บทที่ 6
โครงสร้างข้อมูลแบบง่าย (Simple Data Types)

บทนำ ดังที่ได้กล่าวมาแล้วในบทที่ 1 ถึงการจัดการในพื้นที่ส่วนที่เป็นสมองของคอมพิวเตอร์เพื่อจัดเก็บโปรแกรม และ ข้อมูล ในบทนี้จะกล่าวถึงการจัดการและการจัดเก็บข้อมูลในลักษณะต่างๆของระบบ ซึ่งเป็นส่วนที่เกี่ยวข้องกับการทำนามธรรมข้อมูล (Data Abstraction) ซึ่งจะแบ่งออกเป็น 2 ส่วนคือ โครงสร้างข้อมูลแบบง่าย และโครงสร้างข้อมูลแบบซับซ้อน เพื่อจัดเก็บข้อมูลในการประมวลผลในหน่วยความจำชั่วคราวของระบบเครื่องคอมพิวเตอร์ รวมทั้งการนำโครงสร้างเหล่านี้ไปประยุกต์ในการจัดเก็บข้อมูลแบบถาวรบนสื่อ (Media) ประเภทต่างๆ เช่น เทปแม่เหล็ก (Magnetic Tape) จานแม่เหล็กชนิดอ่อน (Diskette) จานแม่เหล็กชนิดแข็ง (Hard Disk) CD ROM เป็นต้น

การจัดแบ่งประเภทของข้อมูลแบบง่าย (Simple Data Structure or Scalar Data Type)

- **ข้อมูลเชิงคณิตศาสตร์ (Numeric Data Type)**
- **ข้อมูลเชิงตรรกะ (Boolean Data Type)**
- **ข้อมูลเชิงอักขระ (Character Data Type)**

ภาษาโปรแกรมประเภทต่างๆที่ใช้งานนั้นจะมีมาตราฐานของข้อมูลประเภทโครงสร้างแบบง่าย ดังรายการที่ปรากฏมาข้างต้น

ข้อมูลเชิงคณิตศาสตร์ (Numeric Data Type)

จะหมายถึงข้อมูลที่สามารถนำไปดำเนินการประมวลผลทางคณิตศาสตร์ได้ (โดยการใช้เครื่องหมาย + - * /) ดำเนินการได้ ในขณะที่ข้อมูลประเภทอื่นไม่สามารถดำเนินงานดังกล่าวได้

ข้อมูลเชิงคณิตศาสตร์จะแบ่งออกเป็น 2 ประเภท คือ

1. ข้อมูลประเภทเลขจำนวนเต็ม ตัวอย่างเช่นข้อมูลลักษณะต่อไปนี้ 12 ,0 , -34 ,23456 เป็นต้น
2. ข้อมูลประเภทเลขจำนวนจริง ตัวอย่างข้อมูลลักษณะต่อไปนี้ 123.76 , 0.657 , -3.56 , 3.76 E 2

ข้อแตกต่างระหว่างข้อมูลที่เป็นเลขจำนวนเต็มและเลขที่เป็นจำนวนจริงก็คือ เลขจำนวนเต็มจะมีเฉพาะเลขจำนวนเต็มอาจจะเป็นบวก หรือ ลบ หรือ 0 ก็ได้ ในขณะที่เลขจำนวนจริงจะมีการเก็บเลขจำนวนเต็มและส่วนของทศนิยมด้วย (เลข 2 ถ้าเก็บในส่วนของพื้นที่ที่เป็นจำนวนจริง จะจัดเก็บเป็น 2.00 เป็นต้น) ดังนั้นการจัดสรรพื้นที่ในหน่วยความจำสำหรับเลขจำนวนเต็ม

