 5 * 4
3	1 * 5 * 4 * 3
2	1 * 5 * 4 * 3 * 2
1	หยุดกระทำซ้ำ

และส่งค่าผลคูณ ในที่นี้คือ 5! มีค่าเท่ากับ 120 กลับไปให้กับตัวแปร fac

ตัวอย่างที่5.10 โปรแกรมการแปลงองศาเซลเซียสเป็นองศาฟาเรนไฮน์

```
// File : temperatureTable.cpp
// Conversion of celsius to fahrenheit temperature

#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    const int CBEGIN = 10 ;
    const int CLIMIT = -5 ;
    const int CSTEP = 5 ;
    float fahrenheit ;

    // Display the table heading.
    cout <<  "Celsius" << "          Fahrenheit" << endl ;

    // Display the table.
    for (int celsius = CBEGIN ; celsius>=CLIMIT; celsius -= CSTEP )
    {
      fahrenheit = 1.8 * celcius + 32.0 ;
      cout << setw(5) << celsius << setw(15) << fahrenheit << endl ;
    }
    return 0 ;
}
```

ทดสอบ

Celsius	Fahrenheit
10	50.00
5	41.00
0	32.00
-5	23.00

การทำงานของโปรแกรมนี้เป็นการแปลงองศาเซลเซียสเป็นองศาฟาเรนไฮน์ โดยกำหนดค่าเริ่มต้น(CBEGIN) เท่ากับ 10 และค่าสุดท้าย(CLIMIT)เป็น –5 ในการทำงานวนรอบค่าของ celsius จะมีค่าเริ่มต้นเท่ากับ 10 จะทำการแปลงค่าให้เป็นองศาฟาเรนไฮน์มีค่าเท่ากับ 50.00 และพิมพ์ออกมาทางจอภาพต่อจากนั้นค่าของ celsius จะลดค่าลงครั้งละ 5 จากคำสั่ง celsius -= CSTEP ในการทำงานแต่ละรอบ จนกระทั่ง celsius มีค่าเท่ากับ –10 จึงหยุดการทำซ้ำ

ตัวอย่างที่ 5.11 การแบ่งกลุ่มเลขจำนวนเต็ม

โปรแกรมนี้จะให้ผู้ใช้ป้อนกลุ่มของเลขจำนวนเต็มใดๆ 20 จำนวนทางแป้นพิมพ์ นำข้อมูลต่างๆ เหล่านั้นมาแบ่งกลุ่มเพื่อนับจำนวนว่า มีเลขจำนวนเต็มคู่(even) เลขจำนวนเต็มคี่(odd) และเลขศูนย์(zero) อยู่อย่างละกี่จำนวน

```
//*****************************************************************************************
// Program: Counts zeros ,odds , and evens
// This program counts the number of odd and even numbers.
// The program also counts the number of zeros.
//*****************************************************************************************
#include <iostream>
#include <iomanip>

using namespace std;
const int N=20 ;
```

```
int main()
{
   // Declare variables
    int counter ;                          // loop control variable
    int number ;                           // variable to store the new number
    int zeros = 0;
    int odds = 0 ;
    int evens = 0 ;

    cout << "Please enter " << N <<" integers  Positive, negative, or zeros." << endl;
    cout << "the numbers you entered are: " << endl;

    for(counter = 1 ; counter<=N; counter++)
    {
       cin >> number ;
       cout << number << "  " ;
       switch (number %2)
       {
       case 0 : evens++;
                if (number == 0)
                   zeros+=;
                break;
       case 1 :
       case -1 :    odds++ ;
       } // end switch
    } // end for loop
```

```
    cout << endl ;
    cout << "There are " << evens << " evens, which alse includes " << zeros
         << " zeros." << endl;
    cout <<"The number of odd numbers is : " << odds << endl ;

    return 0 ;
}
```

ทดสอบ

```
Please enter 20 integers, positive, negative , or zeros.
The numbers you entered are :

0  0  -2  -3  -5  6  7  8  0  3  0  -23  -8  0  2  9  0  12  67  54
0  0  -2  -3  -5  6  7  8  0  3  0  -23  -8  0  2  9  0  12  67  54
There are 13 evens, which also includes 6 zeros.
The number of odd numbers is : 7
```

การทำงานของโปรแกรมนี้นั้นจะให้ผู้ใช้ป้อนเลขจำนวนเต็มใดๆ 20 จำนวนทางแป้นพิมพ์ เราใช้คำสั่ง for ทำงานวนรอบทั้งหมด 20 ครั้ง โดยในแต่ละรอบจะอ่านข้อมูลทีละค่านำมาตรวจสอบโดยใช้คำสั่ง switch เพื่อเลือกกรณีที่มีค่าเป็น true ซึ่งตัวเลือก(selector) จะทำการหาค่าผลหารของตัวเลข โดยนำค่า 2 เป็นตัวหารจะเหลือเศษที่เป็นไปได้ 3 กรณีคือ –1 , 0 และ 1

