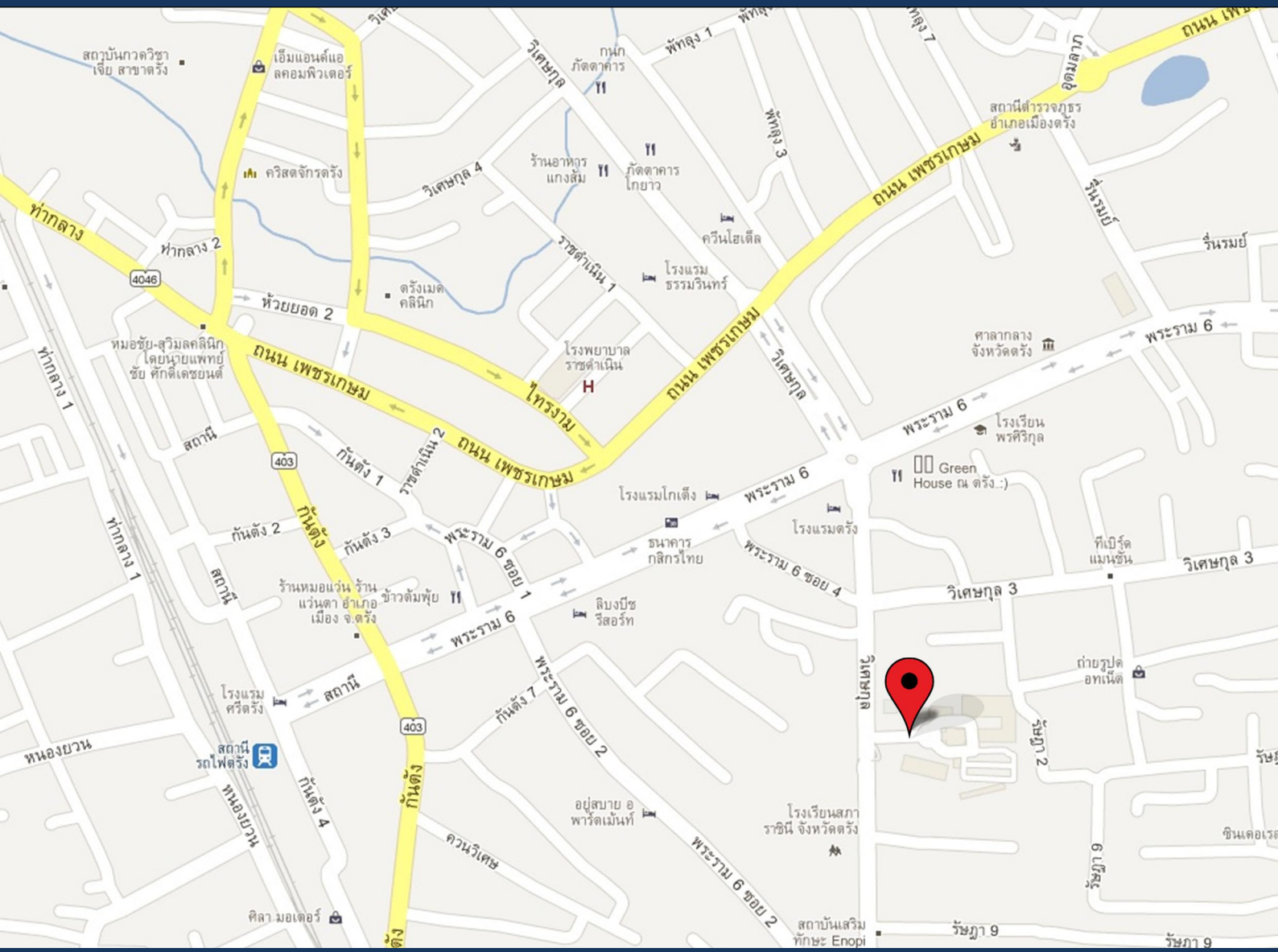
เอกสารประกอบการสอน

รายวิชา 3652204 การโปรแกรมคอมพิวเตอร์ทางธุรกิจเบื้องต้น

(Introduction to Business Computer Programming)



อ.วิทูรย์ คงผล www.dusittrang.com/professor/witoon

คำนำ

เอกสารประกอบการสอนรายวิชา วิชา 3652204 การโปรแกรมคอมพิวเตอร์ทางธุรกิจเบื้องต้น มุ่งเน้นให้ผู้เรียนมีความรู้ความความเข้าใจเกี่ยวกับศึกษาหลักการออกแบบและลงรหัสโปรแกรม รูปแบบไวยากรณ์ภาษา องค์ประกอบของภาษาคอมพิวเตอร์ ที่เกี่ยวข้องกับงานทางด้านธุรกิจ เกี่ยวกับคำสั่งนำเข้าและส่งออก(Input/Output) ชนิดของข้อมูลแบบต่าง ๆ รูปแบบการดำเนินการ(Operation) การวนรอบ(Looping) โปรแกรมย่อย(Procedure) ฟังก์ชัน(Function) และการใช้แฟ้มข้อมูลเบื้องต้น โดยการใช้ภาษาคอมพิวเตอร์ที่ทันสมัยในการฝึกพัฒนาโปรแกรมโดยได้มีการปรับปรุงเนื้อหาให้มีทันตามยุคสมัย และสามารถเรียนรู้ในเรื่องการเขียนโปรแกรมได้เข้าใจได้ง่ายยิ่งขึ้น ทั้งนี้เพื่อให้ผู้เรียนสามารถเรียนรู้และนำไปประยุกต์ใช้งานได้จริง

โดยเนื้อหาในเอกสารประกอบการสอนในเล่มนี้จะประกอบไปด้วยเนื้อหาทั้งหมด 10 บทเรียนได้แก่ ภาษาคอมพิวเตอร์และการพัฒนาโปรแกรมคอมพิวเตอร์ การออกแบบการทำงานโปรแกรมคอมพิวเตอร์ ความเป็นมาของภาษาซีและการใช้งานโปรแกรมภาษาซี โครงสร้างพื้นฐานของภาษาซี ชนิดข้อมูลและการประกาศตัวแปรในภาษาซี ขั้นตอนการพัฒนาโปรแกรมและคำสั่งการทำงานของภาษาซี โครงสร้างการควบคุมการทำงานแบบทางเลือก โครงสร้างการควบคุมการทำงานแบบวนรอบ ตัวแปรชนิดอาร์เรย์และฟังก์ชั่นและการสร้างฟังก์ชั่นในภาษาซี และเนื้อหาการเรียนรู้ทั้งหมดจะก่อให้เกิดองค์ความรู้แก่ผู้เรียนที่หลากหลาย รวมทั้งมีคำถามท้ายบท เพื่อให้ผู้เรียนสามารถเรียนรู้ทบทวนและทดสอบองค์ความรู้ที่ได้เรียนผ่านมา

ผู้จัดทำขอขอบพระคุณทุกท่านที่มีส่วนร่วม ไม่ว่าจะทางตรงหรือทางอ้อมต่อการจัดทำเอกสารประกอบการสอนในครั้งนี้ให้ลุล่วงสำเร็จไปด้วยดี และหากเอกสารประกอบการสอนเล่มนี้ผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้เพื่อแก้ไขต่อไปในโอกาสต่อไป

วิทูรย์ คงผล

มิถุนายน 2555

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญตาราง

ตารางที่ **หน้า**

บทที่ 1

ภาษาคอมพิวเตอร์และการพัฒนาโปรแกรมคอมพิวเตอร์

แผนการสอนประจำบท

หัวข้อเนื้อหา

1.1 ภาษาคอมพิวเตอร์
- 1.1.1 ภาษาเครื่อง (Machine language)
- 1.1.2 ภาษาแอสแซมบลี (Assembly language)
- 1.1.3 ภาษาปาสคาล (Pascal language)
- 1.1.4 ภาษาซี (C language)
- 1.1.5 ภาษาเบสิค (Basic language)
- 1.1.6 ภาษาจาวา (Java language)
- 1.1.7 ภาษา เอช ที เอ็ม แอล (HTML language)
- 1.1.8 ภาษา พี เอช พี (PHP language)

1.2 ระดับของภาษาคอมพิวเตอร์
- 1.2.1 ภาษาระดับต่ำ (Low level language)
- 1.2.2 ภาษาระดับสูง (High level language)

1.3 การพัฒนาโปรแกรมคอมพิวเตอร์
- 1.3.1 ขั้นการวิเคราะห์ปัญหา (Program Analysis)
- 1.3.2 ขั้นการออกแบบโปรแกรม (Program Design)
- 1.3.3 ขั้นการเขียนโปรแกรม (Program Coding)
- 1.3.4 การทดสอบโปรแกรม (Program Testing)
- 1.3.5 ขั้นการจัดทำคู่มือโปรแกรม (Program Manual)

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถอธิบายและเข้าใจภาษาคอมพิวเตอร์ที่มีใช้งานอยู่ในปัจจุบันได้
2. ผู้เรียนสามารถเข้าใจและอธิบายระดับของภาษาคอมพิวเตอร์ในระดับต่าง ๆ ได้
3. ผู้เรียนสามารถเข้าใจขบวนการการพัฒนาโปรแกรมคอมพิวเตอร์ในแต่ละขั้นตอนได้

วิธีการสอน

1. สอนแบบบรรยายโดยใช้เอกสารประกอบการสอน
2. สอนแบบบรรยายโดยใช้ Slide Power Point
3. สอนโดยให้ผู้เรียนเป็นส่วนกลาง โดยใช้วิธีถามตอบกับผู้เรียนระหว่างบรรยาย
4. ให้ผู้เรียนสรุปและอธิบายหัวข้อที่ผู้เรียนที่ได้รับมอบหมาย

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. Slide Power Point
3. ตัวอย่างโปรแกรมหรือบทความ

การวัดผลและประเมินผล

1. การทำแบบฝึกหัด
2. ความตั้งใจในชั้นเรียนและการร่วมมือในการนำเสนอหรืออภิปราย
3. สังเกตจากการตอบคำถามและการร่วมทำกิจกรรมในชั้นเรียน

บทที่ 1

ภาษาคอมพิวเตอร์และการพัฒนาโปรแกรมคอมพิวเตอร์

ในปัจจุบันคอมพิวเตอร์มีบทบาทอย่างยิ่งในการพัฒนางานด้านต่าง ๆ ไม่ว่าจะเป็นงานทางด้านการจัดการเอกสาร งานทางด้านกราฟฟิก งานทางด้านการเก็บฐานข้อมูล หรืองานทางด้านการพัฒนาโปรแกรมในระบบมือถือหรือแท็ปเล็ต จากตัวอย่างที่ได้กล่าวมาข้างต้นถือได้ว่าคอมพิวเตอร์มีบทบาทอย่างยิ่งในการพัฒนางานต่าง ๆ ที่ได้กล่าวมา และที่สำคัญอย่างยิ่งที่จะทำให้งานด้านต่าง ๆ สำเร็จนั้นก็คือ “โปรแกรมคอมพิวเตอร์” โดยจะเห็นได้ว่าในปัจจุบันโปรแกรมคอมพิวเตอร์มีอยู่อย่างมากมายหลากหลายโปรแกรม จนอาจจะกล่าวได้ว่ามีโปรแกรมคอมพิวเตอร์ที่มีใช้อยู่ในปัจจุบันมีมากมายจนไม่สามารถนับได้ทั้งหมด

ดังนั้นสิ่งสำคัญในการพัฒนาโปรแกรมคอมพิวเตอร์ขึ้นมานั้นก็คือ “ภาษาคอมพิวเตอร์” ภาษาคอมพิวเตอร์ถือได้ว่าเป็นหัวใจหลักในการสร้างโปรแกรมคอมพิวเตอร์ที่มีอยู่ในปัจจุบันนี้ขึ้นมา โดยภาษาคอมพิวเตอร์ที่มีอยู่ในปัจจุบันมีอยู่มากมายมายหลายภาษา สามารถนำมาพัฒนาโปรแกรมคอมพิวเตอร์ได้อย่างหลากหลายรูปแบบ ไม่ว่าจะเป็นการพัฒนาโปรแกรมบนเครื่องคอมพิวเตอร์ การพัฒนาโปรแกรมบนระบบเซิร์ฟเวอร์ หรือการพัฒนาโปรแกรมบนโทรศัพท์มือถือหรือแท็ปแล็ต เป็นต้น

1.1 ภาษาคอมพิวเตอร์

1.1.1 ภาษาเครื่อง (Machine language)

ภาษาเครื่องเป็นภาษาของเครื่องคอมพิวเตอร์ โดยเครื่องคอมพิวเตอร์จะสามารถทำงานได้ต้องเป็นภาษาเครื่องเท่านั้น ในงานทางคอมพิวเตอร์ทั่วไปจะไม่เขียนเป็นภาษาเครื่อง เพราะเครื่องคอมพิวเตอร์แต่ละเครื่องจะมีลักษณะแตกต่างกันไป ภาษาเครื่องจะใช้สำหรับผู้ที่ทำหน้าที่ทำงานเกี่ยวกับระบบชุดคำสั่ง เวลาเขียนต้องเขียนในรูปของเลขฐานสอง เมื่อเขียนเสร็จต้องคิดกลับมาเป็นตัวเลขฐานสิบ หรือเป็นตัวอักษร หรือพยัญชนะต่างๆ ที่จะให้เครื่องรับกลับเข้าไปเป็นเลขฐานสอง พร้อมกับประกอบเป็นคำสั่งและรวมกันเป็นชุดคำสั่ง การเขียนชุดคำสั่งภาษาเครื่องไม่จำเป็นต้องใช้ตัวแปลชุดคำสั่งหรือชุดคำสั่งควบคุมในช่วงการทำงาน

```
0001 0001 0000 0011 0000 0101 0101 0110 0111
1000 0000 0000 0011 0000 0101 0101 0101 0101
0110 0111 1000 0001 0001 0000 0011 0100 0100
0101 0110 0111 0000 0000 0000 0000 0100 0101
0110 0111 1000 0000 0000 0011 0000 0101 0101
0101 0101 0110 0111 0000 0000 0000 0011 0100
0100 0101 0000 0111 0000 0000 0000 0011 0000
0000 0000 0000 0000 0001 0001 0001 0010 0011
0011 0100 0101 0110 0111 1000 0001 0001 0100
0011 0100 0101 0110 0111 0000 0000 0000 0011
0011 0011 0100 0101 0110 0111 0000 0000 0000
0011 0000 0101 0101 0101 0101 0110 0111 1000
0000 0000 0011 0011 0000 0101 0101 0101 0101
0110 0111 0000 0000 0000 0000 0000 0101 0101
0101 0101 0110 0111 1000 0000 0000 0000 0100
0000 0110 0111 0000
```

ภาพที่ 1.1 ตัวอย่างชุดคำสั่งภาษาเครื่อง
(ที่มา http://disinfo.s3.amazonaws.com/wp-content/uploads/2013/03/binary-code.jpg)

1.1.2 ภาษาแอสแซมบลี (Assembly language)

ภาษาแอสแซมบลีหรือหลายคนอาจเรียกว่า ภาษาแอสแซมเบลอร์ โดยภาษาแอสแซมเบลอร์เป็นภาษาที่เขียนเป็นตัวพยัญชนะและตัวเลขฐานสิบ เช่นเดียวกับภาษาเครื่อง ต่างกันตรงที่ว่าภาษาแอสแซมเบลอร์เขียนเป็นตัวอักษรโดยไม่คำนึงว่าเลขฐานสองเป็นอย่างไร เมื่อถึงเวลาทำงานยังต้องใช้ชุดคำสั่งควบคุมเข้าช่วยอีกด้วย เมื่อเขียนชุดคำสั่งภาษาแอสแซมเบลอร์เสร็จจะได้ชุดคำสั่งที่เรียกว่าชุดคำสั่งเริ่มต้น ซึ่งจะต้องมีลักษณะที่ตัวแปลชุดคำสั่งภาษาแอสเซมเบลอร์จะรับได้เมื่อเอาชุดคำสั่งเริ่มต้นเข้าเครื่องคอมพิวเตอร์ ตัวแปลชุดคำสั่งภาษาแอสแซมเบลอร์จะแปลชุดคำสั่งภาษาแอสแซมเบลอร์เป็นชุดคำสั่งภาษาเครื่องต่อไป

```
     Assembly                                                  Machine
Label Oper Oper      Comment          Instr.     ILC    OpC Oper
      ation and                       Length           ode  and
---   ----- ---      -------          ------     ----   --- ----
If:   LODD  a0                       1          100*   0
      SUBD  a1                       1          101    3
      JZER  Then                     1          102    5
      JUMP  Next                     1          103    6
Then:LODD   a10                      1          104*   0
      ADDD  one                      1          105    2
EndT:JUMP   Next     /Unneccessary   1          106*   6
Next:       .....                               107*
```

ภาพที่ 1.2 ตัวอย่างชุดคำสั่งภาษาแอสแซมบลี
(ที่มา http://www.johnloomis.org/ece314/notes/carch/img357.gif)

1.1.3 ภาษาปาสคาล (Pascal language)

ภาษาปาสคาลสร้างขึ้นเพื่อใช้พัฒนาโปรแกรมคอมพิวเตอร์ให้เป็นระบบ โดยจะช่วยให้เขียนโปรแกรมได้รวดเร็วเป็นระบบมากยิ่งขึ้นและสามารถแก้ไขปรับปรุงได้ง่าย ทั้งนี้ภาษาปาสคาลเป็นภาษาที่ง่ายต่อการเข้าใจเพราะมีรูปแบบเป็นภาษาอังกฤษ และในการประมวลผลข้อมูลสามารถออกแบบคำสั่งให้มีคุณสมบัติที่แตกต่างกันได้หลายชนิด ทำให้การทำงานของโปรแกรมมีประสิทธิภาพที่ดี สามารถนำมาพัฒนาโปรแกรมคอมพิวเตอร์ในงานด้านต่าง ๆ ได้ คือ งานทางด้านธุรกิจ งานทางด้านคณิตศาสตร์ หรืองานวิทยาศาสตร์ เป็นต้น

```
procedure BeforePost;
var
  Qry: TcQuery;
begin
  Qry := CreateQry('SELECT Price FROM Article WHERE Number=:Number');
  try
    Qry.ParamByName('Number').AsInteger := FieldByName('Article').AsInteger;
    Qry.Execute;

    FieldByName('Price').AsCurrency :=
       Qry['Price'].AsCurrency * FieldByName('Quantity').AsInteger;
```

ภาพที่ 1.3 ตัวอย่างชุดคำสั่งภาษาปาสคาล
(ที่มา http://www.cathedron.com/Content/Afbeeldingen/Cat_Pascal_Script.gif)

1.1.4 ภาษาซี (C language)

ภาษาซีจัดเป็นภาษาที่มีลักษณะเป็นภาษาโครงสร้าง สามารถประยุกต์ใช้ได้กับงานในลักษณะต่างๆ ได้หลากหลาย เช่น ใช้พัฒนางานด้านโปรแกรมคอมพิวเตอร์ หรือนำไปใช้พัฒนาโปรแกรมทางด้านไมโครคอนโทรลเลอร์ เป็นต้น และภาษาซีเป็นภาษาที่ใกล้เคียงกับภาษามนุษย์ คือมีโครงสร้างภาษาเป็นภาษาอังกฤษซึ่งง่ายต่อการเข้าใจ โดยนักพัฒนาโปรแกรมจะสามารถเขียนโปรแกรมได้อย่างคล่องตัวโดยไม่มีข้อจำกัดในการวางตำแหน่งฟังก์ชั่นในโปรแกรม ภาษาซีจึงเป็นภาษาที่ง่ายต่อการเข้าใจและเป็นภาษาที่นิยมนำไปใช้งานในการพัฒนาโปรแกรมต่าง ๆ การสร้างโปรแกรมภาษาซีจะเริ่มจากการเขียนโปรแกรมต้นกำเนิด แล้วนำไปทำการแปลด้วยตัวแปลภาษาซีเกิดเป็นโปรแกรมประสงค์ หลังจากนั้นจึงนำโปรแกรมประสงค์ไปทำการเชื่อมโยง เพื่อให้เกิดเป็นโปรแกรมทำการที่สามารถทำงานได้อย่างรวดเร็ว ดังนั้นภาษาซีอาจจะถือได้ว่าเป็นภาษาหนึ่งที่ผู้เริ่มต้นในการพัฒนาโปรแกรมคอมพิวเตอร์ควรเรียนรู้เป็นพื้นฐานหลักในการเขียนโปรแกรมขั้นพื้น ทั้งนี้ภาษาซีจะเป็นภาษาโครงสร้างพื้นฐานของภาษาอื่นๆ เช่น ภาษา Java หรือภาษา PHP เป็นต้น

```
void mexFunction( int nlhs, mxArray *plhs[],
          int nrhs, const mxArray*prhs[] )

{
    double *yp;
    double *t,*y;
    mwSize m,n;

    /* Check for proper number of arguments */

    if (nrhs != 2) {
    mexErrMsgTxt("Two input arguments required.");
    } else if (nlhs > 1) {
    mexErrMsgTxt("Too many output arguments.");
    }
```

ภาพที่ 1.4 ตัวอย่างชุดคำสั่งภาษาซี
(ที่มา http://www.mathworks.com/help/matlab/matlab_external/msvs_breakpoint.gif)

1.1.5 ภาษาเบสิค (Basic language)

ภาษาเบสิคเป็นภาษาที่มีโครงสร้างที่ไม่ซับซ้อน ง่ายต่อการศึกษาและทำความเข้าใจเหมาะสำหรับ และเหมาะสำหรับผู้ที่เริ่มต้นเขียนโปรแกรมเช่นเดียวกับภาษาซี ภาษาเบสิคใช้ได้กับคอมพิวเตอร์ทุกรุ่นและเป็นภาษาที่มีความยืดหยุ่น สามารถแก้ไขปรับปรุง และพัฒนาโปรแกรมเพื่อนำไปใช้งานได้ง่ายและรวดเร็ว และเหมาะสมกับงานที่ต้องการผลลัพธ์อันรวดเร็ว โดยภาษาเบสิคมีชื่อเรียกต่าง ๆ กัน เช่น QBASIC , BASICA, GWBASIC , VISUAL BASIC เป็นต้น

```
Private Sub Command_1_Click()
On Error Resume Next
Dim strPathFile, yes, No As String
 yes = MsgBox("Do you want to exit", vbQuestion + vbYesNo, "Exit")
If yes = vbYes Then
Text_3.BackColor = vbWhite
Text_4.BackColor = vbWhite
Text_5.BackColor = vbWhite
End If
```

ภาพที่ 1.5 ตัวอย่างชุดคำสั่งภาษาเบสิค

1.1.6 ภาษาจาวา (Java language)

ภาษาจาวาปัจจุบันนี้ถือได้ว่าเป็นภาษาที่ได้รับความนิยมสูง เนื่องจากเป็นภาษาที่มีความยืดหยุ่นสูง สามารถเขียนโปรแกรมและใช้งานได้บนเครื่องคอมพิวเตอร์ทุกประเภท และระบบปฏิบัติการทุกรูปแบบ ในช่วงแรกที่เริ่มมีการนำภาษาจาวามาใช้งานจะเป็นการใช้งานบนเครือข่ายอินเทอร์เน็ตเป็นภาษาที่เน้นการทำงานบนเว็บ แต่ปัจจุบันสามารถสามารถนำมาประยุกต์สร้างโปรแกรมใช้งานทั่วไปได้ นอกจากนี้เมื่อเทคโนโลยีของการสื่อสารก้าวหน้าขึ้น เช่น โทรศัพท์มือถือ หรือแท็บแล็ต ภาษาจาวาถือเป็นภาษาหลักภาษาหนึ่งในการพัฒนาโปรแกรมบนโทรศัพท์มือถือหรือแท็บแล็ต ทั้งนี้ภาษาจาวาจะมีโครงสร้างภาษาใกล้เคียงกับภาษาซี

```
package calendarexample;

import java.util.Calendar;

public class Main {
    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        //int value of the year
        System.out.println(calendar.get(Calendar.YEAR));

        //int value of the month (0-11)
        System.out.println(calendar.get(Calendar.MONTH));

        //int value of the day of the month (1-31)
        System.out.println(calendar.get(Calendar.DAY_OF_MONTH));

        //int value of the day of the week (0-6)
        System.out.println(calendar.get(Calendar.DAY_OF_WEEK));
    }
}
```

ภาพที่ 1.6 ตัวอย่างชุดคำสั่งภาษาจาวา
(ที่มา https://netbeans.org/images_www/articles/70/java/javase-jdk7/converted.png)

1.1.7 ภาษา เอช ที เอ็ม แอล (HTML language)

ภาษา HTML เป็นภาษาคอมพิวเตอร์ที่ใช้ในการแสดงผลของเอกสารบนเว็บไซค์ หรือที่เรียกกันว่าเว็บเพจ ภาษา HTML เป็นภาษาที่สำหรับการสร้างหรือพัฒนาเว็บไซค์ โดยใช้ภาษา HTML สามารถพัฒนาได้โดยใช้โปรแกรม Text Editor ต่างๆ เช่น Notepad, Editplus หรือจะอาศัยโปรแกรมที่เป็นเครื่องมือช่วยสร้างเว็บไซค์ เช่น Microsoft FrontPage, Dream Weaver ซึ่งอำนวยความสะดวกในการสร้างหน้า HTML ส่วนการเรียกใช้งานหรือทดสอบการทำงานของเอกสาร HTML จะใช้โปรแกรมเว็บเบราเซอร์ เช่น Microsoft Internet Explorer , Mozilla Firefox, Safari, Opera, และ Netscape Navigator เป็นต้น

```
 </style>
</head>

<body>
 <table border="0" cellspacing="0" cellpadding="0">
  <tr>
   <td width="600" align="center"><img src="pic/escomp.gif" width="245"
   alt="EditPlus Text Editor"></td>
  </tr>
 </table>

 <table border="0" cellspacing="0" cellpadding="0">
  <tr>
   <td class="c1" width="110" align="center" valign="top">
   </td>

   <td width="5"></td>

   <td width="485" valign="top">
    <table width="100%" cellspacing="0" cellpadding="3" border="0">
     <tr>
     </tr>
    </table>
```

ภาพที่ 1.7 ตัวอย่างชุดคำสั่งภาษา HTML
(ที่มา http://akorra.com/wp-content/uploads/2011/07/HTML.png)

1.1.8 ภาษา พี เอช พี (PHP language)

ภาษาพีเอชพีเป็นภาษาที่เก็บอยู่ในไฟล์ที่เรียกว่าสคริปต์ (script) และเวลาใช้งานต้องอาศัยตัวแปลชุดคำสั่ง ตัวอย่างของภาษาสคริป เช่น JavaScript, Perl เป็นต้น ลักษณะของภาษาพีเอชพีที่แตกต่างจากภาษาสคริปต์แบบอื่น ๆ คือ ภาษาพีเอชพีได้รับการพัฒนาและออกแบบมา เพื่อใช้งานในการสร้างเอกสารแบบ HTML โดยสามารถสอดแทรกหรือแก้ไขเนื้อหาได้โดยอัตโนมัติ ดังนั้นภาษาพีเอชพีถือได้ว่าเป็นเครื่องมือที่สำคัญชนิดหนึ่ง ที่ช่วยให้เราสามารถสร้างเอกสารในรูปแบบ HTML ได้อย่างมีประสิทธิภาพและมีลูกเล่นต่าง ๆ มากยิ่งขึ้น

```php
<?php
interface IPerson    {
    public function Work();
}

class Student implements IPerson  {
    public function Work()    {
        return "The work of student is to study and conduct research.";
    }
}

class Employee implements IPerson {
    public function Work()    {
        return "The work of the employee is to improve productivity in an organization";
    }
}
echo PersonFactory::GetWorkDone('STUDENT');
echo "<hr/>";
echo PersonFactory::GetWorkDone('PROGRAMMER');
?>
```

ภาพที่ 1.8 ตัวอย่างชุดคำสั่งภาษา PHP
(ที่มา http://akorra.com/wp-content/uploads/2011/85/PHP.png)

1.2 ระดับของภาษาคอมพิวเตอร์

จากภาษาคอมพิวเตอร์ที่ได้ยกตัวอย่างไว้ข้างต้น เป็นแค่ส่วนหนึ่งเท่านั้นของภาษาคอมพิวเตอร์ที่มีการใช้งานอยู่ในปัจจุบัน ทั้งนี้ยังมีภาษาคอมพิวเตอร์อีกมากมายที่มีใช้งานอยู่ และจากการยกตัวอย่างของภาษาคอมพิวเตอร์จะสังเกตได้ว่าแต่ละภาษาคอมพิวเตอร์จะมีหน้าที่การทำงานที่มีความโดดเด่นแตกต่าง ๆ กันไป ทั้งนี้ยังมีสิ่งหนึ่งที่ผู้พัฒนาโปรแกรมควรทราบคือ ระดับของภาษาคอมพิวเตอร์ โดยระดับของภาษาคอมพิวเตอร์สามารถแบ่งออกเป็นกว้าง ๆ ได้เป็น 2 ระดับ ดังนี้

1.2.1 ภาษาระดับต่ำ (Low level language)

ภาษาระดับต่ำจะเป็นภาษาที่ใช้เลขฐานสองแทนข้อมูล และคำสั่งต่าง ๆ ทั้งหมดจะเป็นภาษาที่ขึ้นอยู่กับชนิดของเครื่องคอมพิวเตอร์ หรือหน่วยประมวลผลที่ใช้ นั่นคือแต่ละเครื่องก็จะมีรูปแบบของคำสั่งเฉพาะของตนเอง ซึ่งนักคำนวณและนักเขียนโปรแกรมในสมัยก่อนต้องรู้จักวิธีที่จะรวมตัวเลขเพื่อแทนคำสั่งต่าง ๆ ทำให้การเขียนโปรแกรมยุ่งยากมาก ตัวอย่างของภาษาระดับต่ำนั่นก็คือ ภาษาเครื่อง (Machine language) และภาษาแอสแซมบลี (Assembly language) นั่นเอง

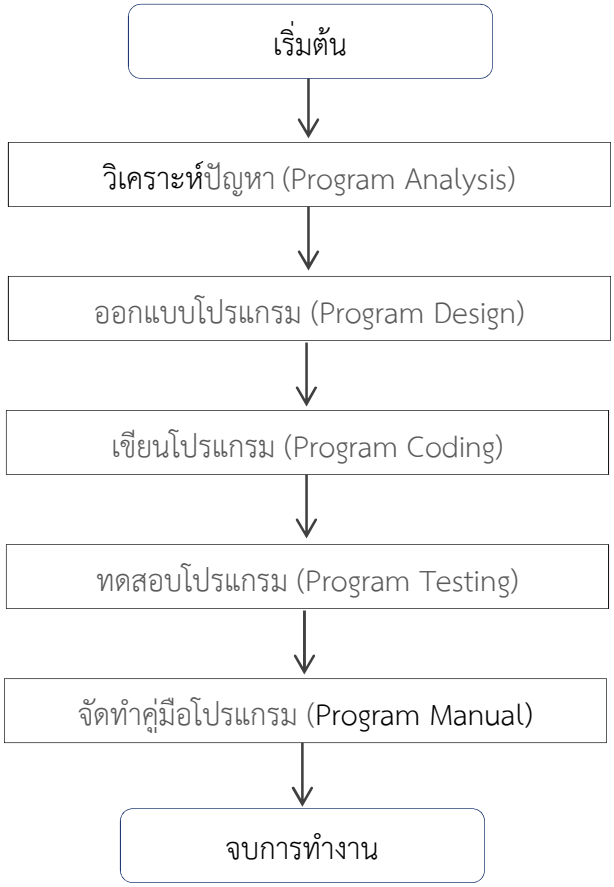
1.2.2 ภาษาระดับสูง (High level language)

ภาษาระดับสูงเป็นภาษาที่ทำความเข้าใจได้ไม่ยากสามารถเข้าใจคำสั่งโปรแกรมได้ง่าย โดยคำสั่งการทำงานมีลักษณะเหมือนภาษาอังกฤษ สามารถสื่อความหมายในการเขียนโปรแกรมหรือคำสั่งได้ง่ายและสะดวกต่อการใช้งานหรือแก้ไขคำสั่งการของโปรแกรม ปัจจุบันเป็นที่นิยมอย่างมากในการพัฒนาโปรแกรมคอมพิวเตอร์ ตัวอย่างของภาษาระดับสูง เช่น ภาษาปาสคาล ภาษาซี ภาษาเบสิก ภาษาจาวา ภาษา HTML ภาษา PHP เป็นต้น

1.3 การพัฒนาโปรแกรมคอมพิวเตอร์

ในการพัฒนาโปรแกรมคอมพิวเตอร์ผู้พัฒนาโปรแกรมคอมพิวเตอร์ควรมีขั้นตอนในการพัฒนาโปรแกรมคอมพิวเตอร์ให้ถูกต้องตามขบวนการ ทั้งนี้จะส่งผลให้การพัฒนาโปรแกรมคอมพิวเตอร์เกิดข้อผิดพลาดน้อยที่สุดในการพัฒนาโปรแกรมคอมพิวเตอร์ในแต่ละครั้ง และสิ่งสำคัญจะทำให้ผู้พัฒนาเข้าใจระบบหรือรูปแบบของโปรแกรมคอมพิวเตอร์ที่จะทำการพัฒนาขึ้นทั้งหมด และจะทำให้ผู้พัฒนาได้เห็นถึงปัญหาหรือจุดบกพร่องในการพัฒนาโปรแกรมคอมพิวเตอร์ได้ในระดับ ส่ผลให้ผู้พัฒนาสามารถแก้ไขปัญหาดังกล่าวได้ทันถ่วงที

โดยขั้นตอนของการพัฒนาโปรแกรมคอมพิวเตอร์ มีขั้นตอนของการพัฒนาการพัฒนาโปรแกรมคอมพิวเตอร์ดังขั้นตอนต่อไปนี้ 1.การวิเคราะห์ปัญหา (Program Analysis) 2.การออกแบบโปรแกรม (Program Design) 3.การเขียนโปรแกรม (Program Coding) 4.การทดสอบโปรแกรม (Program Testing) และ 5.การจัดทำคู่มือโปรแกรม (Program Manual)



ภาพที่ 1.9 โฟร์ชาร์จแสดงขั้นตอนการพัฒนาโปรแกรมคอมพิวเตอร์

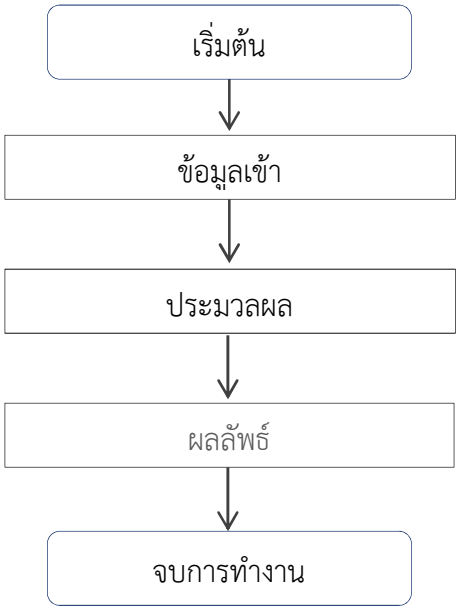
1.3.1 ขั้นการวิเคราะห์ปัญหา (Program Analysis)

การวิเคราะห์ปัญหาในขั้นตอนนี้จะเป็นขั้นตอนแรกสุดที่ผู้พัฒนาโปรแกรมจะต้องลงมือทำก่อนที่จะเขียนโปรแกรมจริง ๆ เพื่อทำความเข้าใจกับปัญหาที่เกิดขึ้นและค้นหาจุดมุ่งหมายหรือสิ่งที่ต้องการวิเคราะห์ปัญหา ทั้งนี้ผู้พัฒนาโปรแกรมจะต้องทำความเข้าใจกับปัญหา โดยสามารถกำหนดให้ได้ว่าปัญหาหรือโจทย์คืออะไร โจทย์ที่ต้องการต้องการอะไร ทำอย่างไรจึงจะแก้ปัญหานั้นได้เพื่อให้ได้ ผลลัพธ์ หรือคำตอบที่ผู้พัฒนาโปรแกรมต้องการ ผู้พัฒนาโปรแกรมจะต้องทำความเข้าใจเกี่ยวกับองค์ประกอบอยู่ 3 องค์ประกอบ ที่จะช่วยในการวิเคราะห์ปัญหา ได้แก่

1.2.1.1 การระบุข้อมูลเข้า (Input Specification) ผู้พัฒนาโปรแกรมต้องรู้ว่ามีข้อมูลอะไรที่จะต้องป้อนเข้าสู่คอมพิวเตอร์พร้อมกับโปรแกรม เพื่อให้โปรแกรมทำ การประมวลผลและออกผลลัพธ์

1.2.1.2 การระบุข้อมูลออก (Output Specification) ผู้พัฒนาโปรแกรมจะพิจารณาว่างานที่ทำ มีเป้าหมายหรือวัตถุประสงค์อะไร ต้องหารผลลัพธ์ที่มีรูปร่างหน้าตาเป็นอย่างไร โดยจะต้องคำนึงถึงผู้ให้งานโปรแกรมเป็นหลักในการออกแบบ

1.2.1.3 กำหนดวิธีการประมวลผล (Process Specification) ผู้พัฒนาโปรแกรมต้องรู้วิธีการประมวลผลเพื่อให้ได้ผลลัพธ์ตามต้องการ ซึ่งจะเป็นวิธีใดนั้นจะต้องทดลองผิดลองถูกไปเรื่อย ๆ จนกว่าจะได้วิธีการประมวลผลโปรแกรมที่สมบูรณ์หรือดีที่สุด



ภาพที่ 1.10 โฟร์ชาร์จแสดงขั้นตอนการวิเคราะห์ปัญหาในการพัฒนาโปรแกรมคอมพิวเตอร์

1.3.2 ขั้นการออกแบบโปรแกรม (Program Design)

หลังจากขั้นตอนการวิเคราะห์ปัญหาแล้ว ขั้นตอนถัดไปคือการออกแบบโปรแกรม โดยใช้เครื่องมือมาช่วยในการออกแบบ ขั้นตอนนี้ยังไม่ได้เป็นการเขียนโปรแกรมจริง ๆ แต่จะช่วยให้การเขียนโปรแกรมทำได้ง่ายยิ่งขึ้น โดยผู้พัฒนาโปรแกรมสามารถเขียนตามขั้นตอนที่ได้ออกแบบไว้ในขั้นตอนนี้ และยังจะช่วยให้การเขียนโปรแกรมมีข้อผิดพลาดน้อยลง นอกจากนี้ยังช่วยในการตรวจสอบการทำงานของโปรแกรม ทำให้ทราบขั้นตอนการทำงานได้อย่างรวดเร็ว โดยไม่ต้องไปไล่ดูจากตัวโปรแกรมจริง ๆ ดังนั้นในขั้นตอนการออกแบบโปรแกรมนี้เป็นการออกแบบการทำงานของโปรแกรม หรือขั้นตอนในการทำปัญหา ซึ่งผู้พัฒนาโปรแกรมต้องมีการออกแบบและสามารถเลือกใช้เครื่องมือมาช่วยในการออกแบบได้หลากหลาย ซึ่งจะส่งผลให้โปรแกรมที่ได้มีประสิทธิภาพการทำงานที่ดี

1.3.2.1 อัลกอริทึม (Algorithm)

อัลกอริทึมเป็นเครื่องมือที่ช่วยในการออกแบบโปรแกรม โดยใช้ข้อความที่เป็นภาษาพูดในการอธิบายทำงานของโปรแกรมที่เป็นลำดับขั้นตอน ในการเขียนอัลกอริทึมนี้แม้จะมีความชัดเจนอยู่แล้ว แต่มีจุดอ่อนอยู่ที่ข้อมูลคำอธิบายที่จะต้องอธิบายให้กะทัดรัดเข้าใจง่าย และถ้าพัฒนาโปรแกรมใช้สำนวนที่อ่านยาก ก็อาจจะส่งผลทำให้ผู้ที่มาศึกษาต่อไม่เข้าใจขั้นตอนการทำงานของโปรแกรมได้ ดังนั้นจึงมีการคิดค้นเครื่องมืออื่น ๆ ที่ช่วยในการออกแบบโปรแกรมแทนอัลกอริทึม อันได้แก่ ผังงาน รหัสจำลอง แผนภูมิโครงสร้าง ฯลฯ

1.3.2.2 ผังงาน (Flowchart)

ผังงานเป็นเครื่องมืออีกแบบหนึ่งที่ใช้รูปภาพแสดงถึงขั้นตอนการทำงานของโปรแกรม หรือขั้นตอนในการแก้ปัญหาทีละขั้น และมีเส้นที่แสดงทิศทางการไหลของข้อมูลตั้งแต่จุดเริ่มต้นจนกระทั่งได้ผลลัพธ์ตามที่ต้องการ ซึ่งจะทำให้ผู้ที่ศึกษาต่อสามารถทำความเข้าใจได้โดยง่าย และผังงานถือได้ว่าเป็นเครื่องหนึ่งที่ได้รับความนิยมมากในการออกแบบพัฒนาโปรแกรมคอมพิวเตอร์

1.3.2.3 รหัสจำลอง (Pseudo code)

รหัสจำลองเป็นเครื่องมือที่ใช้ข้อความที่เป็นภาษาอังกฤษหรือภาษาไทยในการแสดงขั้นตอนการทำงานของโปรแกรม แต่รหัสจำลองจะมีการใช้คำเฉพาะ(reserve words) ที่มีอยู่ในภาษาโปรแกรมมาช่วยในการเขียน โดยโครงของรหัสจำลองจะมีส่วนที่คล้ายกับการเขียนโปรแกรมมาก ดังนั้นรหัสจำลองจึงเป็นเครื่องมืออีกแบบหนึ่งที่นิยมใช้กันมากในการออกแบบพัฒนาโปรแกรมเช่นกัน

1.3.3 ขั้นการเขียนโปรแกรม (Program Coding)

หลังจากที่ผ่านขั้นตอนที่สอง คือ การออกแบบโปรแกรมแล้ว ขั้นต่อไปคือการเขียนโปรแกรมด้วยภาษาคอมพิวเตอร์ ในขั้นตอนนี้ผู้พัฒนาโปรแกรมจะต้องนำเครื่องมือที่ถูกสร้างขึ้นจากขั้นตอนการออกแบบมาแปลให้เป็นโปรแกรมคอมพิวเตอร์ ซึ่งในการสร้างโปรแกรมคอมพิวเตอร์นั้น ผู้พัฒนาโปรแกรมสามารถเลือกใช้ภาษาได้หลายภาษา ตั้งแต่ภาษาระดับต่ำ เช่น ภาษาแอสเซมบลี จนถึงภาษาระดับสูง เช่น ภาษาเบสิค ภาษาซี ภาษาจาวา เป็นต้น ทั้งนี้การเลือกใช้งานโปรแกรมใดในการออกแบบพัฒนาโปรแกรม ผู้พัฒนาโปรแกรมต้องคำนึงถึงคุณสมบัติของโปรแกรมที่นำมาใช้ ว่ามีคุณสมบัติการทำงานสอดคล้องกับงานที่ผู้พัฒนาโปรแกรมต้องการหรือไม่ และแต่ละภาษาก็จะมีรูปแบบโครงสร้างหรือไวยากรณ์ของภาษาที่แตกต่างกันออกไป

ดังนั้นการเขียนโปรแกรมที่ดีนั้น ผู้พัฒนาโปรแกรมควรจะต้องทำตามขั้นตอน คือ เริ่มตั้งแต่วิเคราะห์ปัญหาให้ได้ก่อน แล้วทำการออกแบบโปรแกรมแล้วจึงจะเริ่มเขียนโปรแกรม ซึ่งในการเขียนโปรแกรมนั้นสำหรับผู้ที่ยังไม่มีประสบการณ์การเขียนโปรแกรมที่เพียงพอ ก็ควรจะทดลองเขียนลงในกระดาษก่อน แล้วตรวจสอบจนแน่ใจว่าสามารถทำงานได้แล้วจึงทำการป้อนเข้าสู่เครื่องคอมพิวเตอร์ เพื่อเป็นการประหยัดเวลาและทำให้สามารถทำงานได้เร็วขึ้น

1.3.4 การทดสอบโปรแกรม (Program Testing)

ในบางครั้งโปรแกรมอาจผ่านการแปล โดยไม่มีข้อผิดพลาดใด ๆ แจ้งออกมา แต่เมื่อนำโปรแกรมนั้นไปใช้งานปรากฏว่าได้ผลลัพธ์ที่ไม่เป็นจริง เนื่องจากอาจเกิดข้อผิดพลาดแบบ Logicl Error ขึ้นได้ ดังนั้นผู้พัฒนาโปรแกรมจึงควรจะต้องมีขั้นตอนการทดสอบความถูกต้องของโปรแกรมก่อนนำไปใช้งานจริง ในการทดสอบความถูกต้องของโปรแกรมจะมีอยู่หลายวิธีดังต่อไปนี้

1.) การใส่ข้อมูลที่ถูกต้อง (valid case) เป็นการทดสอบโดยเมื่อมีการรันโปรแกรม โดยผู้พัฒนาโปรแกรมทำการใส่ข้อมูลที่ถูกต้องลงไปในโปรแกรม และดูว่าผลลัพธ์ที่ได้จากโปรแกรมถูกต้องตามความเป็นจริงหรือตรงกับที่ต้องการหรือไม่