จะน้อยกว่าพื้นที่สำหรับการจัดเก็บเลขที่เป็นจำนวนจริง เช่นบางระบบการจัดการจะจัดสรรพื้นที่ให้กับเลขจำนวนเต็มเพียง 2 Bytes และให้เลขจำนวนจริงมีขนาด 4 Bytes ในขณะที่บางระบบการจัดการจะจัดสรรให้ถึง 4 Bytes สำหรับเลขจำนวนเต็มและให้ 8 Bytes สำหรับเลขจำนวนจริง ดังนี้เป็นต้น โครงสร้างการจัดเก็บของเลขจำนวนจริง (Floating Point Format) นั้นจะจัดเก็บในรูปแบบที่เป็น Normalization ดังนี้

$$\text{Real Number} = \text{Mantissa} \times 2^{\text{Exponent}}$$

ภาพที่ 6.1 แสดงการจัดเก็บข้อมูลประเภท จำนวนเต็ม และจำนวนจริงภายในสมองเครื่องคอมพิวเตอร์

Integer Format

Binary Number

Floating Pont Number (Real Format)

Mantissa	Exponent

ตัวอย่างขนาดของเลขจำนวนจริงที่จัดเก็บบนเครื่องไมโครคอมพิวเตอร์ของระบบ IBM จะมีขนาดตั้งแต่ 10^{-39} - 10^{+39} ในขณะที่เลขจำนวนเต็มบวกจะมีขนาดตั้งแต่ 1 ถึง 32767 ส่วนเลขจำนวนเต็มลบจะมีขนาดตั้งแต่ - 32768 ถึง -1 การกำหนดของพื้นที่ในการจัดเก็บนั้นจะขึ้นอยู่กับแต่ละระบบปฏิบัติการ แต่โดยปกติขนาดของพื้นที่ของเลขจำนวนเต็มจะน้อยกว่าพื้นที่ของเลขจำนวนจริงเสมอ

ความผิดพลาดในการจัดการ กับข้อมูลเชิงตัวเลข (Numerical Inaccuracies) การข้อมูลเชิงคณิตศาสตร์ไม่ว่าจะเป็นเรื่องในการจัดเก็บ หรือการคำนวณนั้นจะส่งผลให้ผลลัพธ์ที่จะนำไปใช้ผิดพลาดไปด้วย ดังนั้นผู้เขียนโปรแกรมจึงจำเป็นจะต้องใช้ความเข้าใจนี้เพื่อนำใช้ในการเขียนโปรแกรมและไปป้องกันและแก้ไขข้อผิดพลาดที่อาจเกิดขึ้นในการจัดการกับข้อมูลเชิงคณิตศาสตร์เพื่อ ลดความผิดพลาดไม่ให้มีสูงจนเกินกว่าที่จะนำไปดำเนินการต่อได้ จากการแสดงวิธีการจัด

เก็บข้อมูลสำหรับเลขที่เป็นจำนวนจริงนั้นจะเห็นได้ว่าการจัดเก็บนั้นอาจจะไม่สามารถเก็บได้ถูกต้องทั้งนี้เนื่องจากการที่ขนาดของพื้นที่ที่ใช้ในการจัดเก็บมีขนาดเล็กแกินไป เช่น 1.0/3 = .33333….. ไม่สามารถจัดเก็บได้ครบถ้วนเพราะเลขดังกล่าวเป็นเลขที่หารไม่รู้จบ แต่ก็ยังมีบางกรณีที่เลขจำนวนที่รู้จบบางจำนวนก็ไม่สามารถเก็บครบถ้วนได้ เช่น 1.65265 เมื่อแปลงเป็นเลขฐานสองคือ 1.10101 ภายหลังการทำ Normalize แล้วจะเป็น $(0.110101 \times 2^{1})_2$ สมมุติว่าถ้าเกิดระบบนั้นสามารถจัดเก็บได้โดยให้ส่วนของ Mantissa เป็นขนาด 5 Bits ดังนั้นเลขจำนวนนั้นจะจัดเก็บได้เป็น $(0.11010 \times 2^{1})_2$ ซึ่งมีค่าเท่ากับ $(1.625)_{10}$ ส่งผลให้มีค่าความผิดพลาดสูงถึง $(0.03125)_{10}$ นั่นหมายความว่าเลขจำนวนจริงใดๆที่จัดเก็บในพื้นที่ความจำนั้นๆอาจจะไม่สามารถเก็บได้ครบถ้วนได้ ตัวอย่างการคำนวณด้วยโปรแกรมต่อไปนี้จะเห็นได้ว่าเลขจำนวน 0.1 นั้นไม่สามารถเก็บเป็นเลข 0.1 ได้ เพราะเมื่อโปรแกรมนี้ทำการประมวลผลโดยการนำเลข 0.1 นำมารวมกัน 10 ครั้งนั้นค่าของผลรวมควรจะออกมาเป็น 1.0 ตามหลักของคณิตศาสตร์ คือ