ถ้าเลขจำนวนใดมีเศษเท่ากับ 0 แสดงว่าอาจจะเป็นจำนวนศูนย์ หรือเลขจำนวนเต็มคู่ ดังนั้นต้องตรวจสอบอีกครั้งว่าเป็นเลขจำนวนใดแน่ เพื่อเพิ่มค่าตัวนับของกลุ่มนั้น แต่ในกรณีที่เป็นค่า 1 หรือ –1 แสดงว่าเป็นเลขจำนวนเต็มคี่ จะเพิ่มตัวนับของกลุ่มนี้ หลังจากปฏิบัติงานจนครบ 20 รอบแล้วโปรแกรมจะแสดงจำนวนของเลขจำนวนเต็มแต่ละกลุ่มออกทางจอภาพ

ตัวอย่างที่ 5.12 การหาค่าที่มากที่สุดของกลุ่มตัวเลข 10 จำนวน

สำหรับตัวอย่างนี้มีการใช้คำสั่ง for เรียกฟังก์ชัน larger ทำงาน โดยหาค่าที่มากที่สุดของเลข 10 จำนวนใดๆ

```
// Program Largest
# include <iostream>
using namespace std;

double larger(double x , double y) ;
int main()
{
    double num ;   // variable to hold the current number
    double max ;   // variable to hold the larger number
    int count ;       // loop control variable

   cout << "Enter 10 numbers. " << endl ;
   cin >> num ;
   max = num ;

   for (count = 1 ; count < 10 ; count++)
   {
       cin >> num ;
       max = larger(max , num ) ;
   }
   cout << "The larger number is " << max << endl ;
   return 0 ;
} // end main
```

```
double larger(double x , double y )
{
    if ( x>= y)
        return x ;
    else
        return y ;
}
```

ทดสอบ

```
Enter 10 numbers.
10 56 73 42 22 67 88 26 62 11
The largest number is 88
```

การทำงานนั้นจะมีการกำหนดค่าของข้อมูลตัวแรกให้มีค่ามากที่สุดก่อน ต่อจากนั้นนำมาเปรียบเทียบกับข้อมูลตัวต่อไปซึ่งมีการรับข้อมูลภายในลูปคำสั่ง for ต่อจากนั้นมีการเรียกใช้ฟังก์ชัน larger โดยส่งผ่านค่า max และ num ไปยังฟังก์ชัน โดยเปรียบเทียบค่าที่มากที่สุดของค่าที่เก็บในอากิวเมนต์ส่งค่าที่มากที่สุดกลับมาให้แก่ max เพื่อนำไปเปรียบเทียบกับข้อมูลตัวต่อไป ในการทำงานรอบต่อไป

ตัวอย่างที่ 5.13 การแบ่งกลุ่มตัวเลข

ตัวอย่างนี้เป็นการแสดงการแก้ปัญหาโปรแกรมโดยมีการสร้างฟังก์ชันในหลายๆลักษณะเพื่อรับค่าจำนวนกลุ่มตัวเลข ต่อจากนั้นนำไปจัดกลุ่มว่าเป็นเลขจำนวนเต็มคู่หรือเลขจำนวนเต็มคี่หรือจำนวนเต็มศูนย์ โดยนับจำนวนของแต่ละกลุ่ม และพิมพ์ผลลัพธ์ออกมาทางจอภาพ

```
// Program : Classify Numbers
// This program counts the number of ZEROS ,odd , and even  numbers

#include <iostream>
using namespace std;
const int N = 20 ;

// function prototypes
void initialize(int& zeroCount, int& oddCount , int& evenCount);
void getNumber(int& num);
void classifyNumber(int num ,int&zeroCount ,int& oddCount, int& evenCount);
void printResults(int zeroCount , int oddCount , int evenCount);

int main()
{
   // variable declaration
   int  counter ;    // loop control variable
   int number ;    // variable to store a number
   int zeros ;      // variable to store the number of zeros
   int odds;       //  variable to store the number of odd integers
   int evens;      // variable to store the number of even integers

   initialize(zeros , odds , evens) ;

   cout << "Please endter " << N << " integers." << endl ;
   cout << "The numbers you entered are : " << endl;
```

```
    for (counter = 1 ; counter <= N ; counter++)
    {
        getNumber(number);
        cout <>< number << " " ;
        classifyNumber(number , zeros , odds , evens) ;
    } // end for loop
    cout << endl ;
    printResults (zeros , odds ,evens);

    return 0 ;
}

void  initialize(int& zeroCount , int& oddCount , int& evenCount)
{
    zeroCount = 0 ;
    oddCount = 0 ;
    evenCount = 0;
}

void getNumber(int& num)
{
    cin >> num ;
}

void classifyNumber(int num ,int& zeroCount , int& oddCount , int& evenCount)
{
    switch (num%2)
```

```
    {
    case 0 : evenCount++;
             if (num == 0)
                 zeroCount++;
             break ;
    case 1 :
    case –1 : oddCount++;
    } // end switch
}

void printResults(int zeroCount , int oddCount , int evenCount)
{
    cout << "There are " << evenCount << "evens, "
         << "which also includes " << zeroCount << "zeros" << endl;
    cout << " The number of odd numbers is : " << oddCount << endl ;
}
```

ทดสอบ

```
Please enter 20 integers.

The numbers you entered are:
0  0 12 23 45 7 -2 -8 -3 -9 4 0 1 0 -7 23 -24 0 0 12
0  0 12 23 45 7 -2 -8 -3 -9 4 0 1 0 -7 23 -24 0 0 12

There are 12 evens, which also includes 6 zeros
The number of odd numbers is : 8
```