2.) การใช้ของเขตและความถูกต้องของข้อมูล (Range check and Completeness check) เป็นการทดสอบโดยตรวจสอบขอบเขตของข้อมูลที่ป้อนเข้าสู่โปรแกรม เช่น ถ้าโปรแกรมให้มีการป้อนวันที่ ก็จะต้องตรวจสอบว่าวันที่ที่ป้อนจะไม่เกินวันที่ 31 ถ้าผู้ใช้ป้อนวันที่ที่เป็นเลข 32 โปรแกรมจะต้องไม่ยอมให้ป้อนวันที่นี้ได้ หรือการตรวจสอบความสมบูรณ์ของข้อมูล เช่น การรับข้อมูลที่เป็น วัน/เดือน/ปี ก็จะต้องใส่เป็นตัวเลข 6 ตัวในลักษณะ dd/mm/yy ถ้าใส่น้อยกว่า 6 ตัว โปรแกรมจะต้องไม่รับหรือฟ้องแจ้งเตือนผู้ใช้งานได้

3.) การใช้ความสมเหตุสมผล (Consistency Check) ตัวอย่างเช่น ถ้าโปรแกรมมีการออกแบบให้ผู้ใช้ป้อนข้อมูลลงไปในฟอร์มที่มีข้อมูลแบ่งเป็นเพศหญิงและเพศชาย ดังนั้นรายละเอียดส่วนตัวของแต่ละเพศจะต้องมีส่วนที่แตกต่างกันในบางส่วน เช่นในส่วนวันหยุดกรณีลาคลอด จะต้องมีเฉพาะเพศหญิง แต่ในเพศชายไม่ควรจะมีวันหยุดกรณีลาคลอด เป็นต้น

4.) ข้อมูลที่เป็นตัวเลขและตัวอักษร (Correct No. and Type character check) เป็นการตรวจสอบว่าถ้าโปรแกรมให้ผู้ป้อนข้อมูลในฟิลด์ที่ต้องรับข้อมูลที่เป็นตัวเลขอย่างเช่น ฟิลด์ที่เป็นจำนวนเงิน โดยโปรแกรมควรจะยอมให้ผู้ใช้ป้อนข้อมูลได้เฉพาะตัวเลขเท่านั้น ไม่อนุญาตให้ใส่ตัวอักษรในฟิลด์นั้นได้ หรือถ้าเป็นฟิลด์ที่รับข้อมูลที่เป็นตัวอักษรเช่น ฟิลด์ชื่อนามสกุล ก็จะป้อนได้เฉพาะตัวอักษรเท่านั้น และจะป้อนตัวเลขไม่ได้ เป็นต้น

5.) ข้อมูลเป็นไปตามข้อกำหนด (Existence Check) คือข้อมูลที่ป้อนในฟิลด์ต้องเป็นไปตามที่กำหนดไว้แน่นอนแล้วเท่านั้น เช่น กำหนดให้ฟิลด์นี้ป้อนขอมูลได้เฉพาะตัวเลขที่อยู่ในกลุ่ม 1,2,5,7 ได้เท่านั้น จะป้อนเป็นตัวเลขอื่นที่ไม่อยู่ในกลุ่มนี้ไม่ได้

1.3.5 ขั้นการจัดทำคู่มือโปรแกรม (Program Manual)

ขั้นการจัดทำคู่มือโปรแกรม หมายถึง เอกสารต่าง ๆ ที่ใช้กำกับอธิบายโปรแกรมและช่วยให้ผู้ใช้โปรแกรมทำ งานได้สะดวกขึ้น เช่น คู่มือปฏิบัติงานเครื่อง (Operation Manual) คู่มือผู้ใช้ (User manual) ปัจจุบันเอกสารประกอบโปรแกรม มีอยู่ในหลายสื่อ เช่น มีอยู่ในซอฟต์แวร์ได้แก่ คำ อธิบาย (Help function) โปรแกรมสาธิต (Demo program) เป็นต้น การทำ เอกสารประกอบโปรแกรม คือการอธิบายรายละเอียดของโปรแกรมว่า จุดประสงค์ของโปรแกรมคืออะไร สามารถทำ อะไร

ได้บ้าง และมีขั้นตอนการทำงานของโปรแกรมเป็นอย่างไร เป็นต้น เครื่องมือที่ช่วยในการออกแบบโปรแกรม เช่น ผังงานหรือรหัสจำลอง สามารถนำมาประกอบกันเป็นเอกสารประกอบโปรแกรมได้

โดยผู้พัฒนาโปรแกรมคอมพิวเตอร์ที่ดี ควรจะมีการทำเอกสารประกอบโปรแกรมทุกขั้นตอนของการพัฒนาโปรแกรมไม่ว่าจะเป็นขั้นตอนการออกแบบ การเขียนโปรแกรม หรือขั้นตอนการทดสอบโปรแกรม ซึ่งการทำ เอกสารนี้จะมีประโยชน์อย่างมากต่อหน่วยงาน เนื่องจากบางครั้งอาจต้องการเปลี่ยนแปลงแก้ไขโปรแกรมที่ได้มีการทำ เสร็จไปนานแล้ว เพื่อให้ตรงกับความต้องการที่เปลี่ยนไป จะทำให้เข้าใจโปรแกรมได้ง่ายขึ้นและจะเป็นการสะดวกต่อผู้ที่ต้องเข้ามารับช่วงงานต่อทีหลัง โดยเอกสารประกอบโปรแกรมโดยทั่วไปจะมีอยู่ด้วยกัน 2 แบบ คือ

1.) เอกสารประกอบโปรแกรมสำหรับผู้ใช้ (User Documentation)

เหมาะสำหรับผู้ใช้ที่ไม่ต้องเกี่ยวข้องกับการพัฒนาโปรแกรม แต่เป็นผู้ที่ใช้งานโปรแกรมอย่างเดียว จะเน้นการอธิบายเกี่ยวกับการใช้งานโปรแกรมเป็นหลัก ตัวอย่างเช่น

- โปรแกรมนี้ทำ อะไร ใช้งานในด้านไหน
- ข้อมูลเข้ามีลักษณะอย่างไร
- ข้อมูลออกหรือผลลัพธ์มีลักษณะอย่างไร
- การเรียกใช้โปรแกรมทำ อย่างไร
- คำ สั่ง หรือข้อมูล ที่จำ เป็นให้โปรแกรมเริ่มงานมีอะไรบ้าง
- อธิบายเกี่ยวกับประสิทธิภาพ และความสามารถของโปรแกรม

2.) เอกสารประกอบโปรแกรมสำหรับผู้เขียนโปรแกรม (Technical Documentation) โดยสามารถแบ่งได้ออกเป็น 2 ส่วน คือ

- ส่วนที่เป็นคำอธิบายหรือคอมมนท์ ซึ่งส่วนใหญ่ผู้พัฒนาโปรแกรมคอมพิวเตอร์มักจะเขียนแทรกอยู่ในโปรแกรมอธิบายการทำงานของโปรแกรมเป็นส่วน ๆ เพื่อให้สามารถอธิบายหลักการทำงานของโปรแกรมในส่วนนั้น ๆ
- ส่วนอธิบายด้านเทคนิค ซึ่งส่วนนี้ผู้พัฒนาโปรแกรมคอมพิวเตอร์มักจะจัดทำเป็นเอกสารโดยจะอธิบายในรายละเอียดที่มากขึ้น เช่น ชื่อโปรแกรมย่อยต่างๆ มีอะไรบ้าง แต่ละโปรแกรมย่อยทำ หน้าที่อะไร และคำ อธิบายย่อๆ เกี่ยวกับวัตถุประสงค์ของโปรแกรม เป็นต้น

สรุปท้ายบท

โปรแกรมคอมพิวเตอร์ในปัจจุบันและอนาคตมีบทบาทอย่างยิ่งในการสร้างพัฒนางานด้านต่าง ๆ ไม่ว่าจะเป็นงานทางด้าน การจัดการเอกสาร งานทางด้านกราฟฟิก งานทางด้านการเก็บฐานข้อมูล หรืองานทางด้านการพัฒนาโปรแกรมในระบบมือถือหรือแท็ปเล็ต ดังนั้นสิ่งสำคัญในการพัฒนาโปรแกรมคอมพิวเตอร์ขึ้นมานั้นก็คือ “ภาษาคอมพิวเตอร์” ภาษาคอมพิวเตอร์ถือได้ว่าเป็นหัวใจหลักในการสร้างโปรแกรมคอมพิวเตอร์ที่มีอยู่ในปัจจุบันนี้ขึ้นมา โดยภาษาคอมพิวเตอร์ที่มีอยู่ในปัจจุบันมีอยู่มากมายมายหลายภาษา สามารถนำมาพัฒนาโปรแกรมคอมพิวเตอร์ได้อย่างหลากหลายรูปแบบ ไม่ว่าจะเป็นการพัฒนาโปรแกรมบนเครื่องคอมพิวเตอร์ การพัฒนาโปรแกรมบนระบบเซิร์ฟเวอร์ หรือการพัฒนาโปรแกรมบนโทรศัพท์มือถือหรือแท็ปแล็ต เป็นต้น ทั้งนี้ยังมีสิ่งหนึ่งที่ผู้พัฒนาโปรแกรมควรทราบคือ ระดับของภาษาคอมพิวเตอร์ โดยระดับของภาษาคอมพิวเตอร์สามารถแบ่งออกเป็นกว้าง ๆ ได้เป็น 2 ระดับ ดังนี้ ภาษาระดับต่ำ (Low level language) คือ ภาษาเครื่อง และภาษาแอสแซมบลี และภาษาระดับสูง (High level language) เช่น ภาษาปาสคาล ภาษาซี ภาษาเบสิก ภาษาจาวา ภาษา HTML ภาษา PHP เป็นต้น และสิ่งสำคัญอีกประการ คือ ขั้นตอนของการพัฒนาโปรแกรมคอมพิวเตอร์ ซึ่งขั้นตอนของการพัฒนาการพัฒนาโปรแกรมคอมพิวเตอร์มีขั้นตอนที่ควรปฏิบัติ 5 ขั้นตอนดังนี้ 1.การวิเคราะห์ปัญหา (Program Analysis) 2.การออกแบบโปรแกรม (Program Design) 3.การเขียนโปรแกรม (Program Coding) 4.การทดสอบโปรแกรม (Program Testing) และ 5.การจัดทำคู่มือโปรแกรม (Program Manual)

คำถามทบทวน

1. ภาษาคอมพิวเตอร์สามารถแบ่งออกเป็นกี่ระดับอะไรบ้าง พร้อมทั้งอธิบายลักษณะในแต่ระดับ
2. ข้อแตกต่างของภาษาคอมพิวเตอร์แต่ละระดับมีข้อแตกต่างกันอย่างไรจงอธิบาย
3. จงอธิบายคุณสมบัติของภาษาคอมพิวเตอร์ดังกล่าวนี้ว่ามีคุณสมบัติอย่างไร และภาษาคอมพิวเตอร์ดังกล่าวนำไปพัฒนางานในด้านใด
    - 3.1 ภาษาเครื่อง
    - 3.2 ภาษาภาษาแอสแซมบลี
    - 3.3 ภาษาซี
    - 3.4 ภาษาเบสิค
    - 3.5 ภาษาจาวา
    - 3.6 ภาษา PHP
4. ขั้นตอนของการพัฒนาโปรแกรมคอมพิวเตอร์มีขั้นตอนการพัฒนากี่ขั้นตอน และจงอธิบายคุณสมบัติในแต่ละขั้นตอนว่ามีหน้าที่อย่างไร
5. ถ้าผู้พัฒนาคอมพิวเตอร์ได้มีการพัฒนาโปรแกรม แต่ผู้พัฒนาขาดการวิเคราะห์ปัญหา และขาดการจัดทำคู่มือโปรแกรม หากขาดขั้นตอนดังกล่าวจะส่งผลอย่างไรต่อการพัฒนาโปรแกรมจงอธิบาย

เอกสารอ้างอิง

คะชา ชาญศิลป์. (2548). **ภาษาซีสำหรับผู้เริ่มต้น.** กรุงเทพมหานคร: วิรัตน์เอ็ดดูเคชั่น.

ธีรวัฒน์ ประกอบผล. (2553). **คู่มือการเขียนโปรแกรมภาษา C.** กรุงเทพมหานคร: ซิมพลิฟลาย.

สุระสิทธิ์ ทรงม้า และภูริพจน์ แก้วย่อง. (2552). **การโปรแกรมคอมพิวเตอร์.**: มหาวิทยาลัยราชภัฏสวนดุสิต.

อรพิน ประวัติบริสุทธิ์. (2554). **คู่มือการเรียนภาษา C ฉบับปรับปรุงใหม่** (พิมพ์ครั้งที่ 10)**.** กรุงเทพมหานคร: โปรวิชั่น.

http://www.bcoms.net/php/php01.asp

http:/www.thaigoodview.com/library/contest2552/type2/tech04/22/cit/3_2.html

http://www.cptd.chandra.ac.th/selfstud/it4life/sub%20soft3.htm

http://www.ns2.spw.ac.th/poo/computer54/m354/lesson/lesson_1.html

บทที่ 2

การออกแบบการทำงานโปรแกรมคอมพิวเตอร์

แผนการสอนประจำบท

หัวข้อเนื้อหา

2.1 รหัสจำลอง (Pseudo code)

2.2 ผังงาน (Flowchart)

2.2.1 ประโยชน์ของผังงาน

2.2.2 วิธีการเขียนผังงานที่ดี

2.2.3 สัญลักษณ์ผังงาน (Flowchart)

2.2.4 ลักษณะรูปแบบของผังงาน

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถเข้าใจและสามารถออกแบบรหัสจำลอง (Pseudo code) เพื่อการพัฒนาโปรแกรมคอมพิวเตอร์ได้

2. ผู้เรียนสามารถเข้าใจและสามารถออกแบบผังงาน (Flowchart) เพื่อการพัฒนาโปรแกรมคอมพิวเตอร์ได้

3. ผู้เรียนสามารถวิเคราะห์และประยุกต์รหัสจำลอง (Pseudo code) และผังงาน (Flowchart) เพื่อการพัฒนาโปรแกรมในรูปแบบต่าง ๆ ได้

วิธีการสอน

1. สอนแบบบรรยายโดยใช้เอกสารประกอบการสอน

2. สอนแบบบรรยายโดยใช้ Slide Power Point

3. สอนโดยให้ผู้เรียนเป็นส่วนกลาง โดยใช้วิธีถามตอบกับผู้เรียนระหว่างบรรยาย และใช้วิธีการลงปฏิบัติการทำตามโจทย์คำสั่งด้วยตนเอง

4. ให้ผู้เรียนสรุปและอธิบายหัวข้อที่ผู้เรียนที่ได้รับมอบหมาย

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน

2. Slide Power Point

3. ตัวอย่างโจทย์คำสั่ง

การวัดผลและประเมินผล

1. การทำตามตัวอย่างโจทย์คำสั่งและการทำแบบฝึกหัด

2. ความตั้งใจในชั้นเรียนและการร่วมมือในการนำเสนอหรืออภิปราย

3. สังเกตจากการตอบคำถามและการร่วมทำกิจกรรมในชั้นเรียน

บทที่ 2

การออกแบบการทำงานโปรแกรมคอมพิวเตอร์

ในการพัฒนาโปรแกรมคอมพิวเตอร์ให้สามารถทำงานได้อย่างถูกต้องแม่นยำและมีข้อผิดพลาดของการทำงานน้อยที่สุดนั้น ผู้พัฒนาโปรแกรมคอมพิวเตอร์จำเป็นต้องมีหลักในการเขียนโปรแกรม โดยอาจจะใช้การเขียนอธิบายขบบวนการทำงานหรืออัลกอริทึม (Algorithm) แบบง่าย ๆ ขึ้นมาก่อน หรือผู้พัฒนาโปรแกรมอาจจะใช้การเขียนอธิบายการทำงานของโปรแกรมด้วยผังงาน (Flowchart) หรือยังมีอีกหนึ่งวิธีคือการใช้รหัสจำลอง (Pseudo code) ซึ่งวิธีการต่าง ๆ เหล่านี้จะเป็นวิธีที่จะให้ผู้พัฒนาสามารถมองเห็นการทำงานของโปรแกรมในภาพรวมทั้งหมดได้ และจะสามารถทำให้เราสามารถทราบถึงปัญหาต่าง ๆ ที่อาจจะเกิดขึ้นในการพัฒนาโปรแกรมได้

2.1 รหัสจำลอง (Pseudo code)

รหัสจำลอง (Pseudo code) เป็นการอธิบายขั้นตอนการทำงานของโปรแกรมคอมพิวเตอร์ โดยใช้คำผสมกันระหว่างภาษาอังกฤษและภาษาการเขียนโปรแกรมแบบโครงสร้าง หรือบางครั้งหากผู้พัฒนาไม่มีความถนัดด้านภาษาอังกฤษ ผู้พัฒนาอาจจะใช้ภาษาไทยก็ได้เช่นเดียวกัน แต่หากเป็นไปได้ควรใช้หลักของภาษาอังกฤษเพราะคำสั่งการทำงานของโปรแกรมทั้งหมดเป็นภาษาอังกฤษ การเขียนเขียนรหัสจำลอง (Pseudo code) ที่ดีจะต้องมีความชัดเจน สั้น กะทัดรัดเข้าใจง่าย และได้ใจความ

รูปแบบทั่วไปของรหัสจำลอง (Pseudo code)

Algorithm <ชื่ออัลกอริทึม>

1. .....
2. .....
3. .....
4. .....
5. .....
6. .....

.....

.....

.....

.....

END

ภาพที่ 2.1 ลักษณะรูปแบบทั่วไปของการออกแบบรหัสจำลอง

ตัวอย่างที่ 2.1 จงเขียนรหัสจำลอง (Pseudo code) โดยรับค่าจากผู้ใช้งาน 2 ค่า แล้วนำค่าที่มาทำการบวกกันแล้วแสดงคำตอบ

วิธีทำ

Algorithm < การบวกเลข >

1. ประกาศตัวแปร 3 ตัวแปร
2. รับค่าจากผู้ใช้งานมาใส่ตัวแปรที่ 1
3. รับค่าจากผู้ใช้งานมาใส่ตัวแปรที่ 2
4. ตัวแปรที่ 3 = ตัวแปรที่ 1 + ตัวแปรที่ 2
5. แสดงผลตัวแปรที่ 3
6. จบ

จากตัวอย่างที่ 2.1 จะเห็นว่าหากผู้พัฒนาได้เขียนรหัสจำลอง (Pseudo code) เพื่อออกแบบการทำงานของโปรแกรม จะทำให้ผู้พัฒนาโปรแกรมเห็นว่าจะต้องใช้ตัวแปรใดบ้าง แล้วมีขั้นตอนการทำงานทั้งหมดของโปรแกรมอย่างไร และหากลองเขียนให้เป็นรูปแบบภาษาอังกฤษตามหลักการเขียนโปรแกรมจะได้ ดังนี้

Algorithm < Sum >

1. int a,b,c
2. input a
3. input b
4. c = a + b
5. output c
6. END

จากตัวอย่างข้างต้นในกรณีที่เราเขียนเป็นรูปแบบภาษาอังกฤษจะทำให้ผู้พัฒนาเข้าใจได้ดียิ่งขึ้น เพราะว่าเวลาเขียนโปรแกรมจะมีรูปแบบของคำสั่งใกล้เคียงกับรูปแบบที่เราเขียนรหัสจำลอง (Pseudo code)

ตัวอย่างที่ 2.2 จงเขียนรหัสจำลอง (Pseudo code) โดยรับค่าจากผู้ใช้งาน 2 ค่าแล้วนำค่าทั้งสองมาเปรียบกัน โดยมีเงื่อนไขว่าถ้าค่าใดมากกว่าให้แสดงผลค่านั้นออกทางจอให้ผู้ใช้งานทราบ

วิธีทำ

```
Algorithm < Compare >
1.  int a,b
2.  input a
3.  input b
4.  if a > b {
5.                  output a
6.              }
7. if a < b   {
8.                  output b
9.              }
10 END
```

ตัวอย่างที่ 2.3 จงเขียนรหัสจำลอง (Pseudo code) โดยรับค่าจากผู้ใช้งาน 1 ค่าแล้วนำค่าที่ได้มาแสดงผลตามจำนวนที่ผู้ใช้งานได้ป้อนเข้ามา

ตัวอย่าง เช่น หากผู้ใช้งานป้อน 3 โปรแกรมจะแสดง 1 2 3
หากผู้ใช้งานป้อน 8 โปรแกรมจะแสดง 1 2 3 4 5 6 7 8

วิธีทำ

```
Algorithm < Compare >
1.  int a , i = 1
2.  input a
3.  for 1 to a {
4.        i = i +1
5.        output i
6.              }
7. END
```

2.2 ผังงาน (Flowchart)

ผังงาน (Flowchart) คือ การเขียนแผนภาพที่มีการใช้สัญลักษณ์รูปภาพและลูกศรที่แสดงถึงขั้นตอนการทำงานของโปรแกรมหรือระบบทีละขั้นตอน รวมไปถึงทิศทางการไหลของข้อมูลตั้งแต่แรกจนได้ผลลัพธ์ตามที่ต้องการ การเขียนผังงาน (Flowchart) เป็นรูปแบบการเขียนที่นิยมอย่างมากในการออกแบบเพื่อพัฒนางานทางด้านโปรแกรม เพราะการเขียนผังงาน (Flowchart) ออกแบบง่าย มีความเข้าใจง่ายและเป็นสากล ทำให้บุคคลอื่น ๆ ที่มาศึกษาสามารถเข้าใจระบบงานทั้งหมดได้ง่ายและตรงกัน

2.2.1 ประโยชน์ของผังงาน

1.) ช่วยลำดับขั้นตอนการทำงานของโปรแกรม และสามารถนำไปเขียนโปรแกรมได้โดยไม่สับสน

2.) ช่วยในการตรวจสอบ และแก้ไขโปรแกรมได้ง่ายเมื่อเกิดข้อผิดพลาด

3.) ช่วยให้การดัดแปลง แก้ไข ทำได้อย่างสะดวกและรวดเร็ว

4.) ช่วยให้ผู้อื่นสามารถศึกษาการทำงานของโปรแกรมได้อย่างง่ายและรวดเร็วมากขึ้น

2.2.2 วิธีการเขียนผังงานที่ดี

1.) ใช้สัญลักษณ์ตามที่กำหนดไว้

2.) ใช้ลูกศรแสดงทิศทางการไหลของข้อมูลจากบนลงล่าง หรือจากซ้ายไปขวา

3.) คำอธิบายในภาพควรสั้นกะทัดรัด และเข้าใจง่าย

4.) ทุกแผนภาพต้องมีลูกศรแสดงทิศทางเข้า-ออก

5.) ไม่ควรโยงเส้นเชื่อมผังงานที่อยู่ไกลมาก ๆ ควรใช้สัญลักษณ์จุดเชื่อมต่อแทน

6.) ผังงานควรมีการทดสอบความถูกต้องของการทำงานก่อนนำไปเขียนโปรแกรม

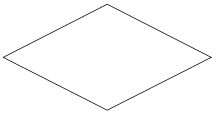
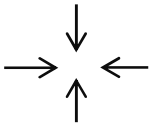
2.2.3 สัญลักษณ์ผังงาน (Flowchart)

การเขียนผังงาน (Flowchart) จะประกอบไปด้วยการใช้สัญลักษณ์มาตรฐานต่าง ๆ ที่เรียกว่า สัญลักษณ์ ANSI (American National Standards Institute) ในการสร้างผังงาน ตัวอย่างที่แสดงในรูปต่อไปนี้เป็นผังงาน (Flowchart) ที่นิยมใช้กันมากในการออกแบบพัฒนาโปรแกรมคอมพิวเตอร์

ตารางที่ 2.1 แสดงสัญลักษณ์ผังงาน

ลำดับ	สัญลักษณ์	คำอธิบาย
1		เริ่มต้น , สิ้นสุด
2		ใช้รับค่าข้อมูลหรือแสดงผลข้อมูล โดยไม่ระบุชนิดของอุปกรณ์
3		ใช้ในการประมวลผลการทำงาน หรือ คำนวณค่าต่าง ๆ

ตารางที่ 2.1 แสดงสัญลักษณ์ผังงาน (ต่อ)

ลำดับ	สัญลักษณ์	คำอธิบาย
4		ใช้ในการตรวจสอบเงื่อนไข
5		ลูกศรแสดงทิศทางการทำงานของโปรแกรม
6		แสดงผลลัพธ์หรือรายงานทางเครื่องพิมพ์
7		จุดเชื่อมต่อของผังงาน

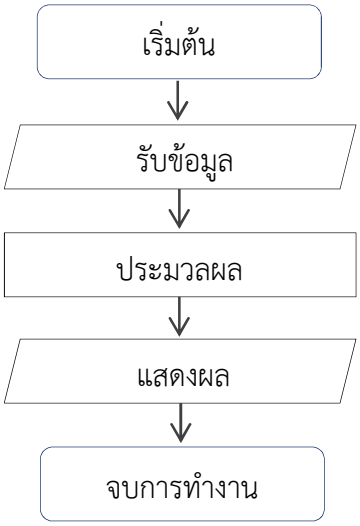
2.2.4 ลักษณะรูปแบบของผังงาน

ลักษณะรูปแบบของผังงานที่ใช้งานในงานทั่ว ๆ ไปจะมีรูปแบบที่นิยมใช้งานกันอยู่หลายรูปแบบ แต่ในการพัฒนาโปรแกรมจะมีรูปแบบการเขียนผังงานอยู่ไม่กี่รูปแบบ ซึ่งรูปแบบการเขียนผังงานในการพัฒนาโปรแกรมโดยทั่วไปจะมีอยู่ 2 แบบดังนี้

1.) รูปแบบผังงานแบบลำดับ
2.) รูปแบบผังงานแบบทางเลือก

2.2.4.1 รูปแบบผังงานแบบลำดับ

รูปแบบผังงานแบบลำดับจะเป็นรูปแบบในลักษณะเรียงลำดับลงไป โดยจะไม่มีการข้ามขั้นหรือย้อนกลับไปทำงานในส่วนอื่น ๆ โดยรูปแบบผังงานแบบลำดับจะมีรูปแบบดังตัวอย่างต่อไปนี้



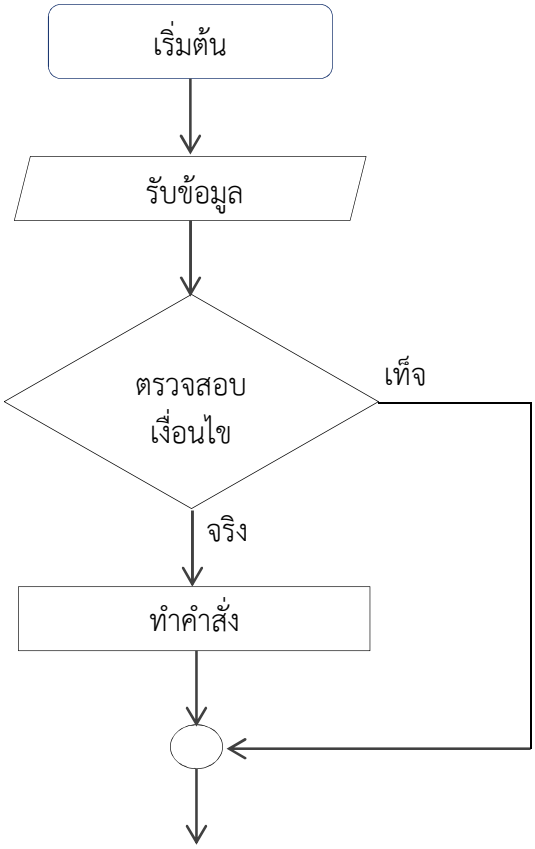
ภาพที่ 2.2 รูปแบบผังงานแบบลำดับ

2.2.4.2 รูปแบบผังงานแบบทางเลือก

รูปแบบผังงานแบบทางเลือกจะเป็นรูปแบบการทำงานที่มีรูปแบบผังงานแบบลำดับผสมรวมกับการตรวจสอบเงื่อนไขของโปรแกรม ทั้งนี้การสอบสอบเงื่อนไขการทำงานของโปรแกรมเพื่อจะเป็นการตรวจสอบทางเลือกให้กับการทำงานของโปรแกรมว่าโปรแกรมจะไปทำงานทางทิศทางใด โดยรูปแบบผังงานแบบลำดับจะมีรูปแบบอยู่ 3 กรณี ดังตัวอย่างต่อไปนี้

1.) ผังงานแบบทางเลือกแบบ 1 เส้นทาง

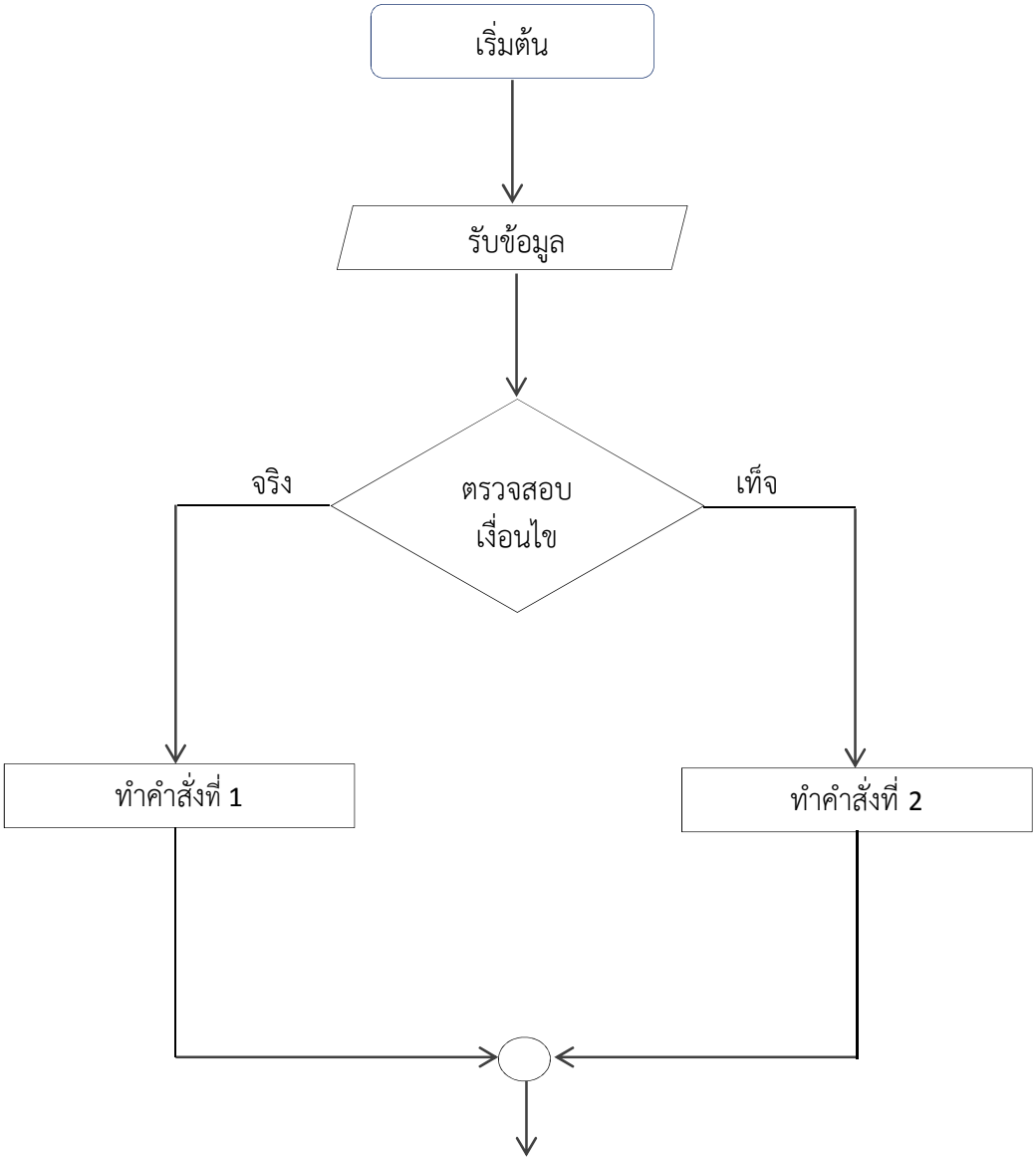
ผังงานแบบทางเลือกแบบ 1 เส้นทางจะทำงานเฉพาะเงื่อนไขที่เป็นจริงเท่านั้น ดังตัวอย่างภาพที่ 2.3



ภาพที่ 2.3 ผังงานแบบทางเลือกแบบ 1 เส้นทาง

2.) ผังงานแบบทางเลือกแบบ 2 เส้นทาง

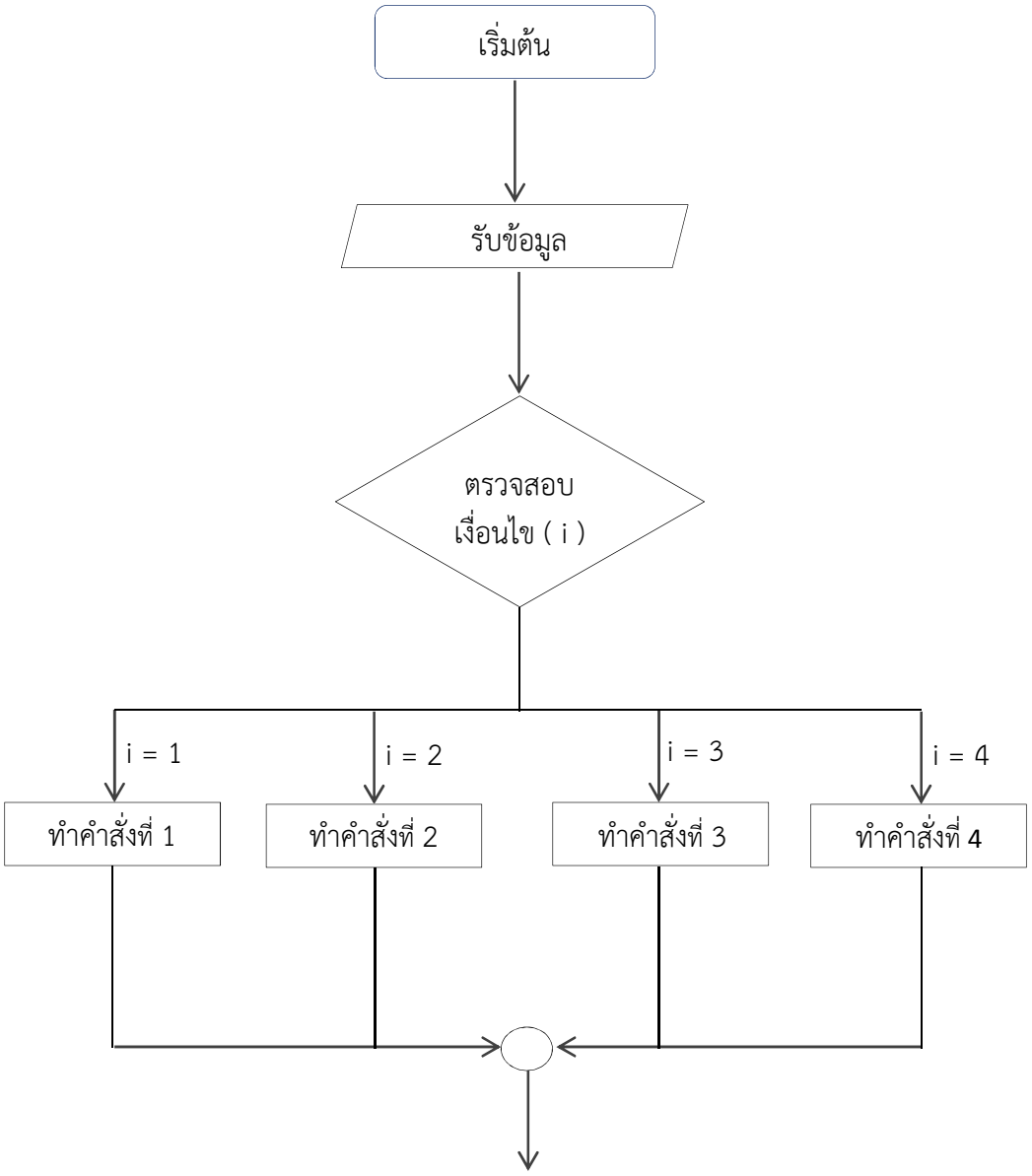
ผังงานแบบทางเลือกแบบ 2 เส้นทางจะทำงานโดยพิจารณาทั้งเงื่อนไขที่เป็นจริงและเป็นเงื่อนไขที่เป็นเท็จ โดยคำสั่งการทำงานจะมีคำสั่งที่แตกต่างกันออกไปในแต่ละเงื่อนไข ดังตัวอย่างภาพที่ 2.4



ภาพที่ 2.4 ผังงานแบบทางเลือกแบบ 2 เส้นทาง

3.) ผังงานแบบทางเลือกแบบหลายเส้นทาง

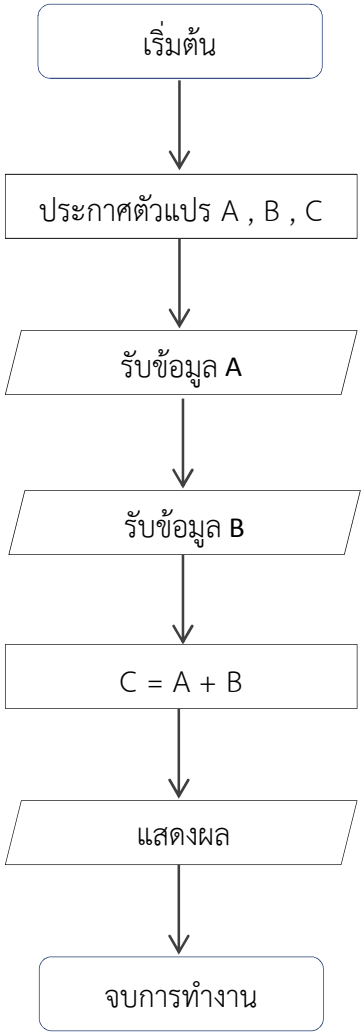
ผังงานแบบทางเลือกแบบหลายเส้นทางจะทำงานโดยพิจารณาทั้งเงื่อนไขที่เกิดขึ้นว่าตรงกับเงื่อนไขใด ๆ โดยคำสั่งการทำงานจะมีคำสั่งที่แตกต่างกันออกไปในแต่ละเงื่อนไข ดังตัวอย่างภาพที่ 2.5



ภาพที่ 2.5 ผังงานแบบทางเลือกแบบหลายเส้นทาง

ตัวอย่างที่ 2.4 จงเขียนผังงาน (Flowchart) โดยรับค่าจากผู้ใช้งาน 2 ค่า แล้วนำค่าที่มาทำการบวกกันแล้วแสดงคำตอบ

วิธีทำ

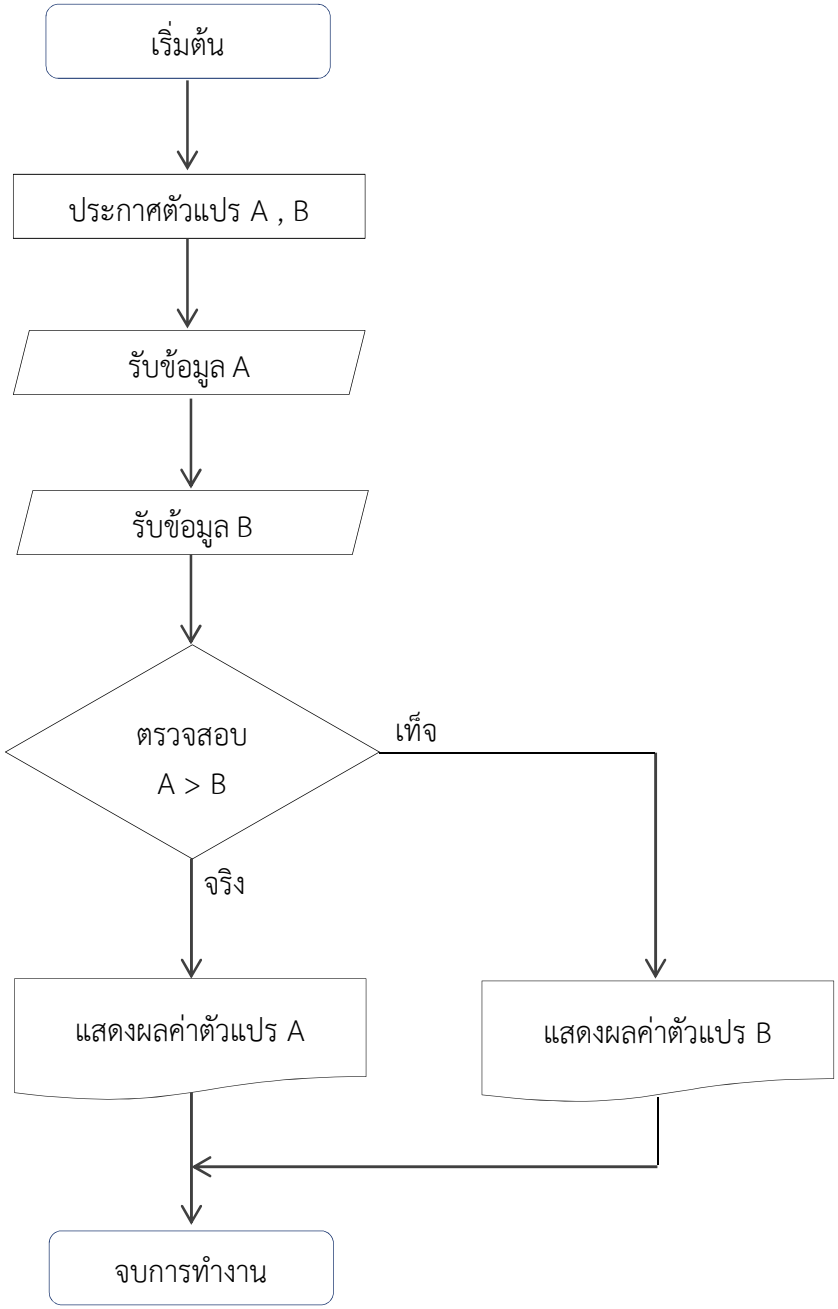


ภาพที่ 2.6 แสดงการออกแบบผังงานในการทำงานของโปรแกรมการบวกเลข

จากตัวอย่างที่ 2.6 เป็นการออกแบบผังงานในการทำงานของโปรแกรมการบวกเลข จะสังเกตได้ว่าจากโจทย์ข้อนี้จะมีเงื่อนไขการตรวจสอบทั้งสิ้น ดังนั้นการออกแบบผังงานในข้อนี้จะมีลักษณะของผังงานที่เป็นรูปแบบผังงานแบบลำดับ

ตัวอย่างที่ 2.5 จงเขียนผังงาน (Flowchart) โดยรับค่าจากผู้ใช้งาน 2 ค่าแล้วนำค่าทั้งสองมาเปรียบกัน โดยมีเงื่อนไขว่าถ้าค่าใดมากกว่าให้แสดงผลค่านั้นออกทางเครื่องพิมพ์

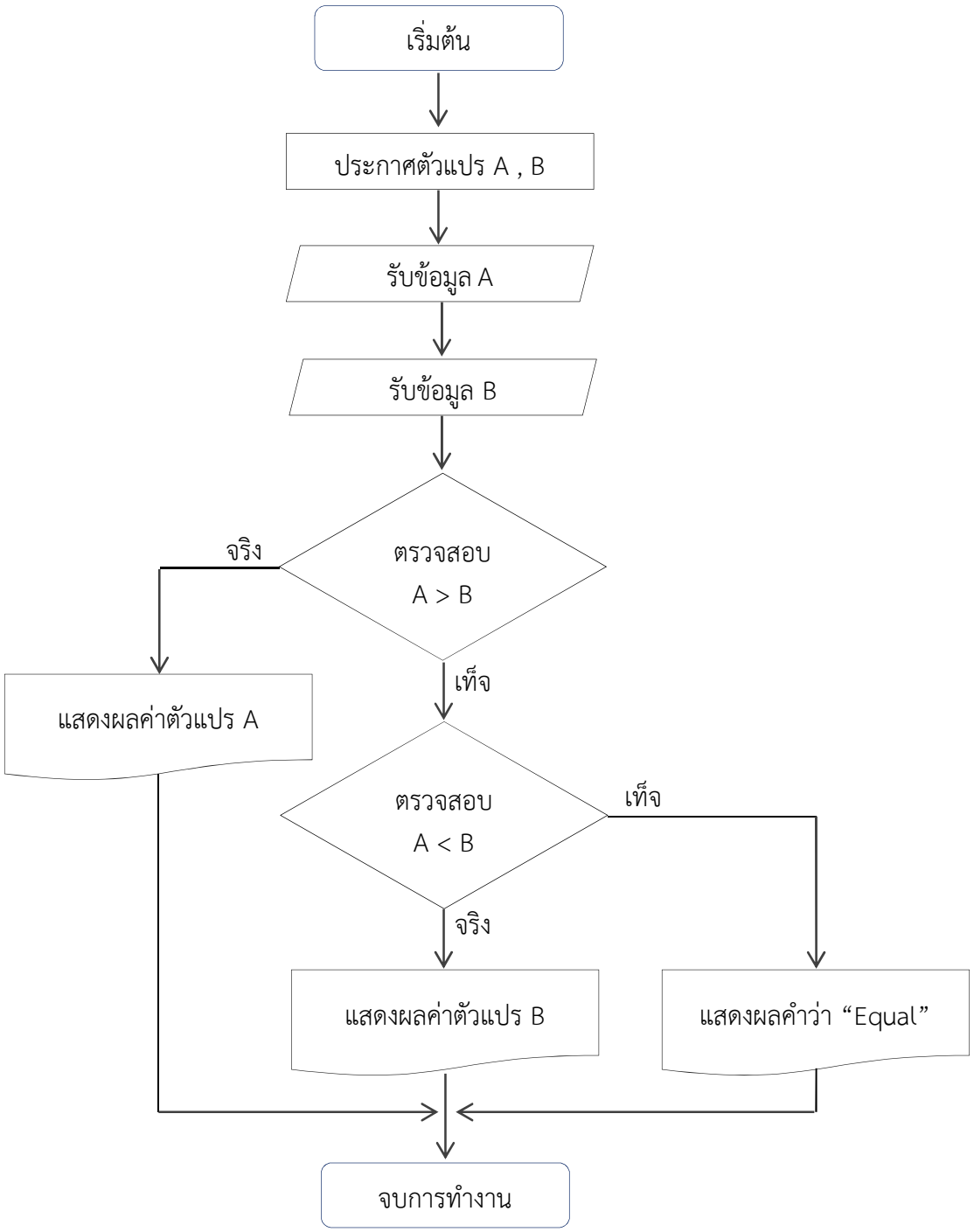
วิธีทำ



ภาพที่ 2.7 แสดงการออกแบบผังงานโดยมีเงื่อนไขเปรียบเทียบแบบสองทาง

ตัวอย่างที่ 2.6 จงเขียนผังงาน (Flowchart) โดยรับค่าจากผู้ใช้งาน 2 ค่าแล้วนำค่าทั้งสองมาเปรียบกัน โดยมีเงื่อนไขดังนี้ ถ้าค่าใดมากกว่าให้แสดงผลค่านั้นออกทางเครื่องพิมพ์ แต่ถ้าค่าทั้งสองเท่ากันให้แสดงคำว่า “Equal” ออกทางเครื่องพิมพ์