$$\sum_{i=1}^{10} 0.1 \quad = \quad 1.0$$

โปรแกรมทำการหาผลรวมของเลขจำนวน 0.1 จำนวน 10 ครั้ง

```
Trial := 0.0 ;
While Trial  < 1.0 Do
        Begin

                Trial       := Trial + 0.1
End ; {* While *}
Writeln ( " Trial =  ", Trial) ;
```

โปรแกรมนี้การทำงานอาจจะมีจำนวนรอบไม่เท่ากัน บางระบบอาจจะวนถึง 10 ครั้ง ในขณะที่บางระบบวนเพียง 11 ครั้ง และบางระบบก็ทำงานวนเพียง 9 ครั้งเท่านั้น ทั้งนี้เนื่องระบบการจัดเก็บและตรวจสอบเลขจำนวนจริงนั้นไม่สามารถจะเก็บเลขจำนวนนั้นได้ครบถ้วนได้ ระบบการจัดการจึงต้องมีการกำหนดค่าของความผิดพลาดให้กับการดำเนินการต่างๆ เช่น อาจจะกำหนดว่า 1.00 มีค่าโดยประมาณเท่ากับ 0.999999 นั่นหมายความว่า ถ้าค่าใดค่าหนึ่งของเลขจริงจำนวนหนึ่งต่างจากเลขจำนวนจริง 1.0 ไม่เกิน E =.0000001 แล้วค่านั้นจะถูกถือเสมือนว่าค่าดังกล่าวมีขนาดเท่ากับ 1.0 ผลที่ตามก็คือโปรแกรมดังกล่าวบางระบบวนทำงานจำนวน 9 รอบ หรือ 10 รอบ แต่บางโปรแกรมวนทำงานถึง 11 รอบเป็น ต้น จากเหตุผลที่กล่าวมานี้ผู้เขียน

โปรแกรมก็ไม่ควรที่จะใช้ตัวแปรข้อมูลที่เป็นเลขจำนวนจริง มาใช้ในการควบคุม Loop ในการทำงาน ไม่ว่าจะเป็น Loop ประเภทใด ระหว่าง For หรือ While หรือ Repeat ... Until เราควรจะใช้เลขจำนวนเต็มในการควบคุม Loop ซึ่งด้วยเลขจำนวนเต็มจะดีกว่า เพราะโอกาสจะผิดพลาดจะไม่ปรากฏ

นอกเหนือจากความผิดพลาดในการจัดเก็บข้อมูลแล้ว การคำนวณก็ส่งผลให้เกิดความผิดพลาดด้วยเช่นกัน เช่นการนำเลขจำนวนจริงสองจำนวนที่มีขนาดแตกต่างกันมาหาผลรวมกันดังนี้ คือ