การทำงานของโปรแกรมจะเริ่มจากให้ผู้ใช้ป้อนค่าเป็นจำนวนของตัวเลขทางแป้นพิมพ์ โดยมีการ call –by – reference ซึ่งการแก้ปัญหานี้มีผลลัพธ์จากการทำงานเพียงค่าเดียวอาจแก้ปัญหาโดยสร้างเป็นฟังก์ชั่นที่มีการส่งผ่านค่ากลับได้ดังนี้

```
int getNumber()
{
  int num ;
  cin >> num ;
  return num ;
}
```

ดังนั้นจึงต้องมีการปรับเปลี่ยนในส่วนของการเรียกใช้

จากเดิม getNumber(number);

เปลี่ยนเป็น number = getNumber();

นอกจากนี้ยังต้องเปลี่ยนต้นแบบของฟังก์ชันในส่วนก่อนการฟังก์ชันหลักด้วย

จากเดิม void getNumber(int& num);

เปลี่ยนเป็น int getNumber();

จะเห็นได้ว่าการแก้ปัญหาโปรแกรมหนึ่งๆนั้นขึ้นอยู่กับวิจารณญาณ ความเชี่ยวชาญประสบการณ์ของโปรแกรมเมอร์ในการออกแบบโปรแกรม โดยการออกแบบโปรแกรมที่ดีนอกจากความถูกต้อง ความน่าเชื่อถือแล้ว การแบ่งปัญหาออกเป็นฟังก์ชันย่อยๆที่เหมาะสมก็เป็นส่วนที่สำคัญอีกประการหนึ่งในการแก้ปัญหาเพราะถ้ามีการออกแบบฟังก์ชันย่อยๆที่ดีจะทำให้ง่ายต่อการทำความเข้าใจ ง่ายต่อการแก้ไข และง่ายต่อการบำรุงรักษา โดยถ้าการทำงานของโปรแกรมนี้เกิดความผิดพลาด โปรแกรมเมอร์สามารถติดตามหรือสืบค้นได้ง่าย การแก้ไขการทำงานในแต่ละฟังก์ชันจะไม่ส่งผลกระทบต่อการทำงานในส่วนอื่นๆ ทำให้เราสามารถแจกจ่ายงานให้แก่โปรแกรมเมอร์ได้หลายๆคนเพื่อช่วยกันในการพัฒนาโปรแกรม

ตัวอย่าง 5.14 Pig Latin Strings

ตัวอย่างนี้เป็นการแก้ปัญหาที่เกี่ยวข้องกับรูปแบบของข้อความที่ต้องการ กล่าวคือผู้ใช้งานต้องป้อนข้อความใดๆทางแป้นพิมพ์ โปรแกรมจะทำการปรับเปลี่ยนให้อยู่ในรูปแบบของ Pig Latin โดยมีกฎดังนี้

1. ถ้าข้อความที่ผู้ใช้ป้อนขึ้นต้นด้วยอักษรภาษาอังกฤษที่เป็นสระ ให้เพิ่มข้อความ "-way" ที่ท้ายข้อความที่ป้อน ตัวอย่าง ถ้าผู้ใช้ป้อนข้อความ "eye" ผลลัพธ์จะพิมพ์ข้อความ "eye-way" ออกทางจอภาพ
2. ถ้าข้อความไม่ได้ขึ้นต้นด้วยอักษรภาษาอังกฤษที่เป็นสระ ให้เพิ่มอักขระ "-" ท้ายข้อความ และนำอักขระตัวแรกของข้อความหมุนมาต่อท้ายข้อความ ต่อจากนั้นพิจารณาอักขระตัวต่อไปถ้าไม่ใช่ตัวอักษรภาษาอังกฤษที่เป็นสระ ให้หมุนนำไปต่อท้ายข้อความ จนกระทั่งตัวอักขระที่พิจารณาเป็นอักษรภาษาอังกฤษที่เป็นสระ ให้เพิ่มอักขระ "ay" ต่อท้ายข้อความ ตัวอย่าง ถ้าผู้ใช้ป้อน "There" ผลลัพธ์จากการทำงานคือ "ere-thay"
3. สำหรับข้อความบางลักษณะ เช่นผู้ใช้ป้อน "by" ซึ่งไม่มีตัวอักษรภาษาอังกฤษที่เป็นสระ จะถือว่าตัวอักษร "y" เป็นสระด้วยตัวหนึ่ง ดังนั้นผลลัพธ์จากการทำงานคือ y-bay
4. ถ้าข้อความที่ป้อนเป็นตัวเลขในตัวอักษรแรก เช่น ผู้ใช้ป้อน "1234" ผลของการทำงานจะนำข้อความนั้นต่อท้ายด้วย "-way" เป็น "1234-way"

```
#include <iostream>
#include <string>
using namespace std;

bool isVowel(char ch);
string rotate(string pStr);
string pigLatinString(string pStr);

int main()
{
    string str;
```

```
    cout << "Enter a string: " ;
    cin >> str ;
    cout << endl ;
    cout << "The pig Latin form of " << str << " is: " << pigLatinString(str) << endl ;
    return 0 ;
}
bool isVowel(char ch)
{
    switch (ch)
    {
    case 'A' : case 'E' : case 'I' : case 'O' : case 'U' : case 'Y' :
    case 'a'  : case 'e' : case 'I' : case 'o'  : case 'u'  : case 'y' :
    case '0' : case '1' : case '2' : case '3' : case '4' : case '5' :
    case '6' : case '7' : case '8' : case '9'  : return true;
    default   : return false ;
    }
}
string rotate(string pStr)
{
    string ::size_type len = pStr.length() ;
    string rStr ;
    rStr = pStr.substr(1 , len -1) + pStr[0] ;
    return rStr ;
}
string pigLatinString(string pStr)
{
    string::size_type len ;
```

```
    bool foundVowel ;
    string::size_type counter ;
    if (isVowel(pStr[0]))
        pStr = pStr + "-way" ;
    else
    {
        pStr = pStr + '-' ;
        pStr = rotate(pStr) ;
        len = pStr.length() ;
        foundVowel = false ;

        for (counter = 1 ; counter < len-1 ; counter++)
            if (isVowel(pStr[0]))
            {
                    foundVowel = true ;
                    break ;
            }
            else
                    pStr = rotate(pStr) ;

     if  (!foundVowel)
         pStr = pStr  + "-way" ;
    else
         pStr = pStr + "ay" ;
    }
    return pStr ;
}
```