วิธีทำ

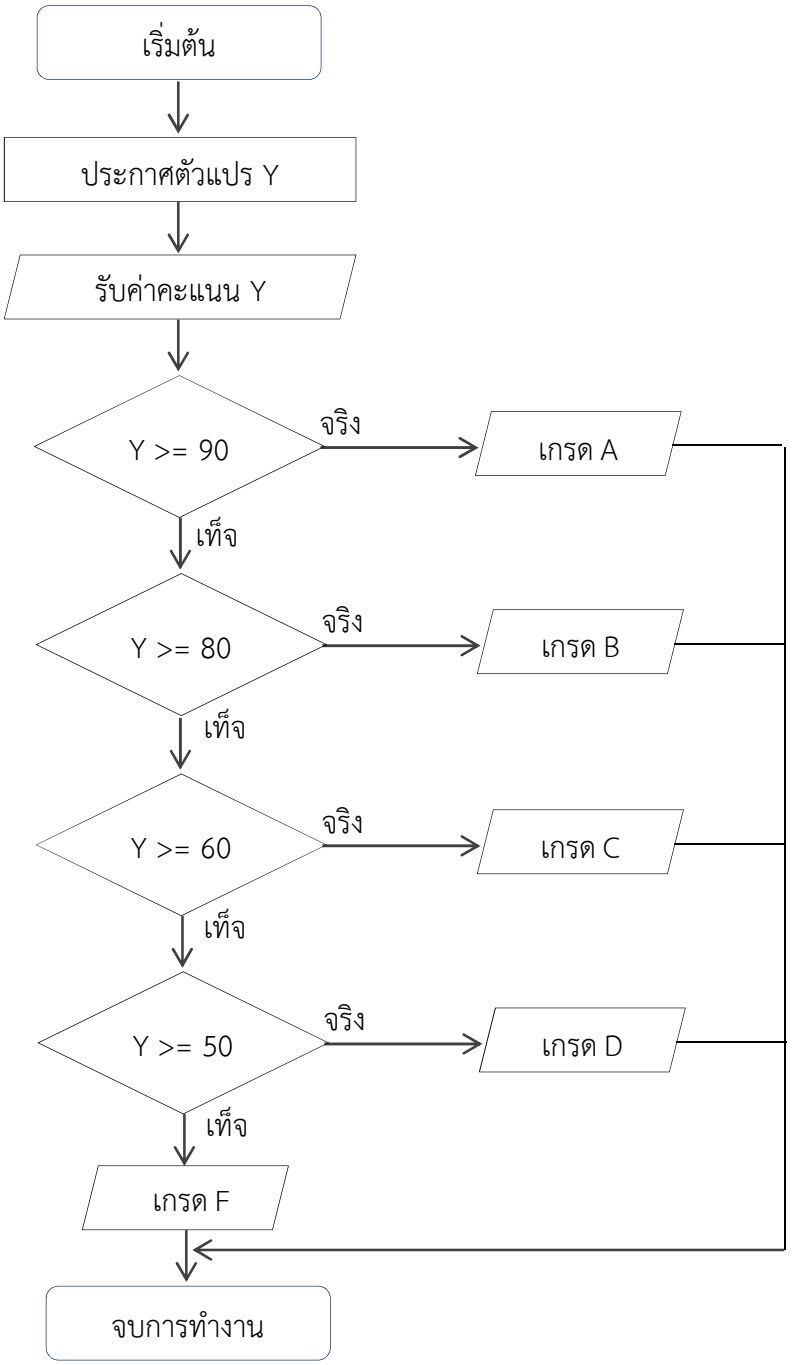


ภาพที่ 2.8 แสดงการออกแบบผังงานโดยมีเงื่อนไขเปรียบเทียบแบบหลายทาง

ตัวอย่างที่ 2.7 จงเขียนผังงาน (Flowchart) โดยรับค่าคะแนนจากนักศึกษา แล้วนำค่าที่นักศึกษาได้ป้อนมาทำตัดเกรด โดยมีเงื่อนไขของค่าคะแนนเกรดดังนี้

1.) ค่าคะแนน 90 – 100 เกรด A
2.) ค่าคะแนน 80 – 89 เกรด B
3.) ค่าคะแนน 60 – 79 เกรด C
4.) ค่าคะแนน 50 – 59 เกรด D
5.) ค่าคะแนนต่ำกว่า 50 เกรด F

<u>วิธีทำ</u>



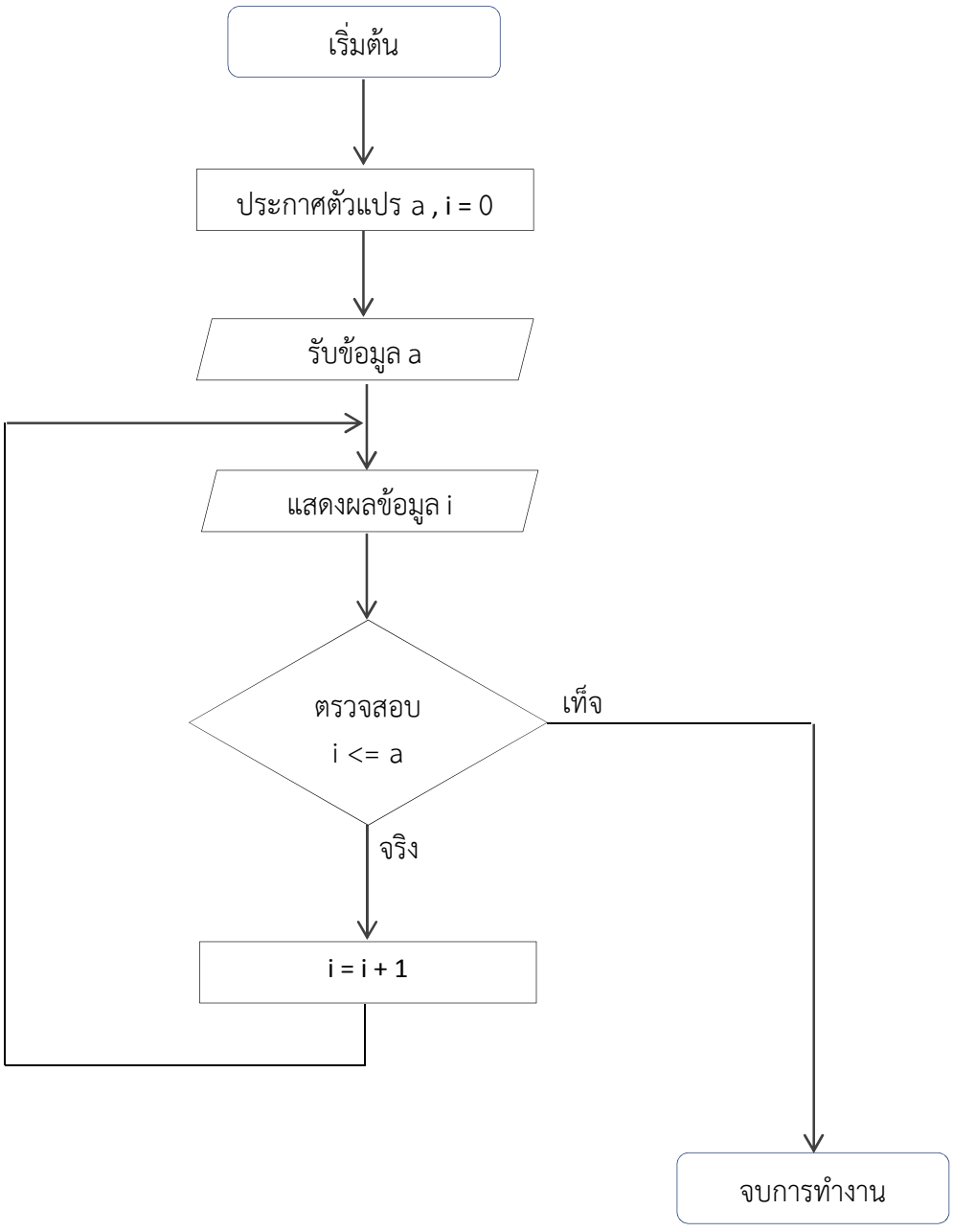
ภาพที่ 2.9 แสดงการออกแบบผังงานการทำงานของโปรแกรมในการตัดเกรดนักศึกษา

ตัวอย่างที่ 2.8 จงเขียนผังงาน (Flowchart) โดยรับค่าจากผู้ใช้งาน 1 ค่าแล้วนำค่าที่ได้มาแสดงผลตามจำนวนที่ผู้ใช้งานได้ป้อนเข้ามา

ตัวอย่าง เช่น หากผู้ใช้งานป้อน 5 โปรแกรมจะแสดง 0 1 2 3 4 5

หากผู้ใช้งานป้อน 9 โปรแกรมจะแสดง 0 1 2 3 4 5 6 7 8 9

วิธีทำ



ภาพที่ 2.10 แสดงการออกแบบผังงานโดยมีเงื่อนไขในการวนวูปกับไปทำงานซ้ำ

สรุปท้ายบท

ในการพัฒนาโปรแกรมคอมพิวเตอร์ผู้พัฒนาโปรแกรมคอมพิวเตอร์จำเป็นจะต้องมีหลักในการเขียนโปรแกรม โดยอาจจะใช้การเขียนอธิบายขบบวนการทำงานหรืออัลกอริทึม (Algorithm) แบบง่าย ๆ หรือผู้พัฒนาโปรแกรมอาจจะใช้การเขียนอธิบายการทำงานของโปรแกรมด้วยผังงาน (Flowchart) หรือยังมีอีกหนึ่งวิธีคือการใช้รหัสจำลอง (Pseudo code) ซึ่งวิธีการต่าง ๆ เหล่านี้จะเป็นวิธีที่จะให้ผู้พัฒนาสามารถมองเห็นการทำงานของโปรแกรมในภาพรวมทั้งหมดได้ และจะสามารถทำให้ผู้พัฒนาโปรแกรมสามารถทราบถึงปัญหาต่าง ๆ ที่อาจจะเกิดขึ้นในการพัฒนาโปรแกรมได้

รหัสจำลอง (Pseudo code) เป็นการอธิบายขั้นตอนการทำงานของโปรแกรมคอมพิวเตอร์ โดยใช้คำผสมกันระหว่างภาษาอังกฤษและภาษาการเขียนโปรแกรมแบบโครงสร้าง หรือบางครั้งหากผู้พัฒนาไม่มีความถนัดด้านภาษาอังกฤษ ผู้พัฒนาอาจจะใช้ภาษาไทยก็ได้เช่นเดียวกัน แต่หากเป็นไปได้ควรใช้หลักของภาษาอังกฤษเพราะคำสั่งการทำงานของโปรแกรมทั้งหมดเป็นภาษาอังกฤษ การเขียนเขียนรหัสจำลอง (Pseudo code) ที่ดีจะต้องมีความชัดเจน สั้น กะทัดรัดเข้าใจง่าย และได้ใจความ

ผังงาน (Flowchart) คือ การเขียนแผนภาพที่มีการใช้สัญลักษณ์รูปภาพและลูกศรที่แสดงถึงขั้นตอนการทำงานของโปรแกรมหรือระบบทีละขั้นตอน รวมไปถึงทิศทางการไหลของข้อมูลตั้งแต่แรกจนได้ผลลัพธ์ตามที่ต้องการ การเขียนผังงาน (Flowchart) เป็นรูปแบบการเขียนที่นิยมอย่างมากในการออกแบบเพื่อพัฒนางานทางด้านโปรแกรม เพราะการเขียนผังงาน (Flowchart) ออกแบบง่าย มีความเข้าใจง่ายและเป็นสากล ทำให้บุคคลอื่น ๆ ที่มาศึกษาสามารถเข้าใจระบบงานทั้งหมดได้ง่ายและตรงกัน โดยลักษณะรูปแบบของผังงานที่ใช้งานในงานทั่ว ๆ ไปจะมีรูปแบบที่นิยมใช้งานกันอยู่หลายรูปแบบ แต่ในการพัฒนาโปรแกรมจะมีรูปแบบการเขียนผังงานอยู่ไม่กี่รูปแบบ ซึ่งรูปแบบการเขียนผังงานในการพัฒนาโปรแกรมโดยทั่วไปจะมีอยู่ 2 แบบ คือ รูปแบบผังงานแบบลำดับ และรูปแบบผังงานแบบทางเลือก โดยรูปแบบผังงานแบบทางเลือก จะมีทางเลือกอยู่ 3 แบบ คือ ผังงานแบบทางเลือกแบบ 1 เส้นทาง ผังงานแบบทางเลือกแบบ 2 เส้นทาง และผังงานแบบทางเลือกแบบหลายเส้นทาง

คำถามทบทวน

1. จงอธิบายคุณสมบัติของรหัสจำลอง (Pseudo code) และผังงาน (Flowchart)

2. จงอธิบายข้อแตกต่างของรหัสจำลอง (Pseudo code) กับผังงาน (Flowchart) ว่ามีความแตกต่างกันอย่างไร

3. จงเขียนรหัสจำลอง (Pseudo code) โดยรับค่าจากผู้ใช้งาน 2 ค่า แล้วนำค่าที่มาทำการคูณกันแล้วแสดงคำตอบ และนำคำตอบมาทำการหาค่า หากมีค่าคำตอบมากกว่า 100 ให้โปรแกรมแสดงคำว่า “Big” แต่ถ้าค่าน้อยกว่าหรือเท่ากับ 100 ให้โปรแกรมแสดงคำว่า “Small”

4. จงเขียนรหัสจำลอง (Pseudo code) โดยรับค่าจากผู้ใช้งาน 3 ค่าแล้วนำค่าทั้งสามมาเปรียบกัน โดยมีเงื่อนไขว่าถ้าค่าใดมีค่ามากที่สุดให้แสดงผลค่านั้นออกทางจอให้ผู้ใช้งานทราบ

5. จงเขียนผังงาน (Flowchart) โดยรับค่าจากผู้ใช้งาน 3 ค่าแล้วนำค่าทั้งสามมาเปรียบกัน โดยมีเงื่อนไขว่าถ้าค่าใดมีค่ามากที่สุดให้แสดงผลค่านั้นออกทางจอให้ผู้ใช้งานทราบ

6. จงเขียนผังงาน (Flowchart) โดยรับค่าจากผู้ใช้งาน 5 ค่าแล้วหาค่าเฉลี่ย

7. จงเขียนผังงาน (Flowchart) โดยรับค่าตัวอักษรจากผู้ใช้งาน หากผู้ใช้งานป้อน ‘A’ ให้โปรแกรมแสดงผลคำว่า “good” และหากผู้ใช้งานป้อน ‘Z’ ให้แสดงคำว่า “bad” หากป้อนค่าอื่น ๆ ให้แสดงคำว่า “Error”

เอกสารอ้างอิง

คะชา ชาญศิลป์. (2548). **ภาษาซีสำหรับผู้เริ่มต้น.** กรุงเทพมหานคร: วิรัตน์เอ็ดดูเคชั่น.

ประภาพร ช่างไม้. (2545). **คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.** กรุงเทพมหานคร: อินโฟเพรส.

ธีรวัฒน์ ประกอบผล. (2553). **คู่มือการเขียนโปรแกรมภาษา C.** กรุงเทพมหานคร: ซิมพลิฟลาย.

สุระสิทธิ์ ทรงม้า และภูริพจน์ แก้วย่อง. (2552). **การโปรแกรมคอมพิวเตอร์.**: มหาวิทยาลัยราชภัฏสวนดุสิต.

สานนท์ เจริญฉาย. (2543). **การเขียนโปรแกรมและอัลกอริทึม.** กรุงเทพมหานคร: มหาจุฬาลงกรณ์ราชวิทยาลัย.

http://www.bcoms.net/site/bit//lesson/2_1.html

http:/www.thaigoodview.com/library/contest2552/type2/tech02.html

http://www.cptd.chandra.ac.th/selfstud/it4life/sub%20soft1.htm

http://ns2.spw.ac.th/poo/computer54/m354/lesson/lesson_2.html

https://sites.google.com/site/bbmm2553/prawati-khwam-pen-ma-khxng-phasa-si.html

http://boingboing.net/2011/10/12/dennis-ritchie-1941-2011creator-c-co-inventor.html

http://images.samartj.multiply.multiplycontent.com

บทที่ 3

ความเป็นมาของภาษาซีและการใช้งานโปรแกรมภาษาซี

แผนการสอนประจำบท

หัวข้อเนื้อหา

3.1 ประวัติภาษาซี
3.2 วิวัฒนาการของภาษาซี
3.3 จุดเด่นของภาษาซี
3.4 การใช้งานโปรแกรมภาษาซีด้วยโปรแกรม Turbo
    3.4.1 การเปิดใช้งานโปรแกรม
    3.4.2 การสร้างหน้าต่างโปรแกรมภาษาซีใหม่
    3.4.3 การป้อนคำสั่งภาษาซี
    3.4.4 การจัดเก็บโปรแกรมที่พัฒนาขึ้น
    3.4.5 การแปลภาษาซีหรือการคอมไพล์โปรแกรม
    3.4.6 การสั่งให้โปรแกรมทำงานหรือการทดสอบการทำงานของโปรแกรม

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถอธิบายประวัติความเป็นมาและวิวัฒนาการของภาษาซีได้
2. ผู้เรียนสามารถอธิบายและเข้าใจถึงจุดเด่นของภาษาซีได้
3. ผู้เรียนสามารถเข้าใจส่วนต่าง ๆ ในการทำงานของโปรแกรม Turbo C ได้

วิธีการสอน

1. สอนแบบบรรยายโดยใช้เอกสารประกอบการสอน
2. สอนแบบบรรยายโดยใช้ Slide Power Point
3. สอนโดยให้ผู้เรียนเป็นส่วนกลาง โดยใช้วิธีถามตอบกับผู้เรียนระหว่างบรรยาย
4. สอนโดยการลงปฏิบัติการใช้งานโปรแกรม
5. ให้ผู้เรียนสรุปและอธิบายหัวข้อที่ผู้เรียนที่ได้รับมอบหมาย

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. Slide Power Point
3. โปรแกรม Turbo C

การวัดผลและประเมินผล

1. การทำตามตัวอย่างโจทย์คำสั่ง การทดลองใช้โปรแกรม และการทำแบบฝึกหัด
2. ความตั้งใจในชั้นเรียน การตอบคำถาม และการร่วมกิจกรรมในชั้นเรียน

บทที่ 3

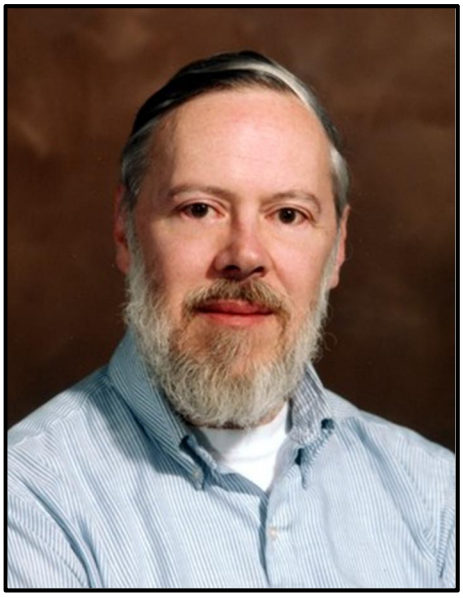
ความเป็นมาของภาษาซีและการใช้งานโปรแกรมภาษาซี

ภาษาซี เป็นภาษาคอมพิวเตอร์ที่ใช้ในการการเขียนโปรแกรมต่าง ๆ โดยผู้พัฒนาโปรแกรมสามารถประยุกต์ใช้กับงานต่าง ๆ ได้อย่างมากมาย เช่น ระบบปฏิบัติการคอมพิวเตอร์ทางคณิตศาสตร์ โปรแกรมทางระบบไฟฟ้า โปรแกรมทางระบบอิเล็กทรอนิกส์ หรือไมโครคอนโทรลเลอร์ โปรแกรมทางด้านระบบฐานข้อมูล เป็นต้น และภาษาซีถือได้ว่าเป็นรากฐานพื้นฐานที่สำคัญของภาษาอื่น ๆ ไม่ว่าภาษา C++ ภาษาจาวา (Java) ภาษา PHP ก็ใช้โครงสร้างคำสั่งแบบเดียวกันกับภาษาซี หรือแม้กระทั่งระบบลีนุกซ์ก็ถูกพัฒนามาจากระบบยูนิกซ์ ซึ่งก็เป็นที่รู้โดยทั่วไปว่าระบบปฏิบัติการตระกูลยูนิกซ์มีการพัฒนามาจากรากฐานของภาษาซี

ภาษาซีเป็นภาษาระดับกลาง คือ ไม่เป็นภาษาระดับต่ำแบบภาษาเครื่อง หรือภาษาแอสเซมบลี เนื่องจากผู้พัฒนาโปรแกรมสามารถจัดการเกี่ยวกับเรื่องของคำสั่งได้อย่างสะดวกและอิสระ และในบางครั้งผู้พัฒนาโปรแกรมสามารถควบคุมฮาร์ดแวร์ผ่านทางภาษาซี ได้ราวกับภาษาแอสเซมบลี ด้วยข้อดีต่าง ๆ ที่หลากหลายทำให้โปรแกรมที่ถูกเขียนด้วยภาษาซีมีความเร็วในการปฏิบัติงานสูงกว่าภาษาทั่ว ๆ ไป

3.1 ประวัติภาษาซี

ภาษาซีเป็นภาษาที่ถูกพัฒนาโดยเดนนิส ริดชี (Dennis Ritche) แห่งห้องทดลอง ที่เมอร์รีฮิลมลรัฐนิวเจอร์ซี่ โดยเดนนิสได้ใช้หลักการของภาษาบีซีพีแอล (BCPL : Basic Combine Programming Language) ซึ่งพัฒนาขึ้นโดยเคน ทอมสัน (Ken Tomson) การออกแบบและพัฒนาภาษาซีของเดนนิส ริดชี มีจุดมุ่งหมายให้เป็นภาษาสำหรับใช้เขียนโปรแกรมปฏิบัติการระบบยูนิกซ์ และได้ตั้งชื่อว่า ซี (C) เพราะเห็นว่า ซี (C) เป็นตัวอักษรต่อจากบี (B) ของภาษา BCPL ภาษาซีถือว่าเป็นภาษาระดับสูงและภาษาระดับต่ำ ทั้งนี้เพราะ ภาษาซีมีวิธีใช้ข้อมูลและมีโครงสร้างการควบคุมการทำงานของโปรแกรมเป็นอย่างเดียวกับภาษาของโปรแกรมระดับสูงอื่น ๆ จึงถือว่าเป็นภาษาระดับสูง ในด้านที่ถือว่าภาษาซีเป็นภาษาระดับต่ำ เพราะภาษาซีมีวิธีการเข้าถึงในระดับต่ำที่สุดของฮาร์ดแวร์ความสามารถทั้งสองด้านของภาษานี้เป็นสิ่งที่เกื้อหนุนซึ่งกันและกัน ความสามารถระดับต่ำทำให้ภาษาซีสามารถใช้เฉพาะเครื่องได้และความสามารถระดับสูง ทำให้ภาษาซีเป็นอิสระจากฮาร์ดแวร์ ภาษาซีสามารถสร้างรหัสภาษาเครื่องซึ่งตรงกับชนิดของข้อมูลนั้นได้เอง ทำให้โปรแกรมที่เขียนด้วยภาษาซีที่เขียนบนเครื่องหนึ่ง สามารถนำไปใช้กับอีกเครื่องหนึ่งได้ ประกอบกับการใช้พอยน์เตอร์ในภาษาซี นับได้ว่าเป็นตัวอย่างที่ดีของการเป็นอิสระจากฮาร์ดแวร์



ภาพที่ 3.1 เดนนิส ริดชี (Dennis Ritche) ผู้พัฒนาภาษาซี
(ที่มา http://media.boingboing.net/wp-content/uploads/2011/10/dennis_ritchie.jpg)

3.2 วิวัฒนาการของภาษาซี

- ค.ศ. 1970 มีการพัฒนาภาษา B โดย Ken Thompson ซึ่งทำงานบนเครื่อง DEC PDP-7 ซึ่ง ทำงานบนเครื่องไมโครคอมพิวเตอร์ไม่ได้ และยังมีข้อจำกัดในการใช้งานอยู่ (ภาษา B สืบทอดมาจากภาษา BCPL ซึ่งเขียนโดย Marth Richards)

- ค.ศ. 1972 Dennis M. Ritchie และ Ken Thompson ได้สร้างภาษา C เพื่อเพิ่มประสิทธิภาพ ภาษา B ให้ดียิ่งขึ้น ในระยะแรกภาษา C ไม่เป็นที่นิยมแก่นักโปรแกรมเมอร์โดยทั่วไปนัก

- ค.ศ. 1978 Brian W. Kernighan และ Dennis M. Ritchie ได้เขียนหนังสือเล่มหนึ่งชื่อว่า The C Programming Language และหนังสือเล่มนี้ทำให้บุคคลทั่วไปรู้จักและนิยมใช้ภาษา C ในการเขียน โปรแกรมมากขึ้น

- ค. ศ. 1981 แต่เดิมภาษา C ใช้ Run บนเครื่องคอมพิวเตอร์ 8 bit ภายใต้ระบบปฏิบัติการ CP/M ของ IBM PC ซึ่งในช่วงปี ค. ศ. 1981 เป็นช่วงของการพัฒนาเครื่องไมโครคอมพิวเตอร์ ภาษา C จึงมี บทบาทสำคัญในการนำมาใช้บนเครื่อง PC ตั้งแต่นั้นเป็นต้นมา และมีการพัฒนาต่อมาอีกหลาย ๆ ค่าย ดังนั้นเพื่อกำหนดทิศทางการใช้ภาษา C ให้เป็นไปแนวทางเดียวกัน ANSI (American National Standard Institute) ได้กำหนดข้อตกลงที่เรียกว่า 3J11 เพื่อสร้างภาษา C มาตรฐานขึ้นมา เรียนว่า ANSI C

- ค.ศ. 1983 Bjarne Stroustrup แห่งห้องปฏิบัติการเบล (Bell Laboratories) ได้พัฒนาภาษา C++ ขึ้นรายละเอียดและความสามารถของ C++ มีส่วนขยายเพิ่มจาก C ที่สำคัญ ๆ ได้แก่ แนวความคิดของการเขียนโปรแกรมแบบกำหนดวัตถุเป้าหมายหรือแบบ OOP (Object Oriented Programming) ซึ่งเป็นแนวการเขียนโปรแกรมที่เหมาะกับการพัฒนาโปรแกรมขนาดใหญ่ที่มีความสลับซับซ้อนมาก มีข้อมูลที่ใช้ในโปรแกรมจำนวนมาก จึงนิยมใช้เทคนิคของการเขียนโปรแกรมแบบ OOP ในการพัฒนาโปรแกรมขนาดใหญ่ในปัจจุบันนี้

3.3 จุดเด่นของภาษาซี

3.3.1 ภาษาซีเป็นภาษาที่มีลักษณะเป็นโครงสร้างจึงเขียนโปรแกรมง่าย โปรแกรมที่เขียนขึ้นจะทำงานได้อย่างมีประสิทธิภาพสูง สั่งงานคอมพิวเตอร์ได้รวดเร็วกว่าภาษาระดับสูงอื่น ๆ

3.3.2. ภาษาซีเป็นภาษาที่สามารถสั่งงานอุปกรณ์ระบบคอมพิวเตอร์ได้เกือบทุกภาคส่วนของฮาร์ดแวร์ ซึ่งภาษาระดับระดับสูงในภาษาอื่นทำงานดังกล่าวได้น้อยกว่า

3.3.3 คอมไพเลอร์ภาษาซีทุกโปรแกรมจะทำงานอ้างอิงมาตรฐาน ANSI (American National Standards Institute) เกือบทั้งหมด จึงทำให้โปรแกรมที่เขียนขึ้นด้วยภาษาซีสามารถนำไปใช้กับคอมพิวเตอร์ได้ทุกรุ่นที่มาตรฐาน ANSI รับรอง

3.3.4 โปรแกรมที่เขียนขึ้นด้วยภาษาซีสามารถนำไปใช้กับเครื่องคอมพิวเตอร์ที่ใช้ซีพียูต่างเบอร์กันได้ หรือกล่าวได้ว่าโปรแกรมมีความยืดหยุ่นสูง

3.3.5 ภาษาซีสามารถนำไปใช้ในการเขียนโปรแกรมประยุกต์ได้หลายระดับ เช่น การเขียนโปรแกรมจัดระบบงาน (OS) คอมไพเลอร์ของภาษาอื่น โปรแกรมสื่อสารข้อมูลโปรแกรมจัดฐานข้อมูล โปรแกรมปัญญาประดิษฐ์ รวมทั้งโปรแกรมคำนวณงานทางด้านวิทยาศาสตร์และวิศวกรรมศาสตร์ เป็นต้น

3.3.6 ภาษาซีมีโปรแกรมช่วย (tool box) ที่ช่วยในการเขียนโปรแกรมมาก และราคาไม่แพงหาซื้อได้ง่าย เช่น vitanin c หรืออื่น ๆ

3.3.7 ภาษาซีสามารถประกาศข้อมูลได้หลายชนิดและหลายรูปแบบ ทำให้สะดวกรวดเร็วต่อการพัฒนาโปรแกรมตามวัตถุประสงค์ของผู้ใช้

3.3.8 ภาษาซีสามารถประยุกต์ใช้ในงานสื่อสารข้อมูล และงานควบคุมที่ต้องการความแม่นยำในเรื่องเวลา (real time application) ได้ดีกว่าภาษาระดับสูงอื่น ๆ หลาย ๆ ภาษา

3.3.9 ภาษาซีสามารถเขียนโปรแกรมด้วยเทคนิคแบบโอโอพี OOP (Object Oriented Programming) ได้หากใช้ภาษาซีรุ่น Turbo C++ ขึ้นไป ทำให้สามารถพัฒนาโปรแกรมประยุกต์เพื่อใช้งานได้กว้างขวางมากยิ่งขึ้นกว่าเดิม

3.4 การใช้งานโปรแกรมภาษาซี

ในการพัฒนาโปรแกรมภาษาซีผู้พัฒนาโปรแกรมจะต้องมีโปรแกรมที่ใช้ในการเขียนและคอมไพ (Compile) ภาษาซีขึ้นมาก่อน จึงจะสามารถพัฒนาโปรแกรมต่าง ๆ ขึ้นที่ต้องมาได้ โดยโปรแกรมที่ใช้ในการเขียนและคอมไพเลอร์ (Compile) ภาษาซีมีหลายโปรแกรมด้วยกัน เช่น Borland C หรืออาจจะเรียกว่า Turbo C , Dev C++ , Visual C++ เป็นต้น ดังนั้นในการเลือกใช้งานว่าผู้พัฒนาโปรแกรมจะใช้งานตัวใดนั้น ต้องจะคำนึงถึงงานที่ผู้พัฒนาโปรแกรมต้องการว่าจะให้งานออกมาในรูปแบบใด แต่สำหรับผู้ที่ต้องการศึกษาภาษาเบื้องต้น ควรใช้โปรแกรม Turbo C ก่อนเนื่องจากโปรแกรม Turbo C จัดเป็นโปรแกรมที่ใช้งานง่าย สะดวก และที่สำคัญจะเป็นโปรแกรมพื้นฐานสุดในการทดลองคำสั่งของภาษาซี

3.4.1 การใช้งานโปรแกรม Turbo C เบื้องต้น

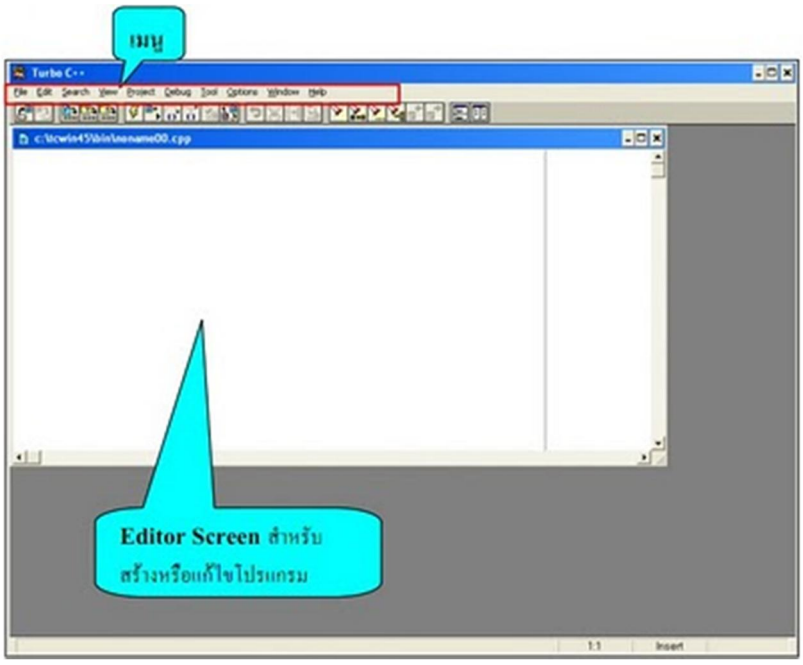
ในการใช้งานโปรแกรม Turbo C ในการทดลองคำสั่งภาษาซีครั้งนี้ จะเลือกใช้งานโปรแกรม Turbo C++ เวอรชั่น 4.5 ซึ่งผลิตโดยบริษัท Borland เป็นซอฟต์แวร์ที่มีหน้าที่ที่สำคัญหลายประการ ดังนี้

1.) ใช้ในการสร้างและแก้ไขโปรแกรมภาษาซี (Create, Edit)
2.) ใช้ในการแปลภาษาซีเป็นภาษาเครื่อง (Compiler)
3.) ใช้ในการทดสอบการทำงานของโปรแกรม (Run)
4.) ใช้ในการตรวจสอบจุดบกพร่องของโปรแกรม

การใช้งานโปรแกรม Turbo C++ 4.5 มีการใช้งานที่ง่ายไม่ซับซ้อน โดยการใช้งานโปรแกรม Turbo C++ 4.5 มีส่วนที่สำคัญของโปรแกรมดังนี้

1.) การเปิดใช้งานโปรแกรม Turbo C++ 4.5

หลังจากการที่ผู้ใช้งานได้ทำการติดตั้งโปรแกรม Turbo C++ 4.5 ดังนั้นผู้ใช้งานสามารถเรียกใช้งาน Turbo C++ 4.5 ขึ้นมาได้ โดยการเข้าไปที่ Start --> Programs --> Turbo C++ 4.5 --> Turbo C++ จากนั้นโปรแกรมจะแสดงโปรแกรม Turbo C++ 4.5 ดังภาพที่ 3.2

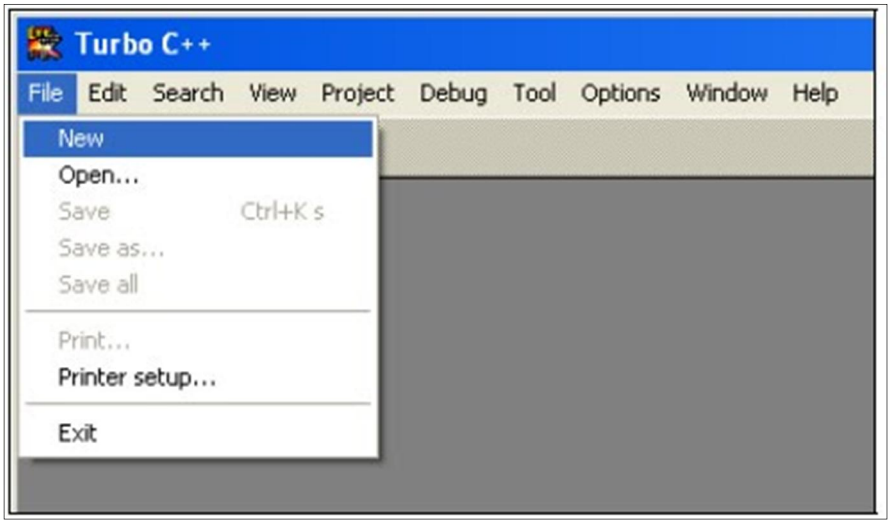


ภาพที่ 3.2 แสดงหน้าหลักโปรแกรม Turbo C++ เวอรชั่น 4.5
(ที่มา http://abarm2007.blogspot.com/2011/01/turbo-c-45.html)

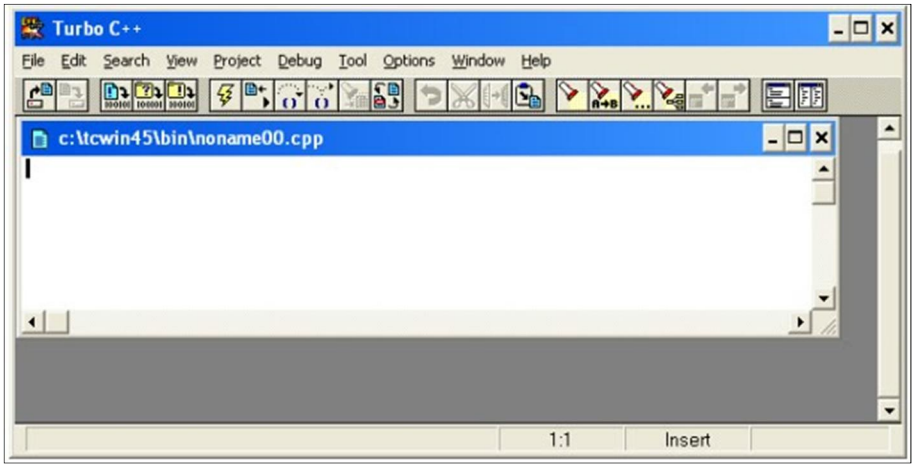
- **Editor Screen** คือหน้าจอสำหรับสร้างหรือแก้ไขโปรแกรมภาษาซี โดยผู้ใช้งานโปรแกรมมีหน้าที่ในการสร้างโปรแกรมในหน้าจอนี้ หรือนำโปรแกรมที่เคยสร้างไว้เรียกคืนมาแก้ไข
- **เมนู** ประกอบด้วยเมนูหลักและเมนูย่อย มีไว้สำหรับสั่งปฏิบัติงานใด ๆ กับโปรแกรมภาษาซีในจอภาพ Editor Screen เช่น Compile, Run หรือ Debug

2.) การสร้างหน้าต่างโปรแกรมภาษาซีใหม่ในโปรแกรม Turbo C++ 4.5

การสร้างหน้าตาโปรแกรมภาษาซีใหม่นั้นจะใช้เมนู File -> New ดังภาพที่ 3.3 และจากนั้นโปรแกรมจะแสดงหน้าตา Editor Screen ขึ้นมาใหม่เพื่อใช้ในการเขียนโปรแกรม ดังภาพที่ 3.4



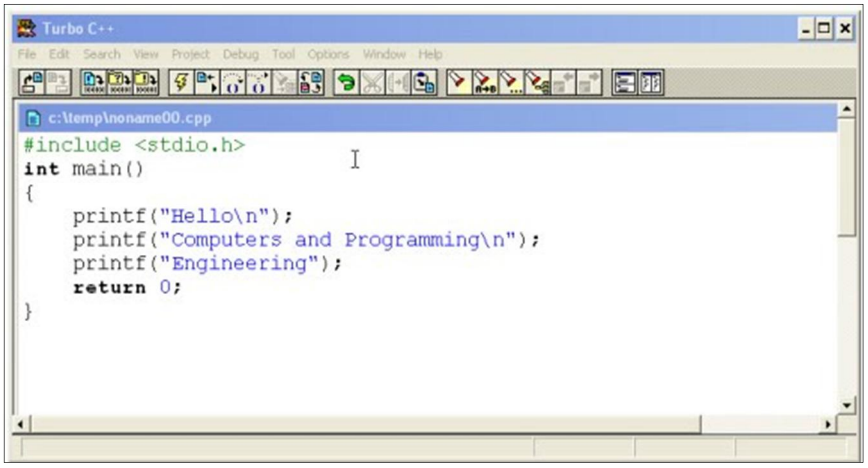
ภาพที่ 3.3 แสดงการสร้างหน้าตาโปรแกรมภาษาซี
(ที่มา http://abarm2007.blogspot.com/2011/01/turbo-c-45.html)



ภาพที่ 3.4 แสดงการสร้างหน้าต่าง Editor Screen
(ที่มา http://abarm2007.blogspot.com/2011/01/turbo-c-45.html)

3.) การป้อนคำสั่งภาษาซีลงในโปรแกรม Turbo C++ 4.5

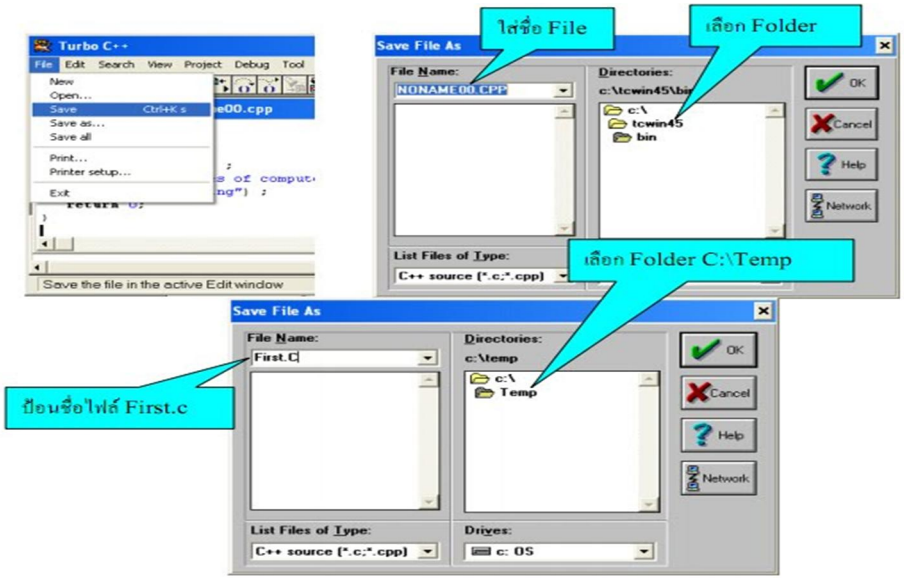
การป้อนคำสั่งภาษาซีลงในโปรแกรม Turbo C++ 4.5 สามารถทำได้โดยการพิมพ์คำสั่งต่าง ๆ ของภาษาซีลงไปในหน้าต่าง Editor Screen ดังภาพที่ 3.5 ทั้งนี้คำสั่งที่พิมพ์ลงไปจะต้องถูกหลักไวยากรณ์ของคำสั่งของภาษาซีที่ได้มีกำหนดเอาไว้



ภาพที่ 3.5 แสดงการป้อนคำสั่งภาษาซีลงในโปรแกรม Turbo C++ 4.5
(ที่มา http://abarm2007.blogspot.com/2011/01/turbo-c-45.html)