X = 100000.0 + 0.000001234 ผลลัพธ์ที่เกิดขึ้นในคอมพิวเตอร์บางระบบจะปรากฏค่า X = 1.00000.0 หรือในกรณีของการนำเลขที่มีค่าเล็กมากๆจนเข้าใกล้ 0 ของเลข 2 จำนวนมาคูณกันก็อาจจะเกิดสภาพของการเกิด Arithmetic Underflow. ในทำนองเดียวกันกับการนำเลข 2 จำนวนที่มีขนาดใหญ่มากๆมาคูณกันก็จะเกิดสภาพของการ Arithmetic Overflow ด้วยเหตุผลดังกล่าวนี้ จึงส่งผลให้คำสั่งบางคำสั่งเกิดปัญหาการทำงานผิดพลาด (Run Time Error) จากสาเหตุของ Arithmetic Underflow หรือ Arithmetic Overflow ตัวอย่างคำสั่งในการแปลงจำนวนชั่วโมงให้เป็นวินาที ดังนี้ SecInDay := Hours * Min * Sec ; คอมพิวเตอร์บางระบบจะเกิดปัญหาเพราะไม่สามารถจัดเก็บเลขที่ใหญ่เกินไปได้ สำหรับเลขจำนวนเต็มในระบบนั้น ดังนั้นการมีความรู้เรื่องการจัดเก็บ การคำนวณกับตัวแปรที่เป็นเลขจำนวนเต็ม หรือเลขจริงที่เป็น Floating Point Format นั้นจะช่วยให้ผู้เขียนโปรแกรมเพิ่มความระมัดระวังในการดำเนินการจัดเก็บและการเขียนคำสั่งในการคำนวณรวมทั้งการควบคุมการทำงานชนิด Repetition ด้วย เช่นการขยายขนาดของเลขจำนวนจริงให้เพิ่มขึ้น โดยบางระบบยอมให้ขยายเพิ่มขึ้นอีกเท่าตัว คือ ถ้าเป็น Real จะให้ 4 Bytes แต่ถ้าเป็น Double จะเพิ่มให้อีกเป็น 8 Bytes เป็นต้น หรืออาจจะขยายให้เป็น Extended ได้คือมีขนาดถึง 16 Bytes และในกรณีของเลขจำนวนเต็มเองก็เช่นกัน ถ้าเรากำหนด เป็น Integer และมีขนาดเล็กเกินไปเราก็อาจจะขอให้ขยายเป็น longint (Loninteger) ได้ซึ่งจะได้พื้นที่เพิ่มอีกเท่าตัวของ Integer

ข้อมูลเชิงตรรกะ หรือข้อมูลแบบบูลลีน (Boolean Data Type)

ข้อมูลเชิงตรรกะจะหมายถึงข้อมูลที่มีสภาพเป็นสัญญาณ True หรือเป็น False เท่านั้น โดยที่สัญญาณนี้เรานำไปใช้ในการควบคุมการทำงานของการควบคุมเช่น การคัดเลือก (Selection) หรือการควบคุมการทำงานซ้ำๆ เช่น While หรือ Repeat ... Until ลักษณะของข้อมูลในตัวแปรที่เป็น Boolean Data Type นั้นอาจกำหนดไว้เป็น 0 หรือ 1 เช่น กำหนดให้ Test : Boolean ; แล้วภายหลังกำหนดให้ Test := True

ดังนั้นข้อมูลของสัญญาณ True ในตัวแปร Test อาจจะปรากฏดังนี้ 00000001 และ ถ้า Test := false ดังนั้นข้อมูลของสัญญาณ False ในตัวแปร Test อาจจะปรากฏ ดังนี้ 00000000 เป็นต้น

ตัวอย่างของการใช้ตัวแปรแบบตรรกะ

```
If Test then
        Call procedure Process A
Else    Call procedure Process B
```

ตัวอย่างการสร้างและการใช้ตัวแปรแบบตรรกะ โดยใช้ภาษาปาสคาล

```
Program    Show  ;
Var
        X , Y : integer ;
        C     : Boolean;
Begin
        Write (" Input X and Y");
        Readln ( X, Y) ;
        C  : =  X  > Y ;
        If  C  Then Writeln (" X is Greater than Y ")
        Else   Writleln (" X  is  Equal   or Less Than Y " );
        Readln
End .
```