ทดสอบ 1

Enter a string: eye

The pig Latin form of eye is: eye-way

ทดสอบ 2

Enter a string: There

The pig Latin form of There is: ere-Thay

ทดสอบ 3

Enter a string: why

The pig Latin form of why is: y-whay

ทดสอบ 4

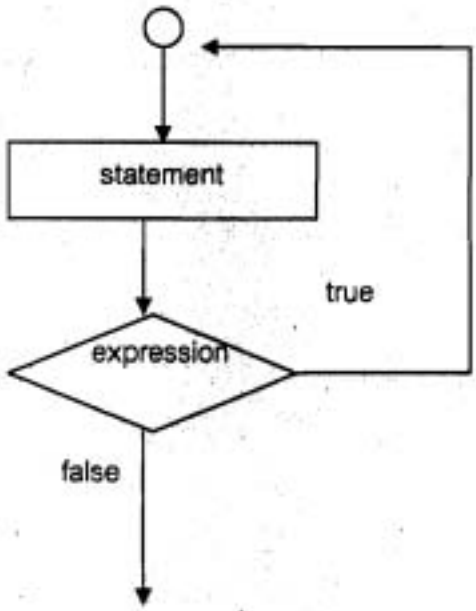
Enter a string: 123456

The pig Latin form of 123456 is: 123456-way

การแก้ปัญหาโปรแกรมนี้มีการสร้าง 3 ฟังก์ชัน โดยฟังก์ชัน isVowel เป็นฟังก์ชันในการตรวจสอบตัวอักขระว่าเป็นสระหรือไม่กรณีถ้าใช่ จะส่ง true กลับไปยังจุดเรียกใช้ ในกรณีอื่นๆจะส่ง false สำหรับ ฟังก์ชัน rotate เป็นฟังก์ชันในการนำตัวอักษรตัวแรกของข้อความที่กำหนดย้ายไปไว้ในลำดับสุดท้ายแทน ส่วนฟังก์ชัน pigLatinString เป็นฟังก์ชันในการแปลงข้อความที่กำหนดให้เป็น Pig Latin form หลักการทำงานนั้นจะนำตัวอักขระแต่ละตัวที่ผู้ใช้ป้อนมาตรวจสอบโดยเริ่มจากอักขระตัวแรก ในกรณีที่เป็นสระหรือตัวเลข ให้หลุดจากการตรวจสอบและนำ "-way" ต่อท้ายข้อความนั้น แต่ถ้ากรณีอื่นๆให้หมุนให้ตัวอักษรในลำดับแรกที่ไม่ใช่สระไปต่อท้ายข้อความเดิมหมุนเวียนไปจนกระทั่งพบสระ จึงหยุดกระทำงาน และนำ "-ay" ต่อท้ายข้อความที่เป็น ผลลัพธ์ออกทางจอภาพ

5.3 คำสั่ง do..while

เป็นคำสั่งทำงานซ้ำโดยมีการตรวจสอบเงื่อนไขการทำงานหลังจากปฏิบัติงาน ซึ่งการทำงานในลักษณะนี้จะมีการทำงานอย่างน้อยก่อน 1 รอบ ดังผังโปรแกรมดังนี้



do-while Statement

รูปแบบ:

```
do
    statement ;
while (expression) ;
```

การใช้คำสั่ง do..while นี้เหมาะสำหรับใช้แก้ปัญหาที่ต้องการตรวจสอบค่าของข้อมูลเข้า (input)ว่าอยู่ในขอบเขตที่ต้องการหรือไม่ ถ้าผู้ใช้ป้อนข้อมูลผิดโปรแกรมจะย้อนกลับให้ป้อนข้อมูลใหม่ที่ถูกต้อง หรืออาจเป็นเมนูเลือกการทำงาน หรือสารสนเทศเพื่อถามผู้ใช้ว่าต้องการทำซ้ำหรือไม่เป็นต้น