4.) การจัดเก็บโปรแกรมที่พัฒนาขึ้นจากโปรแกรม Turbo C++ 4.5

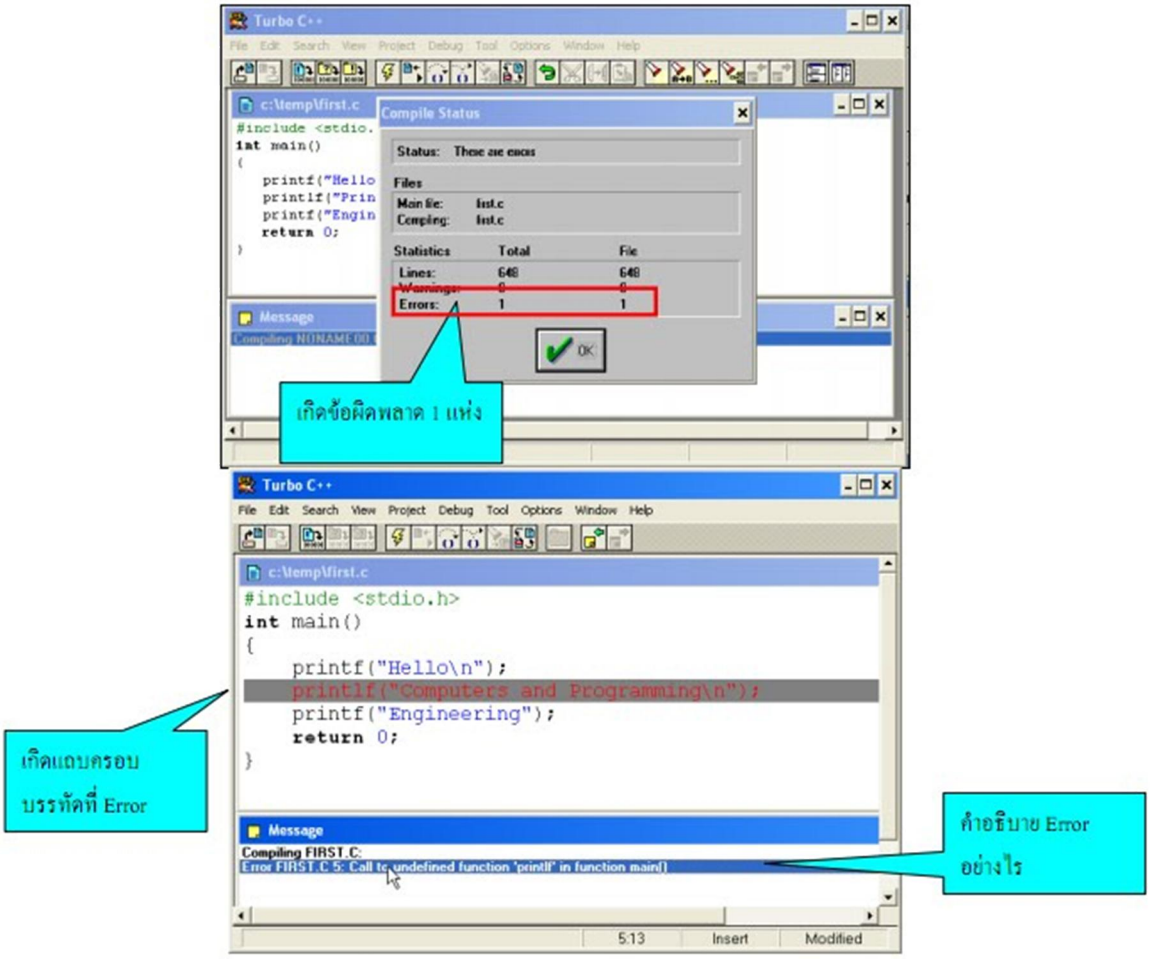
ในการจัดเก็บโปรแกรมที่สร้างหรือพัฒนาขึ้น ผู้ใช้งานสามารถสามารถทำขึ้นเพื่อจัดเก็บโปรแกรมไว้สำหรับการนำกลับมาใช้งานอีกครั้งในภายหลัง ซึ่งขบวนการจัดเก็บจะมีความคล้ายกับการจัดเก็บเอกสารต่าง ๆ โดยการจัดเก็บโปรแกรมที่พัฒนาขึ้นจากโปรแกรม Turbo C++ 4.5 สามารถทำได้โดยการไปที่ใช้เมนู File -> Save ดังภาพที่ 3.6



ภาพที่ 3.6 แสดงการจัดเก็บโปรแกรมที่พัฒนาขึ้นจากโปรแกรม Turbo C++ 4.5
(ที่มา http://abarm2007.blogspot.com/2011/01/turbo-c-45.html)

5.) การแปลภาษาซีหรือการคอมไพล์โปรแกรม

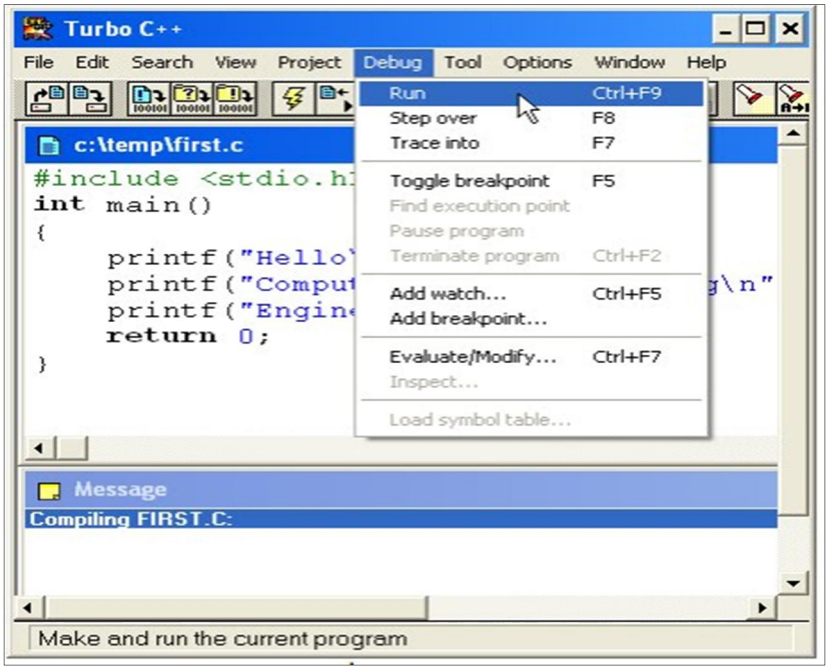
การแปลภาษาซีหรือการคอมไพล์ (Compile) คือการตรวจสอบโปรแกรมที่เขียนขึ้นถูกต้องตามหลักไวยากรณ์ของคำสั่งของภาษาซีหรือไม่ หรือกล่าวอีกนัยหนึ่งว่าการคอมไพล์ในภาษาซี คือ การแปลโปรแกรมภาษาซีให้เป็นภาษาเครื่อง หากแปลได้หมดก็สามารถสั่งให้โปรแกรมทำงานได้ แต่หากแปลไม่ได้ก็จะแจ้งข้อผิดพลาดให้ทราบ ผู้เรียกคอมไพล์ก็ต้องแก้ไขโปรแกรมภาษาซีให้ถูกต้องเสียก่อนจึงเรียกคอมไพล์ใหม่ โดยการแปลภาษาซีหรือการคอมไพล์ (Compile) จากโปรแกรม Turbo C++ 4.5 สามารถทำได้โดยการไปที่ใช้เมนู Project -> Compile ดังภาพที่ 3.7



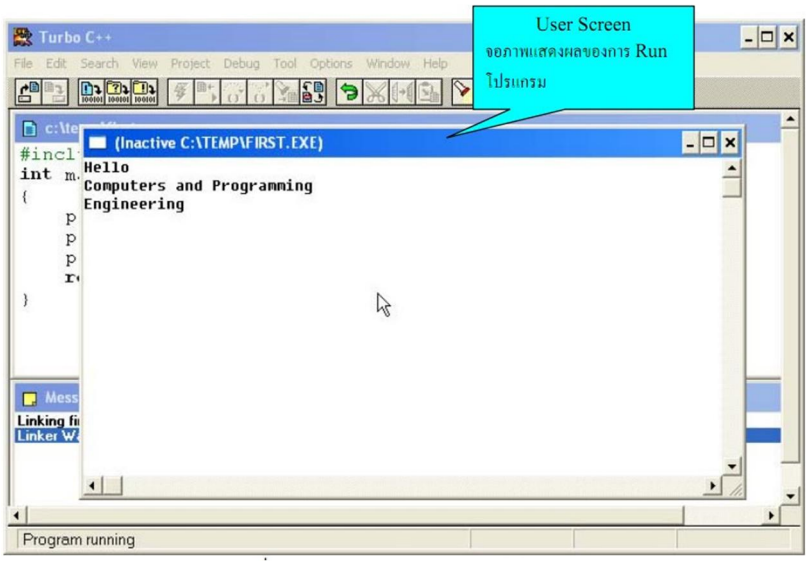
ภาพที่ 3.7 แสดงการแปลภาษาซีหรือการคอมไพล์จากโปรแกรม Turbo C++ 4.5
(ที่มา http://abarm2007.blogspot.com/2011/01/turbo-c-45.html)

6.) การสั่งให้โปรแกรมทำงานหรือการทดสอบการทำงานของโปรแกรม

การสั่งให้โปรแกรมทำงานหรือการทดสอบการทำงานของโปรแกรม หรือหลายคนนิยมเรียกว่าการรันโปนแกรม (RUN) ซึ่งการ Run โปรแกรม คือ การสั่งให้โปรแกรมทำงานตามขั้นตอนที่กำหนดในโปรแกรมภาษาซีที่เขียนขึ้น การ Run โปรแกรมจากโปรแกรม Turbo C++ 4.5 ทำได้โดยเลือกเมนู Debug ->Run ดังภาพที่ 3.8 หลังจากสั่งให้โปรแกรมทำงานแล้วผลลัพธ์ของโปรแกรมจะปรากฏในจอภาพอีกจอภาพหนึ่ง โดยจอภาพดังกล่าวชื่อว่า User Screen ดังภาพที่ 3.9



ภาพที่ 3.8 แสดงการทดสอบการทำงานของโปรแกรม
(ที่มา http://abarm2007.blogspot.com/2011/01/turbo-c-45.html)



ภาพที่ 3.9 แสดงผลลัพธ์ของโปรแกรม
(ที่มา http://abarm2007.blogspot.com/2011/01/turbo-c-45.html)

สรุปท้ายบท

ภาษาซีเป็นภาษาคอมพิวเตอร์ที่ใช้ในการการเขียนโปรแกรมต่าง ๆ โดยผู้พัฒนาโปรแกรมสามารถประยุกต์ใช้กับงานต่าง ๆ ได้อย่างมากมาย เช่น ระบบปฏิบัติการคอมพิวเตอร์ทางคณิตศาสตร์ โปรแกรมทางระบบไฟฟ้า โปรแกรมทางระบบอิเล็กทรอนิกส์ หรือไมโครคอนโทรลเลอร์ โปรแกรมทางด้านระบบฐานข้อมูล เป็นต้น และภาษาซีถือได้ว่าเป็นรากฐานพื้นฐานที่สำคัญของภาษาอื่น ๆ ไม่ว่าภาษา C++ ภาษาจาวา ภาษา PHP ก็ใช้โครงสร้างคำสั่งแบบเดียวกันกับภาษาซี หรือแม้กระทั่งระบบลีนุกซ์ก็ถูกพัฒนามาจากระบบยูนิกซ์ ซึ่งก็เป็นที่รู้โดยทั่วไปว่าระบบปฏิบัติการตระกูลยูนิกซ์มีการพัฒนามาจากรากฐานของภาษาซี ภาษาซีเป็นภาษาระดับกลาง คือ ไม่เป็นภาษาระดับต่ำแบบภาษาเครื่อง หรือภาษาแอสเซมบลี เนื่องจากผู้พัฒนาโปรแกรมสามารถจัดการเกี่ยวกับเรื่องของคำสั่งได้อย่างสะดวกและอิสระ และในบางครั้งผู้พัฒนาโปรแกรมสามารถควบคุมฮาร์ดแวร์ผ่านทางภาษาซี ได้ราวกับภาษาแอสเซมบลี ด้วยข้อดีต่าง ๆ ที่หลากหลายทำให้โปรแกรมที่ถูกเขียนด้วยภาษาซีมีความเร็วในการปฏิบัติงานสูงกว่าภาษาคอมพิวเตอร์ทั่วไป

จุดเด่นของภาษาซี เป็นภาษาที่มีลักษณะเป็นโครงสร้างจึงเขียนโปรแกรมง่าย โปรแกรมที่เขียนขึ้นจะทำงานได้อย่างมีประสิทธิภาพสูง ภาษาซีเป็นภาษาที่สามารถสั่งงานอุปกรณ์ระบบคอมพิวเตอร์ได้เกือบทุกภาคส่วนของฮาร์ดแวร์ คอมไพเลอร์ภาษาซีทุกโปรแกรมจะทำงานอ้างอิงมาตรฐาน ANSI (American National Standards Institute) ภาษาซีสามารถนำไปใช้ในการเขียนโปรแกรมประยุกต์ได้หลายระดับ เช่น การเขียนโปรแกรมจัดระบบงาน (OS) คอมไพเลอร์ของภาษาอื่น โปรแกรมสื่อสารข้อมูลโปรแกรมจัดฐานข้อมูล ภาษาซีมีโปรแกรมช่วยที่ช่วยในการเขียนโปรแกรมได้ง่ายมากขึ้นมาก ภาษาซีสามารถประกาศข้อมูลได้หลายชนิดและหลายรูปแบบ ทำให้สะดวกรวดเร็วต่อการพัฒนาโปรแกรมตามวัตถุประสงค์ของผู้ใช้ ภาษาซีสามารถประยุกต์ใช้ในงานสื่อสารข้อมูล และงานควบคุมที่ต้องการความแม่นยำในเรื่องเวลา ภาษาซีสามารถเขียนโปรแกรมด้วยเทคนิคแบบโอโอพี OOP (Object Oriented Programming) ทำให้สามารถพัฒนาโปรแกรมประยุกต์เพื่อใช้งานได้กว้างขวางมากยิ่งขึ้นกว่าเดิม

คำถามทบทวน

1. จงอธิบายประวัติความเป็นมาของภาษาซีมาให้เข้าใจ
2. จงอธิบายวิวัฒนาการของภาษาซี
3. จงอธิบายจุดเด่นของภาษาซี
4. จงอธิบายเหตุผลใดที่ผู้เรียนจึงต้องเลือกใช้ภาษาซีในการพัฒนาโปรแกรมคอมพิวเตอร์
5. จงอธิบายการใช้งานโปรแกรม Turbo C++ 4.5 มาให้พอสังเขป

เอกสารอ้างอิง

คะชา ชาญศิลป์. (2548). **ภาษาซีสำหรับผู้เริ่มต้น.** กรุงเทพมหานคร: วิรัตน์เอ็ดดูเคชั่น.

ประภาพร ช่างไม้. (2545). **คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.** กรุงเทพมหานคร: อินโฟเพรส.

ธีรวัฒน์ ประกอบผล. (2553). **คู่มือการเขียนโปรแกรมภาษา C.** กรุงเทพมหานคร: ซิมพลิฟลาย.

สุระสิทธิ์ ทรงม้า และภูริพจน์ แก้วย่อง. (2552). **การโปรแกรมคอมพิวเตอร์.**: มหาวิทยาลัยราชภัฏสวนดุสิต.

อรพิน ประวัติบริสุทธิ์. (2554). **คู่มือการเรียนภาษา C ฉบับปรับปรุงใหม่** (พิมพ์ครั้งที่ 10)**.** กรุงเทพมหานคร: โปรวิชั่น.

https://www.sites.google.com/site/bbmm2553/prawati-khwam-pen-ma-khxng.html

http://www.boingboing.net/2009/10/12/dennis-computer-scientist-unix-c-co-inventor.html

http://www.images.samartj.multiply.multiplycontent.com/attachment1-08.html

http://www.iam.hunsa.com/s505420123/article/22094.html

http://www.media.boingboing.net/wp-content/uploads/2011/10/dennis_ritchie.jpg

http://www.abarm2007.blogspot.com/2011/01/turbo-c-45.html

บทที่ 4

โครงสร้างพื้นฐานของภาษาซี

แผนการสอนประจำบท

หัวข้อเนื้อหา

4.1 หัวฟังก์ชัน
4.2 ฟังก์ชั่นหลัก
4.3 ชุดคำสั่ง
4.4 การอธิบายโปรแกรม
4.5 คำสงวน
4.6 ตัวดำเนินการ

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถทำความเข้าใจและแยกแยะส่วนต่าง ๆ ของโครงสร้างพื้นฐานของภาษาซีได้
2. ผู้เรียนสามารถอธิบายและเข้าใจตัวดำเนินการในแต่ละประเภท และสามารถนำตัวดำเนินการในแต่ละประเภทไปใช้งานได้ถูกต้อง

วิธีการสอน

1. สอนแบบบรรยายโดยใช้เอกสารประกอบการสอน
2. สอนแบบบรรยายโดยใช้ Slide Power Point
3. สอนโดยให้ผู้เรียนเป็นส่วนกลาง โดยใช้วิธีถามตอบกับผู้เรียนระหว่างบรรยาย
4. ให้ผู้เรียนสรุปและอธิบายหัวข้อที่ผู้เรียนที่ได้รับมอบหมาย

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. Slide Power Point
3. ตัวอย่างแบบทดสอบ

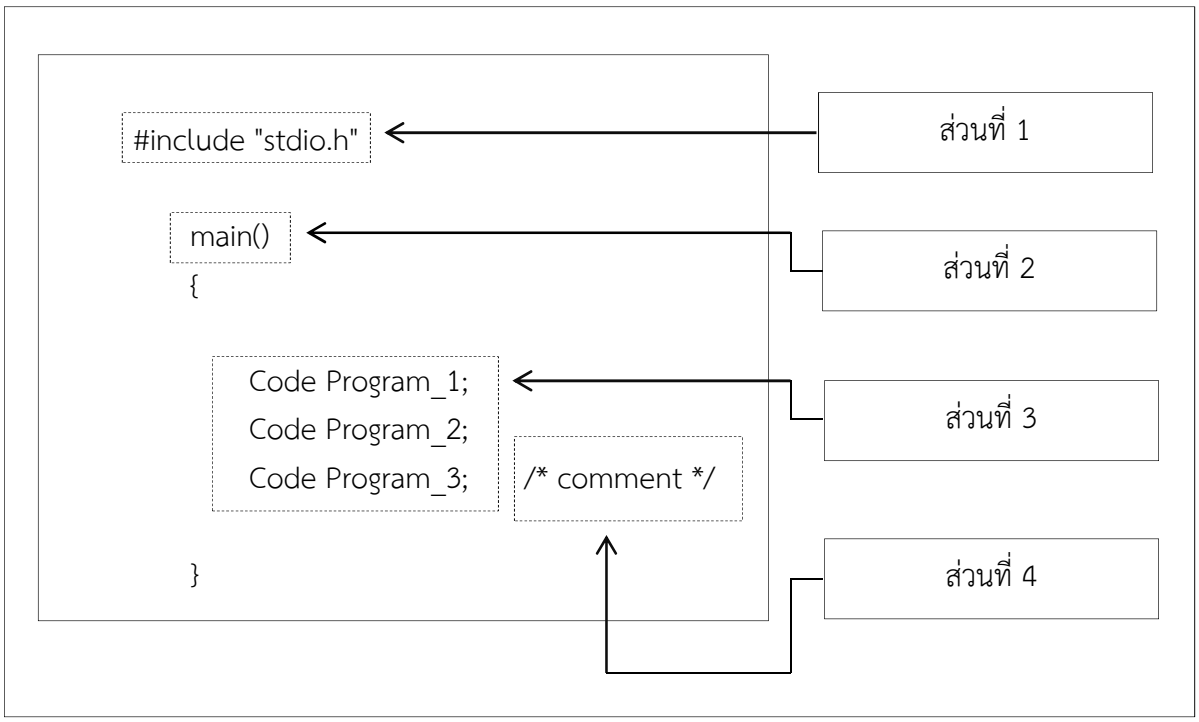
การวัดผลและประเมินผล

1. การทำแบบฝึกหัด
2. ความตั้งใจในชั้นเรียนและการร่วมมือในการนำเสนอหรืออภิปราย
3. สังเกตจากการตอบคำถามและการร่วมทำกิจกรรมในชั้นเรียน

บทที่ 4

โครงสร้างพื้นฐานของภาษาซี

ภาษาคอมทุกภาคจะต้องมีโครงสร้างของภาษา เนื่องจากโครงสร้างของภาษาคอมพิวเตอร์จะเป็นรูปแบบโครงร่างในการเขียนโปรแกรมของภาษานั้น ๆ เช่นเดียวกันกับภาษาซีที่จะต้องมีโครงสร้างพื้นฐานในการพัฒนาโปรแกรมหรือการเขียนโปรแกรม ข้อดีอีกประการหนึ่งของภาษาซี คือ ภาษาซีเป็นภาษาคอมพิวเตอร์ภาษาหนึ่งที่มีโครงสร้างภาษาที่เข้าใจง่าย และง่ายต่อการเขียนโปรแกรม



ภาพที่ 4.1 โครงสร้างพื้นฐานของภาษาซี

4.1 หัวฟังก์ชัน (ส่วนที่ 1)

ส่วนหัวของฟังก์ชันหรืออาจจะเรียกว่า พรีโปรเซสเซอร์ไดเรกทฟี (Preprocessor directives) เป็นส่วนที่ทุกโปรแกรมต้องมีใช้สำหรับเรียกไฟล์ที่โปรแกรมต้องการและกำหนดค่าต่าง ๆ ซึ่งจะต้องเริ่มต้นด้วยเครื่องหมายไดเรกทีฟ (#) และตามด้วยชื่อที่ต้องการกำหนดค่า ตัวอย่างไดเรกทีฟที่ใช้บ่อย คือ คำสั่ง #include เป็นการบอกให้ตัวแปลคำสั่ง (compiler) อ่านไฟล์อื่นเข้ามาร่วมในการคอมไพล์

ตารางที่ 4.1 ตัวอย่างโปรเซสเซอร์ไดเรกทฟีที่มีการใช้งาน

คำสั่ง	คำอธิบาย
#include	Include text from a file
#define	Define a macro
#undef	Undefined a macro
#if	Test if a compile-time condition holed
#ifdef	Test if a symbol is defined
#else	Indicate alternative if a test file fail
#line	Give a line number for compiler message

4.2 ฟังก์ชั่นหลัก (ส่วนที่ 2)

ส่วนของฟังก์ชันหลักหรือในหรือ main program เป็นส่วนที่ภาษาซีทุกโปรแกรมจะต้องมี เพราะจะเป็นส่วนที่จะประกาศให้ตัวแปลคำสั่ง (compiler) รู้ว่าคำสั่งข้างในชุดปีกกา { } ของ main() เป็นชุดคำสั่งที่เป็นการทำงานหลักของโปรแกรม

4.3 ชุดคำสั่ง (ส่วนที่ 3)

ส่วนของชุดคำสั่งเป็นส่วนที่ผู้พัฒนาโปรแกรมจะใช้ในการเขียนโปรแกรมต่าง ๆ ลงไป ทั้งนี้ในทุก ๆ คำสั่งของภาษาซีจะต้องจบคำสั่งด้วยเครื่องหมายเซมิโคลอน (Semicolon)

4.4 การอธิบายโปรแกรม (ส่วนที่ 4)

ส่วนของการอธิบายโปรแกรมหรือคอมเมนท์ (comment) เป็นส่วนที่ใช้ในการอธิบายการทำงานต่าง ๆ ของผู้พัฒนาที่ต้องการขยายความ เพื่อสร้างความเข้าใจของตนหรือเพื่อให้ผู้ที่มาศึกษาต่อสามารถเข้าใจโปรแกรมในส่วนที่ได้อธิบายหรือคอมเมนท์เอาไว้ ทั้งนี้การอธิบายโปรแกรมหรือคอมเมนท์ (comment) จะไม่ผลใด ๆ ต่อการทำงานของโปรแกรม ซึ่งการอธิบายโปรแกรมหรือคอมเมนท์ (comment) ในภาษาซีสามารถทำได้ 2 ลักษณะ ดังนี้

4.4.1 กรณีอธิบายโปรแกรมเพียง 1 บรรทัด

กรณีผู้พัฒนาโปรแกรมต้องการอธิบายโปรแกรมเพียง 1 บรรทัด สามารถทำได้โดยการใส่เครื่องหมาย // (2 อัน) หลังจากเครื่องหมายดังกล่าวก็สามารถใส่คำอธิบายโปรแกรมได้ทันที ดังตัวอย่าง

```
#include "stdio.h"
main()
        {
                int i , a , b;                  // ประกาศตัวแปรแบบ int
                printf (”witoon kongphon”);   // คำสั่งแสดงผล
        }
```

4.4.2 กรณีอธิบายโปรแกรมหลายบรรทัด

กรณีผู้พัฒนาโปรแกรมต้องการอธิบายโปรแกรมหลายบรรทัด สามารถทำได้โดยการใส่เครื่อง /* */ ภายในเครื่องหมายดังกล่าวก็สามารถใส่คำอธิบายโปรแกรมได้ทันที ดังตัวอย่าง

```
#include "stdio.h"
/*  โปรแกรมต่อไปนี้คือโปรแกรมที่ใช้ในการแสดงผลตัวเลข
    โดยมีการประกาศตัวแปรเพื่อรับค่าแล้วนำค่านั้นแสดงผล  */
main()
        {
                int num ;
                scanf(“%d”, &num);
                printf (“Number :”, num”);
        }
```

ทั้งนี้ในการอธิบายโปรแกรมหรือคอมเมนท์ (comment) ผู้พัฒนาโปรแกรมสามารถนำทั้ง 2 วิธีมาใช้ในโปรแกรมเดียวได้ ดังตัวอย่าง

```
#include "stdio.h"
/*  โปรแกรมต่อไปนี้คือโปรแกรมที่ใช้ในการแสดงผลตัวเลข
    โดยมีการประกาศตัวแปรเพื่อรับค่าแล้วนำค่านั้นแสดงผล  */
main()
        {
                int num ;  // ประกาศตัวแปรแบบ int
                scanf(“%d”, &num);    // รับค่าจากผู้ใช้งาน
                printf (“Number :”, num”); // แสดงผลข้อมูล
        }
```

4.5 คำสงวน

คำสงวน หมายถึง คำที่สงวนไว้สำหรับใช้ในคำสั่งภาษาซีเป็นการเฉพาะ ทั้งนี้คำสงวนเหล่านี้ผู้พัฒนาโปรแกรมไม่สามารถนำใช้ในการประกาศตัวแปรต่าง ๆ ได้ โดยในภาษาซีจะมีคำสงวนไว้ดังนี้

คำสงวนในภาษาซี

auto	default	float	register
struct	volatile	break	do
for	return	switch	while
case	double	goto	short
typedef	char	else	if
signed	union	const	enum
int	sizeof	unsigned	continue
extern	long	static	void

4.6 ตัวดำเนินการ

ตัวดำเนินการบางครั้งเรียกว่า “เครื่องหมาย” จะเข้าใจง่ายกว่า ในภาษา C สามารถแบ่งตัวดำเนินการได้หลายประเภท ดังนี้

4.6.1 ตัวดำเนินการคณิตศาสตร์

ตารางที่ 4.2 แสดงตัวดำเนินการคณิตศาสตร์

สัญลักษณ์ (symbol)	ตัวดำเนินการ (operators)	ตัวอย่าง
+	บวก (addition)	a+b
-	ลบ (subtraction)	a-b
*	คูณ (multiplication)	a*b
/	หาร (division)	a/b
%	หารเอาเศษ (remainder)	a%b

โดยผลลัพธ์ที่ได้จากการคำนวณทางคณิตศาสตร์จะอยู่ในรูปของตัวเลข

4.6.2 ตัวดำเนินการความสัมพันธ์

ตารางที่ 4.3 แสดงตัวดำเนินการความสัมพันธ์

สัญลักษณ์ (symbol)	ตัวดำเนินการ (operators)	ตัวอย่าง
<	น้อยกว่า (less than)	A<b
>	มากกว่า (greater than)	a>b
<=	น้อยกว่าหรือเท่ากับ (less than or equal)	A<=b
>=	มากกว่าหรือเท่ากับ (greater than or equal)	a>=b
==	เท่ากับ (equal)	A==b
!=	ไม่เท่ากับ (not equal)	a!=b

โดยผลลัพธ์ที่ได้จากตัวดำเนินการความสัมพันธ์ จะได้ค่าจริง (1) หรือค่าเท็จ (0) เท่านั้น

4.6.3 ตัวดำเนินการเชิงตรรกะ

ตารางที่ 4.4 แสดงตัวดำเนินการเชิงตรรกะ

สัญลักษณ์ (symbol)	ตัวดำเนินการ (operators)	ตัวอย่าง
&&	และ (AND)	A<b && c>d
||	หรือ (OR)	a<b || c>d
!	ไม่ (NOT)	!(a<b)

โดยผลลัพธ์ที่ได้จากตัวดำเนินการเชิงตรรกะ จะได้ค่าจริง (1) หรือค่าเท็จ (0) เท่านั้น

4.6.4 ตัวดำเนินการเพิ่มค่าและลดค่า

ตารางที่ 4.5 แสดงตัวดำเนินการเพิ่มค่าและลดค่า

สัญลักษณ์ (symbol)	ตัวดำเนินการ (operators)	ตัวอย่าง
++	เพิ่มค่า (increment)	a++ หรือ ++a
--	ลดค่า (decrement)	a-- หรือ --a

โดยผลลัพธ์ที่ได้จากการเพิ่มค่าและลดค่าจะอยู่ในรูปของค่าตัวเลข

4.6.5 ตัวดำเนินการบิตไวส์

ตารางที่ 4.6 แสดงตัวดำเนินการบิตไวส์

สัญลักษณ์ (symbol)	ตัวดำเนินการ (operators)	ตัวอย่าง
&	AND	a&b
|	inclusive OR	a|b
^	exclusive OR	a^b
~	Complement	~a
>>	right shift	a>>2
<<	left shift	a<<3

4.6.6 ตัวดำเนินการกำหนดค่า

ตารางที่ 4.7 แสดงตัวดำเนินการกำหนดค่า

สัญลักษณ์ (symbol)	ตัวดำเนินการ (operators)	ตัวอย่าง
=	Assignment	a=b
+=	Addition	a+=b หมายถึง (a=a+b)
-=	Subtraction	a-=b หมายถึง (a=a-b)
*=	Multiplication	a*=b หมายถึง (a=a*b)
/=	Division	a/=b หมายถึง (a=a/b)
%=	Remainder	a%=b หมายถึง (a=a%b)
&=	bitwise AND	a&=b หมายถึง (a=a&b)
|=	bitwise Inclusive OR	a|=b หมายถึง (a=a|b)
^=	bitwise exclusive OR	a^=b หมายถึง (a=a^b)
<<=	right shift	a<<2 หมายถึง (a=a<<2)
>>=	left shift	a>>3 หมายถึง (a=a>>3)

4.6.8 ลำดับการทำงานของตัวดำเนินการ

ตารางที่ 4.8 แสดงลำดับการทำงานของตัวดำเนินการในภาษาซี

ลำดับที่	ตัวดำเนินการ
1	( ) [ ] . ->
2	- ~ ! * &
3	++ - -
4	* / %
5	+ -
6	<< >>
7	< > < = >=
8	= = !=
9	& (bitwise AND)
10	^ (bitwise exclu OR)
11	| (bitwise inclu OR)
12	&&
13	||
14	?:
15	= += -= /= %=
16	<<= >>=

ตัวดำเนินการที่มีลำดับการทำงานอันดับที่ 1 จะทำงานก่อนอันดับที่ 2 โดยการทำงานจะมีการทำงานไปเรื่อย ๆ จนกระทั่งหมดตัวดำเนินการต่อการประมวลนั้น ๆ ส่วนลักษณะการทำงานของตัวดำเนินการที่อยู่ในอันดับเดียวกันจะเป็นการทำงานจากซ้ายไปขวาเป็นหลัก

ตัวอย่างที่แสดงขั้นตอนการทำงานของตัวดำเนินการ

ตัวอย่างที่ 4.1

A = 8 + 5 * 3

A = 4 + 15

A = 19

ตัวอย่างที่ 4.2

B = 10 / 2 + 5 – 3

B = 5 + 5 – 3

B = 10 - 3

B = 7

ตัวอย่างที่ 4.3

C = 9 * 3 - 20 / 5 + 6

C = 27 - 20 / 5 + 6

C = 27 - 4 + 6

C = 23 + 6

C = 29

ตัวอย่างที่ 4.4

D = 16 * (3 – 2) / (2 + 6)

D = 16 * 1/ (2 + 6)

D = 16 * 1/ 8

D = 16 / 8

D = 2

สรุปท้ายบท

ภาษาคอมทุกภาคจะต้องมีโครงสร้างของภาษา เนื่องจากโครงสร้างของภาษาคอมพิวเตอร์จะเป็นรูปแบบโครงร่างในการเขียนโปรแกรมของภาษานั้น ๆ เช่นเดียวกันกับภาษาซีที่จะต้องมีโครงสร้างพื้นฐานในการพัฒนาโปรแกรมหรือการเขียนโปรแกรม ข้อดีอีกประการหนึ่งของภาษาซี คือ ภาษาซีเป็นภาษาคอมพิวเตอร์ภาษาหนึ่งที่มีโครงสร้างภาษาที่เข้าใจง่าย และง่ายต่อการเขียนโปรแกรม โดยโครงสร้างของภาษาซีมีโครงสร้างที่สำคัญดังนี้

- หัวฟังก์ชัน เป็นส่วนหัวของฟังก์ชันหรืออาจจะเรียกว่า พรีโปรเซสเซอร์ไดเรกทฟี (Preprocessor directives) เป็นส่วนที่ทุกโปรแกรมต้องมีใช้สำหรับเรียกไฟล์ที่โปรแกรมต้องการและกำหนดค่าต่าง ๆ ซึ่งจะต้องเริ่มต้นด้วยเครื่องหมายไดเรกทีฟ (#) และตามด้วยชื่อที่ต้องการกำหนดค่า ตัวอย่างไดเรกทีฟที่ใช้บ่อย คือ คำสั่ง #include เป็นการบอกให้ตัวแปลคำสั่ง (compiler) อ่านไฟล์อื่นเข้ามาร่วมในการคอมไพล์

- ฟังก์ชั่นหลัก เป็นส่วนของฟังก์ชันหลักหรือในหรือ main program เป็นส่วนที่ภาษาซีทุกโปรแกรมจะต้องมี เพราะจะเป็นส่วนที่จะประกาศให้ตัวแปลคำสั่ง (compiler) รู้ว่าคำสั่งข้างในชุดปีกกา { } ของ main() เป็นชุดคำสั่งที่เป็นการทำงานหลักของโปรแกรม

- ชุดคำสั่ง เป็นส่วนของชุดคำสั่งเป็นส่วนที่ผู้พัฒนาโปรแกรมจะใช้ในการเขียนโปรแกรมต่าง ๆ ลงไป ทั้งนี้ในทุก ๆ คำสั่งของภาษาซีจะต้องจบคำสั่งด้วยเครื่องหมายเซมิโคลอน (Semicolon)

- การอธิบายโปรแกรม เป็นส่วนของการอธิบายโปรแกรมหรือคอมเมนท์ (comment) เป็นส่วนที่ใช้ในการอธิบายการทำงานต่าง ๆ ของผู้พัฒนาที่ต้องการขยายความ เพื่อสร้างความเข้าใจของตนหรือเพื่อให้ผู้ที่มาศึกษาต่อสามารถเข้าใจโปรแกรมในส่วนที่ได้อธิบายหรือคอมเมนท์เอาไว้ ทั้งนี้การอธิบายโปรแกรมหรือคอมเมนท์ (comment) จะไม่ผลใด ๆ ต่อการทำงานของโปรแกรม ซึ่งการอธิบายโปรแกรมหรือคอมเมนท์

- คำสงวน เป็นคำที่สงวนไว้สำหรับใช้ในคำสั่งภาษาซีเป็นการเฉพาะ ทั้งนี้คำสงวนเหล่านี้ผู้พัฒนาโปรแกรมไม่สามารถนำใช้ในการประกาศตัวแปรต่าง ๆ ได้

- ตัวดำเนินการ บางครั้งเรียกว่า “เครื่องหมาย” เป็นเครื่องหมายที่จะทำให้การเขียนโปรแกรมเข้าใจง่ายและสะดวกในการพัฒนาโปรแกรม ในภาษา C สามารถแบ่งตัวดำเนินการได้หลายประเภท ดังนี้

- ตัวดำเนินการคณิตศาสตร์
- ตัวดำเนินการความสัมพันธ์
- ตัวดำเนินการเชิงตรรกะ
- ตัวดำเนินการเพิ่มค่าและลดค่า
- ตัวดำเนินการบิตไวส์
- ตัวดำเนินการกำหนดค่า
- ลำดับการทำงานของตัวดำเนินการ

คำถามทบทวน

1. โครงสร้างของภาษาซีมีโครงสร้างที่สำคัญใดบ้างจงอธิบาย
2. การอธิบายโปรแกรมหรือคอมเมนท์ (comment) ของภาษาซีมีการอธิบายโปรแกรมในรูปแบบใดบ้างจงอธิบาย
3. จงอธิบายตัวดำเนินการแต่ละประเภทว่ามีคุณสมบัติที่สำคัญอย่างไรต่อการเขียนโปรแกรม
4. จงหาคำตอบของโจทย์ตัวอย่างต่อไปนี้
    4.1 A = 50 – 3 + 2 – 1
    4.2 B = 5 * (2+3) – 1 * (1-1)
    4.3 C = (5+ (2+3)) – (1+2) * 2
    4.4 D = (3-3 + ((2-2) + (1-1))) – 2 * 3
    4.5 F = ((3.5 + 3.1 – 9) + (5-2.2) + (8-1)) * 0

เอกสารอ้างอิง

คะชา ชาญศิลป์. (2548). **ภาษาซีสำหรับผู้เริ่มต้น.** กรุงเทพมหานคร: วิรัตน์เอ็ดดูเคชั่น.

ประภาพร ช่างไม้. (2545). **คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.** กรุงเทพมหานคร: อินโฟเพรส.

ธีรวัฒน์ ประกอบผล. (2553). **คู่มือการเขียนโปรแกรมภาษา C.** กรุงเทพมหานคร: ซิมพลิฟลาย.

สุระสิทธิ์ ทรงม้า และภูริพจน์ แก้วย่อง. (2552). **การโปรแกรมคอมพิวเตอร์.**: มหาวิทยาลัยราชภัฏสวนดุสิต.

สานนท์ เจริญฉาย. (2543). **การเขียนโปรแกรมและอัลกอริทึม.** กรุงเทพมหานคร: มหาจุฬาลงกรณ์ราชวิทยาลัย.

อรพิน ประวัติบริสุทธิ์. (2554). **คู่มือการเรียนภาษา C ฉบับปรับปรุงใหม่** (พิมพ์ครั้งที่ 10)**.** กรุงเทพมหานคร: โปรวิชั่น.

http://www.mwit.ac.th/~cs/download/tech30101/TECH30101_ch6.html

http://www.eng.su.ac.th/ee/618240/variable.html

http://www.e-learning.snru.ac.th/els/program1/lesson2/page2_5.html

บทที่ 5

ชนิดข้อมูลและการประกาศตัวแปรในภาษาซี

แผนการสอนประจำบท

หัวข้อเนื้อหา

5.1 ชนิดข้อมูล
5.2 ตัวแปร
5.3 กฎของการตั้งชื่อตัวแปรในภาษาซี
5.4 รูปแบบการประกาศตัวแปรในภาษาซี
5.5 การแปลงชนิดข้อมูลของภาษาซี

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถอธิบายและเข้าใจชนิดข้อมูลและสามารถเลือกไปใช้งานได้อย่างถูกต้อง
2. ผู้เรียนสามารถอธิบายและให้ความหมายของตัวแปรได้อย่างถูกต้อง
3. ผู้เรียนสามารถกำหนดรูปแบบของตัวแปร และเลือกชนิดของตัวแปรได้อย่างถูกต้องและเหมาะสม
4. ผู้เรียนสามารถอธิบายและเข้าใจผลลัพธ์ของการแปลงชนิดข้อมูลของภาษาซีได้อย่างถูกต้อง

วิธีการสอน

1. สอนแบบบรรยายโดยใช้เอกสารประกอบการสอน
2. สอนแบบบรรยายโดยใช้ Slide Power Point
3. สอนโดยให้ผู้เรียนเป็นส่วนกลาง โดยใช้วิธีถามตอบกับผู้เรียนระหว่างบรรยาย
4. ให้ผู้เรียนสรุปและอธิบายหัวข้อที่ผู้เรียนที่ได้รับมอบหมาย

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. Slide Power Point
3. ตัวอย่างแบบทดสอบ

การวัดผลและประเมินผล

1. การทำแบบฝึกหัด
2. ความตั้งใจในชั้นเรียนและการร่วมมือในการนำเสนอหรืออภิปราย
3. สังเกตจากการตอบคำถามและการร่วมทำกิจกรรมในชั้นเรียน

บทที่ 5

ชนิดข้อมูลและการประกาศตัวแปรในภาษาซี

5.1 ชนิดข้อมูล

ในการเขียนโปรแกรมคอมพิวเตอร์จะมีข้อมูลต่าง ๆ เข้ามาเกี่ยวข้องหลากหลายส่วน ซึ่งในแต่ละส่วนของโปรแกรมอาจมีชนิดข้อมูลที่แตกต่างกัน เช่น การนับจำนวนการวนลูป ซึ่งในการทำงานจะใช้ข้อมูลชนิดจำนวนเต็มเป็นตัวนับ หรือการแสดงข้อความโดยใช้ข้อมูลชนิดตัวอักษรเป็นส่วนที่ใช้ในการแสดงผล เป็นต้น ดังนั้นจะเห็นว่าข้อมูลต่าง ๆ ถูกแบ่งออกเป็นหลายชนิดตามวัตถุประสงค์ของการใช้งาน นอกจากนี้ข้อมูลแต่ละชนิดจะมีการใช้พื้นที่หน่วยความจำ (memory) ไม่เท่ากัน

โดยชนิดข้อมูลที่สำคัญในการใช้งานในภาษาซี มีดังนี้

5.1.1 ข้อมูลชนิดตัวอักษร (Character) คือ ข้อมูลที่เป็นรหัสแทนตัวอักษรหรือค่าจำนวนเต็มได้แก่ ตัวอักษร ตัวเลข และกลุ่มตัวอักขระพิเศษใช้พื้นที่ในการเก็บข้อมูล 1 ไบต์

5.1.2 ข้อมูลชนิดจำนวนเต็ม (Integer) คือ ข้อมูลที่เป็นเลขจำนวนเต็ม ได้แก่ จำนวนเต็มบวก จำนวนเต็มลบ ศูนย์ ใช้พื้นที่ในการเก็บ 2 ไบต์

5.1.3 ข้อมูลชนิดจำนวนเต็มที่มีขนาด 2 เท่า (Long Integer) คือ ข้อมูลที่มีเลขเป็นจำนวนเต็ม ใช้พื้นที่ 4 ไบต์

5.1.4 ข้อมูลชนิดเลขทศนิยม (Float) คือ ข้อมูลที่เป็นเลขทศนิยม ขนาด 4 ไบต์

5.1.5 ข้อมูลชนิดเลขทศนิยมอย่างละเอียด (Double) คือข้อมูลที่เป็นเลขทศนิยม ใช้พื้นที่ในการเก็บ 8 ไบต์

โดยในภาษาซีได้มีการแบ่งชนิดของข้อมูลออกเป็นหลายชนิดดังตารางที่ 5.1

ตารางที่ 5.1 ชนิดข้อมูลในภาษาซี

ชนิดข้อมูล	ขนาด (ไบต์)	ขอบเขต	ข้อมูลที่เก็บ
char	1	-128 ถึง 127	ข้อมูลชนิดอักขระ
unsigned char	1	0 ถึง 255	ข้อมูลชนิดอักขระ
int	2	-32,768 ถึง 32,767	ข้อมูลชนิดจำนวนเต็ม
unsigned int	2	0 ถึง 65,535	ข้อมูลชนิดจำนวนเต็ม
short	1	-128 ถึง 127	ข้อมูลชนิดจำนวนเต็มแบบสั้น
unsigned short	1	0 ถึง 255	ข้อมูลชนิดจำนวนเต็มแบบสั้น
long	4	-2,147,483,648 ถึง 2,147,483,649	ข้อมูลชนิดจำนวนเต็มแบบยาว
unsigned long	4	0 ถึง 4,294,967,296	ข้อมูลชนิดจำนวนเต็มแบบยาว
float	4	3.4*10e(-38) ถึง 3.4*10e(38)	ข้อมูลชนิดเลขทศนิยม
double	8	3.4*10e(-308) ถึง 3.4*10e(308)	ข้อมูลชนิดเลขทศนิยม
long double	16	3.4*10e(-4032) ถึง 1.1*10e(4032)	ข้อมูลชนิดเลขทศนิยม

5.2 ตัวแปร

ตัวแปร (Variable) คือ การจองพื้นที่ในหน่วยความจำของคอมพิวเตอร์สำหรับเก็บข้อมูลที่ต้องใช้ในการทำงานของโปรแกรม โดยมีการตั้งชื่อเรียกหน่วยความจำในตำแหน่งนั้นด้วย เพื่อความสะดวกในการเรียกใช้ข้อมูล ถ้าจะใช้ข้อมูลใดก็ให้เรียกผ่านชื่อของตัวแปรที่เก็บเอาไว้ โดยปกติการเขียนโปรแกรมที่ดี ควรจะตั้งชื่อตัวแปรให้สอดคล้องกับการทำงานหรือหน้าที่ของตัวแปรนั้น ๆ เพราะเมื่อถึงเวลาต้องมาทำการปรับปรุงแก้ไขโปรแกรมจะสามารถทำได้โดยไม่ยากนัก

5.3 กฎของการตั้งชื่อตัวแปรในภาษาซี

5.2.1 ห้ามตั้งชื่อตรงกับคำสงวน และต้องไม่มีเครื่องหมายคำนวณใด ๆ

5.2.2 ต้องขึ้นต้นด้วยอักษร และห้ามขึ้นต้นด้วยตัวเลข

5.2.3 ห้ามเว้นวรรค หากต้องการเว้นวรรคต้องใช้เครื่องหมาย _ (underscore)