การใช้ตัวแปรเชิงตรรกะจะมีประโยชน์ต่อการเขียนโปรแกรมในส่วนของการควบคุม ซึ่งอาจจะมีการเปลี่ยนแปลงบ่อยๆครั้งโดยที่ทำให้การปรับปรุงโปรแกรมจะทำได้ง่ายเข้า

ตัวอย่าง เช่นถ้าในระบบโปรแกรมหนึ่ง มีการควบคุมด้วยเงื่อนไขว่า ถ้าอายุของคนงานมากกว่า 35 และ เงินเดือนสูงกว่า 6500 บาท แล้วจะส่งผลให้มีการสั่งให้ดำเนินงานบางอย่างดังลักษณะการเขียนโปรแกรมต่อไปนี้

```
Program Demo ;
Var
        Age : Integer ;
        Salary : Longint ;


Begin

        If (Age > 35 and Salary >= 6500 ) Then
                Begin


                End ;

        While (Age > 35 and Salary >= 6500 ) Do
                Begin


                End ;

End.
```

เราอาจเขียนโปรแกรมเสียใหม่โดยการใช้ตัวแปรตรรกะเข้าช่วยดังนี้คือ

```
Program Demo ;
Var
Age : Integer ;
Salary : Longint ;
Test : Boolean ;

Begin
    Test := Age > 35 and Salary >= 6500 ;
    If Test Then
        Begin


        End;

    While (Test ) Do
        Begin


        End;

End
```

การเขียนโปรแกรมลักษณะดังกล่าวที่ปรับใหม่นี้ โดยใช้ตัวแปรตรรกะเข้าช่วยส่งผลให้การเขียนโปรแกรมสั้นเข้า และในกรณีที่จะมีการเปลี่ยนแปลงเงื่อนไขก็สามารถกระทำได้ง่าย โดยเพียงแต่ปรับปรุงที่ คำสั่ง Test1 := Age > 35 and Salary >= 6500 ; เพียงคำสั่งเดียวเท่านั้น ไม่ต้องกระทำหลายๆคำสั่ง ที่ควบคุมด้วยเงื่อนไขดังกล่าวนี้ เพราะถ้าเราแก้ไขไม่หมดก็จะเกิดความผิดพลาดได้

ภาษาบางภาษาเช่นภาษา C การกำหนดข้อมูลที่ เป็น True หรือ False อาจจะกำหนดว่า ถ้าข้อมูลในตัวแปรแบบตรรกะใดที่มีค่าทุก Bit ในตัวแปรเป็น 0 หมดให้ถือว่าเป็น False แต่ถ้า Bit ใด Bit หนึ่งเป็น 1 ให้ถือว่าเป็นสัญญาณของ True ก็ได้

ข้อมูลเชิงอักขระ (Character Data Type)

ข้อมูลเชิงอักขระ (Character Data Type) จะหมายถึงข้อมูลที่เป็นอักขระหนึ่งอักขระ ซึ่งจะใช้พื้นที่ในหน่วยความจำเท่ากับ 1 Byte โดยที่ระบบรหัสที่จัดเก็บบนระบบเครื่องไมโครคอมพิวเตอร์จะใช้ ระบบ ASCII Code ระบบที่ใช้กันทั่วๆไป คือ "One Byte Character " ในกรณีของการภาษาอังกฤษนั้นจะไม่มีปัญหาเกิดขึ้น แต่สำหรับระบบที่ใช้ภาษาอื่น เช่นภาษาจีน นั้นจะมีปัญหาในกรณีที่จำนวนอักขระจะไม่พอใช้งาน ดังนั้นจึงได้มีการพยายามพัฒนาระบบที่เรียกว่า "Two Bytes Character " ขึ้นมาใช้งาน นั่นหมายความว่าในระบบนั้นจะใช้ 2 Bytes สำหรับ 1 ตัวอักขระ