ตัวอย่างที่ 5.15 ฟังก์ชันในการตรวจสอบค่า

ฟังก์ชันนี้เป็นการตรวจสอบค่าที่ผู้ใช้ป้อนว่าอยู่ในช่วงที่ต้องการหรือไม่ โดยให้ค่าที่น้อยที่สุดเก็บในตัวแปร min และค่าที่มากที่สุดเก็บในตัวแปร max ฟังก์ชันจะส่งค่าตัวเลขที่อยู่ในช่วงของข้อมูลที่กำหนดกลับไปยังจุดเรียกใช้

```
// Returns the first integer in the range min through max
// Pre : min <= max
int getIntRange(int min , int max)                    // range boundaries
{
    int  nextInt ;                                    // next number read

    // enter data until a number between min and max is read
    do
    {
        cout << "Enter a number between " << min << " and " << max << " : " ;
        cin  >> nextInt ;
    } while ((nextInt < min) || (nextInt > max)) ;

    return nextInt ;
}
```

การเรียกใช้ `Value = getIntRange(1 , 100 ) ;`

การทำงานของฟังก์ชันจะอ่านค่าจำนวนเต็ม 1 จำนวนจากผู้ใช้ เพื่อนำมาตรวจสอบเงื่อนไขในการทำงานในกรณีที่อยู่นอกขอบเขตที่กำหนดในที่นี้อยู่ระหว่างค่า1 ถึงค่า 100 ในกรณีที่เงื่อนไขเป็น true จะย้อนกลับไปกระทำซ้ำโดยอ่านค่าจำนวนเต็มค่าใหม่เพราะค่าที่ป้อนอยู่นอกขอบเขตที่กำหนด ต่อจากนั้นจะนำค่าใหม่ที่ป้อนมาตรวจสอบว่าอยู่ในขอบเขตหรือไม่ ถ้าตรวจสอบเงื่อนไขแล้วมีค่าเป็น false จะหยุดการทำงานซ้ำ แสดงว่าค่าที่ได้อยู่ในขอบเขตที่กำหนด จะส่งตัวเลขจำนวนเต็มนั้นกลับไปยังจุดเรียกใช้

ตัวอย่างที่ 5.16 โปรแกรมเลือกการทำงาน

คำสั่ง do..while เหมาะสำหรับการแก้ปัญหาที่มีการใช้เมนูเพื่อควบคุมการทำงานที่ต้องการ โดยมีทางเลือกให้แก่ผู้ใช้งาน เช่น

```
List of edit operations :
D - Delete a substring.
F – Find a string .
I – Insert a string
R – Replace a string
Q – Quit
Enter D , F, I , R , Q as your selection :
```

เมื่อโปรแกรมทำงานจะมีเมนูให้ผู้ใช้เลือกการทำงานที่ต้องการ โดยการทำงานจะปฏิบัติงานตามทางเลือกที่ผู้ใช้เลือกเมื่อกระทำงานจบ โปรแกรมจะวนรอบเพื่อให้ผู้ใช้เลือกเมนูที่ต้องการกระทำงานอีก การทำงานจะวนรอบจนเมื่อผู้ใช้เลือก Q จึงจะหยุดการทำงาน ดังรูปแบบดังนี้

```
do
   Display the menu
   Read the user's choice
   Perform the user's choice
   Display the edited string
while the choice is not Quit
```

เราสามารถเขียนโปรแกรมหลักในการทำงานได้ดังนี้

```
#include <iostream>
#include <string>
using namespace std;
```

```
// Insert function subprogram prototypes here.

int main()
{
    const char SENTINAL = 'Q' ;
    char choice ;                                   // input –edit operation
    string textString ;                             // input/output – string to edit

    cout << "Enter string to edit : " ;
    getline(cin , textString) ;

    do
    {
       displayMenu();                               // display the menu & prompt
       cin >> choice ;
       textString = edit(textString ,chioce);       // edit string
       cout << "New string is " << textString << endl ;
    } while (choice != SENTINEL);

    return 0 ;
}

// insert function subprograms here.
```

โปรแกรมนี้ยังไม่สมบูรณ์ต้องมีการเพิ่มโปรแกรมย่อยไปในโปรแกรม ในที่นี้คือฟังก์ชัน displayMenu() ซึ่งทำหน้าที่แสดงสารสนเทศต่อผู้ใช้เป็นเมนูการทำงาน และฟังก์ชัน edit ซึ่งเป็นฟังก์ชันในการแก้ไขข้อความที่ต้องการ

ตัวอย่างที่ 5.17 โปรแกรมหาค่าที่มากที่สุด

โปรแกรมนี้เป็นโปรแกรมที่หาค่าที่มากที่สุดโดยใช้คำสั่ง do..while

```
// File: largest.cpp
// Finds the largest number in a sequence of integer values

#include <iostream>
#include <limits>                          // needed for INT_MIN
using namespace std;

int main()
{
    int itemValue ;                        // input – each data value
    int largestSoFar ;                     // output – largest values so far
    int minValue ;                         // the smallest integer

    // Initialize largestSoFar to the smallest integer .
    minValue = INT_MIN ;
    largestSoFar = minValue ;

    // Save the largest number encountered so far.
    cout << "Finding the largest value in a sequence :" << endl ;
    do
    {
       cout << "Enter an integer or " << minValue << " to stop : " ;
       cin >> itemValue ;
```

```
    if (itemValue > largestSoFar)
        largestSoFar = ItemValue ;                    // save new largest number
    } while (itemValue != minValue ) ;

    cout << "The largest value entered was " << largestSoFar << endl ;

    return 0 ;
}
```