5.2.4 ตัวอักษรใหญ่และตัวอักษรเล็ก ถือเป็นคนละตัวแปร

5.4 รูปแบบการประกาศตัวแปรในภาษาซี

```
ชนิดข้อมูล   ชื่อตัวแปร;
```

5.4.1 ตัวอย่างการประกาศตัวแปรในภาษาซี

1.) **กรณีการประกาศตัวแปรแบบตัวแปรเดียว**

```
int num;
int num_1 = 1;
char name[ ];
char name[10];
float id_2;
float id_2 = 2.80;
```

2.) **กรณีการประกาศตัวแปรแบบหลายตัวแปร**

```
int num , num_1;
int num1 = 1 , num2 = 2;
char name[ ] , address[];
char name1[10] , address2[20];
float id_2 , sum , sum1;
float id1 = 1 , sum2 = 2.0 , sum3 = 3.00;
```

นอกจากนี้ในภาษาซี ชื่อตัวแปร ที่ประกอบไปด้วยอักษรเล็ก หรือใหญ่ ก็มีความแตกต่างกัน หรือที่เรียกว่า Case sensitive ตัวอย่าง เช่น

‘X’ และ ‘x’	เป็นตัวแปรต่างกัน
‘Ex1’ และ ‘EX1’	เป็นตัวแปรต่างกัน
‘bookno1’ และ ‘bookNo1’	เป็นตัวแปรต่างกัน
‘XTREME’ และ ‘xtreme’	เป็นตัวแปรต่างกัน
‘x_1’ และ ‘x1’	เป็นตัวแปรต่างกัน
‘Witoon’ และ ‘witOOn’	เป็นตัวแปรต่างกัน

ดังนั้นในการประกาศตัวแปรในภาษาซีจะต้องคำนึงถึงกฎของการตั้งชื่อตัวแปรในภาษาซีอย่างเคร่งครัด ทั้งนี้ในการเลือกใช้ชนิดข้อมูลจะต้องเลือกใช้ชนิดข้อมูลให้ถูกต้องกับข้อมูลที่ผู้พัฒนาต้องการจัดเก็บ และเมื่อจบการประกาศตัวแปรทุกครั้งจะต้องจบด้วยเครื่องหมายเซมิโคลอน ; (Semicolon) เพราะหากประกาศตัวแปรผิดรูปแบบที่กำหนดจะส่งผลให้โปรแกรมมีปัญหาในการทดสอบโปรแกรมได้

5.4.2 ตัวอย่างการประกาศตัวแปรในภาษาซีพร้อมคำอธิบาย

1.) char d, c[30];

ประกาศตัวแปรชื่อ d และ c มีชนิดเป็น character และตัวแปรสตริง c มีขนาด 30 bytes

2.) int a, b;

ประกาศตัวแปรชื่อ a และ b มีชนิดเป็น int คือใช้เนื้อที่ในหน่วยความจำ 2 bytes และสามารถเก็บตัวเลขจำนวนเต็มที่มีค่าอยู่ในช่วง -32768 ถึง 32767

3.) float k, m, n;

ประกาศตัวแปรชื่อ k, m และ n มีชนิดเป็น float คือใช้เนื้อที่ในหน่วยความจำ 4 bytes สามารถเก็บจำนวนทศนิยมและตัวเลขที่อยู่ในรูป E ยกกำลังได้

4.) double w, x, y, z

ประกาศตัวแปรชื่อ w, x, y และ z มีชนิดเป็น double คือใช้เนื้อที่ในหน่วยความจำ 8 bytes สามารถเก็บจำนวนทศนิยมหรือตัวเลขที่อยู่ในรูป E ยกกำลังที่มีความละเอียดสูงกว่าชนิด float

5.5 การแปลงชนิดข้อมูลของภาษาซี

เมื่อผู้พัฒนาได้เขียนโปรแกรมมักจะพบว่า ในแต่ละโปรแกรมที่ซับซ้อนมักมีตัวดำเนินการกับตัวแปรของชนิดข้อมูลที่แตกต่างกัน ดังนั้นเพื่ออำนวยความสะดวกในการเขียนโปรแกรม ภาษาซีจึงได้กำหนดกฎเกณฑ์การแปลงชนิดข้อมูลไว้อย่างอัตโนมัติ ดังนี้ ถ้าค่าตัวแปรหรือค่าคงที่ต่างชนิดกัน ให้ทำการเปลี่ยนชนิดของข้อมูลที่มีขนาดเล็กให้เป็นชนิดของข้อมูลที่ใหญ่ขึ้น โดยกฎเกณฑ์การแปลงชนิดข้อมูลสามารถศึกษารายละเอียดในตารางที่ 5.1

ตารางที่ 5.2 การแปลงชนิดข้อมูลของภาษาซี

ชนิดของข้อมูล x	ชนิดของข้อมูล y	ผลลัพธ์ที่ได้จากการแปลง
int	long	long
char	int	int
int	float	float
int	double	double
float	double	double
long	double	double
ชนิดข้อมูลต่าง ๆ	long double	long double

จากตารางที่ 5.2 แถวที่ 1 จะสังเกตเห็นว่าตัวแปร x เป็นตัวแปรชนิดข้อมูลแบบ int และตัวแปร y เป็นตัวแปรข้อมูลแบบ long เมื่อนำ x และ y มาทำการบวก ลบ คูณ หรือหารกัน ผลลัพธ์ที่ได้จะเป็นชนิดข้อมูลแบบ long เพราะ long มีพิสัยในการเก็บข้อมูลกว้างกว่า int

ตัวอย่างเช่น

```
int x=20;
float y = 10;
        ถ้า x/2 จะได้ผลลัพธ์เป็น 10 (เป็นชนิดข้อมูลแบบ int)
        ถ้า x/y จะได้ผลลัพธ์เป็น 10.000000 (เป็นชนิดข้อมูลแบบ float)
```

สรุปท้ายบท

ในการพัฒนาหรือเขียนโปรแกรมในภาษาซีส่วนที่สำคัญอีกประการก็คือ ชนิดข้อมูล การประกาศตัวแปร ในการเขียนโปรแกรมคอมพิวเตอร์จะมีข้อมูลต่าง ๆ เข้ามาเกี่ยวข้องหลากหลายส่วน ซึ่งในแต่ละส่วนของโปรแกรมอาจมีชนิดข้อมูลที่แตกต่างกัน เช่น การนับจำนวนการวนลูป ซึ่งในการทำงานจะใช้ข้อมูลชนิดจำนวนเต็มเป็นตัวนับ หรือการแสดงข้อความโดยใช้ข้อมูลชนิดตัวอักษรเป็นส่วนที่ใช้ในการแสดงผล เป็นต้น ดังนั้นจะเห็นว่าข้อมูลต่าง ๆ ถูกแบ่งออกเป็นหลายชนิดตามวัตถุประสงค์ของการใช้งานนอกจากนี้ข้อมูลแต่ละชนิดจะมีการใช้พื้นที่หน่วยความจำ (memory) ไม่เท่ากัน

โดยชนิดข้อมูลที่สำคัญในการใช้งานในภาษาซี มีดังนี้

- ข้อมูลชนิดตัวอักษร (Character) คือ ข้อมูลที่เป็นรหัสแทนตัวอักษรหรือค่าจำนวนเต็ม ได้แก่ ตัวอักษร ตัวเลข และกลุ่มตัวอักขระพิเศษใช้พื้นที่ในการเก็บข้อมูล 1 ไบต์

- ข้อมูลชนิดจำนวนเต็ม (Integer) คือ ข้อมูลที่เป็นเลขจำนวนเต็ม ได้แก่ จำนวนเต็มบวก จำนวนเต็มลบ ศูนย์ ใช้พื้นที่ในการเก็บ 2 ไบต์
- ข้อมูลชนิดจำนวนเต็มที่มีขนาด 2 เท่า (Long Integer) คือ ข้อมูลที่มีเลขเป็นจำนวนเต็ม ใช้พื้นที่ 4 ไบต์
- ข้อมูลชนิดเลขทศนิยม (Float) คือ ข้อมูลที่เป็นเลขทศนิยม ขนาด 4 ไบต์
- ข้อมูลชนิดเลขทศนิยมอย่างละเอียด (Double) คือข้อมูลที่เป็นเลขทศนิยม ใช้พื้นที่ในการเก็บ 8 ไบต์

ตัวแปร (Variable) คือ การจองพื้นที่ในหน่วยความจำของคอมพิวเตอร์สำหรับเก็บข้อมูลที่ต้องใช้ในการทำงานของโปรแกรม โดยมีการตั้งชื่อเรียกหน่วยความจำในตำแหน่งนั้นด้วย เพื่อความสะดวกในการเรียกใช้ข้อมูล ถ้าจะใช้ข้อมูลใดก็ให้เรียกผ่านชื่อของตัวแปรที่เก็บเอาไว้ โดยปกติการเขียนโปรแกรมที่ดี ควรจะตั้งชื่อตัวแปรให้สอดคล้องกับการทำงานหรือหน้าที่ของตัวแปรนั้น ๆ เพราะเมื่อถึงเวลาต้องมาทำการปรับปรุงแก้ไขโปรแกรมจะสามารถทำได้โดยไม่ยากนัก โดยกฎของการตั้งชื่อตัวแปรในภาษาซีมีกฎพื้นฐานที่ต้องเข้าใจดังนี้

- ห้ามตั้งชื่อตรงกับคำสงวน และต้องไม่มีเครื่องหมายคำนวณใด ๆ
- ต้องขึ้นต้นด้วยอักษร และห้ามขึ้นต้นด้วยตัวเลข
- ห้ามเว้นวรรค หากต้องการเว้นวรรคต้องใช้เครื่องหมาย _ (underscore)
- ตัวอักษรใหญ่และตัวอักษรเล็ก ถือเป็นคนละตัวแปร
- ตัวอย่างการประกาศตัวแปรในภาษาซีพร้อมคำอธิบาย

ตัวอย่างการประกาศตัวแปรในภาษาซี

```
char x, y[10];
```

เป็นการประกาศตัวแปรชื่อ x และ y มีชนิดเป็น character และตัวแปรสตริงมีขนาด 10 bytes

```
int A1, A_1;
```

เป็นการประกาศตัวแปรชื่อ A1 และ A_1 มีชนิดเป็น int คือใช้เนื้อที่ในหน่วยความจำ 2 bytes และสามารถเก็บตัวเลขจำนวนเต็มที่มีค่าอยู่ในช่วง -32768 ถึง 32767

```
float number, monney;
```

เป็นการประกาศตัวแปรชื่อ number และ monney มีชนิดเป็น float คือใช้เนื้อที่ในหน่วยความจำ 4 bytes สามารถเก็บจำนวนทศนิยมและตัวเลขที่อยู่ในรูป E ยกกำลังได้

```
double  dusit_trang, witoon;
```

เป็นการประกาศตัวแปรชื่อ dusit_trang และ witoon มีชนิดเป็น double คือใช้เนื้อที่ในหน่วยความจำ 8 bytes สามารถเก็บจำนวนทศนิยมหรือตัวเลขที่อยู่ในรูป E ยกกำลังที่มีความละเอียดสูงกว่าชนิด float

รูปแบบการประกาศตัวแปรในภาษาซี

```
ชนิดข้อมูล  ชื่อตัวแปร;
```

คำถามทบทวน

1. จงอธิบายข้อแตกต่างระหว่างชนิดข้อมูล int กับชนิดข้อมูลแบบ long มีความแตกต่างกันอย่างไร

2. จงอธิบายข้อแตกต่างระหว่างชนิดข้อมูล float กับชนิดข้อมูลแบบ double มีความแตกต่างกันอย่างไร

3. ตัวแปรคืออะไรมีความสำคัญอย่างไรต่อการเขียนโปรแกรม

4. ในการเขียนโปรแกรมทุกโปรแกรมจำเป็นที่ต้องมีการประกาศตัวแปรหรือไม่จงอธิบาย

5. จงประกาศตัวแปรเพื่อเก็บค่าตัวเลข 3 หลัก

6. จงประกาศตัวแปรเพื่อเก็บค่าตัวเลข 7 หลัก

7. จงประกาศตัวแปรเพื่อเก็บค่าตัวเลขแบบทศนิยม โดยตัวแปรจะต้องมีตัวเลขประกอบด้วย

8. จงประกาศตัวแปรเพื่อเก็บตัวอักษร โดยตัวแปรจะต้องมีตัวเลขประกอบด้วย

9. จงประกาศตัวแปร 5 ตัวเพื่อเก็บค่าตัวเลข

10. จงประกาศตัวแปร 4 ตัวเพื่อเก็บค่าตัวเลขแบบทศนิยม 7 ตำแหน่ง

เอกสารอ้างอิง

คะชา ชาญศิลป์. (2548). **ภาษาซีสำหรับผู้เริ่มต้น.** กรุงเทพมหานคร: วิรัตน์เอ็ดดูเคชั่น.

ประภาพร ช่างไม้. (2545). **คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.** กรุงเทพมหานคร: อินโฟเพรส.

ธีรวัฒน์ ประกอบผล. (2553). **คู่มือการเขียนโปรแกรมภาษา C.** กรุงเทพมหานคร: ซิมพลิฟลาย.

สุระสิทธิ์ ทรงม้า และภูริพจน์ แก้วย่อง. (2552). **การโปรแกรมคอมพิวเตอร์.**: มหาวิทยาลัยราชภัฏสวนดุสิต.

สานนท์ เจริญฉาย. (2543). **การเขียนโปรแกรมและอัลกอริทึม.** กรุงเทพมหานคร: มหาจุฬาลงกรณ์ราชวิทยาลัย.

อรพิน ประวัติบริสุทธิ์. (2554). **คู่มือการเรียนภาษา C ฉบับปรับปรุงใหม่** (พิมพ์ครั้งที่ 10)**.** กรุงเทพมหานคร: โปรวิชั่น.

https://www.sites.google.com/site/khasangphasac/tawpaer-phasac.html

http://www.e-learning.snru.ac.th/els/program1/lesson2/page2_5.html

บทที่ 6

ขั้นตอนการพัฒนาโปรแกรมและคำสั่งการทำงานของภาษาซี

แผนการสอนประจำบท

หัวข้อเนื้อหา

6.1 ขั้นตอนการพัฒนาโปรแกรมของภาษาซี

6.2 คำสั่งการทำงานของภาษาซี

- 6.2.1 คำสั่ง printf
- 6.2.2 คำสั่ง putchar
- 6.2.3 คำสั่ง puts
- 6.2.4 คำสั่ง scanf
- 6.2.5 คำสั่ง getchar
- 6.2.6 คำสั่ง getch
- 6.2.7 คำสั่ง clrscr
- 6.2.8 คำสั่ง goto

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถอธิบายและเข้าใจขั้นตอนการพัฒนาโปรแกรมของภาษาซีได้อย่างถูกต้อง
2. ผู้เรียนสามารถเข้าใจและอธิบายการทำงานของภาษาซีได้อย่างถูกต้อง
3. ผู้เรียนสามารถเลือกนำคำสั่งการทำงานของภาษาซีไปพัฒนาโปรแกรมได้อย่างถูกต้อง

วิธีการสอน

1. สอนแบบบรรยายโดยใช้เอกสารประกอบการสอน
2. สอนแบบบรรยายโดยใช้ Slide Power Point
3. สอนโดยใช้โปรแกรม Turbo C++ 4.5 เพื่อลงมือปฏิบัติในการเขียนโปรแกรมภาษาซี
4. สอนโดยให้ผู้เรียนเป็นส่วนกลาง โดยใช้วิธีถามตอบกับผู้เรียนระหว่างบรรยาย
5. ให้ผู้เรียนสรุปและอธิบายหัวข้อที่ผู้เรียนที่ได้รับมอบหมาย

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. Slide Power Point
3. โปรแกรม Turbo C++ 4.5
4. ตัวอย่างแบบทดสอบ

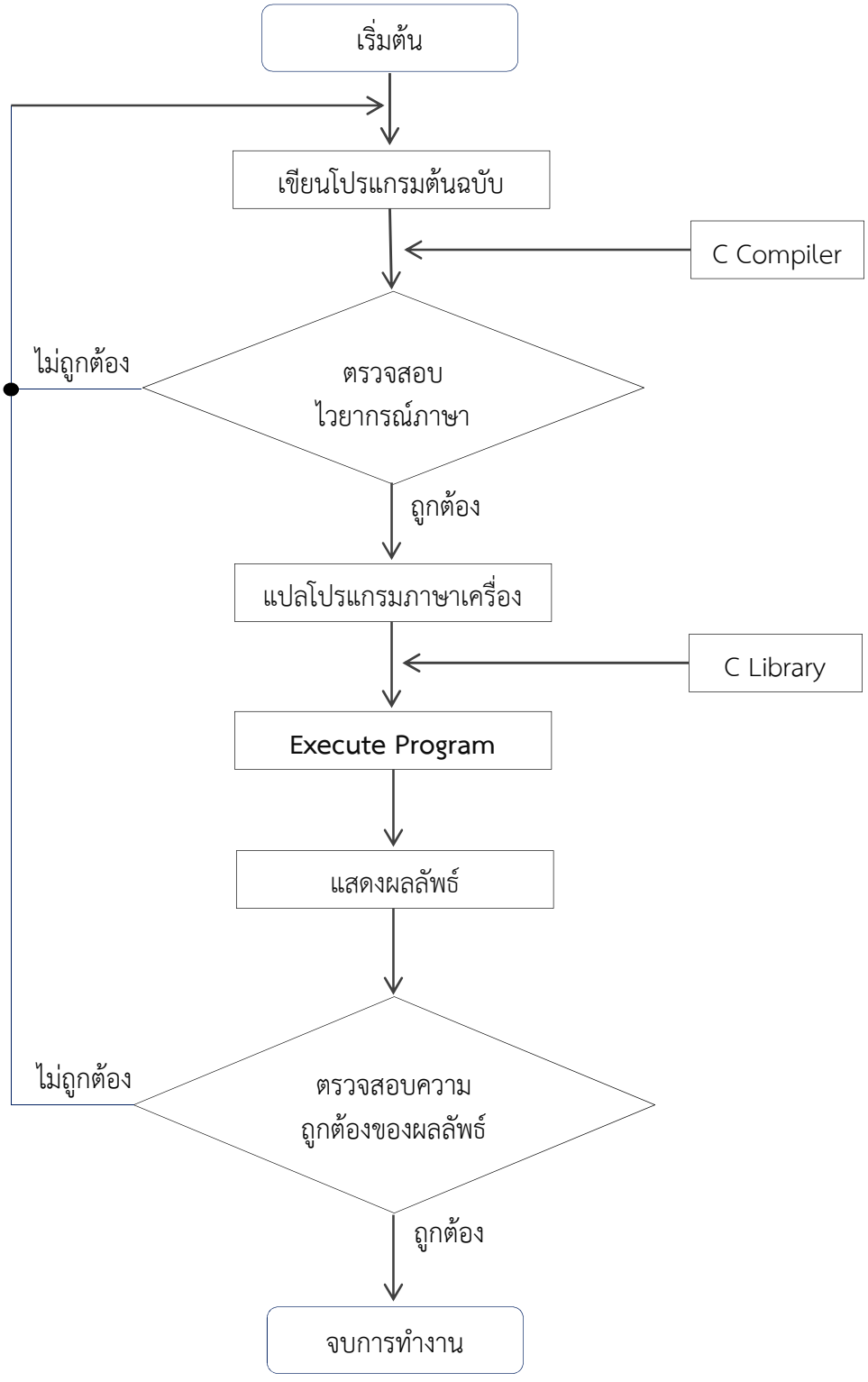
การวัดผลและประเมินผล

1. การทำโจทย์ทดลอง และการทำแบบฝึกหัด
2. ความตั้งใจในชั้นเรียน การตอบคำถามและการร่วมทำกิจกรรมในชั้นเรียน

บทที่ 6

ขั้นตอนการพัฒนาโปรแกรมและคำสั่งการทำงานของภาษาซี

6.1 ขั้นตอนการพัฒนาโปรแกรมของภาษาซี



ภาพที่ 6.1 ผังงานแสดงขั้นตอนการพัฒนาโปรแกรมภาษาซี

จากภาพที่ 6.1 เป็นผังงานแสดงขั้นตอนการพัฒนาโปรแกรมภาษาซี ซึ่งสามารถอธิบายในการทำงานในแต่ละมีขั้นตอน ดังนี้

6.1.1 เขียนโปรแกรมต้นฉบับด้วยภาษาซี

ใช้การเขียนโปรแกรมภาษาซีในการศึกษาครั้งนี้จะใช้โปรแกรม Turbo C ในการเขียนและพัฒนาโปรแกรม ซึ่งการเขียนโปรแกรมผู้พัฒนาจะต้องเขียนโปรแกรมต้นฉบับด้วยภาษาซีขึ้นมาก่อนหนึ่งโปรแกรม โดยการเขียนจะต้องคำนึงถึงหลักไวยากรณ์ของภาษาเป็นหลัก จากนั้นทำการบันทึกโปรแกรมพร้อมกับตั้งชื่อแฟ้มไว้ โดยแฟ้มที่ได้จะมีนามสกุล *.c หรือ เช่น simple1.c เป็นต้น

6.1.2 แปลโปรแกรมภาษาซีไปเป็นโปรแกรมภาษาเครื่อง

ในการแปลโปรแกรมภาษาซีไปเป็นโปรแกรมภาษาเครื่อง เป็นการใช้คำสั่ง compile เพื่อแปลโปรแกรมภาษาซีไปเป็นโปรแกรมภาษาเครื่อง แฟ้มที่ได้จะมีนามสกุล *.obj ซึ่งในขั้นตอนนี้โปรแกรมต้นฉบับอาจเกิดความผิดพลาดทางไวยากรณ์ภาษาขึ้นได้ จึงต้องย้อนกลับไปแก้ไขโปรแกรมต้นฉบับตามขั้นตอนที่หนึ่งให้ถูกต้องเสียก่อน

6.1.3 เชื่อมโยงโปรแกรมภาษาเครื่องเข้ากับ library function ของภาษาซี

การเชื่อมโยงโปรแกรมภาษาเครื่องเข้ากับ library function ของภาษาซี จะได้เป็นการ execute program โดยใช้คำสั่ง link แฟ้มที่ได้จะมีนามสกุล *.exe

6.1.4 การ execute program เพื่อแสดงผลลัพธ์ออกมา

ในการ execute program เพื่อแสดงผลลัพธ์ออกมา โดยใช้คำสั่ง run ในขั้นตอนนี้ผู้พัฒนาโปรแกรมควรตรวจสอบผลลัพธ์ที่ได้จากโปรแกรม ว่าตรงกับความต้องการของเราหรือไม่ ถ้าผลลัพธ์ที่ได้ไม่ตรงกับความต้องการให้กลับไปแก้ไขโปรแกรมต้นฉบับในขั้นตอนที่หนึ่งใหม่อีกครั้ง เสร็จแล้วทำขั้นตอนที่สองถึงขั้นตอนที่สี่ซ้ำอีกครั้ง และต้องทำซ้ำเช่นนี้จนกว่าจะได้ผลลัพธ์ที่ถูกต้อง

6.2 คำสั่งการทำงานของภาษาซี

คำสั่งการทำงานของภาษาซี คือ ชุดคำสั่งที่ผู้พัฒนาโปรแกรมต้องการให้โปรแกรมคอมพิวเตอร์ทำงานตามผลลัพธ์ที่ต้องการให้แสดง โดยคำสั่งการงานในภาษาซีมีมากมายหลายคำสั่ง ซึ่งจะมีการศึกษาคำสั่งจากคำสั่งพื้นฐานจนถึงคำสั่งระดับสูงขึ้นไป

6.2.1 คำสั่ง printf( )

คำสั่ง printf ถือว่าเป็นคำสั่งพื้นฐานที่สุดในการแสดงผลข้อมูลทุกชนิดออกทางหน้าจอ ไม่ว่าจะเป็นจำนวนเต็ม (int) , ทศนิยม (float) , ข้อความ (string) หรืออักขระ นอกจากนี้คำสั่งยังมีความยืดหยุ่นสูง โดยผู้พัฒนาโปรแกรมสามารถกำหนดหรือจัดรูปแบบการแสดงผลให้มีระเบียบหรือเหมาะสมตามความต้องการ ดังตารางแสดงรหัสควบคุมดังตารางที่ 6.1 และนอกจากนี้ผู้พัฒนาโปรแกรมยังสามารถจัดรูปแบบการแสดงผลให้ดูเป็นระเบียบมากขึ้น เช่น การขึ้นบรรทัดใหม่หลังแสดงข้อความ หรือเว้นระยะแท็บระหว่างข้อความ โดยใช้อักขระควบคุมการแสดงผลร่วมกับคำสั่ง printf ในภาษาซีมีอักขระควบคุมการแสดงผลหลายรูปแบบด้วยกัน ดังแสดงไว้ในตารางที่ 6.2

ตารางที่ 6.1 รหัสควบคุมรูปแบบการแสดงผลค่าของตัวแปรออกทางหน้าจอ

รหัสควบคุมรูปแบบ	การนำไปใช้งาน
%d	สำหรับแสดงผลค่าของตัวแปรชนิดจำนวนเต็ม (int, short, unsigned short, long, unsigned long)
%u	สำหรับแสดงผลตัวเลขจำนวนเต็มบวก (unsigned short, unsigned long)
%o	สำหรับแสดงผลออกมาในรูปแบบของเลขฐานแปด
%x	สำหรับแสดงผลออกมาในรูปแบบของเลขฐานสิบหก
%f	สำหรับแสดงผลค่าของตัวแปรชนิดจำนวนทศนิยม (float, double, long double)
%e	สำหรับแสดงผลตัวเลขทศนิยมออกมาในรูปแบบของ (E หรือ e) ยกกำลัง (float, double, long double)
%c	สำหรับแสดงผลอักขระ 1 ตัว (char)
%s	สำหรับแสดงผลข้อความ (string หรืออักขระมากกว่า 1 ตัว)
%p	สำหรับแสดงผลตัวชี้ตำแหน่ง (pointer)

ตารางที่ 6.2 การจัดระแบบระเบียบข้อความด้วยอักขระควบคุมการแสดงผล

อักขระควบคุมการแสดงผล	ความหมาย
\n	ขึ้นบรรทัดใหม่
\t	เว้นช่องว่างเป็นระยะ 1 แท็บ (6 ตัวอักษร)
\r	กำหนดให้เคอร์เซอร์ไปอยู่ต้นบรรทัด
\f	เว้นช่องว่างเป็นระยะ 1 หน้าจอ
\b	ลบอักขระสุดท้ายออก 1 ตัว

ในการใช้เรียกใช้งานคำสั่ง printf( ) จะมีรูปแบบการเรียกใช้คำสั่ง printf( ) อยู่ 2 กรณีหลัก โดยแสดงวิธีการใช้งานคำสั่ง printf( ) ทั้งสองกรณีดังนี้

6.2.1.1 กรณีแสดงข้อความอย่างเดียว

```
printf(“ format ”);
```

format คือ ข้อความที่ต้องการให้แสดงออกทางหน้าจอโดยข้อมูลนี้ต้องเขียนไว้ในเครื่องหมาย “ ” ซึ่งการทำงานแสดงผลไว้ดังตัวอย่าง

ตัวอย่างที่ 6.1 แสดงการทำงานของคำสั่ง printf( ) กรณีแสดงข้อความอย่างเดียว

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                printf (“Dusittrang”);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Dusittrang
```

ตัวอย่างที่ 6.2 แสดงการทำงานของคำสั่ง printf( ) กรณีแสดงข้อความอย่างเดียวแต่แสดงหลายข้อความ

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                printf (“Dusittrang”);
                printf (“        Dusittrang”);
                printf (“                Dusittrang”);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Dusittrang          Dusittrang                    Dusittrang
```

ตัวอย่างที่ 6.3 แสดงการทำงานของคำสั่ง printf( ) กรณีแสดงข้อความอย่างเดียว แต่แสดงหลายข้อความ และมีการเว้นให้ขึ้นบรรทัดใหม่เมื่อมีข้อความใหม่ โดยการใช้อักขระควบคุม \n

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )

        {
                printf (“Dusittrang\n”);
                printf (“          Dusittrang\n”);
                printf (“                    Dusittrang\n”);
                printf (“                              Dusittrang”);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Dusittrang
    Dusittrang
        Dusittrang
            Dusittrang
```

6.2.1.2 กรณีแสดงข้อความพร้อมทั้งแสดงข้อมูลในตัวแปร

```
printf(“ format ” , variable);
```

format คือ ข้อมูลที่ต้องการแสดงออกทางหน้าจอโดยข้อมูลนี้ต้องเขียนไว้ในเครื่องหมาย “ ” ข้อมูลที่สามารถแสดงผลได้มีอยู่ 2 ประเภท คือ ข้อความธรรมดา และค่าที่เก็บไว้ในตัวแปร ซึ่งถ้าเป็นค่าที่เก็บไว้ในตัวแปรต้องใส่รหัสควบคุมรูปแบบให้ตรงกับชนิดของข้อมูลที่เก็บไว้ในตัวแปรนั้นด้วย

variable คือ ตัวแปรหรือนิพจน์ที่ต้องการนำค่าไปแสดงผลให้ตรงกับรหัสควบคุมรูปแบบที่กำหนดไว้

ตัวอย่างที่ 6.4 แสดงการทำงานของคำสั่ง printf( ) กรณีแสดงข้อความพร้อมทั้งแสดงข้อมูลในตัวแปร

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )

        {
                int num = 10;
                printf (“Ans : %d” , num);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Ans : 10
```

ตัวอย่างที่ 6.5 แสดงการทำงานของคำสั่ง printf( ) กรณีแสดงหลายตัวแปร

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int num1 = 10 , num2 = 20 num3 = 30;
                    printf (“Ans : %d,%d,%d ” , num1,num2,num3);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Ans : 102030
```

ตัวอย่าง 6.6 แสดงการทำงานของคำสั่ง printf( ) กรณีแสดงหลายตัวแปรแต่ต้องการเว้นช่องว่างโดยการใช้อักขระควบคุม \t

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int num1 = 10 , num2 = 20 num3 = 30;
                printf (“Ans : %d\t,%d\t,%d ” , num1,num2,num3);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Ans : 10      20      30
```

ตัวอย่างที่ 6.7 แสดงการทำงานของคำสั่ง printf( ) กรณีแสดงข้อความพร้อมทั้งแสดงข้อมูลในตัวแปรที่เป็นตัวเลขทศนิยม

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                float num = 10.00;
                printf (“Ans : %f” , num);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Ans : 10.000000
```

ตัวอย่างที่ 6.8 แสดงการทำงานของคำสั่ง printf( ) กรณีแสดงข้อความพร้อมทั้งแสดงข้อมูลในตัวแปรที่เป็นตัวเลขทศนิยม แต่กำหนดจำนวนเลขทศนิยม โดยการใช้ %._f เป็นตัวกำหนดจำนวนทศนิยม

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                float num = 35.4589;
                printf (“Ans : %f\n” , num);
                printf (“Ans : %.2f\n” , num);
                printf (“Ans : %.3f\n” , num);
                printf (“Ans : %.4f\n” , num);
                printf (“Ans : %.5f\n” , num);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Ans : 35.458900
Ans : 35.45
Ans : 35.458
Ans : 35.4589
Ans : 35.45890
```

ตัวอย่างที่ 6.9 แสดงการทำงานของคำสั่ง printf( ) กรณีแสดงข้อความพร้อมทั้งแสดงข้อมูลในตัวแปรที่เป็นตัวเลขทศนิยม แต่ตัดตัวเลขทศนิยมตัวสุดท้ายออกด้วยการใช้ โดยการใช้อักขระควบคุม \b

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                float num = 12.4567;
                printf (“Ans : %f\n” , num);
                printf (“Ans : %\b\n” , num);
                printf (“Ans : %f\b\b\n” , num);
                printf (“Ans : %f\b\b\b\n” , num);
                printf (“Ans : %f\b\b\b\b\n” , num);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Ans : 12.345670
Ans : 12.34567
Ans : 12.3456
Ans : 12.345
Ans : 12.34
```

6.2.2 คำสั่ง putchar()

ในการแสดงผลตัวอักษรหรืออักขระ (char) ออกทางหน้าจอ นอกจากใช้คำสั่ง printf( ) พร้อมกับกำหนดรหัสควบคุมรูปแบบ %c แล้ว เราสามารถเรียกใช้คำสั่งสำหรับแสดงตัวอักษรหรืออักขระโดยเฉพาะได้อีกด้วย โดยคำสั่งนั้น คือ คำสั่ง putchar( ) ซึ่งมีรูปแบบการเรียกใช้คำสั่งแสดงไว้ดังต่อไปนี้

```
putchar(ch);
```

ch คือ ตัวอักษรหรืออักขระเขียนอยู่ภายในเครื่องหมาย ‘c' หรือตัวแปรชนิด char

ตัวอย่างที่ 6.10 แสดงการทำงานของคำสั่ง putchar( )

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                putchar(‘w’);
                putchar(‘i’);
                putchar(‘t’);
                putchar(‘o’);
                putchar(‘o’);
                putchar(‘n’);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
witoon
```

6.2.3 คำสั่ง puts()

คำสั่ง puts() คือ คำสั่งในการแสดงผลข้อความหรืออักษรออกทางหน้าจอคอมพิวเตอร์ ดังนั้นข้อแตกต่างระหว่างคำสั่ง putchar( ) กับคำสั่ง puts() คือ putchar( ) จะสามารถแสดงผลแค่ตัวอักษรหรืออักขระ แต่ puts() สามารถแสดงผลเป็นข้อความได้ ซึ่งมีรูปแบบการเรียกใช้คำสั่ง puts() แสดงไว้ดังต่อไปนี้

```
puts( );
```

ตัวอย่างที่ 6.11 แสดงการทำงานของคำสั่ง puts( )

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                puts(“witoon”);
                puts(“kongphon”);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
witoon
kongphon
```

ตัวอย่างที่ 6.12 แสดงการทำงานของคำสั่ง puts( ) โดยมีการแสดงจากตัวแปร

```
// โค้ดโปรแกรม
#include "stdio.h"
main( ) {
        char name1[ ] = “Somsak”;
        char name2[ ] = “Ratthai”;
                puts(name1);
                puts(name2);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Somsak
Ratthai
```

6.2.4 คำสั่ง scanf( )

คำสั่ง scanf( ) เป็นคำสั่งพื้นฐานที่ใช้ในการรับผลข้อมูลจากผู้ใช้งาน และในภาษาซี การรับข้อมูลจากคีย์บอร์ดสามารถทำได้โดยการเรียกใช้ฟังก์ชัน scanf( ) ซึ่งเป็นฟังก์ชันมาตรฐานสำหรับรับข้อมูลจากคีย์บอร์ด โดยสามารถรับข้อมูลได้ทุกประเภท ไม่ว่าจะเป็นจำนวนเต็ม (int) , ทศนิยม (float) ,อักขระ (char) หรือข้อความก็ตาม รูปแบบการเรียกใช้คำสั่ง scanf( ) คล้ายกับการเรียกใช้คำสั่ง printf( ) ดังแสดงรูปแบบของคำสั่ง scanf( ) ไว้ดังต่อไปนี้

```
scanf(“ format ” , &variable);
```

format คือ การกำหนดรหัสควบคุมรูปแบบ เพื่อกำหนดชนิดของข้อมูลที่จะรับเข้ามาจากคีย์บอร์ด โดยรหัสควบคุมรูปแบบจะใช้ชุดเดียวกับรหัสควบคุมรูปแบบของคำสั่ง printf( )

variable คือ ตัวแปรที่จะใช้เก็บค่าข้อมูลที่รับเข้ามาจากคีย์บอร์ด โดยชนิดของตัวแปรจะต้องตรงกับรหัสควบคุมรูปแบบที่กำหนดไว้ นอกจากนี้ หน้าชื่อของตัวแปรจะต้องนำหน้าด้วยเครื่องหมาย & ยกเว้นตัวแปรสตริงสำหรับเก็บข้อความเท่านั้นที่ไม่ต้องนำหน้าด้วยเครื่องหมาย &

ตัวอย่างที่ 6.13 แสดงการทำงานของคำสั่ง scanf( ) กรณีรับข้อมูลชนิดเลขจำนวนเต็ม (int)

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int num;
                printf("Enter number :");
                scanf("%d" ,&num);
                printf("\nAns = %d", num);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter number : 12345

Ans : 12345
```

ตัวอย่างที่ 6.14 แสดงการทำงานของคำสั่ง scanf( ) กรณีรับข้อมูลชนิดเลขทศนิยม (float)

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                float num;
                printf("Enter number :");
                scanf("%f", &num);
                printf("\nAns = %f", num);
                printf("\nAns = %.2f", num);

        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter number : 987.56

Ans : 987.559998
Ans : 987.56
```

ตัวอย่าง 6.15 แสดงการทำงานของคำสั่ง scanf( ) กรณีรับข้อมูลชนิดตัวอักษร (char)

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                char ch[20];
                printf("Name :");
                scanf("%s" ,&ch);
                printf("\n Wellcome %s",ch);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Name : Witoon
Wellcome : Witoon
```

6.2.5 คำสั่ง getchar( )

ในการรับค่าในภาษาซีนอกจากใช้คำสั่ง scanff( ) แล้ว ผู้พัฒนาโปรแกรมสามารถเรียกใช้คำสั่ง getchar( ) เพื่อการรับค่าได้อีกด้วย โดยรูปแบบการเรียกใช้คำสั่ง getchar( ) แสดงไว้ดังต่อไปนี้

```
ch = getchar( );
```

ch คือ ตัวแปรชนิด char เพื่อใช้เก็บค่าของอักขระหรือตัวอักษรที่รับเข้ามา

ตัวอย่างที่ 6.16 แสดงการทำงานของคำสั่ง getchar( )

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                char ch;
                printf("name :");
                ch = getchar();
                printf("\n Hello :%s",ch);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
name : witoon
Hello : witoon
```

6.2.6 คำสั่ง getch( )

ในการรับค่าอักษระในภาษาซี นอกจากใช้คำสั่ง scanff( ) และคำสั่ง getchar( ) แล้ว ผู้พัฒนาโปรแกรมสามารถเรียกใช้คำสั่ง getch( ) เพื่อใช้ในการรับข้อมูลได้อีกคำสั่งหนึ่ง ข้อสังเกต ถึงแม้ว่าทั้งคำสั่ง getchar( ) และ getch( ) จะใช้สำหรับรับข้อมูลชนิดอักขระเหมือนกัน แต่ทั้ง 2 คำสั่งนี้มีความแตกต่างกันอยู่ตรงที่ คำสั่ง getchar( ) เมื่อป้อนอักขระเข้ามาแล้ว จะต้องกดปุ่ม <Enter> โปรแกรมจึงจะกลับไปทำงานต่อได้ และตัวอักขระที่เราป้อนจะแสดงขึ้นมาให้เห็นบนหน้าจอด้วย ส่วนคำสั่ง getch( ) ไม่ต้องกดปุ่ม <Enter> เพียงแค่ผู้ใช้งานป้อนอักขระเข้ามา 1 ตัว โปรแกรมจะกลับไปทำงานต่อทันที และตัวอักขระที่เราป้อนจะไม่แสดงขึ้นมาให้เห็น โดยรูปแบบการเรียกใช้คำสั่ง getchar( ) แสดงไว้ดังต่อไปนี้

```
ch = getch( );
```

ch คือ ตัวแปรชนิด char เพื่อใช้เก็บค่าของอักขระหรือตัวอักษรที่รับเข้ามา

ตัวอย่างที่ 6.17 แสดงการทำงานของคำสั่ง getch( )

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                char ch;
                printf("Enter :");
                ch = getch();
                printf("\n Ans :%c",ch);
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter : 1
Ans : 1
```

6.2.7 คำสั่ง clrscr( )

คำสั่ง clrscr( ) เป็นคำสั่งที่ใช้ในการเคลียร์หรือล้างหน้าจอ ประโยชน์หลักของคำสั่งนี้ คือ ผู้พัฒนาโปรแกรมสามารถที่จะทำการล้างหน้าจอรายงานต่าง ๆ ของข้อมูลออกไปได้ เพราะถ้าไม่มีคำสั่งนี้โปรแกรมจะทดสอบและยังมีผลพัลธ์หรือคำตอบแสดงค่าหน้าจอ อาจจะทำให้ผู้ใช้งานมีความสับสนได้ โดยรูปแบบการเรียกใช้คำสั่ง clrscr( ) แสดงไว้ดังต่อไปนี้

```
clrscr( );
```

ตัวอย่างที่ 6.18 แสดงการทำงานของคำสั่ง clrscr( )

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )

        {
                char ch;
                printf("Hello witoon");
                ch = getch();
                clrscr( );
                printf("Wellcome to Thailand");
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Hello witoon
```

และหากกดปุ่มคีย์บอร์ดใด ๆ โปรแกรมจะทำการเคลียร์หน้าจอแล้วแสดงผลลัพธ์ต่อไป

```
Wellcome to Thailand
```

6.2.8 คำสั่ง goto

คำสั่ง goto เป็นคำสั่งที่ใช้ในการข้ามหรือกระโดดกลับไปทำงานในคำสั่งอื่น ๆ ตามที่ผู้พัฒนาโปรแกรมต้องการ โดยรูปแบบการเรียกใช้คำสั่ง goto แสดงไว้ดังต่อไปนี้

```
goto loop;
```

loop คือ ตำแหน่งที่ต้องการให้โปรแกรมกระโดดไปทำงานในตำแหน่งนั้น ๆ

ตัวอย่างที่ 6.19 แสดงการทำงานของคำสั่ง goto

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )

        {
                printf("1\n");
                goto loop1;
                printf("2\n");
        loop1: printf("3\n");
                goto loop2;
                printf("4\n");
        loop2: printf("5\n");
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
1
3
5
```

จากโปรแกรมจะสังเกตว่า เมื่อโปรแกรมเจอคำสั่ง goto เมื่อไร โปรแกรมจะทำการกระโดดไปทำงานตามที่ตำแหน่ง loop กำหนดไว้

สรุปท้ายบท

ขั้นตอนของการพัฒนาโปรแกรมภาษาซี

ในขั้นตอนของการพัฒนาโปรแกรมภาษาซี จะมีขั้นตอนในการทำงาน ดังนี้

1. เขียนโปรแกรมต้นฉบับด้วยภาษาซี

ซึ่งการเขียนโปรแกรมผู้พัฒนาจะต้องเขียนโปรแกรมต้นฉบับด้วยภาษาซีขึ้นมาก่อนหนึ่งโปรแกรม โดยการเขียนจะต้องคำนึงถึงหลักไวยากรณ์ของภาษาเป็นหลัก จากนั้นทำการบันทึกโปรแกรมพร้อมกับตั้งชื่อแฟ้มไว้ โดยแฟ้มที่ได้จะมีนามสกุล *.c หรือ เช่น simple1.c เป็นต้น

2. แปลโปรแกรมภาษาซีไปเป็นโปรแกรมภาษาเครื่อง

ในการแปลโปรแกรมภาษาซีไปเป็นโปรแกรมภาษาเครื่อง เป็นการใช้คำสั่ง compile เพื่อแปลโปรแกรมภาษาซีไปเป็นโปรแกรมภาษาเครื่อง แฟ้มที่ได้จะมีนามสกุล *.obj ซึ่งในขั้นตอนนี้โปรแกรมต้นฉบับอาจเกิดความผิดพลาดทางไวยากรณ์ภาษาขึ้นได้ จึงต้องย้อนกลับไปแก้ไขโปรแกรมต้นฉบับตามขั้นตอนที่หนึ่งให้ถูกต้องเสียก่อน

3. เชื่อมโยงโปรแกรมภาษาเครื่องเข้ากับ library function ของภาษาซี

การเชื่อมโยงโปรแกรมภาษาเครื่องเข้ากับ library function ของภาษาซี จะได้เป็นการ execute program โดยใช้คำสั่ง link แฟ้มที่ได้จะมีนามสกุล *.exe

4. การ execute program เพื่อแสดงผลลัพธ์ออกมา

ในการ execute program เพื่อแสดงผลลัพธ์ออกมา โดยใช้คำสั่ง run ในขั้นตอนนี้ผู้พัฒนาโปรแกรมควรตรวจสอบผลลัพธ์ที่ได้จากโปรแกรม ว่าตรงกับความต้องการของเราหรือไม่ ถ้าผลลัพธ์ที่ได้ไม่ตรงกับความต้องการให้กลับไปแก้ไขโปรแกรมต้นฉบับในขั้นตอนที่หนึ่งใหม่อีกครั้ง เสร็จแล้วทำขั้นตอนที่สองถึงขั้นตอนที่สี่ซ้ำอีกครั้ง และต้องทำซ้ำเช่นนี้จนกว่าจะได้ผลลัพธ์ที่ถูกต้อง

คำสั่งพื้นฐานในการทำงานของภาษาซี

คำสั่ง printf( ) ถือว่าเป็นคำสั่งพื้นฐานที่สุดในการแสดงผลข้อมูลทุกชนิดออกทางหน้าจอ ไม่ว่าจะเป็นจำนวนเต็ม (int) , ทศนิยม (float) , ข้อความ (string) หรืออักขระ

คำสั่ง putchar( ) คือ คำสั่งในการแสดงผลตัวอักษรหรืออักขระ (char) ออกทางหน้าจอ นอกจากใช้คำสั่ง printf( ) พร้อมกับกำหนดรหัสควบคุมรูปแบบ %c แล้ว เราสามารถเรียกใช้คำสั่งสำหรับแสดงตัวอักษรหรืออักขระโดยเฉพาะได้อีกด้วย

คำสั่ง puts( ) คือ คำสั่งในการแสดงผลข้อความหรืออักษรออกทางหน้าจอคอมพิวเตอร์ ดังนั้นข้อแตกต่างระหว่างคำสั่ง putchar( ) กับคำสั่ง puts() คือ putchar( ) จะสามารถแสดงผลแค่ตัวอักษรหรืออักขระ แต่ puts( ) สามารถแสดงผลเป็นข้อความได้

คำสั่ง scanf( ) เป็นคำสั่งพื้นฐานที่ใช้ในการรับผลข้อมูลจากผู้ใช้งาน และในภาษาซี การรับข้อมูลจากคีย์บอร์ดสามารถทำได้โดยการเรียกใช้ฟังก์ชัน scanf( ) ซึ่งเป็นฟังก์ชันมาตรฐานสำหรับรับข้อมูลจากคีย์บอร์ด โดยสามารถรับข้อมูลได้ทุกประเภท ไม่ว่าจะเป็นจำนวนเต็ม (int) , ทศนิยม (float) , อักขระ (char) หรือข้อความก็ตาม

คำสั่ง getchar( ) โดยในการรับค่าข้อมูลต่าง ๆ ในภาษาซีนอกจากใช้คำสั่ง scanff( ) แล้ว ผู้พัฒนาโปรแกรมสามารถเรียกใช้คำสั่ง getchar( ) เพื่อการรับค่าได้อีกด้วย

คำสั่ง getch( ) ในการรับค่าอักษระในภาษาซี นอกจากใช้คำสั่ง scanff( ) และคำสั่ง getchar( ) แล้ว ผู้พัฒนาโปรแกรมสามารถเรียกใช้คำสั่ง getch( ) เพื่อใช้ในการรับข้อมูลได้อีกคำสั่งหนึ่ง ข้อสังเกตถึงแม้ว่าทั้งคำสั่ง getchar( ) และ getch( ) จะใช้สำหรับรับข้อมูลชนิดอักขระเหมือนกัน แต่ทั้ง 2

คำสั่งนี้มีความแตกต่างกันอยู่ตรงที่ คำสั่ง getchar( ) เมื่อป้อนอักขระเข้ามาแล้ว จะต้องกดปุ่ม <Enter> โปรแกรมจึงจะกลับไปทำงานต่อได้ และตัวอักขระที่เราป้อนจะแสดงขึ้นมาให้เห็นบนหน้าจอด้วย ส่วนคำสั่ง getch( ) ไม่ต้องกดปุ่ม <Enter> เพียงแค่ผู้ใช้งานป้อนอักขระเข้ามา 1 ตัว โปรแกรมจะกลับไปทำงานต่อทันที และตัวอักขระที่เราป้อนจะไม่แสดงขึ้นมาให้เห็น

คำสั่ง clrscr( ) เป็นคำสั่งที่ใช้ในการเคลียร์หรือล้างหน้าจอ ประโยชน์หลักของคำสั่งนี้ คือ ผู้พัฒนาโปรแกรมสามารถที่จะทำการล้างหน้าจอรายงานต่าง ๆ ของข้อมูลออกไปได้

คำสั่ง goto เป็นคำสั่งที่ใช้ในการข้ามหรือกระโดดกลับไปทำงานในคำสั่งอื่น ๆ ตามที่ผู้พัฒนาโปรแกรมต้องการ

คำถามทบทวน

1. จงอธิบายขั้นตอนของการพัฒนาโปรแกรมภาษาซีว่ามีกี่ขั้นตอนอะไรบ้าง และแต่ละขั้นตอนมีความสำคัญอย่างไรต่อการเขียนโปรแกรมคอมพิวเตอร์

2. จงเขียนโปรแกรมให้แสดงผลข้อมูลดังนี้

I Love Dusittrang.