สำหรับระบบ " One Byte Character " มีหลักการในการสร้างดังนี้คือ ใช้ 1 Byte ที่ประกอบด้วย 8 Bits ในการสร้างหนึ่งอักขระ ซึ่งความสามารถในการสร้างจำนวนอักขระที่เป็นไปได้คือ $2^8 = 256$ อักขระ โดยที่อักขระเหล่านี้จะแบ่งออกเป็นกลุ่มย่อยๆดังนี้คือ

- **กลุ่มที่ 1** เป็นอักขระที่ไม่ปรากฏให้เห็น แต่ใช้เป็นอักขระควบคุม เรามักจะใช้อักขระกลุ่มนี้ในการควบคุมระบบ (System Program) เช่นการจัดการเรื่องอุปกรณ์ เช่น FF , Cls , Bell เป็นต้น
- **กลุ่มที่ 2** เป็นรหัสที่เป็นตัวอักขระที่ปรากฏให้เห็น เช่นตัวอักษร 'A' ตัวเลข '1' หรืออักขระพิเศษต่างๆเช่น 'ไ" ที่ใช้ในภาษาไทย เราจะแบ่งเป็น 3 กลุ่มย่อย ดังนี้ คือ
  - **กลุ่มที่ 2.1** ตัวอักษรภาษาอังกฤษ (Upper Case and Lower Case) อันประกอบด้วย 'A','B",....'a',...'z'
  - **กลุ่มที่ 2.2** ตัวเลข ประกอบด้วยอักขระที่เป็นตัวเลข คือ 0 ,1,...,9
  - **กลุ่มที่ 2.3** เป็นอักขระพิเศษ เช่น เครื่องหมาย หรือตัวอักขระพิเศษ หรือตัวอักขระที่เป็น Graphic เช่น '#' , '*' , ...,'ค',.....

การเก็บรหัสที่เป็นอักขระนี้จะเก็บตามข้อตกลงของระบบเช่นระบบรหัส ASCII จะเก็บตัวอักษร 'A" ที่ 65 และ 'B' ที่ 66 ด้วยวิธีการเช่นนี้จึงทำให้การทดสอบเป็นไปลักษณะของมูลค่าการจัดเก็บของรหัส ดังนี้คือ

'0' < '1' < '2' < '3' < ....< '9'

'A' < 'B' < 'C' < 'D' < ....< 'Z'

'a' < 'b' < 'c' < 'd' < ....< 'z'

'A' < 'a'

ตัวแปรกลุ่มนี้อันประกอบด้วยตัวแปรเชิงตัวเลขข้อมูลเชิงตรรกะ ข้อมูลตัวอักขระ อาจจะจัดอยู่ในกลุ่มของข้อมูลที่เรียกว่า Ordinal Data Type ซึ่งหมายความว่าการจัดเก็บข้อมูลในกลุ่มนี้จะมีการจัดเรียงลำดับ ซึ่งส่งผลให้การจัดการข้อมูลเหล่านี้สามารถใช้ข้อตกลงหรือคำสั่งพิเศษเข้าช่วยในการดำเนินการได้เช่นใช้ คำว่า MAXINT เป็นค่าสูงสุดของระบบเลขจำนวนเต็มของระบบได้ หรือการใช้คำสั่งพิเศษบางอย่างเช่น Ord , Pred ในภาษาปาสคาล เพื่อดำเนินการเกี่ยวกับข้อมูล ได้ ตัวอย่างเช่น Ord (True) = 1 และ Ord (True) = 1เป็นต้น ส่วนการเรียงลำดับของตัวแปรที่เป็นตรรกะนั้นจะกำหนดให้ เป็นลำดับดังนี้คือ False ,True (นั่นคือ False มีค่าน้อยกว่า True)

ตัวอย่างการใช้คำสั่ง Ord , Pred . Succ ในภาษาปาสคาล

Parameter	**Ord**	**Succ**	**Pred**
15	15	16	14
0	0	1	-1
Flase	0	True	Undefined
MaxInt	MaxInt	Undefined	MaxInt -1
‘C’	67	‘0’	‘B’
‘c’	99	‘d’	‘b’
‘0’	48	‘1’	‘/’
‘7’	55	‘8’	‘6’
‘ ‘	32	‘!’	Unprintable