ทดสอบ

```
Finding the largest value in a sequence :
Enter an integer or –2147483648 to stop : -999
Enter an integer or –2147483648 to stop : 500
Enter an integer or –2147483648 to stop : 100
Enter an integer or –2147483648 to stop : -21474863648
The largest value entered was 500
```

การทำงานของโปรแกรมนี้มีการใช้ฟังก์ชัน INT_MIN ซึ่งให้ค่าจำนวนเต็มที่น้อยที่สุดที่เครื่องคอมพิวเตอร์สามารถกระทำงานได้ฟังก์ชันนี้บรรจุในคลาส limits จึงต้องมีการ include ไว้ก่อนนำมาใช้งาน โปรแกรมมีคำสั่งกำหนดค่านี้ให้แก่ตัวแปร minValue เพื่อใช้เป็น SENTINEL โดยโปรแกรมจะหยุดการทำซ้ำเมื่อผู้ใช้ป้อนค่า –21474863648 เป็นลำดับสุดท้าย การทำงานภายในลูปนั้นโปรแกรมจะให้ผู้ใช้ป้อนข้อมูลเป็นเลขจำนวนเต็มเป็นลำดับโดยนำไปตรวจสอบเพื่อหาว่าค่าใดที่มีค่ามากที่สุดโดยจะนำไปเก็บไว้ในตัวแปร largestSoFar ต่อจากนั้นตรวจสอบเงื่อนไขหลังจากตรวจสอบค่าในกรณีที่ไม่ใช่ค่า minValue จะย้อนกลับไปกระทำซ้ำให้ผู้ใช้ป้อนค่าใหม่และกระทำงานเป็นลำดับ วนรอบไปจนกระทั่งผู้ใช้ป้อนค่าที่เท่ากับค่าของ minValue โปรแกรมจะหยุดการทำซ้ำ หลุดจากการทำงานวนรอบ และพิมพ์ค่าที่มากที่สุดของกลุ่มข้อมูลชุดนี้ออกมาทางจอภาพ

การแก้ปัญหาในบางกรณีสามารถใช้ คำสั่งกระทำซ้ำซ้อนๆกันได้ โดยการทำงานนั้นจะกระทำเริ่มจาก ลูปนอกก่อน และ จะกระทำลูปใน ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 5.18 การใช้คำสั่ง for ซ้อน for

โปรแกรมนี้แสดงการทำงานของการใช้คำสั่ง for ซ้อนคำสั่ง for โดยการทำงานจะเริ่มจาก for ลูปนอกก่อน และกระทำในลูปในจะกระทั่งจบการทำงาน จึงย้อนกลับมาลูปนอก

```
// File : nestedLoops.cpp
// Illustrates a pair of nested for loops
#include <iostream>
#include <iomanip>
using namespace std ;

int main()
{
    // Display heading
    cout << setw(12) << "i" << setw(60 << "j" << endl ;
    for (int i = 0 ; i < 4 ; i ++ )
    {
        cout << "Outer" << setw(7) << i << endl ;
        for (int j = 0 ; j < i ; j ++ )
                cout << " Inner " << setw(10) << j << endl ;
    }  // end outer loop

    return 0 ;
}
```

ทดสอบ

	i	j
Outer	0	
Outer	1	
Inner		0
Outer	2	
Inner		0
Inner		1
Outer	3	
Inner		0
Inner		1
Inner		2

การทำงานของโปรแกรมจะถูกควบคุมโดยตัวแปรที่ใช้ในการนับรอบ โดยลูปนอกจะกระทำทั้งหมด 4 รอบ แต่ลูปในจะถูกควบคุมการทำงานด้วยค่าของตัวแปร i นั่นคือถ้า i = 0 ลูปในจะไม่กระทำงาน แต่ถ้า i = 1 ลูปในจะกระทำงาน 1 รอบ ถ้า i = 3 ลูปในจะกระทำงานซ้ำ 3 รอบ เมื่อค่า i = 4 จะหยุดการทำงาน

ตัวอย่างที่ 5.19 โปรแกรมพิมพ์ตารางผลคูณ

โปรแกรมนี้จะพิมพ์ค่าของผลคูณของค่าในแนวตั้งและแนวนอน โดยกำหนดให้ค่าในแนวตั้งและแนวนอนมีค่า เริ่มจาก 0 , 1 , 2 , ... 9 นำค่าของแนวตั้งและแนวนอนมาคูณกัน พิมพ์ออกมาในตำแหน่งตัดกันของแนวตั้งและแนวนอน

กำหนดให้ rowVal เก็บค่าของแนวนอน และ colVal เก็บค่าของแนวตั้ง การแก้ปัญหานี้เราใช้คำสั่ง for ซ้อน for โดยให้ลูปนอกทำงานของแนวนอนหรือเปรียบเสมือนแถว และให้ลูปในแทนการทำงานของแนวตั้งซึ่งเปรียบเสมือนสดมภ์นั่นเอง

```
// File: multiplication.cpp
```

```
// Displays the multiplication table
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    // Display table heading
    cout << "   |" ;
    for (int cilHead = 0 ; colHead < 10 ; colHead++)
        cout << setw(30 << colHead ;
    cout << endl ;
    cout << " ----------------------------------------------" << endl ;

    // Display table , row-by-row
    for (int rowVal = 0 ; rowVal < 10 ; rowVal++)
    {
        cout << setw(3) << rowVal << '|' ;

        // Display all columns of current row
        for ( int colVal = 0 ; colVal < 10 ; colVal++)
            cout << setw(3) <, rowVal * colVal ;
        cout << endl ;
    }

    return 0 ;
}
```