3. จงเขียนโปรแกรมให้แสดงผลข้อมูลดังนี้

C Programming C Programming
C Programming C Programming
C Programming C Programming

4. จงเขียนโปรแกรมเพื่อรับค่าตัวอักขระพร้อมทั้งแสดงผลตัวอักขระที่รับเข้ามา แล้วให้โปรแกรมขึ้นไปรับค่าใหม่ ดังตัวอย่างต่อไปนี้

Enter Character : a
Show Character : a

Enter Character : Z
Show Character : Z

Enter Character : y
Show Character : y

5. จงเขียนโปรแกรมเพื่อรับค่าตัวเลขพร้อมทั้งแสดงผลตัวเลขที่รับเข้ามา จากนั้นให้โปรแกรมรอรับค่าใด ๆ จากคีย์บอร์ด หากมีการป้อนค่าใด ๆ จากคีย์บอร์ดให้โปรแกรมจะล้างข้อมูลที่หน้าจอใหม่ แล้วให้โปรแกรมขึ้นไปรับค่าใหม่อีกครั้ง ดังตัวอย่างต่อไปนี้

Enter Number : 55.01
Numbe = 55.01

หากมีกดคีย์บอร์ด (โปรแกรมจะล้างข้อมูลหน้าจอใหม่ แล้วให้โปรแกรมขึ้นไปรับค่าใหม่อีกครั้ง)

Enter Number :

6. จงเขียนโปรแกรมแปลงค่าจากหน่วยเมตรเป็นหน่วยเซนติเมตร ดังตัวอย่าง

Meter = 1
Centimeter = 100

Meter = 2.5
Centimeter = 2500

7. จงเขียนโปรแกรมคำนวณหาพื้นที่สามเหลี่ยม โดยโปรแกรมทำการรับค่า ฐาน และ สูง แล้วนำมาคำนวณตามสูตรพื้นที่สามเหลี่ยม = (1/2) x ฐาน x สูง ดังตัวอย่าง

base = 10
high = 20
triangle = 100.00

base = 8
high = 15
triangle = 60.00

เอกสารอ้างอิง

คะชา ชาญศิลป์. (2548). **ภาษาซีสำหรับผู้เริ่มต้น.** กรุงเทพมหานคร: วิรัตน์เอ็ดดูเคชั่น.

ประภาพร ช่างไม้. (2545). **คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.** กรุงเทพมหานคร: อินโฟเพรส.

ธีรวัฒน์ ประกอบผล. (2553). **คู่มือการเขียนโปรแกรมภาษา C.** กรุงเทพมหานคร: ซิมพลิฟลาย.

สุระสิทธิ์ ทรงม้า และภูริพจน์ แก้วย่อง. (2552). **การโปรแกรมคอมพิวเตอร์.**: มหาวิทยาลัยราชภัฏสวนดุสิต.

สานนท์ เจริญฉาย. (2543). **การเขียนโปรแกรมและอัลกอริทึม.** กรุงเทพมหานคร: มหาจุฬาลงกรณ์ราชวิทยาลัย.

สมชาย รัตนเลิศนุสรณ์. (2553). **การเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี.** กรุงเทพมหานคร: สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) ส.ส.ท.

อรพิน ประวัติบริสุทธิ์. (2554). **คู่มือการเรียนภาษา C ฉบับปรับปรุงใหม่** (พิมพ์ครั้งที่ 10)**.** กรุงเทพมหานคร: โปรวิชั่น.

http://www.mwit.ac.th/~cs/download/tech30101/TECH30101_ch7.html

http://www.e-learning.snru.ac.th/els/program1/lesson2/page2_4.html

https://www.sites.google.com/site/cprogramingbypanus/kar-saedng-phllaphth/printf1.html

http://www.krubpk.com/com_2/pan7/pan7.html

บทที่ 7

โครงสร้างการควบคุมการทำงานแบบทางเลือก

แผนการสอนประจำบท

หัวข้อเนื้อหา

7.1 คำสั่ง if

- 7.1.1 คำสั่ง if แบบเลือกทำทางเดียว (if)
- 7.1.2 คำสั่ง if แบบเลือกทำสองทาง (if... else)
- 7.1.3 คำสั่ง if แบบเลือกทำหลายทาง (if... else if)

7.2 คำสั่ง switch

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถเข้าใจหลักการทำงานของคำสั่ง if ได้อย่างเข้าใจและถูกต้อง
2. ผู้เรียนสามารถนำคำสั่ง if ไปใช้งานในการเขียนโปรแกรมในกรณีต่าง ๆ ได้อย่างถูกต้อง
3. ผู้เรียนสามารถเข้าใจหลักการทำงานของคำสั่ง switch ได้อย่างเข้าใจและถูกต้อง
4. ผู้เรียนสามารถนำคำสั่ง switch ไปใช้งานในการเขียนโปรแกรมในกรณีต่าง ๆ ได้อย่างถูกต้อง

วิธีการสอน

1. สอนแบบบรรยายโดยใช้เอกสารประกอบการสอน
2. สอนแบบบรรยายโดยใช้ Slide Power Point
3. สอนโดยใช้โปรแกรม Turbo C++ 4.5 เพื่อลงมือปฏิบัติในการเขียนโปรแกรมภาษาซี
4. สอนโดยให้ผู้เรียนเป็นส่วนกลาง โดยใช้วิธีถามตอบกับผู้เรียนระหว่างบรรยาย
5. ให้ผู้เรียนสรุปและอธิบายหัวข้อที่ผู้เรียนที่ได้รับมอบหมาย

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. Slide Power Point
3. โปรแกรม Turbo C++ 4.5
4. ตัวอย่างแบบทดสอบ

การวัดผลและประเมินผล

1. การทำตามตัวอย่างโจทย์คำสั่งและการทำแบบฝึกหัด
2. ความตั้งใจในชั้นเรียนและการร่วมมือในการนำเสนอหรืออภิปราย
3. สังเกตจากการตอบคำถามและการร่วมทำกิจกรรมในชั้นเรียน

บทที่ 7

โครงสร้างการควบคุมการทำงานแบบทางเลือก

ในการพัฒนาหรือเขียนโปรแกรมคอมพิวเตอร์ไม่ว่าจะเป็นภาษาคอมพิวเตอร์ใด ๆ ก็ตามเงื่อนไขหนึ่งที่มีการเรียกใช้งานอย่างบ่อยครั้ง คือ เงื่อนไขแบบทางเลือก โดยในภาษาซีจะมีคำสั่งที่เป็นการเลือกเงื่อนไขการทำงานอยู่ 2 คำสั่ง คือ คำสั่ง if และคำสั่ง switch แต่คำสั่งที่มีการใช้งานมากนั้นก็คือคำสั่ง if ดังนั้นนั้นผู้ที่คิดจะพัฒนาโปรแกรมคอมพิวเตอร์จะต้องมีความเชียวชาญในการใช้คำสั่ง if ให้มาก เพราะเป็นคำสั่งที่ต้องมาใช้ในการตัดสินใจในทางเลือกทางทำงานให้กับโปรแกรม ซึ่งในการพัฒนาโปรแกรมคอมพิวเตอร์ ผู้พัฒนาจะมีการเรียกใช้งานคำสั่ง if ได้หลายรูปแบบขึ้นอยู่กับความเหมาะสมกับเงื่อนไขนั้น ๆ โดยคำสั่ง if จะมีรูปแบบหลักอยู่ 3 รูปแบบ คือ

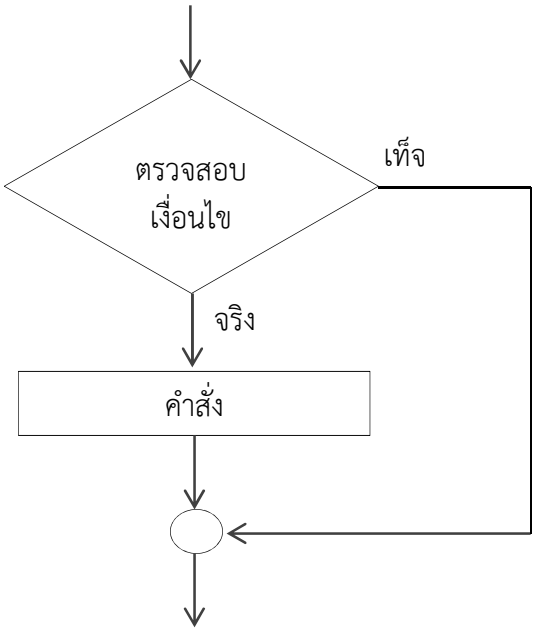
1.) คำสั่ง if แบบเลือกทำทางเดียว (if)
2.) คำสั่ง if แบบเลือกทำสองทาง (if... else)
3.) คำสั่ง if แบบเลือกทำหลายทาง (if... else if)

7.1 คำสั่ง if

7.1.1 คำสั่ง if แบบเลือกทำทางเดียว (if)

คำสั่ง if แบบเลือกทำทางเดียว เป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไขของผู้ใช้งาน โดยจะตรวจสอบแบบทางเดียว คือ จะทำงานเฉพาะกรณีที่เงื่อนไขเป็นจริงเท่านั้น หากเงื่อนไขเป็นเท็จจะไม่ทำงาน โดยเงื่อนไขที่เป็นจริง คือ คำสั่งที่อยู่ใน { } ของคำสั่ง if เท่านั้น ซึ่งคำสั่ง if แบบเลือกทำทางเดียวมีรูปแบบการทำงานของคำสั่ง if แบบเลือกทำทางเดียวแสดงไว้ ดังนี้

```
If (เงื่อนไข)
  {
      // ชุดคำสั่งที่เป็นจริง
      คำสั่ง..... 1
      คำสั่ง..... 2
      คำสั่ง..... 3
      คำสั่ง.....
      คำสั่ง.....
      คำสั่ง.....
  }
```



ภาพที่ 7.1 ผังงานแสดงการทำงานของคำสั่ง if แบบเลือกทำทางเดียว

ตัวอย่างการใช้คำสั่ง if แบบเลือกทำทางเดียว

ตัวอย่างที่ 7.1 ให้โปรแกรมรับค่าน้ำหนักจากผู้ใช้งาน แล้วให้ตรวจสอบว่าค่าน้ำหนักที่รับรับมามีค่ามากกว่า 100 หรือไม่ ถ้ามากกว่าหรือเท่ากับ 80 ให้โปรแกรมแสดงคำว่า “Fat” ออกมาทางหน้าจอ

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                float weight;
                printf("Please fill weight : ");
                scanf(“%f”,& weight);
                if (weight >= 80)
                {
                        printf("\nYou fat.");
                }
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Please fill weight :
```

และหากกรณีใส่ค่าน้ำหนักมากกว่า 80 โปรแกรมจะแสดงผล ดังนี้

```
Please fill weight : 90
You fat.
```

ตัวอย่างที่ 7.2 ให้โปรแกรมรับค่าตัวเลขจำนวนเต็มจากผู้ใช้งาน 2 ค่า แล้วให้โปรแกรมตรวจสอบว่าหากค่าแรกที่ป้อนเข้ามามีค่ามากกว่าให้แสดงผลออกทางหน้าจอว่า “Bingo” จากนั้นให้ผู้ใช้งานกดคีย์บอร์ดใด ๆ แล้วให้โปรแกรมขึ้นรับค่าใหม่แล้วตรวจสอบเงื่อนไขเช่นเดิมอีกครั้ง

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int num1 , num2;
                char ch;
        loop1:  printf("Please Number1: ");
                scanf("%d",&num1);
                printf("\nPlease Number2: ");
                scanf("%d",&num2);
                if (num1 > num2)
                {
                        printf("\nBiggo ");
                }
                ch = getch();
                goto loop1;
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

- กรณีใส่ค่าตัวแรกมากกว่ากว่าตัวที่สอง จากนั้นกดคีย์บอร์ด 1 ครั้ง

```
Please Number1: 100

Please Number2: 50

Biggo

Please Number1: _
```

- กรณีใส่ค่าตัวแรกน้อยกว่ากว่าตัวที่สอง จากนั้นกดคีย์บอร์ด 1 ครั้ง

```
Please Number1: 3000

Please Number2: 5000

Please Number1: _
```

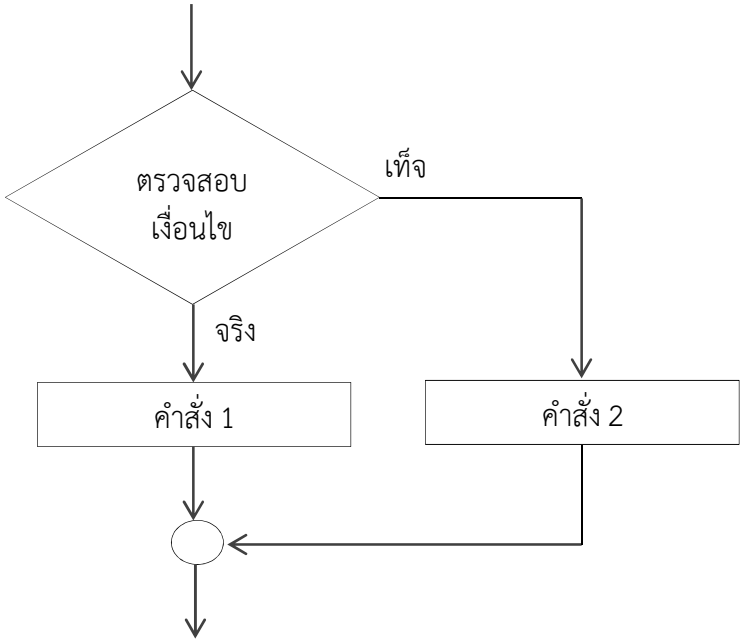
- กรณีใส่ค่าตัวแรกมากกว่ากว่าตัวที่สอง จากนั้นกดคีย์บอร์ด 1 ครั้ง และใส่ค่าตัวแรกมากกว่ากว่าตัวที่สองอีกครั้ง

```
Please Number1: 100

Please Number2: 50

Biggo

Please Number1: 8000

Please Number2: 7999

Biggo
```

7.1.2 คำสั่ง if แบบเลือกทำสองทาง (if... else)

คำสั่ง if แบบเลือกทำสองทางเป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไขของผู้ใช้งาน โดยจะตรวจสอบการทำงานทั้งสองเงื่อนไข คือ กรณีที่เงื่อนไขเป็นจริงก็จะทำงานคำสั่งที่อยู่ใน { } ของเงื่อนไขที่เป็นจริง แต่หากกรณีที่เป็นเท็จจะทำงานคำสั่งที่อยู่ใน { } ภายใต้เงื่อนไข else รูปแบบการทำงานของคำสั่ง if แบบเลือกทำสองทางแสดงไว้ดังนี้

```
If (เงื่อนไข)
  {
        // ชุดคำสั่งที่เป็นจริง
        คำสั่ง..... 1
        คำสั่ง..... 2
        คำสั่ง..... 3
        คำสั่ง.....
  }
  else
  {
        // ชุดคำสั่งที่เป็นเท็จ
        คำสั่ง..... 1
        คำสั่ง..... 2
        คำสั่ง..... 3
        คำสั่ง.....
  }
```



ภาพที่ 7.2 ผังงานแสดงการทำงานของคำสั่ง if แบบเลือกทำสองทาง

ตัวอย่างการใช้คำสั่ง if แบบเลือกทำสองทาง

ตัวอย่างที่ 7.3 ให้โปรแกรมรับค่าตัวเลขจำนวนเต็มจากผู้ใช้งาน 2 ค่า แล้วให้โปรแกรมตรวจสอบว่าหากค่าใดมากกว่ากันให้แสดงคำตอบค่านั้นออกทางหน้าจอ

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int num1 , num2;
                printf("Please Number1: ");
                scanf("%d",&num1);
                printf("\nPlease Number2: ");
                scanf("%d",&num2);
                if (num1 > num2)
                {
                        printf("\n%d",num1);
                }
                else
                {
                        printf("\n%d",num2);
                }
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

- กรณีใส่ค่าตัวหนึ่งมากกว่าตัวที่สอง

```
Please Number1: 450

Please Number2: 150

450
```

- กรณีใส่ค่าตัวสองมากกว่าตัวที่หนึ่ง

```
Please Number1: 1999

Please Number2: 2001

2001
```

- กรณีใส่ค่าตัวแรกเท่ากับตัวที่สอง

```
Please Number1: 100

Please Number2: 100

100
```

ตัวอย่างที่ 7.4 ให้โปรแกรมรับค่าตัวเลขจำนวนเต็ม 1 จำนวน แล้วตรวจสอบว่าค่าเท่ากับ 100 หรือไม่ ถ้าค่าที่รับเข้ามามีค่าเท่ากับ 100 ให้โปรแกรมแสดงคำว่า “Biggo” ออกมาทางหน้าจอ แต่ถ้าหากไม่ใช่ให้โปรแกรมเคลียร์หน้าจอ แล้วขึ้นไปรับค่าใหม่แล้ว

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int num;
  loop_Y:       printf("Enter Number : ");
                scanf("%d",&num);
                if (num == 100)
                {
                        printf("\nBiggo");
                }
                else
                {
                        clrscr();
                        goto loop_Y;
                }
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

- กรณีใส่ค่าตัวเลขจำนวนเต็มใด ๆ ที่ไม่เท่ากับ 100

```
Enter Number : 300
```

จากนั้นโปรแกรมจะทำเคลียร์หน้าจอแล้วขึ้นไปรับค่าใหม่อีกครั้ง

```
Enter Number :
```

- กรณีใส่ค่าตัวเลขจำนวนเต็มเท่ากับ 100

```
Enter Number : 300

Biggo
```

ตัวอย่างที่ 7.5 ให้โปรแกรมแสดงข้อความว่า “You are a man.” ถ้าผู้ใช้โปรแกรมตอบ y ให้แสดงคำว่า “Man” แต่ถ้าตอบอื่น ๆ ให้แสดงคำว่า “Women” จากนั้นให้โปรแกรมขึ้นไปรับค่าใหม่อีกครั้ง

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                char ch;
 loop1:         printf("\nYou are a man. (y) :  ");
                ch = getch();
                if (ch == ‘y’)
                {
                        printf("\nMen");
                }
                else
                {
                        printf("\nWomen");
                }
                goto loop1;
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

- กรณีใส่ค่าอักษระที่ไม่ใช่ “y” (ตัวพิมพ์เล็ก)

```
You are a man. (y) :  m

Women

You are a man. (y) :  a

Women

You are a man. (y) :  Y

Women

You are a man. (y) :
```

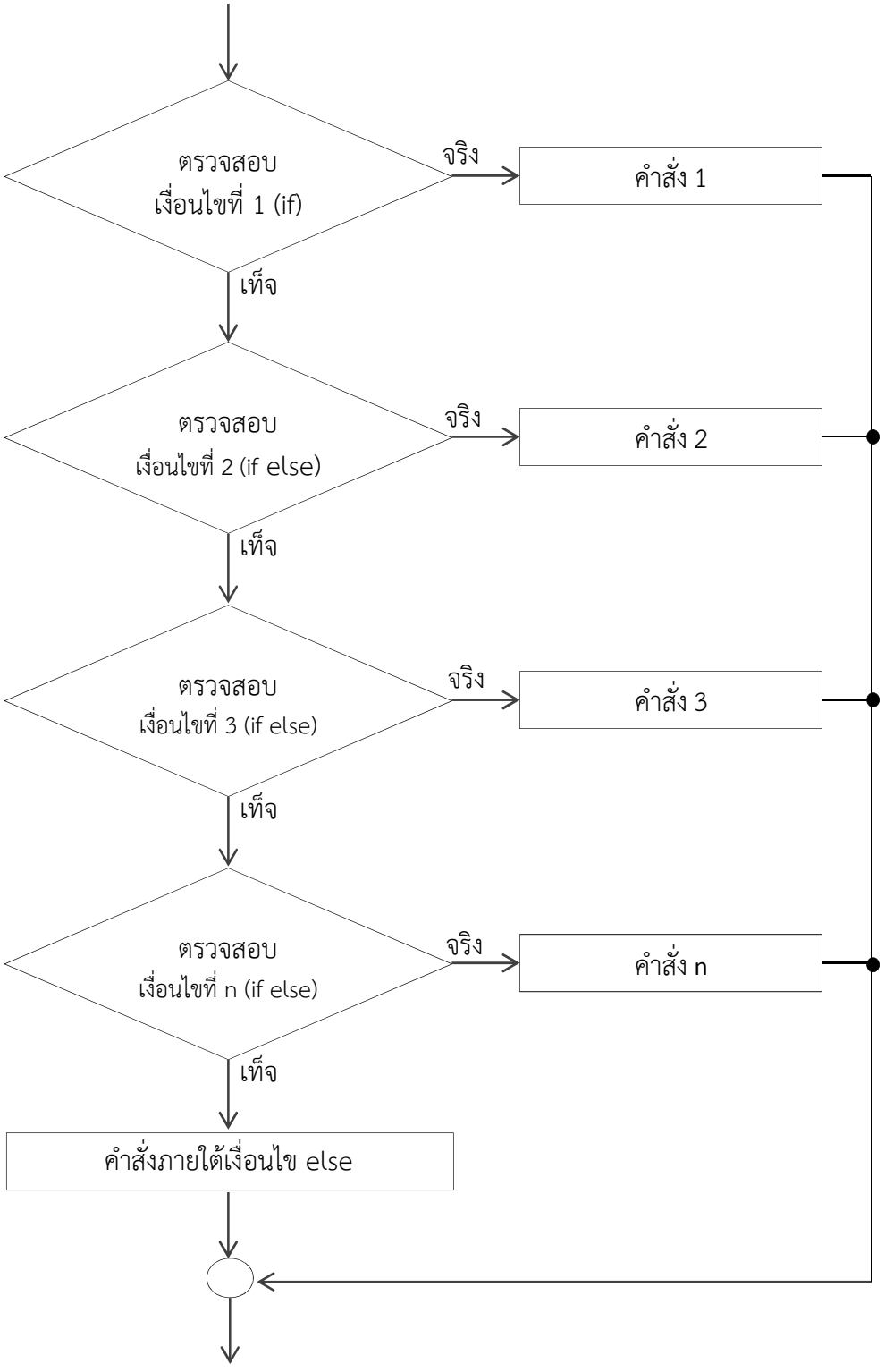
- กรณีใส่ค่าอักษระที่เป็นตัว “y” (ตัวพิมพ์เล็ก)

```
You are a man. (y) :  m

Men

You are a man. (y) :
```

7.1.3 คำสั่ง if แบบเลือกทำหลายทาง (if... else if)

คำสั่ง if แบบเลือกทำหลายทาง จะเป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไขของผู้ใช้งานที่มีเงื่อนไขมากกว่าสองทางเลือก โดยการตรวจสอบการทำงานในเงื่อนไขใด ๆ ที่เป็นจริงเท่านั้น แต่ถ้าไม่มีเงื่อนไขใด ๆ เป็นจริง โปรแกรมจะไปทำงานภายใต้เงื่อนไข else รูปแบบการทำงานของคำสั่ง if แบบเลือกทำหลายทางแสดงไว้ดังนี้

```
If (เงื่อนไขที่ 1)
  {
        // ชุดคำสั่งของเงื่อนไขที่ 1
        คำสั่ง.....
        คำสั่ง.....
  }
else If (เงื่อนไขที่ 2)
  {
        // ชุดคำสั่งของเงื่อนไขที่ 2
        คำสั่ง.....
        คำสั่ง.....
  }
else If (เงื่อนไขที่ 3)
  {
        // ชุดคำสั่งของเงื่อนไขที่ 3
        คำสั่ง.....
        คำสั่ง.....
  }
else If (เงื่อนไขที่ n)
  {
        // ชุดคำสั่งของเงื่อนไขที่ n
        คำสั่ง.....
        คำสั่ง.....
  }
  else
  {
        // ชุดคำสั่งที่เป็นเท็จ (หากกรณีที่เงื่อน if ใด ๆ เป็นเท็จทั้งหมด)
        คำสั่ง.....
        คำสั่ง.....
  }
```



ภาพที่ 7.3 ผังงานแสดงการทำงานของคำสั่ง if แบบเลือกทำหลายทาง

ตัวอย่างการใช้คำสั่ง if แบบเลือกทำหลายทาง

ตัวอย่างที่ 7.6 ให้โปรแกรมรับค่าตัวเลขจำนวนเต็มจากผู้ใช้งาน 2 ค่า แล้วให้โปรแกรมตรวจสอบ โดยมีเงื่อนไขดังนี้

- หากค่าแรกที่รับเข้ามามีค่ามากกว่าค่าที่สองให้แสดงคำว่า “Number 1 : Bingo”
- หากค่าแรกที่รับเข้ามามีค่าน้อยกว่าค่าที่สองให้แสดงคำว่า “Number 2 : Bingo”
- หากค่าแรกที่รับเข้ามามีค่าเท่ากับค่าที่สองให้แสดงคำว่า “Error”
- เมื่อโปรแกรมแสดงผลใด ๆ เสร็จสิ้นให้โปรแกรมขึ้นรับค่าใหม่อีกครั้ง

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int num1 , num2;
   loop_new:    printf("\n\nPlease Number1: ");
                scanf("%d",&num1);
                printf("\nPlease Number2: ");
                scanf("%d",&num2);
                if (num1 > num2)
                {
                        printf("\nNumber 1 : Bingo");
                }
                else if (num1 < num2)
                {
                        printf("\nNumber 2 : Bingo");
                }
                else
                {
                        printf("\nError");
                }
                goto loop_new;
         }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Please Number1: 111

Please Number2: 101

Number 1 : Bingo


Please Number1: 9876

Please Number2: 9998

Number 2 : Bingo


Please Number1: 1001

Please Number2: 1001

Error


Please Number1: 434

Please Number2: 343

Number 1 : Bingo


Please Number1: _
```

ตัวอย่างที่ 7.7 โปรแกรมตัดเกรดโดยให้รับค่าจากผู้ใช้งาน 3 ค่า ซึ่งค่าที่ 1 คือ คะแนนเก็บ ค่าที่ 2 คือ คะแนนสอบกลางภาค และค่าที่ 3 คือ คะแนนปลายภาค จากนั้นนำค่าทั้ง 3 มารวมกัน จากนั้นนำคะแนนรวมที่ได้มาตรวจสอบเกรดแล้วให้แสดงผลคะแนนเกรดดังระดับค่าคะแนน ดังนี้

ค่าคะแนนรวม 90 - 100 = A ค่าคะแนนรวม 70 - 89 = B
ค่าคะแนนรวม 60 - 69 = C ค่าคะแนนรวม 50 - 59 = D
ค่าคะแนนรวม 0 - 49 = F

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int num1 , num2 , num3 , num4;
                printf("\nCollection points.: ");
                scanf("%d",&num1);
                printf("\nCentral points.: ");
                scanf("%d",&num2);
                printf("\nFinal score.: ");
                scanf("%d",&num3);
                num4 = num1+num2+num3;
                printf("\nAggregate.: %d " ,num4);
                if (num4>= 90 && num4<= 100)
                {
                        printf("\n\nGrade 'A'");
                }
                else if (num4>= 70 && num4<= 89)
                {
                        printf("\n\nGrade 'B'");
                }
                else if (num4>= 60 && num4<= 69)
                {
                        printf("\n\nGrade 'C'");
                }
                else if (num4>= 50 && num4<= 59)
                {
                        printf("\n\nGrade 'D'");
                }
                else
                {
                        printf("\n\nGrade 'F'");
                }
```

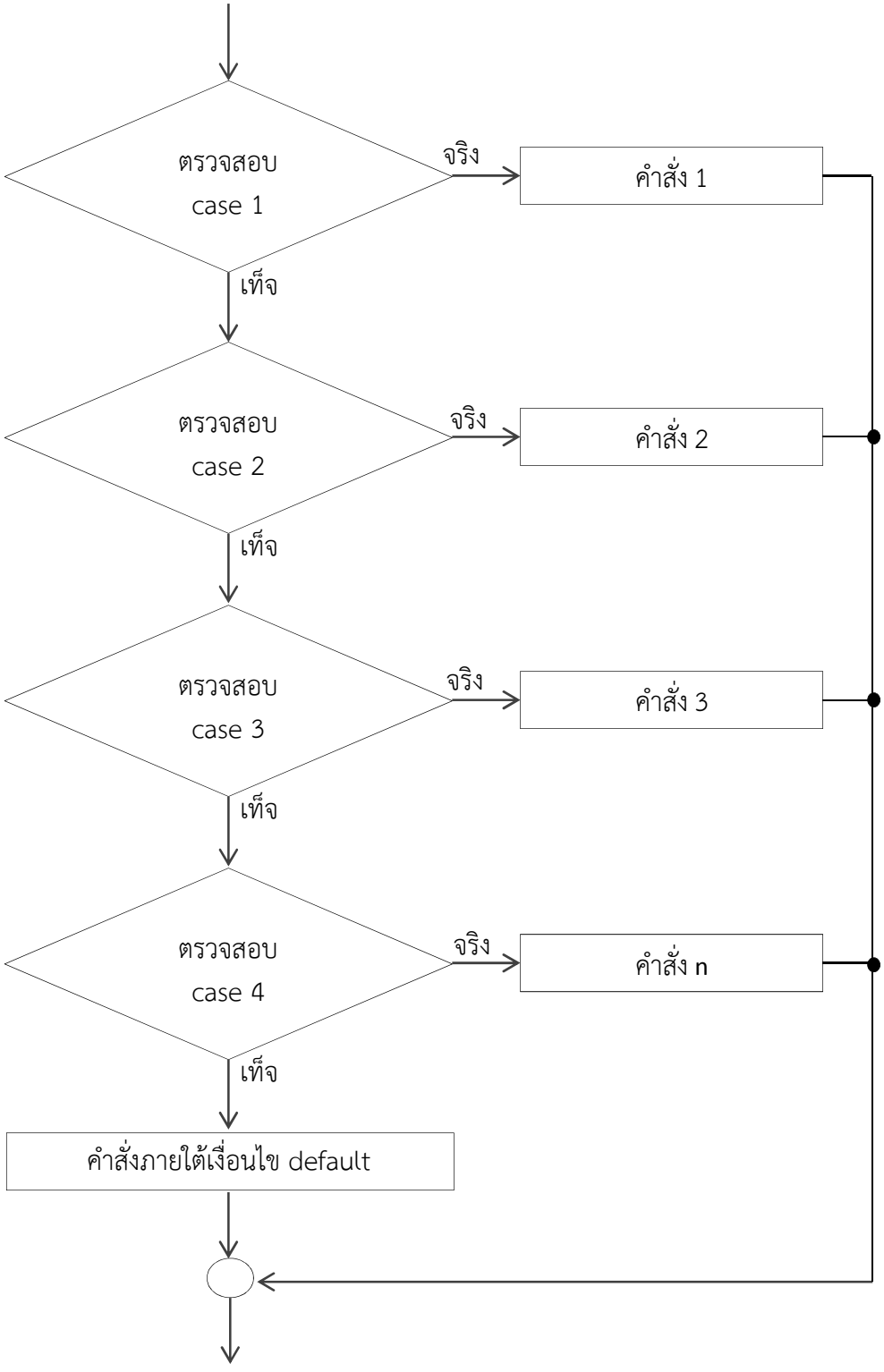
ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Collection points.: 10
Central points.: 30
Final score.: 30
Aggregate.: 70
Grade 'B'

Collection points.: 25
Central points.: 40
Final score.: 30
Aggregate.: 95
Grade 'A'

Collection points.: 15
Central points.: 15
Final score.: 10
Aggregate.: 40
Grade 'F'

Collection points.: 25
Central points.: 5
Final score.: 25
Aggregate.: 55
Grade 'D'

Collection points.: 22
Central points.: 22
Final score.: 22
Aggregate.: 66
Grade 'C'

Collection points.: _
```

7.2 คำสั่ง switch

คำสั่ง switch เป็นคำสั่งที่มีทางเลือกแบบหลายทางเช่นเดียวกับคำสั่ง if แบบเลือกทำหลายทาง แต่คำสั่ง switch เป็นคำสั่งที่ตัดสินใจทางเลือกกรณีที่มีตัวแปรเดียว และเมื่อจบการทำงานในคำสั่งใด ๆ ในเงื่อนไขนั้น ๆ จะต้องจบการทำงานด้วยคำสั่ง break; โดยรูปแบบการทำงานของคำสั่ง switch แสดงไว้ดังนี้

```
switch (ตัวแปร)
  {
        case 1 :        // ชุดคำสั่งของเงื่อนไขที่ 1
                        คำสั่ง.....
                        คำสั่ง.....
                        break;

        case 2 :        // ชุดคำสั่งของเงื่อนไขที่ 2
                        คำสั่ง.....
                        คำสั่ง.....
                        break;

        case 3 :        // ชุดคำสั่งของเงื่อนไขที่ 3
                        คำสั่ง.....
                        คำสั่ง.....
                        break;

        case 4 :        // ชุดคำสั่งของเงื่อนไขที่ 4
                        คำสั่ง.....
                        คำสั่ง.....
                        break;

        case n :        // ชุดคำสั่งของเงื่อนไขที่ n
                        คำสั่ง.....
                        คำสั่ง.....
                        break;

        default :       /* ชุดคำสั่งกรณีที่เงื่อนใน case ใดๆ
                           ไม่มีเงื่อนไขใดเป็นจริง */
                        คำสั่ง.....
                        คำสั่ง.....
  }
```



ภาพที่ 7.4 ผังงานแสดงการทำงานของคำสั่ง switch

ตัวอย่างที่ 7.8 จงเขียนโปรแกรมให้รับค่าตัวเลขจำนวนเต็มจากผู้ใช้งานหนึ่งตัวเลข โดยให้โปรแกรมรับค่าตัวเลขจำนวนเต็มเลข 1 ถึง 10 จากนั้นโปรแกรมจะพิมพ์คำของตัวเลขที่ผู้ใช้งานได้กรอกเข้าไป แต่หากตัวเลขจำนวนเต็มไม่ได้อยู่ในช่วงที่กำหนด โปรแกรมจะแสดงคำว่า “Error” เมื่อโปรแกรมแสดงผลใด ๆ เสร็จสิ้นโปรแกรมจะขึ้นไปค่าตัวเลขใหม่อีกครั้ง

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int num;
        loop1:  printf("\n\nEnter Number : ");
                scanf("%d",&num);
                switch(num)
                {
                  case 1: printf("\none");
                                break;
                  case 2: printf("\ntwo");
                                break;
                  case 3: printf("\nthree");
                                break;
                  case 4: printf("\nfour");
                                break;
                  case 5: printf("\nfive");
                                break;
                  case 6: printf("\nsix");
                                break;
                  case 7: printf("\nseven");
                                break;
                  case 8: printf("\neigth");
                                break;
                  case 9: printf("\nnine");
                                break;
                  case 10: printf("\nten");
                                break;
                  default: printf("\nerror");
                }
                        goto loop1;
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter Number : 1
one

Enter Number : 2
two

Enter Number : 3
three

Enter Number : 4
four

Enter Number : 5
five

Enter Number : 6
six

Enter Number : 7
seven

Enter Number : 8
eight

Enter Number : 9
nine

Enter Number : 10
ten

Enter Number : 11
error

Enter Number : _
```

ตัวอย่างที่ 7.9 ตัวอย่างนี้เป็นการใช้โจทย์ทดลองจากโจทย์ตัวอย่างที่ 7.8 แต่ครั้งจะทำการทดลองตัดคำสั่ง break; ในบางบรรทัด ทั้งนี้เพื่อจะให้ผู้พัฒนาผู้โปรแกรมเห็นว่าหากไม่มีคำสั่ง break; โปรแกรมจะไม่หยุดการทำงาน โดยโปรแกรมจะทำการแสดงผลไปเรื่อย ๆ จนกว่าจะเจอคำสั่ง break; ดังตัวอย่างโค้ดคำสั่งต่อไปนี้

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )

        {
                int num;
        loop1:  printf("\n\nEnter Number : ");
                scanf("%d",&num);
                switch(num)
                {
                  case 1: printf("\none");

                  case 2: printf("\ntwo");

                  case 3: printf("\nthree");
                                break;
                  case 4: printf("\nfour");
                                break;
                  case 5: printf("\nfive");

                  case 6: printf("\nsix");
                                break;
                  case 7: printf("\nseven");

                  case 8: printf("\neight");

                  case 9: printf("\nnine");

                  case 10: printf("\nten");

                  default: printf("\nerror");
                }
                        goto loop1;
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter Number : 1
one
two
three

Enter Number : 2
two
three

Enter Number : 3
three

Enter Number : 4
four

Enter Number : 5
five
six

Enter Number : 6
six

Enter Number : 7
seven
eight
nine
ten
error

Enter Number : 10
ten
error

Enter Number : 11
error

Enter Number : _
```

สรุปท้ายบท

ในการพัฒนาหรือเขียนโปรแกรมคอมพิวเตอร์ไม่ว่าจะเป็นภาษาคอมพิวเตอร์ใด ๆ ก็ตามเงื่อนไขหนึ่งที่มีการเรียกใช้งานอย่างบ่อยครั้ง คือ เงื่อนไขแบบทางเลือก โดยในภาษาซีจะมีคำสั่งที่เป็นการเลือกเงื่อนไขการทำงานอยู่ 2 คำสั่ง คือ คำสั่ง if และคำสั่ง switch แต่คำสั่งที่มีการใช้งานมากนั้นก็คือคำสั่ง if ดังนั้นนั้นผู้ที่คิดจะพัฒนาโปรแกรมคอมพิวเตอร์จะต้องมีความเชียวชาญในการใช้คำสั่ง if ให้มาก เพราะเป็นคำสั่งที่ต้องมาใช้ในการตัดสินใจในทางเลือกทางทำงานให้กับโปรแกรม

คำสั่ง if

คำสั่ง if มีหลายรูปแบบขึ้นอยู่กับความเหมาะสมกับเงื่อนไขนั้น ๆ โดยคำสั่ง if จะมีรูปแบบหลักอยู่ 3 รูปแบบ คือ

1.) คำสั่ง if แบบเลือกทำทางเดียว (if)

คำสั่ง if แบบเลือกทำทางเดียว เป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไขของผู้ใช้งาน โดยจะตรวจสอบแบบทางเดียว คือ จะทำงานเฉพาะกรณีที่เงื่อนไขเป็นจริงเท่านั้น หากเงื่อนไขเป็นเท็จจะไม่ทำงาน โดยเงื่อนไขที่เป็นจริง คือ คำสั่งที่อยู่ใน { } ของคำสั่ง if เท่านั้น

2.) คำสั่ง if แบบเลือกทำสองทาง (if... else)

คำสั่ง if แบบเลือกทำสองทางเป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไขของผู้ใช้งาน โดยจะตรวจสอบการทำงานทั้งสองเงื่อนไข คือ กรณีที่เงื่อนไขเป็นจริงก็จะทำงานคำสั่งที่อยู่ใน { } ของเงื่อนไขที่เป็นจริง แต่หากกรณีที่เป็นเท็จจะทำงานคำสั่งที่อยู่ใน { } ภายใต้เงื่อนไข else

3.) คำสั่ง if แบบเลือกทำหลายทาง (if... else if)

คำสั่ง if แบบเลือกทำหลายทาง จะเป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไขของผู้ใช้งานที่มีเงื่อนไขมากกว่าสองทางเลือก โดยการตรวจสอบการทำงานในเงื่อนไขใด ๆ ที่เป็นจริงเท่านั้น แต่ถ้าไม่มีเงื่อนไขใด ๆ เป็นจริง โปรแกรมจะไปทำงานภายใต้เงื่อนไข else

คำสั่ง switch

คำสั่ง switch เป็นคำสั่งที่มีทางเลือกแบบหลายทางเช่นเดียวกับคำสั่ง if แบบเลือกทำหลายทาง แต่คำสั่ง switch เป็นคำสั่งที่ตัดสินใจทางเลือกกรณีที่มีตัวแปรเดียว และเมื่อจบการทำงานในคำสั่งใด ๆ ในเงื่อนไขนั้น ๆ จะต้องจบการทำงานด้วยคำสั่ง break;

คำถามทบทวน

1. จงเขียนโปรแกรมตรวจสอบค่าความสูง หากความสูงมากกว่าหรือเท่ากับ 180 ให้โปรแกรมแสดงคำว่า “High” และหากความสูงน้อยกว่า 180 ให้โปรแกรมแสดงคำว่า “Short”

2. จงเขียนโปรแกรมเรียงลำดับจากน้อยไปมาก โดยให้รับค่าตัวเลขจำนวนเต็ม 3 จำนวน

3. จงเขียนโปรแกรมเรียงลำดับจากมากไปน้อย โดยให้รับค่าตัวเลขจำนวนเต็ม 5 จำนวน

4. จงเขียนโปรแกรมตัดเกรดโดยให้รับค่าจากผู้ใช้งาน 1 ค่า แล้วให้แสดงผลคะแนนเกรดดังระดับค่าคะแนน ดังนี้

ค่าคะแนนรวม 90 - 100 = A

ค่าคะแนนรวม 85 - 89 = B+

ค่าคะแนนรวม 75 - 84 = B

ค่าคะแนนรวม 70 - 74 = C+
ค่าคะแนนรวม 60 - 69 = C
ค่าคะแนนรวม 55 - 59 = D+
ค่าคะแนนรวม 54 - 50 = D
ค่าคะแนนรวม 0 - 49 = F
และหากผู้ใช้งานป้อนค่าคะแนนไม่อยู่ช่วงที่กำหนดให้โปรแกรมแสดงคำว่า “Error”

5. จงเขียนโปรแกรมโดยใช้คำสั่ง switch โดยให้โปรแกรมรับค่าตัวอักขระจากผู้ใช้งาน แล้วแสดงผลดังนี้

ถ้าผู้ใช้งานป้อน ‘A’ โปรแกรมจะแสดงคำว่า “Apple”
ถ้าผู้ใช้งานป้อน ‘B’ โปรแกรมจะแสดงคำว่า “Book”
ถ้าผู้ใช้งานป้อน ‘C’ โปรแกรมจะแสดงคำว่า “Cat”
ถ้าผู้ใช้งานป้อน ‘D’ โปรแกรมจะแสดงคำว่า “Dusit”
ถ้าผู้ใช้งานป้อน ‘E’ โปรแกรมจะแสดงคำว่า “Eagle”
และหากผู้ใช้งานป้อนค่าตัวอักขระไม่อยู่ในค่าที่กำหนดให้โปรแกรมแสดงคำว่า “Error”

เอกสารอ้างอิง

คะชา ชาญศิลป์. (2548). **ภาษาซีสำหรับผู้เริ่มต้น.** กรุงเทพมหานคร: วิรัตน์เอ็ดดูเคชั่น.