ตัวอย่างการเขียนคำสั่งในการใช้ฟังก์ชันในการดำเนินงานของตัวอักขระ

```
UpCase := Chr(Ord(‘c’) – Ord(‘a’) |+ Ord(‘A’))
```

โปรแกรมตัวอย่างในการแสดงผลตัวอักขระ

```
Program  Collate  (Input,  Output);
Const
        Min = 32 ;
        Max = 90 ;
Var
        NextOrd : Integer ;
Begin
        For NextOrd := Mii to Max do
                Write (Chr(NextOrd)) ;
                Writeln ;
        End.
```

คำถามท้ายบท

1. ฟังก์ชันต่อไปนี้จะส่งผลกลับไปให้ผู้เรียกเป็นอะไร ถ้ากำหนดให้ Argument คือ 5,7

```
Function ThisDoesWhat(First,Second:Integer):Boolean;
Begin
        ThisDoesWhat := (First div Second) = 0
End ;
```

2. กำหนดให้ นิพจน์คณิตศาสตร์ 3* (1.0/3.0) ท่านคิดว่าค่าที่คำนวณได้จะมีค่าเท่ากับ 1 หรือไม่ ถ้าไม่ได้ค่าดังกล่าวที่คำนวณได้คืออะไร
3. ตัวแปรที่เป็นข้อมูลประเภทจำนวนเต็มจะมีข้อดีที่เหนือกว่าเลขจำนวนจริงคืออะไร
4. กำหนดให้ ตัวแปรเลขจำนวนเต็มของระบบเครื่องระบบหนึ่งถูกกำหนดให้มีขนาดเท่ากับ 2 Bytes
   - ตามเงื่อนไขนี้ถ้ากำหนดว่าตัวแปรนั้นสามารถจัดเก็บได้ทั้งเลขจำนวนเต็มที่เป็นทั้งบวกทั้งลบแล้ว ตัวแปรนั้นจะมีค่า ต่ำสุด และสูงสุดเป็นเท่าไรในการจัดเก็บ
   - ตามเงื่อนไขนี้ถ้ากำหนดว่าตัวแปรนั้นสามารถจัดเก็บได้ทั้งเลขจำนวนเต็มที่เป็นเฉพาะเลขบวกแล้ว ตัวแปรนั้นจะมีค่า ต่ำสุด และสูงสุดเป็นเท่าไรในการจัดเก็บ
5. ในกรณีที่เราจะเก็บเลขจำนวนเต็มที่มีค่าสูงกว่าที่ระบบนั้นกำหนดไว้คือ 2 Bytes เราจะสามารถจัดการเรื่องดังกล่าวได้อย่างไร
6. จงอธิบายถึงวิธีการจัดเก็บเลขจำนวนจริงหนึ่งจำนวนคือ 5.342 ในรูปแบบฐานสองที่เป็น Normalized Form โดยมีข้อกำหนดว่าให้เป็น Mantissa 3 Bytes และ Exponent 1 Byte
7. ภายใต้ข้อกำหนดในข้อ 6. นิพจน์คณิตศาสตร์ต่อไปนี้จะให้ค่าเป็นเท่าไร และผิดพลาดไปจากค่าจริงเท่าไร

   .000321 + 34567.345
8. ข้อผิดพลาดที่เกิดขึ้นในข้อ 7 .นั้นมีสาเหตุความผิดพลาดเกิดจากอะไรบ้าง
9. เลข จำนวนจริง 1.00 จะเก็บภายในสมองเครื่องคอมพิวเตอร์เป็นค่าใด

10. สาเหตุของ RunTime Error นั้นส่วนหนึ่งจะเกิดจากเรื่องของการในเชิงคณิตศาสตร์ ให้อธิบายข้อความนี้ พร้อมกับยกตัวอย่าง