ทดสอบ

```
   | 0  1  2  3  4  5  6  7  8  9
---------------------------------------------
0| 0  0  0  0  0  0  0  0  0  0
1| 0  1  2  3  4  5  6  7  8  9
2| 0  2  4  6  8 10 12 14 16 18
3| 0  3  6  9 12 15 18 21 24 27
4| 0  4  8 12 16 20 24 28 32 36
5| 0  5 10 15 20 25 30 35 40 45
6| 0  6 12 18 24 30 36 42 48 54
7| 0  7 14 21 28 35 42 49 56 63
8| 0  8 16 24 32 40 48 56 64 72
9| 0  9 18 27 36 45 54 63 72 81
```

ตัวอย่าง 5.20 การพิมพ์รูปที่กำหนด

ต้องการพิมพ์รูปดังนี้ทางจอภาพ

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

เมื่อพิจารณารูปที่ต้องการแล้วมีการทำงานในลักษณะของแถวและสดมภ์ นั่นคือมีทั้งหมด 5 แถวและในแต่ละแถวมีอักขระ "*" เท่ากับจำนวนค่าของแถว เราสามารถนำคำสั่ง for มาแก้ปัญหานี้ได้ ดังนี้

```
// File : create pattern
#include <iostream>
using namespace std ;
```

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

  for (i = 1 ; i<= 5 ; i++)                    // Line 1
  {
      for (j=1 ; j <= i ; j++)                 // Line 3
          cout << "*" ;
      cout << endl ;
  }

  return 0 ;
}
```

ถ้าเปลี่ยนคำสั่ง for ของ Line 1 เป็นดังนี้

```
for (i=5 ; i >= 1 ; i-- )
```

ผลเป็นอย่างไร ?

ถ้าเปลี่ยนคำสั่ง for ของ Line 3 เป็นดังนี้

```
for (j = 1 ; j <= 5 ; j ++ )
```

ผลเป็นอย่างไร ?

สรุป

1. คำสั่งในการกระทำซ้ำที่ภาษา C++ เตรียมไว้ให้ได้แก่ คำสั่ง while , for และ do..while
2. รูปแบบของคำสั่ง while คือ

   while (expression)

   statement ;

   การทำงานของคำสั่งนี้จะตรวจสอบนิพจน์ตรรก(expression) ถ้ามีค่าเป็น true จะกระทำตามคำสั่ง(statement)ที่กำหนดและย้อนกลับไปตรวจสอบเงื่อนไขหรือนิพจน์ตรรก โดยกระทำคำสั่งซ้ำถ้าเป็น true และหยุดกระทำซ้ำเมื่อเงื่อนไขเป็น false
3. รูปแบบของคำสั่ง for คือ

   for (initialize statement ; loop condition ; update statement)

   statement ;

   การทำงานของคำสั่งนี้จะกำหนดค่าเริ่มต้นให้กับตัวแปรในการควบคุมการกระทำซ้ำ (initialize statement) ต่อจากนั้นจะตรวจสอบเงื่อนไขการทำงานโดยมีผลที่สามารถเป็นไปได้ทั้ง true และ false ในกรณีที่เป็น true จะกระทำงานตามคำสั่งที่กำหนด(statement) และเปลี่ยนค่าตัวแปรในการควบคุม(update statement) เพื่อไปตรวจสอบเงื่อนไขในการทำงานเพื่อกระทำงานในรอบต่อไป การทำงานจะกระทำเมื่อเงื่อนไขเป็น true และหยุดการกระทำงานซ้ำเมื่อเงื่อนไขเป็น false
4. รูปแบบของคำสั่ง do..while

   do

   statement

   while (expression) ;

   เป็นการกระทำงานซึ่งขึ้นอยู่กับเงื่อนไข โดยมีการตรวจสอบเงื่อนไขเมื่อทำงานไปแล้ว ดังนั้นการปฏิบัติงานนั้นต้องกระทำตามคำสั่ง(statement) ก่อนอย่างน้อย 1 ครั้ง จึงมาตรวจสอบเงื่อนไขถ้าเงื่อนไขเป็น true จะกระทำซ้ำในรอบต่อไป การทำงานจะวนรอบทำซ้ำจนกระทั่งเงื่อนไข เป็น false

แบบฝึกหัด

1. ผลลัพธ์ของส่วนของคำสั่งต่อไปนี้ เป็นอย่างไร

1.1
```
count = 1 ;
y = 100 ;
while (count < 100)
{
  y = y – 1 ;
  count ++ ;
}
cout << "y=" <, y << " and count = " << count << endl ;
```

1.2
```
num = 5;
while (num < 10)
{
  cout << num << " " ;
  num = num + 2 ;
}
cout << endl ;
```

1.3
```
ch = 'D' ;
while ( 'A' <= ch && ch <= 'Z')
   ch = static_cast<char>(static_cast<int>(ch)+1);
```

1.4 กำหนดให้ ข้อมูลเข้าเป็นดังนี้ 38 45 71 4 -1
```
cin >> sum ;
cin >> num ;
for (j=1 ; j<=3 ; j++)
{
   cin >> num ;
```