ประภาพร ช่างไม้. (2545). **คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.** กรุงเทพมหานคร: อินโฟเพรส.

ธีรวัฒน์ ประกอบผล. (2553). **คู่มือการเขียนโปรแกรมภาษา C.** กรุงเทพมหานคร: ซิมพลิฟลาย.

สุระสิทธิ์ ทรงม้า และภูริพจน์ แก้วย่อง. (2552). **การโปรแกรมคอมพิวเตอร์.**: มหาวิทยาลัยราชภัฏสวนดุสิต.

สมชาย รัตนเลิศนุสรณ์. (2553). **การเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี.** กรุงเทพมหานคร: สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) ส.ส.ท.

สานนท์ เจริญฉาย. (2543). **การเขียนโปรแกรมและอัลกอริทึม.** กรุงเทพมหานคร: มหาจุฬาลงกรณ์ราชวิทยาลัย.

อรพิน ประวัติบริสุทธิ์. (2554). **คู่มือการเรียนภาษา C ฉบับปรับปรุงใหม่** (พิมพ์ครั้งที่ 10)**.** กรุงเทพมหานคร: โปรวิชั่น.

http://www.mwit.ac.th/~cs/download/tech30101/TECH30101_ch8.html

http://www.e-learning.snru.ac.th/els/program1/lesson2/page5_1.html

https://www.sites.google.com/site/cprogramingbypanus/kar-saedng-phllaphth/printf4.html

http://www.krubpk.com/com_2/pan7/pan7_1.html

บทที่ 8

โครงสร้างการควบคุมการทำงานแบบวนรอบ

แผนการสอนประจำบท

หัวข้อเนื้อหา

8.1 คำสั่ง for loop
8.2 คำสั่ง while loop
8.3 คำสั่ง do while

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถเข้าใจหลักการทำงานของคำสั่ง for loop ได้อย่างเข้าใจและถูกต้อง
2. ผู้เรียนสามารถนำคำสั่ง for loop ไปใช้งานในการเขียนโปรแกรมในกรณีต่าง ๆ ได้อย่างถูกต้อง
3. ผู้เรียนสามารถเข้าใจหลักการทำงานของคำสั่ง while loop ได้อย่างเข้าใจและถูกต้อง
4. ผู้เรียนสามารถนำคำสั่ง while loop ไปใช้งานในการเขียนโปรแกรมในกรณีต่าง ๆ ได้อย่างถูกต้อง
5. ผู้เรียนสามารถเข้าใจหลักการทำงานของคำสั่ง do while ได้อย่างเข้าใจและถูกต้อง
6. ผู้เรียนสามารถนำคำสั่ง do while ไปใช้งานในการเขียนโปรแกรมในกรณีต่าง ๆ ได้อย่างถูกต้อง

วิธีการสอน

1. สอนแบบบรรยายโดยใช้เอกสารประกอบการสอน
2. สอนแบบบรรยายโดยใช้ Slide Power Point
3. สอนโดยใช้โปรแกรม Turbo C++ 4.5 เพื่อลงมือปฏิบัติในการเขียนโปรแกรมภาษาซี
4. สอนโดยให้ผู้เรียนเป็นส่วนกลาง โดยใช้วิธีถามตอบกับผู้เรียนระหว่างบรรยาย
5. ให้ผู้เรียนสรุปและอธิบายหัวข้อที่ผู้เรียนที่ได้รับมอบหมาย

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. Slide Power Point
3. โปรแกรม Turbo C++ 4.5
4. ตัวอย่างแบบทดสอบ

การวัดผลและประเมินผล

1. การทำตามตัวอย่างโจทย์คำสั่งและการทำแบบฝึกหัด
2. ความตั้งใจในชั้นเรียนและการร่วมมือในการนำเสนอหรืออภิปราย
3. สังเกตจากการตอบคำถามและการร่วมทำกิจกรรมในชั้นเรียน

บทที่ 8

โครงสร้างการควบคุมการทำงานแบบวนรอบ

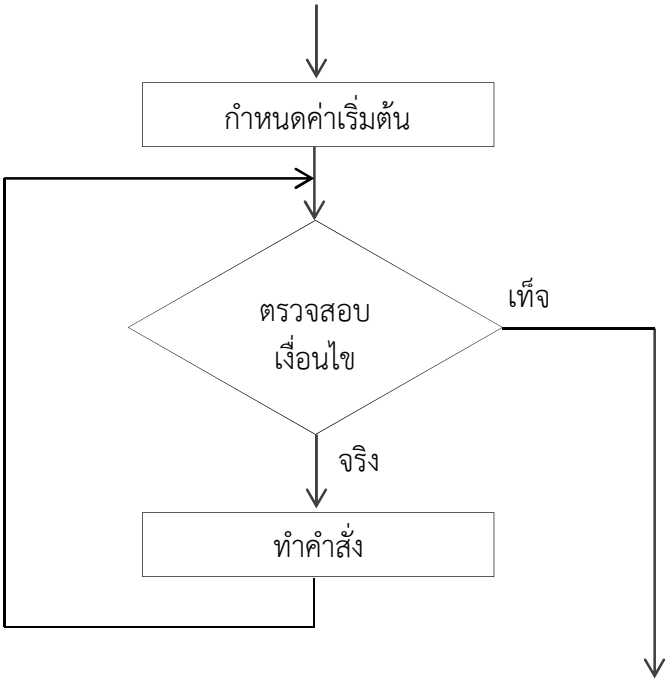
ในการพัฒนาหรือเขียนโปรแกรมคอมพิวเตอร์เพื่อให้โปรแกรมคอมพิวเตอร์ทำงานอะไรซ้ำ ๆ หลาย ๆ ครั้งคำตอบที่งานที่สุด คือ การเรียกใช้งานคำสั่งโปรแกรมที่เรียกว่า ลูป (loop) ซึ่งลูปมีหลายรูปแบบ ทั้ง for loop while loop และ do while และในภาษาซีการใช้งานคำสั่งวนลูปมีการใช้งานที่ง่าย และมีการกำหนดรอบทำซ้ำได้แน่นอนชัดเจน ดังนั้นในเนื้อบทนี้ผู้พัฒนาโปแกรมจะได้เรียนรู้คำสั่งวนลูป (loop) ต่าง ๆ ดังนี้

1.) คำสั่ง for loop
2.) คำสั่ง while loop
3.) คำสั่ง do while

8.1 คำสั่ง for loop

คำสั่ง for loop เป็นคำสั่งที่ใช้งานง่ายและมีการกำหนดรอบทำซ้ำที่แน่นอนว่าจะต้องทำซ้ำกี่รอบ อีกทั้งรูปแบบของ for loop จะเป็นรูปแบบเดียวกันกับภาษาที่พัฒนามาจากภาษาซี เช่น php, java เป็นต้น ซึ่งรูปแบบคำสั่ง for loop มีรูปแบบการใช้งานดังนี้

```
for (ค่าเริ่มต้น ; เงื่อนไขตรวจสอบ ; เพิ่มหรือลดจำนวน )
  {
        // ชุดคำสั่งกรณีที่เงื่อนไขตรวจสอบเป็นจริง
        คำสั่ง..... 1
        คำสั่ง..... 2
        คำสั่ง..... 3
        คำสั่ง..... 4
        คำสั่ง.....
        คำสั่ง..... n
  }
```



ภาพที่ 8.1 ผังงานแสดงการทำงานของคำสั่ง for loop

ตัวอย่างการใช้คำสั่ง for loop

ตัวอย่างที่ 8.1 ให้โปรแกรมแสดงตัวเลข 1 ถึง 10 โดยการใช้คำสั่ง for loop

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int i;
                for (i=1; i<=10; i++)
                {
                 printf("%d   ",i);
                }
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
1   2   3   4   5   6   7   8   9   10
```

ตัวอย่างที่ 8.2 ให้โปรแกรมรับค่าตัวเลขจำนวนเต็มจากผู้ใช้งาน 1 ค่า จากนั้นให้โปรแกรมแสดงจำนวนตัวเลขจาก 1 ถึง จำนวนที่ผู้ใช้ป้อนเข้าไป

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int i , num;
    loop1:      printf("\n\nEnter number : ");
                scanf("%d",&num);
                for (i=1; i<=num; i++)
                {
                 printf("%d   ",i);
                }
                goto loop1;
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter number : 10
1   2   3   4   5   6   7   8   9   10

Enter number : 5
1   2   3   4

Enter number : 1
1

Enter number : 0


Enter number : 3
1   2   3

Enter number :_
```

ตัวอย่างที่ 8.3 โปรแกรมแสดงผลเลขคู่เลขคี่ โดยมีเงื่อนไขคือหากผู้ใช้งานป้อนค่า ‘x’ โปรแกรมจะทำการแสดงเลขคู่จาก 0 - 20 และหากผู้ใช้งานป้อนค่า ‘y’ โปรแกรมจะทำการแสดงเลขคี่จาก 1 - 19 แต่หากผู้ใช้งานป้อนค่าอื่น ๆ โปรแกรมจะทำการแสดงคำว่า “Error” ออกมา

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
                int i;
                char ch;
    loop1:      printf("\n\nEnter number : ");
                ch = getch();
                if(ch == 'x')
                {       // ส่วนการแสดงเลขคู่
                        for (i=0; i<=20; i= i+2)
                        {
                        printf("%d   ",i);
                        }
                }

                else if(ch == 'y')
                {       // ส่วนการแสดงเลขคี่
                        for (i=1; i<=20; i= i+2)
                        {
                        printf("%d   ",i);
                        }
                }
                else
                {
                        printf("Error");
                }
                goto loop1;
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
// กรณีผู้ใช้งานป้อน ‘x’
Enter number : 0  2  4  6  8  10  12  14  16  18  20

// กรณีผู้ใช้งานป้อน ‘y’
Enter number : 1  3  5  7  9  11  13  15  17  19

// กรณีผู้ใช้งานป้อนค่าอื่น ๆ ที่ไม่ใช่ ‘x’ หรือ ‘y’
Enter number : Error

Enter number :
```

ตัวอย่างที่ 8.4 การใช้คำสั่ง for ในการเขียนโปรแกรม Bingo โดยรับค่าตัวเลขจำนวนเต็มจากผู้ใช้งาน 1 ค่า แล้วตรวจสอบค่าตัวเลขที่รับมีค่าเท่ากับ “10” หรือไม่ ถ้าเท่ากับ “10” ให้โปรแกรมแสดงคำว่า “Bingo” ออกมา และโปรแกรมสามารถบอกจำนวนครั้งที่ผู้ใช้งานป้อนไปกี่ครั้งถึงจะ Bingo ได้

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
        int i, x;
        for (i=1; i<=5000; i++)  //กำหนดค่าให้มากไว้เพราะไม่ทราบจำนวนลูปที่แน่นอน
          {
                printf("\n\nEnter number : ");
                scanf("%d",&x);
                if(x == 10)
                        {
                        printf("\nBingo\n");
                        printf("Number of times: %d\n",i);
                        i = 5001; // กำหนดค่าเพื่อให้ออกจาก for loop
                        }
          }
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter number : 1

Enter number : 2

Enter number : 3

Enter number : 4

Enter number : 5

Enter number : 6

Enter number : 7

Enter number : 8

Enter number : 9

Enter number : 10
Number of times: 10
```

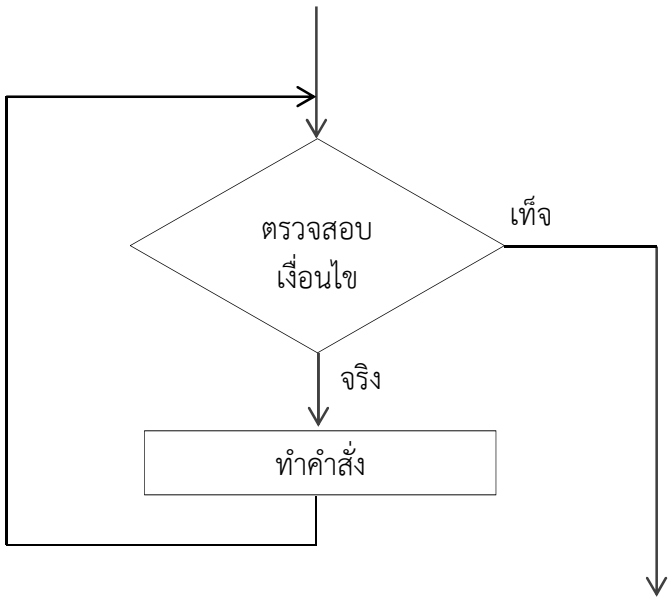
<u>หมายเหตุ</u>

จากตัวอย่างข้างต้นเป็นการนำคำสั่ง for loop มาใช้ในกรณีที่ไม่ทราบจำนวนลูปที่แน่นอน ซึ่งจริง ๆ แล้วหากผู้พัฒนาโปรแกรมไม่ทราบจำนวนลูปที่แน่นอน ส่วนใหญ่จะไม่นิยมนำคำสั่ง for loop มาใช้งาน เพราะคำสั่ง for loop เหมาะสำหรับวนลูปที่มีจำนวนลูปที่แน่นอน ดังนั้นหากจะวนลูปที่มีจำนวนลูปที่ไม่แน่นอน ผู้พัฒนาโปรแกรมส่วนใหญ่จะนิยมใช้คำสั่ง while เป็นคำสั่งในการทำงานในกรณีดังกล่าว

8.2 คำสั่ง while loop

คำสั่ง while loop เป็นคำสั่งที่ใช้ในกรณีที่ผู้พัฒนาโปรแกรมไม่ทราบจำนวนลูปที่แน่นอน ที่แน่นอนว่าจะต้องทำซ้ำกี่รอบ ลักษณะเด่นอย่างหนึ่งของคำสั่ง while คือ จะมีตรวจสอบเงื่อนไขก่อนว่าเป็นจริงหรือเท็จ ซึ่งในการตรวจสอบครั้งแรก ถ้าเงื่อนไขตรวจสอบเป็นเท็จ ก็จะไม่เข้าไปทำในวนรอบของการทำซ้ำเลย แต่ในทำนองเดียวกันถ้าตัวตรวจสอบเงื่อนไขเป็นจริงตลอด โปรแกรมจะทำซ้ำไปไม่สิ้นสุด ดังนั้นข้อสำคัญของการใช้คำสั่ง while loop คือ ผู้พัฒนาโปรแกรมจะต้องระมัดระวัง โดยควรเขียนโปรแกรมให้มีโอกาสออกจากการวนรอบทำซ้ำให้ได้ เพราะถ้าวนลูปค้างจะส่งผลให้โปรแกรมทำงานค้างหรือหยุดการทำงานไปเลยก็ได้ ซึ่งรูปแบบคำสั่ง while loop มีรูปแบบการใช้งานดังนี้

```
while (เงื่อนไขตรวจสอบ)
  {
        // ชุดคำสั่งกรณีที่เงื่อนไขตรวจสอบเป็นจริง
        คำสั่ง..... 1
        คำสั่ง..... 2
        คำสั่ง..... 3
        คำสั่ง..... 4
        คำสั่ง.....
        คำสั่ง..... n
  }
```



ภาพที่ 8.2 ผังงานแสดงการทำงานของคำสั่ง while loop

ตัวอย่างการใช้คำสั่ง while loop

ตัวอย่างที่ 8.5 การใช้คำสั่ง while loop ในการเขียนโปรแกรม Bingo โดยรับค่าตัวเลขจำนวนเต็มจากผู้ใช้งาน 1 ค่า แล้วตรวจสอบค่าตัวเลขที่รับมีค่าเท่ากับ “10” หรือไม่ ถ้าเท่ากับ “10” ให้โปรแกรมแสดงคำว่า “Bingo” ออกมา และโปรแกรมสามารถบอกจำนวนครั้งที่ผู้ใช้งานป้อนไปกี่ครั้งถึงจะ Bingo ได้ (เป็นโจทย์ตัวอย่างเดียวกับตัวอย่างที่ 8.4 แต่ตัวอย่างนี้จะใช้คำสั่ง while แทนคำสั่ง for)

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
 {
        int i , x , w;
        i = x = w =0;
                while(w == 0)
                 {
                        printf("\n\nEnter number : ");
                        scanf("%d",&x);
                        i++;
                        if(x == 5)
                         {
                                printf("\nBingo\n");
                                printf("Number of times: %d\n",i);
                                w = 1;
                         }
                 }
  }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter number : 7

Enter number : 8

Enter number : 9

Enter number : 10
Number of times: 4
```

ตัวอย่างที่ 8.6 การใช้คำสั่ง while loop ในการเขียนโปรแกรมรหัสลับแบบ 2 ชั้น โดยชั้นแรกผู้ใช้งานจะต้องป้อนรหัสคำว่า ‘A’ หากป้อนถูกโปรแกรมจะให้ไปถอดรหัสชั้นที่สอง โดยมีรหัสคำว่า ‘Z’ หากป้อนถูกทั้งสองชั้นโปรแกรมจะแสดงคำว่า “*** WINNING ***” ซึ่งในแต่ละชั้นผู้ใช้งานสามารถป้อนกี่ครั้งก็ได้จนกว่าจะป้อนตรงรหัสลับที่ตั้งไว้

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
 {
        int i=0 , x=0;
        char ch;
        // ตรวจสอบรหัสลับชั้นที่ 1
        while(i == 0)
                {
                        printf("\n\nEnter Text (1) : ");
                        scanf("%s",&ch);
                        x++;
                        if (ch == 'A')
                                {
                                  i = 1;
                                }
                  }
                        printf("\n*** Pass (1) ***\n");
                        printf("Number of times: %d\n",x);
                        x = 0;
        // ตรวจสอบรหัสลับชั้นที่ 2
        while(i == 1)
                {
                        printf("\n\nEnter Text (2) : ");
                        scanf("%s",&ch);
                        x++;
                        if (ch == 'Z')
                                {
                                   i = 2;
                                }
                }
                        printf("\n*** WINNING ***");
                        printf("Number of times: %d\n",x);
 }
```

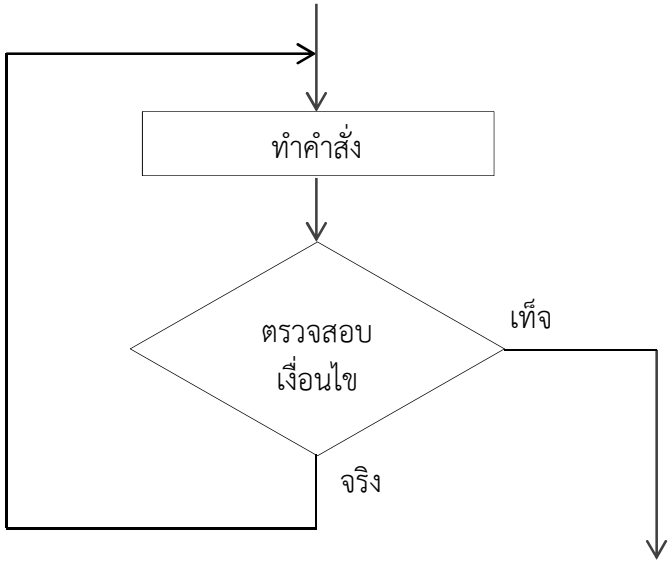
ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter Text (1) : a

Enter Text (1) : b

Enter Text (1) : c

Enter Text (1) : d

Enter Text (1) : x

Enter Text (1) : V

Enter Text (1) : Z

Enter Text (1) : A

*** Pass (1) ***
Number of times: 8

Enter Text (2) : F

Enter Text (2) : D

Enter Text (2) : z

Enter Text (2) : Z

*** WINNING ***
"Number of times: 4
```

8.3 คำสั่ง do while

คำสั่ง do while เป็นคำสั่งให้มีการทำซ้ำเป็นลูป(loop) โดยมีลักษณะการทำงานทำนองเดียวกับคำสั่ง while ซึ่งจะแตกต่างกันตรงที่คำสั่ง do while นี้จะมีการทำงานตามคำสั่งไป 1 รอบก่อนที่จะทดสอบเงื่อนไข ถ้าเงื่อนไขที่เป็นจริงจะทำงานต่อไป ถ้าเงื่อนไขเป็นเท็จจึงจะออกจากคำสั่ง ข้อควรระวังของคำสั่ง do while ก็เช่นเดียวกันกับคำสั่ง while ซึ่งผู้พัฒนาจะต้องกำหนดการวนลูปให้มีโอกาสที่เงื่อนสามารถหลุดออกจากลูปได้ มิฉะนั้นจะมีปัญหาที่โปรแกรมทำงานแบบวนซ้ำแบบไม่มีที่สิ้นสุด ซึ่งรูปแบบคำสั่ง do while มีรูปแบบการใช้งานดังนี้

```
do
  {
        คำสั่ง..... 1
        คำสั่ง..... 2
        คำสั่ง..... 3
        คำสั่ง..... 4
        คำสั่ง.....
        คำสั่ง..... n
  } while (เงื่อนไขตรวจสอบ);
```



ภาพที่ 8.3 ผังงานแสดงการทำงานของคำสั่ง do while

ตัวอย่างการใช้คำสั่ง do while

ตัวอย่างที่ 8.7 การใช้คำสั่ง do loop เพื่อตรวจสอบรหัสลับ โดยรหัสลับที่ตั้งไว้คือ ‘9’ ดังนั้นหากผู้ใช้งานป้อนรหัสตรงกับ ‘9’ โปรแกรมจะทำการแสดงคำว่า “Bingo”

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
  {
  int i=0;
  char ch;
                    do {
                              printf("\nEnter Code : ");
                              ch = getch();
                              putchar(ch);
                               if (ch == '9')
                                        {
                                            i = 1;
                                        }
                          }
                    while (i == 0);
          printf("\n\n+++ Bingo +++");
    }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter Code : a

Enter Code : 8

Enter Code : c

Enter Code : 9


+++ Bingo +++
```

ตัวอย่างที่ 8.8 การใช้คำสั่ง do loop โดยให้โปรแกรมรับค่าตัวเลขจำนวนเต็มจากผู้ใช้งาน 1 ค่า จากนั้นให้โปรแกรมแสดงจำนวนตัวเลขจาก 1 ถึง จำนวนที่ผู้ใช้ป้อนเข้าไป

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
 {
   int i,x;
loop1:  i=1;
   x=0;
   printf("\n\nEnter Number :  ");
   scanf("%d",&x);
        do      {
                  printf("%d   ",i);
                  i++;
                }
        while(i<=x);
        goto loop1;
 }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter Number :  5
1  2  3  4  5

Enter Number :  8
1  2  3  4  5  6  7  8

Enter Number :  11
1  2  3  4  5  6  7  8  9  10  11

Enter Number :  3
1  2  3

Enter Number : _
```

สรุปท้ายบท

ในการพัฒนาหรือเขียนโปรแกรมคอมพิวเตอร์เพื่อให้โปรแกรมคอมพิวเตอร์ทำงานอะไรซ้ำ ๆ หลาย ๆ ครั้งคำตอบที่งานที่สุด คือ การเรียกใช้งานคำสั่งโปรแกรมที่เรียกว่า ลูป (loop) ซึ่งลูปมีหลายรูปแบบ ทั้ง for loop while loop และ do while และในภาษาซีการใช้งานคำสั่งวนลูปมีการใช้งานที่ง่าย และมีการกำหนดรอบทำซ้ำได้แน่นอนชัดเจน โดยคำสั่งวนลูปในภาษามีคำสั่งในการทำงานของโปรแกรม ดังนี้

คำสั่ง for loop

คำสั่ง for loop เป็นคำสั่งที่ใช้งานง่ายและมีการกำหนดรอบทำซ้ำที่แน่นอนว่าจะต้องทำซ้ำกี่รอบ อีกทั้งรูปแบบของ for loop จะเป็นรูปแบบที่เข้าใจง่าย และแก้ไขเงื่อนไขต่าง ๆ ได้สะดวก

คำสั่ง while loop

คำสั่ง while loop เป็นคำสั่งที่ใช้ในกรณีที่ผู้พัฒนาโปรแกรมไม่ทราบจำนวนลูปที่แน่นอนที่แน่นอนว่าจะต้องทำซ้ำกี่รอบ ลักษณะเด่นอย่างหนึ่งของคำสั่ง while คือ จะมีตรวจสอบเงื่อนไขก่อนว่าเป็นจริงหรือเท็จ ซึ่งในการตรวจสอบครั้งแรก ถ้าเงื่อนไขตรวจสอบเป็นเท็จ ก็จะไม่เข้าไปทำในวนรอบของการทำซ้ำเลย แต่ในทำนองเดียวกันถ้าตัวตรวจสอบเงื่อนไขเป็นจริงตลอด โปรแกรมจะทำซ้ำไปไม่สิ้นสุด ดังนั้นข้อสำคัญของการใช้คำสั่ง while loop คือ ผู้พัฒนาโปรแกรมจะต้องระมัดระวัง โดยควรเขียนโปรแกรมให้มีโอกาสออกจากการวนรอบทำซ้ำให้ได้ เพราะถ้าวนลูปค้างจะส่งผลให้โปรแกรมทำงานค้างหรือหยุดการทำงานไปเลยก็ได้

คำสั่ง do while

คำสั่ง do while เป็นคำสั่งให้มีการทำซ้ำเป็นลูป(loop) โดยมีลักษณะการทำงานทำนองเดียวกับคำสั่ง while ซึ่งจะแตกต่างกันตรงที่คำสั่ง do while นี้จะมีการทำงานตามคำสั่งไป 1 รอบก่อนที่จะทดสอบเงื่อนไข ถ้าเงื่อนไขที่เป็นจริงจะทำงานต่อไป ถ้าเงื่อนไขเป็นเท็จจึงจะออกจากคำสั่ง ข้อควรระวังของคำสั่ง do while ก็เช่นเดียวกันกับคำสั่ง while ซึ่งผู้พัฒนาจะต้องกำหนดการวนลูปให้มีโอกาสที่เงื่อนสามารถหลุดออกจากลูปได้ มิฉะนั้นจะมีปัญหาที่โปรแกรมทำงานแบบวนซ้ำแบบไม่มีที่สิ้นสุด

คำถามทบทวน

1. จงเขียนโปรแกรมรวมค่าตัวเลขที่ผู้ใช้งานป้อนเข้ามา โดยผู้ใช้งานสามารถป้อนได้ 10 ตัวเลข
2. จงเขียนโปรแกรมแสดงรับค่าตัวเลขจำนวนเต็มจากผู้ใช้งาน 6 ค่า แล้วให้โปรแกรมแสดงค่าตัวเลขออกเป็นกลุ่มโดยแยกเป็นเฉพาะกลุ่มเลขคู่ และเฉพาะกลุ่มเลขคี่
3. จงเขียนโปรแกรมตรวจรหัส 3 หลัก โดยเงื่อน คือ จะต้องตรวจสอบรหัสในหลักนั้น ๆ ให้ผ่านก่อนจึงจะข้ามไปตรวจสอบรหัสในหลักต่อไปได้

โดย หลักที่หนึ่งรหัสตัวเลข 9
หลักที่สองรหัสตัวเลข 8
หลักที่สามรหัสตัวเลข 7

4. จงเขียนโปรแกรมตรวจรหัส 3 หลัก โดยเงื่อน คือ จะต้องตรวจสอบรหัสในหลักนั้น ๆ ให้ผ่านก่อนจึงจะข้ามไปตรวจสอบรหัสในหลักต่อไปได้ พร้อมสรุปผลจำนวนครั้งทั้งหมดว่าใส่รหัสไปทั้งหมดกี่ครั้งจึงจะสามารถถอดรหัสผ่านดังกล่าวได้

โดย หลักที่หนึ่งรหัสตัวเลข 11
หลักที่สองรหัสตัวอักขระ A
หลักที่สามรหัสตัวอักขระ z

5. จงเขียนโปรแกรมโดยให้ผู้ใช้งานคนที่หนึ่งสามารถตั้งค่ารหัสผ่านได้ 1 ค่า จากนั้นให้โปรแกรมทำการรับค่าจากผู้ใช้งานคนที่สอง เพื่อทำการถอดรหัสผ่านดังกล่าว และหากใส่รหัสผ่านสำเร็จให้แสดงคำว่า “excellently” พร้อมสรุปผลจำนวนครั้งทั้งหมดว่าใส่รหัสไปทั้งหมดกี่ครั้งจึงจะสามารถถอดรหัสผ่านดังกล่าวได้

เอกสารอ้างอิง

คะชา ชาญศิลป์. (2548). **ภาษาซีสำหรับผู้เริ่มต้น.** กรุงเทพมหานคร: วิรัตน์เอ็ดดูเคชั่น.

ประภาพร ช่างไม้. (2545). **คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.** กรุงเทพมหานคร: อินโฟเพรส.

ธีรวัฒน์ ประกอบผล. (2553). **คู่มือการเขียนโปรแกรมภาษา C.** กรุงเทพมหานคร: ซิมพลิฟลาย.

สุระสิทธิ์ ทรงม้า และภูริพจน์ แก้วย่อง. (2552). **การโปรแกรมคอมพิวเตอร์.**: มหาวิทยาลัยราชภัฏสวนดุสิต.

สมชาย รัตนเลิศนุสรณ์. (2553). **การเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี.** กรุงเทพมหานคร: สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) ส.ส.ท.

สานนท์ เจริญฉาย. (2543). **การเขียนโปรแกรมและอัลกอริทึม.** กรุงเทพมหานคร: มหาจุฬาลงกรณ์ราชวิทยาลัย.

อรพิน ประวัติบริสุทธิ์. (2554). **คู่มือการเรียนภาษา C ฉบับปรับปรุงใหม่** (พิมพ์ครั้งที่ 10)**.** กรุงเทพมหานคร: โปรวิชั่น.

http://www.61.7.214.35/c/webbase/unit8/while.php

http://www.mwit.ac.th/~cs/download/tech30101/TECH30101_ch9.html

http://www.e-learning.snru.ac.th/els/program1/lesson2/page6_1.html

http://www.krubpk.com/com_2/pan7/pan8_2.html

บทที่ 9

ตัวแปรชนิดอาร์เรย์

แผนการสอนประจำบท

หัวข้อเนื้อหา

9.1 ตัวแปรอาร์เรย์แบบ 1 มิติ

9.2 ตัวแปรอาร์เรย์แบบ 2 มิติ

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถเข้าใจหลักการทำงานของตัวแปรอาร์เรย์แบบ 1 มิติได้อย่างเข้าใจและถูกต้อง
2. ผู้เรียนสามารถประกาศตัวแปรอาร์เรย์แบบ 1 มิติ ได้อย่างถูกต้องและสามารถนำตัวแปรอาร์เรย์แบบ 1 มิติไปใช้งานในการเขียนโปรแกรมในกรณีต่าง ๆ ได้อย่างถูกต้อง
3. ผู้เรียนสามารถเข้าใจหลักการทำงานของตัวแปรอาร์เรย์แบบ 2 มิติได้อย่างเข้าใจและถูกต้อง
4. ผู้เรียนสามารถประกาศตัวแปรอาร์เรย์แบบ 2 มิติ ได้อย่างถูกต้องและสามารถนำตัวแปรอาร์เรย์แบบ 2 มิติไปใช้งานในการเขียนโปรแกรมในกรณีต่าง ๆ ได้อย่างถูกต้อง

วิธีการสอน

1. สอนแบบบรรยายโดยใช้เอกสารประกอบการสอน
2. สอนแบบบรรยายโดยใช้ Slide Power Point
3. สอนโดยใช้โปรแกรม Turbo C++ 4.5 เพื่อลงมือปฏิบัติในการเขียนโปรแกรมภาษาซี
4. สอนโดยให้ผู้เรียนเป็นส่วนกลาง โดยใช้วิธีถามตอบกับผู้เรียนระหว่างบรรยาย
5. ให้ผู้เรียนสรุปและอธิบายหัวข้อที่ผู้เรียนที่ได้รับมอบหมาย

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. Slide Power Point
3. โปรแกรม Turbo C++ 4.5
4. ตัวอย่างแบบทดสอบ

การวัดผลและประเมินผล

1. การทำตามตัวอย่างโจทย์คำสั่งและการทำแบบฝึกหัด
2. ความตั้งใจในชั้นเรียนและการร่วมมือในการนำเสนอหรืออภิปราย
3. สังเกตจากการตอบคำถามและการร่วมทำกิจกรรมในชั้นเรียน

บทที่ 9

ตัวแปรชนิดอาร์เรย์

อาร์เรย์ (Arrays) เป็นโครงสร้างข้อมูลของกลุ่มสมาชิกที่ใช้เก็บข้อมูลชนิดเดียวกัน แล้วมาจัดเรียงกันอย่างต่อเนื่อง โดยข้อมูลชนิดเดียวกัน คือ ข้อมูลทุกตัวที่อยู่ในอาร์เรย์จะต้องเป็นข้อมูลชนิดเดียวกันเท่านั้น เช่น ถ้าเป็นอาร์เรย์ชนิดจำนวนเต็ม ข้อมูลทุกตัวในอาร์เรย์ก็ต้องเป็นชนิดจำนวนเต็มไม่สามารถเก็บข้อมูลต่างชนิดกันได้ โดยอาร์เรย์จะจัดเก็บชุดข้อมูลในหน่วยความจำในแบบต่อเนื่องกันไป ซึ่งแตกต่างจากตัวแปรธรรมดา โดยข้อมูลแต่ละตัวจะมีลำดับของสมาชิก เรียกว่า element ซึ่งแต่ละ element จะมีอินเด็กซ์ (index) เป็นตัวชี้ตำแหน่งของข้อมูลในหน่วยความจำ โดยที่อินเด็กซ์ในตำแหน่งแรกจะเริ่มจากศูนย์ (0)

9.1 ตัวแปรอาร์เรย์แบบ 1 มิติ

ตัวแปรอาร์เรย์แบบ 1 มิติ จะเป็นการเก็บข้อมูลต่อเนื่องกันไปเป็นแถว การประกาศชื่อตัวแปรอาร์เรย์แบบ 1 มิตจะใช้ชื่อเพียงชื่อเดียวตามด้วยเครื่องหมาย [ ] คร่อมตัวเลขที่บอกถึงจำนวนของข้อมูลที่ต้องการจองพื้นที่เอาไว้

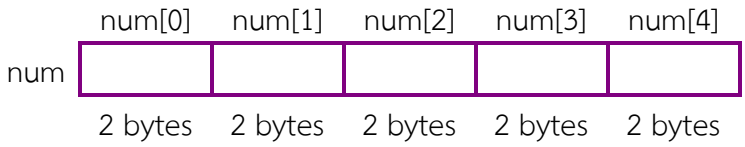
รูปแบบการประกาศตัวแปร

ชนิดข้อมูล ชื่อตัวแปร [ขนาดของอาร์เรย์];

ตัวอย่างการประกาศตัวแปรอาร์เรย์แบบ 1 มิติ (แบบจองพื้นที่อย่างเดียว)

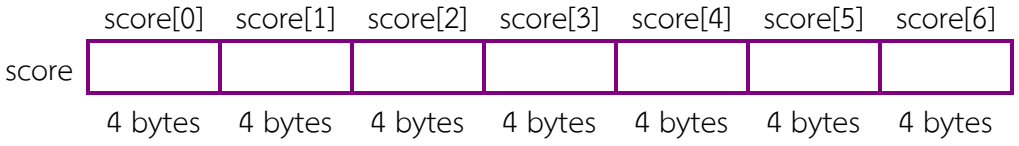
ประกาศตัวแปรอาร์เรย์ชนิด int

```
int  num[5];
```



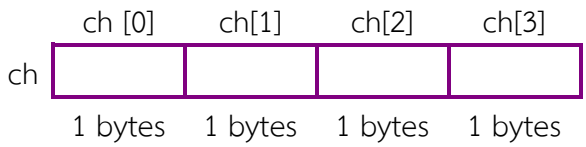
ประกาศตัวแปรอาร์เรย์ชนิด float

```
float  score[7];
```



ประกาศตัวแปรอาร์เรย์ชนิด char

char ch[4];

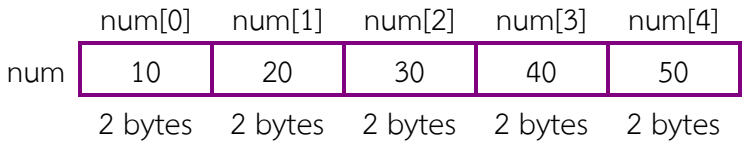


การประกาศตัวแปรอาร์เรย์นอกจากจะประกาศแบบจองพื้นที่เพียงอย่างเดียว ผู้พัฒนาโปรแกรมสามารถประกาศตัวแปรแบบอาร์เรย์พร้อมทั้งกำหนดค่าเริ่มต้นให้กับอาร์เรย์แบบ 1 มิติได้ โดยการกำหนดค่าเริ่มต้นให้กับ array ได้ตั้งแต่ตอนประกาศตัวแปรค่าที่กำหนดต้องอยู่ในเครื่องหมาย { } และถ้ามีมากกว่า 1 ค่า ต้องแยกจากกันด้วยเครื่องหมาย , (comma)

ตัวอย่างการประกาศตัวแปรอาร์เรย์แบบ 1 มิติ (แบบกำหนดค่าในพื้นที่)

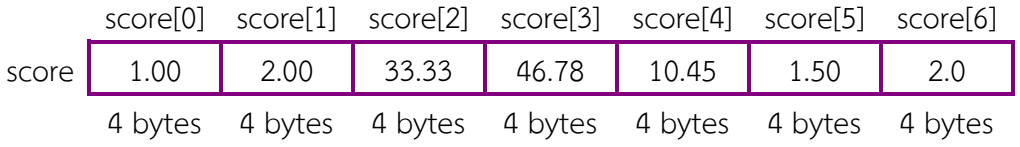
ประกาศตัวแปรอาร์เรย์ชนิด int

int num[5] = { 10,20,30,40,50 } ;



ประกาศตัวแปรอาร์เรย์ชนิด float

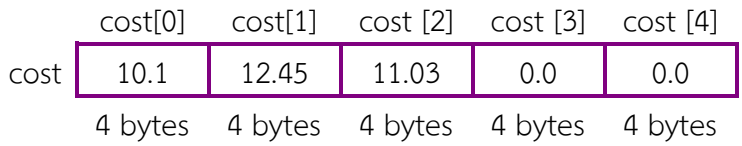
float score[7] = { 1.00,2.00,33.33, 46.78,10.45,1.50,2.0 } ;



ถ้าในตอนประกาศตัวแปรอาร์เรย์ไม่กำหนดค่าเริ่มต้นให้กับมันแล้ว ค่าที่อยู่ในตัวแปรจะเป็นค่าที่ค้างอยู่ในหน่วยความจำช่วงที่เราจองไว้เป็นอาร์เรย์นั้น

ถ้ากำหนดค่าเริ่มต้นตั้งแต่ตอนประกาศตัวแปรแต่กำหนดไม่ครบ ในกรณีที่เป็นอาร์เรย์แบบตัวเลขทั้งจำนวนเต็มและจำนวนจริง ค่าที่เหลือจะถูกกำหนดเป็น 0 โดยอัตโนมัติ

เช่น float cost[5] = {10.1,12.45,11.3} ;



บางครั้งถ้ากำหนดค่าเริ่มต้นให้แก่อาร์เรย์เลย เราไม่จำเป็นต้องใส่ขนาดของอาร์เรย์ก็ได้

เช่น int num[ ] = {1,2,3,4,5} ;

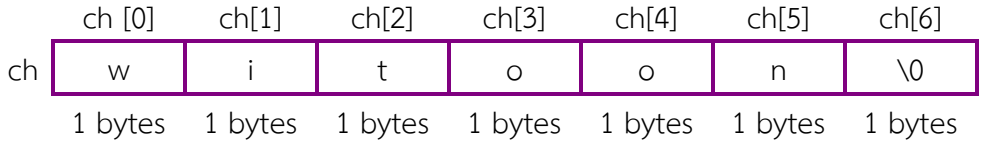
ความหมายคือ เป็นการกำหนดตัวแปรอาร์เรย์ของจำนวนจริงแบบ float ขนาด 5 ช่อง แต่ถ้ากำหนดตัวแปรอาร์เรย์โดยไม่ใส่ขนาดของอาร์เรย์ และไม่ได้กำหนดค่าเริ่มต้นให้กับมัน

เช่น int num[ ] ; // เป็นการประกาศตัวแปรแบบอาร์เรย์ที่ผิด

วิธีการแบบนี้ไม่สามารถประกาศตัวแปรอาร์เรย์โดยไม่ใส่ขนาดของอาร์เรย์ได้ ยกเว้นมีการกำหนดค่าเริ่มต้นให้กับมันตั้งแต่แรก

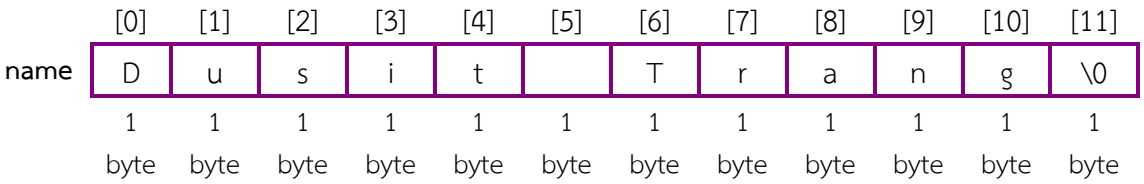
ประกาศตัวแปรอาร์เรย์ชนิด char

char ch[7] = “witoon” ;



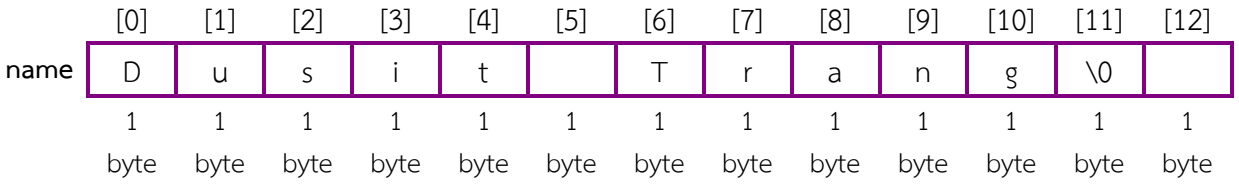
ในการประกาศตัวแปรแบบอาร์เรย์ชนิด char โดยที่ตำแหน่ง สุดท้ายของข้อมูลระบบจะเก็บค่า \0 ไว้อัตโนมัติ เพื่อแสดงการสิ้นสุดข้อความ

char name[ ] = “Dusit Trang”; จะได้



โดยที่ตำแหน่ง name[11] จะเก็บค่า \0 ไว้อัตโนมัติ เพื่อแสดงการสิ้นสุดข้อความ

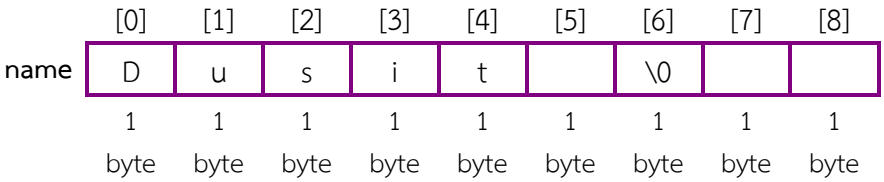
หรือ char name[13] = “Dusit Trang”; จะได้



โดยที่ตำแหน่ง name[11] จะเก็บค่า \0 ไว้อัตโนมัติ เพื่อแสดงการสิ้นสุดข้อความเช่นเดียวกัน และตำแหน่งที่ 12 จะว่าง

หรือถ้าต้องการกำหนดค่าให้กับอาร์เรย์เป็นอักขระสามารถทำได้ดังนี้

char name[9] = { ‘D’, ‘u’, ‘s’, ‘i’, ‘t’, ‘ ’, } ; จะได้



โดยที่ตำแหน่ง name[6] จะเก็บค่า \0 ไว้อัตโนมัติ เพื่อแสดงการสิ้นสุดข้อความ เช่นเดียวกัน และตำแหน่งที่ 7 และ 8 จะว่างเนื่องจากไม่มีข้อมูลใด ๆ

ดังนั้นในการประกาศให้ตัวแปรที่เป็นสตริง จะต้องคำนึงถึงจำนวนตัวอักขระที่ต้องการจัดเก็บด้วย เช่น หากต้องการเก็บชื่อซึ่งมีความยาวไม่เกิน 20 ตัวอักษร จะต้องประกาศตัวแปรให้เป็นอาร์เรย์ขนาด 21 ช่อง หรือมากกว่าเล็กน้อย เพื่อมีพื้นที่สำหรับเก็บค่า \0 หรือค่า null)

ตัวอย่าง การทดลองการประกาศตัวแปรอาร์เรย์แบบ 1 มิติ

ตัวอย่างที่ 9.1 โปรแกรมแสดงตัวเลขที่ได้เก็บไว้ในตัวแปรอาร์เรย์แบบ 1 มิติ

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
      {
         int n[5] = { 5, 4 , 3, 2, 1 } ;
         int  i ;
            for ( i = 0; i < 5; i++ )
             {
                  printf( "%d   ",n[ i ] );

             }
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
5   4   3   2   1
```

ตัวอย่างที่ 9.2 โปรแกรมแสดงอักขระที่ได้เก็บไว้ในตัวแปรอาร์เรย์แบบ 1 มิติ

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )

    {
        char ch[ ] = "witoon";
         int i;
         for ( i = 0; i < 6; i++ )
         {
                  printf( "%c  ",ch[ i ] );
         }
     }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
w  i  t  o  o  n
```

ตัวอย่างที่ 9.3 โปรแกรมแสดงอักขระที่ได้เก็บไว้ในตัวแปรอาร์เรย์แบบ 1 มิติ พร้อมแสดงผลตำแหน่งในอาร์เรย์แต่ละตัวที่ได้จัดเก็บ

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
    {
        char ch[ ] = "Dusit Trang";
        int i;
        for ( i = 0; i < 11; i++ )
        {
            printf( "\n ch[%d] = %c ", i , ch[i] );
        }
     }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
ch[0] = D
ch[1] = u
ch[2] = s
ch[3] = i
ch[4] = t
ch[5] =
ch[6] = T
ch[7] = r
ch[8] = a
ch[9] = n
ch[10] = g
```

ตัวอย่างที่ 9.4 โปรแกรมแสดงการเก็บค่าตัวเลขไว้ในตัวแปรอาร์เรย์แบบ 1 มิติ และเมื่อเก็บครบตามจำนวน โปรแกรมจะทำการแสดงผลข้อมูลตามตำแหน่งในอาร์เรย์แต่ละตัวที่ได้จัดเก็บ

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
    {
        int x[10];
        int i;
        for ( i = 0; i < 10; i++ )
        {
                printf( "Enter Number : " );
                scanf("%d",&x[i]);
        }
        for ( i = 0; i < 10; i++ )
        {
                printf( "\n x[%d] = %d ", i , x[i] );
        }
     }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter Number : 9
Enter Number : 8
Enter Number : 7
Enter Number : 6
Enter Number : 5
Enter Number : 4
Enter Number : 3
Enter Number : 2
Enter Number : 1
Enter Number : 0
x[0] = 9
x[1] = 8
x[2] = 7
x[3] = 6
x[4] = 5
x[5] = 4
x[6] = 3
x[7] = 2
x[8] = 1
x[9] = 0
```

9.2 ตัวแปรอาร์เรย์แบบ 2 มิติ

ตัวแปรอาร์เรย์แบบ 2 มิติ จะเป็นการเก็บข้อมูลในแนวตั้งและแนวนอน หรือแนวแถวและคอลัมน์ ซึ่งการนำตัวแปรอาร์เรย์ 2 มิติมาใช้งานจะเหมาะสำหรับการเก็บข้อมูลในบางประเภท เช่น การเก็บคะแนนนักศึกษา การเก็บข้อมูลประวัติ เป็นต้น ดังแสดงไว้ตัวอย่างข้างล่าง

	วิชา A	วิชา B	วิชา C	วิชา D	วิชา E
นศ.คนที่ 1					
นศ.คนที่ 2					
นศ.คนที่ 3					
นศ.คนที่ 4					
นศ.คนที่ 5					
นศ.คนที่ 6					
นศ.คนที่ 7					
นศ.คนที่ 8					
นศ.คนที่ 9					
นศ.คนที่ 10					

รูปแบบการประกาศตัวแปร

ชนิดข้อมูล ชื่อตัวแปร [row][column];

ตัวอย่างการประกาศตัวแปรอาร์เรย์แบบ 2 มิติ (แบบจองพื้นที่อย่างเดียว)

ประกาศตัวแปรอาร์เรย์ชนิด int

```
int  b[5][4];
```

Column (คอลัมน์)

Row (แถว)

b	[0]	[1]	[2]	[3]
[0]	b[0][0]	b[0][1]	b[0][2]	b[0][3]
[1]	b[1][0]	b[1][1]	b[1][2]	b[1][3]
[2]	b[2][0]	b[2][1]	b[2][2]	b[2][3]
[3]	b[3][0]	b[3][1]	b[3][2]	b[3][3]
[4]	b[4][0]	b[4][1]	b[4][2]	b[4][3]

ประกาศตัวแปรอาร์เรย์ชนิด float

int float cost[10][5];

Column (คอลัมน์)

cost	[0]	[1]	[2]	[3]	[4]
[0]	[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1]	[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2]	[2][0]	[2][1]	[2][2]	[2][3]	[2][4]
[3]	[3][0]	[3][1]	[3][2]	[3][3]	[3][4]
[4]	[4[0]	[4][1]	[4][2]	[4][3]	[4][4]
[5]	[5][0]	[5][1]	[5][2]	[5][3]	[5][4]
[6]	[6][0]	[1][1]	[1][2]	[1][3]	[1][3]
[7]	[7][0]	[7][1]	[7][2]	[7][3]	[7][4]
[8]	[8][0]	[8][1]	[8][2]	[8][3]	[8][4]
[9]	[9][0]	[9][1]	[9][2]	[9][3]	[9][4]

Row (แถว)

ประกาศตัวแปรอาร์เรย์ชนิด char

int char name[15][2];

Column (คอลัมน์)

name	[0]	[1]
[0]	name [0][0]	name [0][1]
[1]	name [1][0]	name [1][1]
[2]	name [2][0]	name [2][1]
[3]	name [3][0]	name [3][1]
[4]	name [4[0]	name [4][1]
[5]	name [5][0]	name [5][1]
[6]	name [6][0]	name [1][1]
[7]	name [7][0]	name [7][1]
[8]	name [8][0]	name [8][1]
[9]	name [9][0]	name [9][1]
[10]	name [10][0]	name [10][1]
[11]	name [11][0]	name [11][1]
[12]	name [12][0]	name [12][1]
[13]	name [13][0]	name [13][1]
[14]	name [14][0]	name [14][1]

Row (แถว)

ตัวอย่างการประกาศตัวแปรอาร์เรย์แบบ 2 มิติ (แบบกำหนดค่าในพื้นที่)

ประกาศตัวแปรอาร์เรย์ชนิด int

```
int  num[2][4] = { {1,2,3,4},
                   {5,6,7,8} };
```

Column

Row / num	[0]	[1]	[2]	[3]
[0]	1	2	3	4
[1]	5	6	7	8

ประกาศตัวแปรอาร์เรย์ชนิด float

```
float  cost[5][5] ={ {1.0,2.0,3.0,4.0,5.0},
                     {11.1,22.22,33.33,44.44,55.55},
                     {0,0,3.3,4.4,5.5},
                     {0,0,0,0,0} };
```

Column

Row / cost	[0]	[1]	[2]	[3]	[4]
[0]	1.0	2.0	3.0	4.0	5.0
[1]	11.1	22.22	33.33	44.44	55.55
[3]	0	0	3.3	4.4	5.5
[4]	0	0	0	0	0

ประกาศตัวแปรอาร์เรย์ชนิด char

```
char ch[6][5] ={ {'W','i','t','o','o'},
                 {'n', '  ' ,'D','u','s'},
                 {'I','t'} };
```

Column

Row / ch	[0]	[1]	[2]	[3]	[4]
[0]	W	i	t	o	o
[1]	n		D	u	s
[3]	i	t			
[4]					
[5]					

ตัวอย่าง การทดลองการประกาศตัวแปรอาร์เรย์แบบ 2 มิติ

ตัวอย่างที่ 9.5 โปรแกรมแสดงตัวเลขที่ได้เก็บไว้ในตัวแปรอาร์เรย์แบบ 2 มิติ

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
          int i , j ;
          int num[3][3] = { { 1,2,3 } ,
                            { 4,5,6 } ,
                            { 7,8,9 } } ;
          for ( j = 0 ;  j <= 2 ; j++ )
          {
                  for ( i = 0 ;  i <= 2 ; i++ )
                          {
                                  printf( " %d \t " , num[i][j] ) ;
                          }
                  printf( "\n" ) ;
          }
          }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
1       2       3

4       5       6

7       8       9
```

ตัวอย่างที่ 9.6 โปรแกรมแสดงตัวอักขระที่ได้เก็บไว้ในตัวแปรอาร์เรย์แบบ 2 มิติ พร้อมทั้งแสดงตำแหน่งอาร์เรย์ที่จัดเก็บ

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
          char ch[2][4] ={ {'W','i','t','o'},
                                {'0','n',} };
          for ( j = 0 ;  j <= 1 ; j++ )
            {
                    for ( i = 0 ;  i <= 3 ; i++ )
                    {
                      printf( " ch[%d][%d] = %c \t " , j,i,ch[j][i] ) ;
                    }
                    printf( "\n" ) ;
             }
          }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
ch[0][0] = W    ch[0][1] = i      ch[0][2] = t      ch[0][3] = o

ch[1][0] = o    ch[1][1] = n      ch[1][2] =        ch[1][3] =
```

ตัวอย่างที่ 9.7 โปรแกรมแสดงตัวอักขระที่ได้เก็บไว้ในตัวแปรอาร์เรย์แบบ 2 มิติ พร้อมทั้งแสดงตำแหน่งอาร์เรย์ที่จัดเก็บ

```
// โค้ดโปรแกรม
#include "stdio.h"
main( )
        {
          int i , j ;
          char ch[3][3];
          for ( j = 0 ;  j <= 2 ; j++ )
                    {
                    for ( i = 0 ;  i <= 2 ; i++ )
                              {
                               printf("\n\n Enter Character :" );
                               ch[j][i] = getch();
                               putchar(ch[j][i]);
                              }
                    }
          printf( "\n" ) ;
          for ( j = 0 ;  j <= 2 ; j++ )
                    {
                    for ( i = 0 ;  i <= 2 ; i++ )
                              {
                      printf( " ch[%d][%d] = %c \t " , j , i , ch[j][i] );
                              }
                    printf( "\n" ) ;
                    }
          }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter Character : A
Enter Character : B
Enter Character : C
Enter Character : 1
Enter Character : 2
Enter Character : 3
Enter Character : x
Enter Character : y
Enter Character : z
ch[0][0] = A        ch[0][1] = B        ch[0][2] = C
ch[1][0] = 1        ch[1][1] = 2        ch[1][2] = 3
ch[2][0] = x        ch[2][1] = y        ch[2][2] = z
```

สรุปท้ายบท

อาร์เรย์ (Arrays) เป็นโครงสร้างข้อมูลของกลุ่มสมาชิกที่ใช้เก็บข้อมูลชนิดเดียวกัน แล้วมาจัดเรียงกันอย่างต่อเนื่อง โดยข้อมูลชนิดเดียวกัน คือ ข้อมูลทุกตัวที่อยู่ในอาร์เรย์จะต้องเป็นข้อมูลชนิดเดียวกันเท่านั้น เช่น ถ้าเป็นอาร์เรย์ชนิดจำนวนเต็ม ข้อมูลทุกตัวในอาร์เรย์ก็ต้องเป็นชนิดจำนวนเต็มไม่สามารถเก็บข้อมูลต่างชนิดกันได้ โดยอาร์เรย์จะจัดเก็บชุดข้อมูลในหน่วยความจำในแบบต่อเนื่องกันไป ซึ่งแตกต่างจากตัวแปรธรรมดา โดยข้อมูลแต่ละตัวจะมีลำดับของสมาชิก เรียกว่า element ซึ่งแต่ละ element จะมีอินเด็กซ์ (index) เป็นตัวชี้ตำแหน่งของข้อมูลในหน่วยความจำ โดยที่อินเด็กซ์ในตำแหน่งแรกจะเริ่มจากศูนย์ (0)

1. ตัวแปรอาร์เรย์แบบ 1 มิติ

ตัวแปรอาร์เรย์แบบ 1 มิติ จะเป็นการเก็บข้อมูลต่อเนื่องกันไปเป็นแถว การประกาศชื่อตัวแปรอาร์เรย์แบบ 1 มิตจะใช้ชื่อเพียงชื่อเดียวตามด้วยเครื่องหมาย [ ] คร่อมตัวเลขที่บอกถึงจำนวนของข้อมูลที่ต้องการจองพื้นที่เอาไว้

รูปแบบการประกาศตัวแปรอาร์เรย์แบบ 1 มิติ

ชนิดข้อมูล ชื่อตัวแปร [ขนาดของอาร์เรย์];

ตัวอย่างการประกาศตัวแปรอาร์เรย์แบบ 1 มิติ (แบบจองพื้นที่อย่างเดียว)

```
ประกาศตัวแปรอาร์เรย์ชนิด int
        int  num[5];

ประกาศตัวแปรอาร์เรย์ชนิด float
        float  score[7];

ประกาศตัวแปรอาร์เรย์ชนิด char
        char ch[4];
```

ตัวอย่างการประกาศตัวแปรอาร์เรย์แบบ 1 มิติ (แบบกำหนดค่าในพื้นที่)

```
ประกาศตัวแปรอาร์เรย์ชนิด int
        int  num[5] = { 10,20,30,40,50 } ;

ประกาศตัวแปรอาร์เรย์ชนิด float
        float  score[7] = { 1.00,2.00,33.33, 46.78,10.45,1.50,2.0 } ;

ประกาศตัวแปรอาร์เรย์ชนิด char
        char ch[7] = “witoon” ;
```

หรือถ้าต้องการกำหนดค่าให้กับอาร์เรย์เป็นอักขระสามารถทำได้ดังนี้

```
char name[9] = { ‘D’, ‘u’, ‘s’, ‘i’, ‘t’, ‘ ’, } ;
```

2. ตัวแปรอาร์เรย์แบบ 2 มิติ

ตัวแปรอาร์เรย์แบบ 2 มิติ จะเป็นการเก็บข้อมูลในแนวตั้งและแนวนอน หรือแนวแถวและคอลัมน์ ซึ่งการนำตัวแปรอาร์เรย์ 2 มิติมาใช้งานจะเหมาะสำหรับการเก็บข้อมูลในบางประเภท เช่น การเก็บคะแนนนักศึกษา การเก็บข้อมูลประวัติ เป็นต้น ดังแสดงไว้ตัวอย่างข้างล่าง

รูปแบบการประกาศตัวแปรอาร์เรย์แบบ 2 มิติ

ชนิดข้อมูล ชื่อตัวแปร [row][column];

ตัวอย่างการประกาศตัวแปรอาร์เรย์แบบ 2 มิติ (แบบจองพื้นที่อย่างเดียว)

ประกาศตัวแปรอาร์เรย์ชนิด int

```
int  b[5][4];
```

ประกาศตัวแปรอาร์เรย์ชนิด float

```
int  float  cost[10][5];
```

ประกาศตัวแปรอาร์เรย์ชนิด char

```
int  char  name[15][2];
```

ตัวอย่างการประกาศตัวแปรอาร์เรย์แบบ 2 มิติ (แบบกำหนดค่าในพื้นที่)

ประกาศตัวแปรอาร์เรย์ชนิด int

```
int  num[2][4] = { {1,2,3,4},
                   {5,6,7,8} };
```

ประกาศตัวแปรอาร์เรย์ชนิด float

```
float  cost[5][5] = { {1.0,2.0,3.0,4.0,5.0},
                      {11.1,22.22,33.33,44.44,55.55},
                      {0,0,3.3,4.4,5.5},
                      {0,0,0,0,0} };
```

ประกาศตัวแปรอาร์เรย์ชนิด char

```
char ch[6][5] = { {‘W’,’i’,’t’,’o’,’o’},
                  {‘n’, ’  ‘ ,’D’,’u’,’s’},
                  {‘l’,’t’} };
```

คำถามทบทวน

1. จงเขียนโปรแกรมเก็บค่าตัวเลขด้วยตัวแปรอาร์เรย์ โดยผู้ใช้งานสามารถป้อนได้ 10 ตัวเลข เมื่อเก็บเสร็จให้โปรแกรมทำการหาค่าเฉลี่ยของตัวเลขทั้งหมด

2. จงเขียนโปรแกรมเก็บค่าตัวอักษรจากผู้ใช้งานจำนวน 12 ตัวอักษรไว้ในตัวแปรแบบอาร์เรย์ จากนั้นเมื่อผู้ใช้งานป้อนเสร็จ ให้โปรแกรมแสดงค่าตัวอักษรในอาร์เรย์ออกมาทั้งหมด

3. จงเขียนโปรแกรมเก็บข้อมูลของนักศึกษา 5 คน โดยข้อมูลประกอบไปด้วย ชื่อ อายุ คะแนนวิชา A คะแนนวิชา B เมื่อโปรแกรมรับข้อมูลเสร็จให้โปรแกรมแสดงข้อมูลทั้งหมดออกมา

4. จงเขียนโปรแกรมเก็บข้อมูลของนักศึกษา 5 คน โดยข้อมูลประกอบไปด้วย ชื่อ อายุ คะแนนวิชา A คะแนนวิชา B และเกรดวิชา A และเกรดวิชา B เมื่อโปรแกรมรับข้อมูลเสร็จสิ้นให้โปรแกรมถามชื่อว่าต้องการดูข้อมูลของใคร และเมื่อผู้ใช้งานป้อนชื่อคนนั้นให้โปรแกรมแสดงข้อมูลคนนั้นออกมา

เอกสารอ้างอิง

คะชา ชาญศิลป์. (2548). **ภาษาซีสำหรับผู้เริ่มต้น.** กรุงเทพมหานคร: วิรัตน์เอ็ดดูเคชั่น.

ประภาพร ช่างไม้. (2545). **คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.** กรุงเทพมหานคร: อินโฟเพรส.

ธีรวัฒน์ ประกอบผล. (2553). **คู่มือการเขียนโปรแกรมภาษา C.** กรุงเทพมหานคร: ซิมพลิฟลาย.

สุระสิทธิ์ ทรงม้า และภูริพจน์ แก้วย่อง. (2552). **การโปรแกรมคอมพิวเตอร์.**: มหาวิทยาลัยราชภัฏสวนดุสิต.

สมชาย รัตนเลิศนุสรณ์. (2553). **การเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี.** กรุงเทพมหานคร: สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) ส.ส.ท.

สานนท์ เจริญฉาย. (2543). **การเขียนโปรแกรมและอัลกอริทึม.** กรุงเทพมหานคร: มหาจุฬาลงกรณ์ราชวิทยาลัย.

อรพิน ประวัติบริสุทธิ์. (2554). **คู่มือการเรียนภาษา C ฉบับปรับปรุงใหม่** (พิมพ์ครั้งที่ 10)**.** กรุงเทพมหานคร: โปรวิชั่น.

http://www.61.7.214.35/c/webbase/unit9/arrays.php

http://www.mwit.ac.th/~cs/download/tech30101/ch12.html

http://www.krubpk.com/com_3/pan8/pan9_3.html

http://www.bcoms.net/php/php08.asp

http://www.thaigoodview.com/library/contest2552/type2/tech04/22/cit/6_1.html

บทที่ 10

ฟังก์ชั่นและการสร้างฟังก์ชั่นในภาษาซี

แผนการสอนประจำบท

หัวข้อเนื้อหา

10.1 ฟังก์ชั่นมาตรฐานในภาษาซี
10.2 ฟังก์ชั่นที่ผู้พัฒนาโปรแกรมสร้างขึ้น

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถเข้าใจหลักการทำงานของคำสั่ง if ได้อย่างเข้าใจและถูกต้อง
2. ผู้เรียนสามารถนำคำสั่ง if ไปใช้งานในการเขียนโปรแกรมในกรณีต่าง ๆ ได้อย่างถูกต้อง
3. ผู้เรียนสามารถเข้าใจหลักการทำงานของคำสั่ง switch ได้อย่างเข้าใจและถูกต้อง
4. ผู้เรียนสามารถนำคำสั่ง switch ไปใช้งานในการเขียนโปรแกรมในกรณีต่าง ๆ ได้อย่างถูกต้อง

วิธีการสอน

1. สอนแบบบรรยายโดยใช้เอกสารประกอบการสอน
2. สอนแบบบรรยายโดยใช้ Slide Power Point
3. สอนโดยใช้โปรแกรม Turbo C++ 4.5 เพื่อลงมือปฏิบัติในการเขียนโปรแกรมภาษาซี
4. สอนโดยให้ผู้เรียนเป็นส่วนกลาง โดยใช้วิธีถามตอบกับผู้เรียนระหว่างบรรยาย
5. ให้ผู้เรียนสรุปและอธิบายหัวข้อที่ผู้เรียนที่ได้รับมอบหมาย

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. Slide Power Point
3. โปรแกรม Turbo C++ 4.5
4. ตัวอย่างแบบทดสอบ

การวัดผลและประเมินผล

1. การทำตามตัวอย่างโจทย์คำสั่งและการทำแบบฝึกหัด
2. ความตั้งใจในชั้นเรียนและการร่วมมือในการนำเสนอหรืออภิปราย
3. สังเกตจากการตอบคำถามและการร่วมทำกิจกรรมในชั้นเรียน

บทที่ 10

ฟังก์ชั่นและการสร้างฟังก์ชั่นในภาษาซี

ฟังก์ชั่น (Function) หรือหลายคนอาจจะเรียกว่าโปรแกรมย่อย ซึ่งเป็นกลุ่มคำสั่งที่สร้างขึ้นมาเพื่อให้โปรแกรมทำงานอย่างใดอย่างหนึ่ง และสามารถเรียกใช้งานขึ้นมาได้อย่างสะดวก และส่งผลให้โปรแกรมมีความซับซ้อนน้อยลงและสามารถแก้ไขโปรแกรมได้อย่างง่ายขึ้น โดยฟังก์ชันในภาษาซี สามารถแบ่งตามแหล่งที่มาได้ 2 ประเภท คือ

10.1 ฟังก์ชั่นมาตรฐานในภาษาซี

ฟังก์ชันมาตรฐานในภาษาซี ซึ่งจะอยู่ในไลบรารีภาษาซีมาตรฐาน (C Standard Library) ไลบรารีภาษาซีมาตรฐานประกอบด้วยฟังก์ชันต่าง ๆ มากมาย ไม่ว่าจะใช้สำหรับการคำนวณทางคณิตศาสตร์ การจัดการกับข้อความ การจัดการกับ input/output และอื่นๆ ซึ่งจะทำให้งานของโปรแกรมเมอร์ง่ายขึ้น โดยการใช้งานฟังก์ชันประเภทนี้จะต้องรวม (include) ไลบรารีที่ต้องการใช้งานเพื่อให้ตัวแปลภาษารู้ว่าฟังก์ชันที่โปรแกรมเมอร์ต้องการใช้อยู่ในไลบรารีมาตรฐานตัวใด

ในบางครั้งอาจเรียกว่า library functions ปกติฟังก์ชันเหล่านี้จะจัดเก็บไว้ใน header files ดังนั้นผู้ใช้จะต้องรู้ว่าฟังก์ชันนั้นอยู่ใน header file ใด จึงจะนำไปเรียกใช้ในส่วนต้นของโปรแกรมด้วย #include <header file.h> ได้ เช่น #include <stdio.h> ตัวอย่าง เช่น หากต้องการใช้ฟังก์ชัน printf () ซึ่งอยู่ในไลบรารีมาตรฐานสำหรับเกี่ยวกับอินพุตและเอาต์พุต (standard input/output) ที่ชื่อ stdio โดยจะต้องเรียกใช้ฟังก์ชั่น #include<stdio.h> โดยที่ stdio.h เป็นชื่อ header file ของไลบรารีมาตรฐาน stdio หรือหากต้องการใช้ฟังก์ชัน clrscr() ของไลบรารีมาตรฐาน conio เพื่อล้างหน้าจอ โดยจะต้องเรียกใช้ฟังก์ชั่น #include<conio.h>

ตารางที่ 10.1 ตัวอย่างฟังก์ชันมาตรฐานในภาษาซี

ไลบรารี	ฟังก์ชั่น	คำอธิบาย
Stdio.h	printf( )	คำสั่งใช้ในการแสดงผล
	scanf( )	คำสั่งใช้ในการรับข้อมูล
conio.h	getchar( )	คำสั่งใช้ในการรับค่าข้อมูลอักขระ
	getch( )	คำสั่งใช้ในการรับค่าข้อมูลอักขระ
	putchar( )	คำสั่งใช้ในการแสดงผลอักขระ
	clrscr( )	คำสั่งเคลียร์หรือล้างข้อมูลหน้าจอคอมพิวเตอร์
math.h	abs(X)	คำนวณค่าสัมบูรณ์ของ X
	arctan(X)	คำนวณค่า arctan ของ X
	cos(X)	คำนวณค่า cosine ของ X
	exp(X)	คำนวณค่า e^{X} โดย e = 2.71828 เป็นเลขฐานของลอการิทึม

ตารางที่ 10.1 ตัวอย่างฟังก์ชันมาตรฐานในภาษาซี (ต่อ)

ไลบรารี	ฟังก์ชั่น	คำอธิบาย
math.h	Ln(X)	คำนวณค่าลอการิธึมธรรมชาติของ X โดย X > 0
	Odd(X)	ทดสอบว่า X เป็นเลขจำนวนคี่หรือไม่โดยให้ค่าเป็นจริง ถ้า X เป็นเลขจำนวนคี่มิฉะนั้นให้ค่าเป็นเท็จ
	Sin(X)	คำนวณค่า sine ของ X
	Round(X)	ให้ค่าเลขจำนวนเต็มที่ใกล้ X ที่สุด โดยเศษตั้งแต่ 0.5 ขึ้นไปปัดขึ้น
	Sqr(X)	คำนวณค่ายกกำลังที่สองของ X
	Sqrt(X)	คำนวณค่ารากที่สองของ X
	Trunc(X)	ให้ค่าเลขจำนวนเต็มที่ปัดเศษทศนิยมของเลขจำนวนจริง X ทิ้ง
	Chr(X)	ให้ค่าตัวอักขระที่มีรหัสแอสกีเป็น X
	Ord(X)	ให้ค่ารหัสแอสกีที่ตรงกับตัวอักขระ X
	Pred(X)	ให้ค่าข้อมูลลำดับก่อนหน้า X ตามลำดับในตารางรหัสแอสกี
	Succ(X)	ให้ค่าข้อมูลลำดับถัดจาก X ตามลำดับในตารางรหัสแอสกี
	Upcase(X)	ให้ค่าตัวอักขระใหญ่ของ X ในกรณีที่ X เป็นตัวอักขระเล็ก มิฉะนั้นให้ค่าเป็นตัวอักขระ X ตามเดิม
	Floor(X)	ให้ค่าจำนวนเต็มที่มากที่สุดที่น้อยกว่า X
	Ceiling(X)	ให้ค่าจำนวนเต็มที่น้อยที่สุดที่มากกว่า X
	Pi	ให้ค่าคงตัวของ p = 3.1417

10.2 ฟังก์ชั่นที่ผู้พัฒนาโปรแกรมสร้างขึ้น

ฟังก์ชั่นที่ผู้พัฒนาโปรแกรมสร้างขึ้น มีไว้เพื่อในกรณีที่การเรียกใช้งานชุดคำสั่งนั้น ๆ บ่อยครั้งหรือเป็นชุดคำสั่งพิเศษ ทั้งนี้จะช่วยให้การพัฒนาโปรแกรมมีความซับซ้อนของโปรแกรมน้อยลง และง่ายต่อการเข้าใจและง่ายต่อการแก้ไขโปรแกรม ซึ่งการสร้างฟังก์ชั่นมีรูปแบบ ดังนี้

```
ประเภทข้อมูล  ชื่อฟังก์ชั่น(พารามิเตอร์)
        {
                คำสั่ง..... 1;
                คำสั่ง..... 2;
                คำสั่ง..... 3;
                คำสั่ง.....
                คำสั่ง.....
                คำสั่ง..... n;
                return ;
        }
```

ประเภทข้อมูล หมายถึง ประเภทของข้อมูลที่ส่งกลับ ส่วนนี้จะมีหรือไม่มีก็ได้ขึ้นอยู่กับการทำงานของฟังก์ชั่นนั้น ๆ ถ้าหากฟังก์ชั่นไม่มีการส่งค่ากลับส่วนนี้จะประกาศเป็น Void

ชื่อฟังก์ชั่น หมายถึง ชื่อของฟังก์ชั่นที่ได้สร้างขึ้นมา โดยผู้ใช้งานสามารถเรียกใช้งานฟังก์ชั่นนี้ด้วยชื่อที่ได้ตั้งขึ้นนี้ และการตั้งชื่อต้องยึดหลักตามกฎการตั้งชื่อตัวแปรของภาษาซีที่ได้เรียนรู้ไปแล้วในเนื้อในบทที่ 5

พารามิเตอร์ หมายถึง ตัวแปรที่ใช้ในการรับค่าเข้ามาในฟังก์ชั่น ส่วนนี้จะมีหรือไม่มีก็ได้ขึ้นอยู่กับการทำงานของฟังก์ชั่นนั้น ๆ

return หมายถึง ส่วนที่ใช้ในการส่งค่ากลับไปยังส่วนอื่น ๆ ส่วนนี้จะมีหรือไม่มีก็ได้ขึ้นอยู่กับการทำงานของฟังก์ชั่นนั้น ๆ

ตัวอย่างการสร้างฟังก์ชั่นในภาษาซี

```
void  name( )
        {
                printf (“I love Dusit Trang”);
        }
```

จากตัวอย่างข้างต้น ชื่อฟังก์ชั่นนี้มีชื่อว่า name() จะไม่มีข้อมูลในการส่งกลับ และไม่มีการผ่านค่าพารามิเตอร์ การทำงานก็คือเมื่อมีเรียกใช้งานฟังก์ชั่น name() โปรแกรมจะพิมพ์คำว่า I love Dusit Trang ออกทางหน้าจอคอมพิวเตอร์

```
int  number(int a ,int b)
        {
                Int x;
                x = a + b;
                return x;
        }
```

จากตัวอย่างข้างต้น เป็นฟังก์ชั่นที่ใช้ในการบวกเลข โดยมีชื่อฟังก์ชั่นว่า number( ) และมีการผ่านค่าพารามิเตอร์ทางตัวแปร a และตัวแปร b ในการทำงานก็คือเมื่อมีเรียกใช้งานฟังก์ชั่น number( ) โปรแกรมจะนำค่าส่งผ่านมามาเก็บค่าไว้ในตัวแปร a และ b จากนั้นก็จะคำนวณตามที่โปรแกรมสั่งงานไว้ และเมื่อคำนวณเสร็จสิ้น โปรแกรมจะทำการคืนค่าผลบวกกลับไปผ่านทางตัวแปร x

ตัวอย่างที่ 10.1 การเรียกใช้งานฟังก์ชั่นย่อย โดยในฟังก์ชั่น main จะมีการเรียกใช้งานฟังก์ชั่นย่อยที่ชื่อว่า name( ) และในคำสั่งของฟังก์ชั่นย่อยจะมีการแสดงผลข้อมูลคำว่า I love Dusit Trang จำนวน 5 ข้อความ

```
// โค้ดโปรแกรม
#include "stdio.h"
void name( )     // ฟังก์ชั่นย่อย
        {
         int i;
                for (i=0;i<5;i++)
                {
                printf ("I love Dusit Trang\n\n");
                }
        }
main( )         // ฟังก์ชั่นหลัก
        {
         name( );
        }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
I love Dusit Trang

I love Dusit Trang

I love Dusit Trang

I love Dusit Trang

I love Dusit Trang
```

ตัวอย่างที่ 10.2 การเรียกใช้งานฟังก์ชั่นย่อย โดยเป็นการทดลองเขียนโปรแกรมบวกเลข โดยมีการผ่านค่าไปยังค่าพารามิเตอร์ และมีการส่งค่า return กลับมาแสดงผล

```
// โค้ดโปรแกรม
#include "stdio.h"
int  sum(int a ,int b)   // ฟังก์ชั่นย่อยใช้ในการคำนวณการบวก
        {
                int c;
                c = a + b;
                return c;
        }
main()                  // ฟังก์ชั่นหลัก
        {
                int x,y;
        loop1:  printf("\nEnter : ");
                scanf("%d",&x);
                printf("\nEnter : ");
                scanf("%d",&y);
                printf("\n%d + %d = %d\n",x,y,sum(x,y));
                goto loop1;
                }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
Enter : 1
Enter : 2
1 + 2 = 3

Enter : 10
Enter : 20
10 + 20 = 30

Enter : 88
Enter : 99
88 + 99 = 187

Enter : _
```

ตัวอย่างที่ 10.3 การเรียกใช้งานฟังก์ชั่นย่อย โดยการทดลองจะเป็นการเขียนโปรแกรมบวกเลข ลบเลข โดยจะมีฟังก์ชั่นย่อย 2 ฟังก์ชั่น คือ ฟังก์ชั่น summation ใช้ในการบวก และฟังก์ชั่น Deletion ใช้ในการลบ

```
// โค้ดโปรแกรม
#include "stdio.h"
int  summation(int a1 ,int b1) { //ฟังก์ชั่นย่อยสำหรับการบวก
                int c1;
                c1 = a1 + b1;
                return c1;
        }
int  deletion(int a2 ,int b2) {   //ฟังก์ชั่นย่อยสำหรับการลบ
                int c2;
                c2 = a2 - b2;
                return c2;
        }
main()  {                       //ฟังก์ชั่นหลัก
        int x,y,z;
loop1: printf("\n\nYou want to + (1) or - (2) : ");
        scanf("%d",&z);
        printf("\nEnter  : ");
        scanf("%d",&x);
        printf("\nEnter  : ");
        scanf("%d",&y);
if(z==1) {
        printf("\n%d + %d = %d\n",x,y,summation(x,y));
          }
else if (z==2)   {
        printf("\n%d - %d = %d\n",x,y,deletion(x,y));
                }
Else {
         printf("\nYou put the wrong value.");
                }
                goto loop1;
            }
```

ผลลัพธ์เมื่อทดสอบโปรแกรม

```
You want to + (1) or - (2) : 1
Enter : 99
Enter : 1
99 + 1 = 100

You want to + (1) or - (2) : 1
Enter : 1000
Enter : 1001
1000 + 1001 = 2001

You want to + (1) or - (2) : 2
Enter : 100
Enter : 20
100 - 20 = 80

You want to + (1) or - (2) : 2
Enter : 590
Enter : 876
590 - 876 = -286

You want to + (1) or - (2) : 3
Enter : 88
Enter : 99
You put the wrong value.

You want to + (1) or - (2) : 100
Enter : 245
Enter : 123
You put the wrong value.
```

สรุปท้ายบท

ฟังก์ชั่น (Function) หรือหลายคนอาจจะเรียกว่าโปรแกรมย่อย ซึ่งเป็นกลุ่มคำสั่งที่สร้างขึ้นมาเพื่อให้โปรแกรมทำงานอย่างใดอย่างหนึ่ง และสามารถเรียกใช้งานขึ้นมาได้อย่างสะดวก และส่งผลให้โปรแกรมมีความซับซ้อนน้อยลงและสามารถแก้ไขโปรแกรมได้อย่างง่ายขึ้น โดยฟังก์ชันในภาษาซี สามารถแบ่งตามแหล่งที่มาได้ 2 ประเภท คือ

1. ฟังก์ชั่นมาตรฐานในภาษาซี

ฟังก์ชันมาตรฐานในภาษาซี ซึ่งจะอยู่ในไลบรารีภาษาซีมาตรฐาน (C Standard Library) ไลบรารีภาษาซีมาตรฐานประกอบด้วยฟังก์ชันต่าง ๆ มากมาย ไม่ว่าจะใช้สำหรับการคำนวณทางคณิตศาสตร์ การจัดการกับข้อความ การจัดการกับ input/output และอื่นๆ ซึ่งจะทำให้งานของโปรแกรมเมอร์ง่ายขึ้น โดยการใช้งานฟังก์ชันประเภทนี้จะต้องรวม (include) ไลบรารีที่ต้องการใช้งานเพื่อให้ตัวแปลภาษารู้ว่าฟังก์ชันที่โปรแกรมเมอร์ต้องการใช้อยู่ในไลบรารีมาตรฐานตัวใด ในบางครั้งอาจเรียกว่า library functions ปกติฟังก์ชันเหล่านี้จะจัดเก็บไว้ใน header files ดังนั้นผู้ใช้จะต้องรู้ว่าฟังก์ชันนั้นอยู่ใน header file ใด จึงจะนำไปเรียกใช้ในส่วนต้นของโปรแกรมด้วย #include <header file.h> ได้ เช่น #include <stdio.h> , #include<conio.h> เป็นต้น

2. ฟังก์ชั่นที่ผู้พัฒนาโปรแกรมสร้างขึ้น

ฟังก์ชั่นที่ผู้พัฒนาโปรแกรมสร้างขึ้น มีไว้เพื่อในกรณีที่การเรียกใช้งานชุดคำสั่งนั้น ๆ บ่อยครั้งหรือเป็นชุดคำสั่งพิเศษ ทั้งนี้จะช่วยให้การพัฒนาโปรแกรมมีความซับซ้อนของโปรแกรมน้อยลง และง่ายต่อการเข้าใจและง่ายต่อการแก้ไขโปรแกรม ซึ่งการสร้างฟังก์ชั่นมีรูปแบบ ดังนี้

```
ประเภทข้อมูล  ชื่อฟังก์ชั่น(พารามิเตอร์)
        {
                คำสั่ง..... 1;
                คำสั่ง..... 2;
                คำสั่ง..... n;
                return ;
        }
```

ประเภทข้อมูล หมายถึง ประเภทของข้อมูลที่ส่งกลับ ส่วนนี้จะมีหรือไม่มีก็ได้ขึ้นอยู่กับการทำงานของฟังก์ชั่นนั้น ๆ ถ้าหากฟังก์ชั่นไม่มีการส่งค่ากลับส่วนนี้จะประกาศเป็น Void

ชื่อฟังก์ชั่น หมายถึง ชื่อของฟังก์ชั่นที่ได้สร้างขึ้นมา โดยผู้ใช้งานสามารถเรียกใช้งานฟังก์ชั่นนี้ด้วยชื่อที่ได้ตั้งขึ้นนี้ และการตั้งชื่อต้องยึดหลักตามกฎการตั้งชื่อตัวแปรของภาษาซีที่ได้เรียนรู้ไปแล้วในเนื้อในบทที่ 5

พารามิเตอร์ หมายถึง ตัวแปรที่ใช้ในการรับค่าเข้ามาในฟังก์ชั่น ส่วนนี้จะมีหรือไม่มีก็ได้ขึ้นอยู่กับการทำงานของฟังก์ชั่นนั้น ๆ

return หมายถึง ส่วนที่ใช้ในการส่งค่ากลับไปยังส่วนอื่น ๆ ส่วนนี้จะมีหรือไม่มีก็ได้ขึ้นอยู่กับการทำงานของฟังก์ชั่นนั้น ๆ

คำถามทบทวน

1. จงเขียนโปรแกรมรับค่าตัวเลขจากผู้ใช้งาน 1 ค่า จากนั้นให้โปรแกรมคำนวณค่ารากที่สองของตัวเลขที่รับเข้ามา

2. จงเขียนโปรแกรมรับค่าตัวเลขจากผู้ใช้งาน 1 ค่า จากนั้นให้โปรแกรมคำนวณค่า cosine ของตัวเลขที่รับเข้ามา

3. จงเขียนโปรแกรมโดยสร้างฟังก์ชั่นย่อย โดยให้พิมพ์ชื่อที่ผู้ใช้งานได้ป้อนเข้าไปจำนวน 10 ครั้ง

4. จงเขียนโปรแกรมโดยสร้างฟังก์ชั่นย่อย โดยให้รับข้อมูลตัวเลขจากผู้ใช้งาน แล้วให้โปรแกรมตอบว่าเป็นตัวเลขที่รับเข้ามาเป็นเลขคู่หรือเลขคี่

5. จงเขียนโปรแกรมโดยสร้างฟังก์ชั่นย่อย โดยให้รับข้อมูลตัวเลขจากผู้ใช้งาน 1 ค่า จากนั้นให้โปรแกรมแสดง * ตามจำนวนที่ผู้ใช้งานได้ป้อนเข้าไป

6. จงเขียนโปรแกรมโดยสร้างฟังก์ชั่นย่อย โดยโปรแกรมจะทำการรับค่าจากผู้ใช้งาน 2 ค่า แล้วโปรแกรมจะให้มีการเลือกว่าผู้ใช้งานต้องการนำตัวเลขมาบวก ลบ คูณ หรือหาร จากนั้นให้โปรแกรมแสดงคำตอบของการคำนวณนั้น ๆ ออกมา

เอกสารอ้างอิง

คะชา ชาญศิลป์. (2548). **ภาษาซีสำหรับผู้เริ่มต้น.** กรุงเทพมหานคร: วิรัตน์เอ็ดดูเคชั่น.

ประภาพร ช่างไม้. (2545). **คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น.** กรุงเทพมหานคร: อินโฟเพรส.

ธีรวัฒน์ ประกอบผล. (2553). **คู่มือการเขียนโปรแกรมภาษา C.** กรุงเทพมหานคร: ซิมพลิฟลาย.

สุระสิทธิ์ ทรงม้า และภูริพจน์ แก้วย่อง. (2552). **การโปรแกรมคอมพิวเตอร์.**: มหาวิทยาลัยราชภัฏสวนดุสิต.

สมชาย รัตนเลิศนุสรณ์. (2553). **การเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี.** กรุงเทพมหานคร: สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) ส.ส.ท.

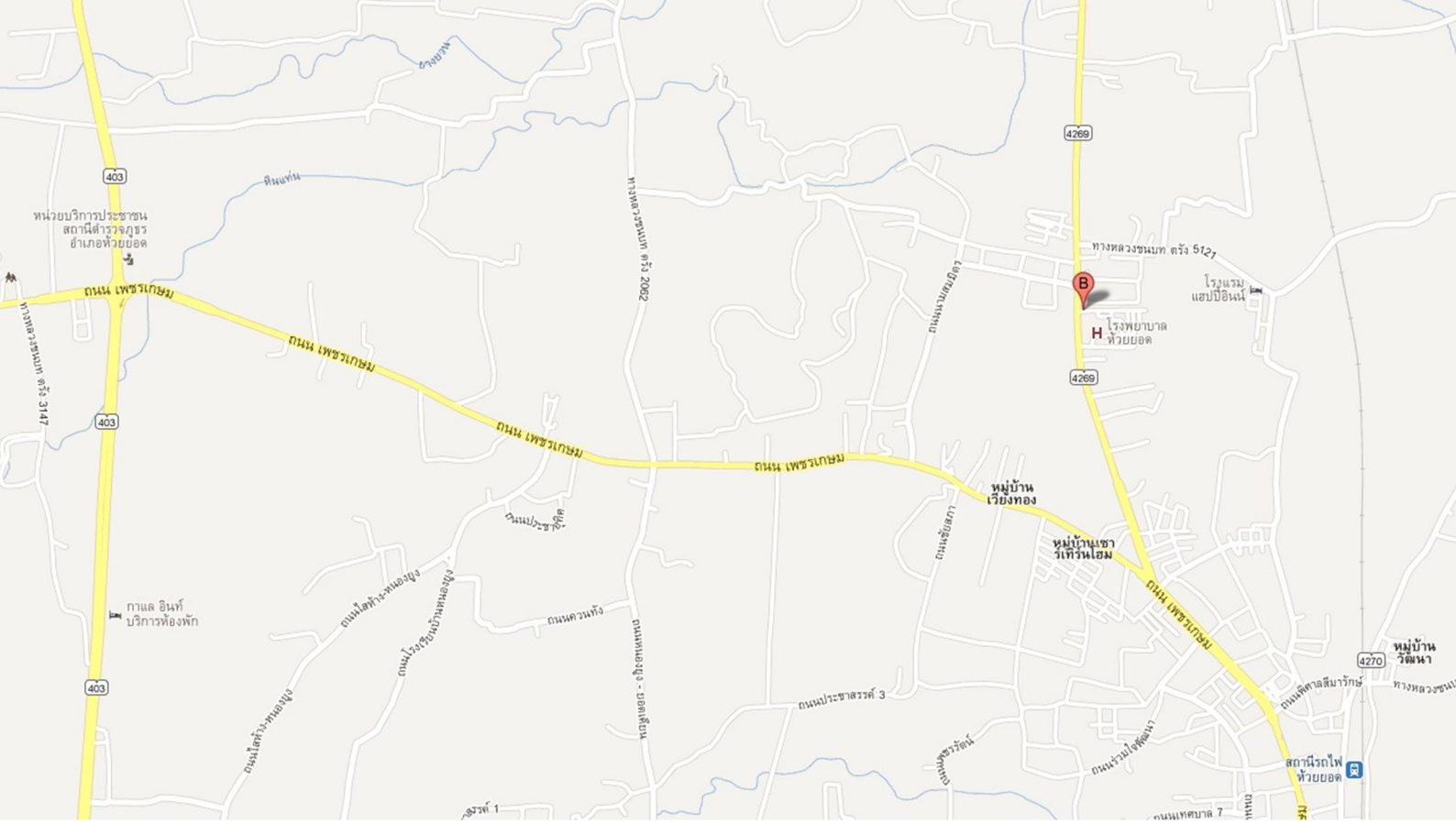
สานนท์ เจริญฉาย. (2543). **การเขียนโปรแกรมและอัลกอริทึม.** กรุงเทพมหานคร: มหาจุฬาลงกรณ์ราชวิทยาลัย.

อรพิน ประวัติบริสุทธิ์. (2554). **คู่มือการเรียนภาษา C ฉบับปรับปรุงใหม่** (พิมพ์ครั้งที่ 10)**.** กรุงเทพมหานคร: โปรวิชั่น.

http://www.bitcom/com1/pan8/cp_11.html

http://www.bcoms.net/php/php12.asp

http://www.thaigoodview.com/library/contest2552/type2/tech04/22/cit/8_2.html



ความพยายามอยู่ที่ไหน

ความสำเร็จอยู่ที่นั่น