```
    sum = sum + num ;
  }
  cout << " Sum = " << sum << endl ;
```

1.5 กำหนดให้ข้อมูลเข้าเป็นดังนี้ 38 45 71 4 -1

```
  cin >> sum ;
  cin >> num ;
  while (num != -1 )
  {
    sum = sum + num ;
    cin >> num ;
  }
  cout << "Sum = " << sum << endl ;
```

1.6 กำหนดให้ข้อมูลเข้าเป็นดังนี้ 38 45 71 4 -1

```
  cin >> num ;
  sum =  num ;
  while (num != -1 )
  {
    cin >> num ;
    sum = sum + num ;
  }
  cout << "Sum = " << sum << endl ;
```

1.7 กำหนดให้ข้อมูลเข้าเป็นดังนี้ 38 45 71 4 -1

```
  sum = 0 ;
  cin >> num ;
  while (num != -1 )
  {
    sum = sum + num ;
```

```
        cin >> num ;
    }
    cout << "Sum = " << sum << endl ;
```

2. โปรแกรมต่อไปนี้แสดงผลลัพธ์เป็นอย่างไร

```
#include <iostream>
using namespace std;
int main()
{
   int x , y , z ;
   x = 4 ;  y = 5 ;
   z = y + 6 ;
   while (( z-x) % 40 != 0)
   {
      cout << z << "  " ;
      z = z+7 ;
   }
   cout << endl ;

   return 0 ;
}
```

3. โปรแกรมต่อไปนี้แสดงผลลัพธ์เป็นอย่างไร

```
#include <iostrem>
using namespace std ;
int main()
{
   int counter ;
```

```
for (counter = 7 ; counter <= 16 ; counter++)
    switch (counter %10)
    {
    case 0 :  cout << " , " ;
              break ;
    case 1 : cout << "OFTEN " ;
              break ;
    case 2 : case 8 : cout << " IS " ;
              break ;
    case 3 : cout << "NOT " ;
              break ;
    case 4 : case 9 : cout << "DONE " ;
              break ;
    case 5 : cout << "WELL" ;
             break ;
    case 6 : cout <<"  . " ;
              break ;
    case 7 : cout << "WHAT  " '
    default : cout << "Bad number. " ;
    }
  cout << endl ;
  return 0 ;
}
```

4. โปรแกรมต่อไปนี้แสดงผลลัพธ์เป็นอย่างไร

```
#include <iostream>
using namespace std ;
```

```
int main ()
{
   int total = 0 , count = 0 , number ;
   do
   {
      cin >> number ;
      total = total + number ;
      count ++ ;
   } while (number != -1);
   cout << "The number of data read is " << count << endl ;
   cout << "The sum of the numbers entered is " << total << endl ;

   return 0 ;
}
```

5. จงเขียนโปรแกรมพิมพ์สูตรคูณใดๆ โดยป้อนเลขที่ต้องการทางแป้นพิมพ์ 1 จำนวน

INPUT : 2

OUTPUT :

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
...

2 * 12 = 24
```

6. จงเขียนโปรแกรมภาษา C++ ในการหาผลรวมของเลขจำนวนเต็มที่หารด้วย 9 ลงตัว

   6.1 กำหนดให้เลขชุดนี้มีทั้งหมด 10 จำนวน

   6.2 กำหนดให้ SENTINEL มีค่าเท่ากับ –999 เพราะไม่ทราบจำนวนของข้อมูล

7. จงเขียนโปรแกรมหาค่าแฟคตอเรียลของเลขจำนวนเต็มใดๆ

   INPUT : 5

   OUTPUT : 5! = 120

8. จงเขียนโปรแกรมในการพิมพ์รูปแบบดังต่อไปนี้โดยใช้ nest loop

   1 2 3 4 5

   2 3 4 5 6

   3 4 5 6 7

   4 5 6 7 8

   5 6 7 8 9

9. จงเขียนโปรแกรมในการพิมพ์รูปแบบดังต่อไปนี้โดยใช้ nest loop

   * * * * *

   * * * *

   * * *

   * *

   *

10. จงเขียนโปรแกรมสร้างเมนูเพื่อให้ผู้ใช้เลือกการทำงานและปฏิบัติงานได้ดังนี้ดังนี้

    List of Process operations :

    C - Circle.

    T – Triangle .

    R – Rectangle

    S – Square

    Q – Quit

    Enter C , T, R , S , Q as your selection :

    เป็นเมนูให้ผู้ใช้เลือกเพื่อทำการคำนวณหาพื้นที่ของรูปแบบใดๆ อาจเป็นวงกลม หรือ สามเหลี่ยม หรือ สี่เหลี่ยมผืนผ้า หรือ สี่เหลี่ยมจัตุรัส โดยวนรอบเพื่อคำนวณพื้นที่ของรูปต่างๆ จนกระทั่งผู้ใช้เลือก Q จึงหยุดการทำงาน