บทที่ 1
บทนำ

1.1 ทำไมต้องเรียนรู้ภาษา C

ความจริงแล้วเราเรียนรู้ภาษาเพียง 1-2 ภาษาก็นับว่าเพียงพอแล้ว แต่เนื่องจากภาษา C เป็นภาษาที่กำลังได้รับความสนใจจากวงการคอมพิวเตอร์ เนื่องจากเป็นภาษาที่ใช้ร่วมกันได้กับเครื่องคอมพิวเตอร์ ตั้งแต่ระดับเมนเฟรม จนถึงระดับไมโคร และมีข้อได้เปรียบหลายประการดังต่อไปนี้ซึ่งอาจถือไว้เป็นประเด็นสำคัญที่ชวนให้น่าศึกษาภาษาดังกล่าว

1. ภาษา C มีตัวดำเนินการ (operator) และคอมมานด์ให้ใช้มากทำให้เราสามารถพัฒนาโปรแกรมต่าง ๆ ได้ง่าย ไม่ติดขัด เนื่องจากมีเครื่องสนับสนุนพร้อมมูล

2. ภาษา C เป็นภาษาที่เอื้อเฟื้อให้สามารถวิ่งโปรแกรมบนคอมพิวเตอร์ต่าง

รุ่นต่างขนาดกันได้โดยอาจแก้ไขโปรแกรมเพียงเล็กน้อย หรืออาจไม่ต้องแก้ไขเลย ถ้าใช้คอมไพเลอร์ภาษา C แบบเดียวกัน เช่น คอมไพเลอร์ภายใต้ CP/M หรือ Z80 หรือ UNIX หรือคอมไพเลอร์สำหรับเครื่อง IBM

3. ภาษา C ทำคำสั่งต่าง ๆ (execution) ได้รวดเร็วมากอย่างน่าพิศวง เช่น โปรแกรมเพิ่มค่าตัวแปร x จาก 0 ถึง 30000 ดังต่อไปนี้

```
/* increment variable to 30000 */
# define BELL 7 /* ASCII code for terminal bell */
main ( )
  {
   int x;
   putchar (BELL);
   x = 0;
   while (x!=30000)
         ++x;
   putchar (BELL);
   printf(" loop finished \n at the bell.");
  }
```

โปรแกรมนี้ถ้าเขียนเป็นภาษาเบสิคจะใช้เวลาทำคำสั่งนานถึง 96 วินาที แต่ในภาษา C จะใช้เวลาทำคำสั่งเพียง 2 วินาทีเท่านั้น

เกี่ยวกับโปรแกรมข้างต้นมีข้อที่ควรแนะนำดังนี้

1. ข้อความภายในเครื่องหมาย /* */ เช่น /* increment - variable to 30000 */ หมายถึง Comment หรือ remark

2. # define BELL 7 คำสั่ง # difine ใช้สำหรับนิยาม Symbolic constant รูปของคำสั่งปรากฏดังนี้คือ

```
# define NAME xxxxx
```

NAME หมายถึง symbol หรือ Symbolic constant ที่เราต้องการใช้แทน และ xxxxx หมายถึงสตริง (string) ดังนั้นคำสั่ง# define BELL 7 จึงเป็นคำสั่งให้ BELL (คือให้เครื่องส่งเสียงออด) โดยให้คอมไพเลอร์นำ ASCII 7 แทนลงในโปร-แกรมเมื่ออ่านพบคำว่า BELL ทั้งนี้ ASCII 7 คือเสียงออด

3) putchar (BELL) เฉพาะ putchar(c) เป็นฟังก์ชันใน Compiler library ใช้สั่งให้พิมพ์อักขระลงบนจอภาพ ในที่นี้ c เป็น argument (ตามคำสั่ง BELL คือ argument)

4) x ! = 30000 หมายถึง $x \neq 30000$ และ ++ x หมายถึง x = x+1 หรือ NEXT x และ printf ( ) เป็นฟังก์ชันให้พิมพ์ข้อความบนจอภาพ \ n หมายถึงให้ขึ้นบรรทัดใหม่ ดังนั้นฟังก์ชัน printf (" loop finished\n at the bell."); จะพิมพ์ดังนี้

```
loop  finished
at  the  bell
```

4. ภาษา c ผูกพันตัวเองไว้ด้วยโปรแกรมโครงสร้างจึงเป็นการบังคับให้เราต้องทำงานเป็น modul หรือ block หรือฟังก์ชัน การที่ภาษา c มีลักษณะเช่นนี้ทำให้การโปรแกรมกระทำตามลักษณะของโมดูล คือ เชื่อมโยงฟังก์ชันเข้าด้วยกันด้วยอากิว-เมนต์ (argument) การตรวจและแก้ไขโปรแกรมจึงกระทำได้ง่าย

5. ภาษา c เป็นภาษาที่น่าสนุก เพราะเป็นภาษาที่มีลักษณะของmodular approach คือเชื่อมโยงฟังก์ชันต่าง ๆ เข้าเป็นโปรแกรมด้วยอาร์กิวเมนต์ ถ้าเราไม่สนใจฟังก์ชันใน Compiler library เราก็นิยามฟังก์ชันขึ้นใช้เอง โดยเก็บ (save) ไว้ในดิสก์ เป็นภาษา c ของเราเอง นับว่าท้าทายความสามารถไม่น้อยทีเดียว

1.2 ลักษณะของโปรแกรมภาษา C

โปรแกรมในภาษา C ก็คือกลุ่มของฟังก์ชันที่เชื่อมโยงรับ-ส่งค่ากันด้วยอาร์กิวเมนต์ หรือแอดเดรส เช่น [1/]

```
/* the C program prints  a massage on sgreen */
main ( )
  {
   printf (" this is my first program. \ n");
  }
```

จากโปรแกรมพบว่าเราใช้ฟังก์ชันคือ main( ) กับ printf ( ) และมีหมายเหตุเป็นข้อความอยู่ในเครื่องหมาย /* */ ฟังก์ชัน main ( ) เป็นฟังก์ชันที่ใช้แจ้งให้คอมไพเลอร์รับทราบว่าให้เริ่มทำคำสั่ง (execution) ที่นี่หรือ ณ ที่นี้เป็นต้นไป ทุกโปรแกรมในภาษา C จึงต้องมีฟังก์ชันนี้ โดยเราจะต้องมีเพียงครั้งเดียว คือ แจ้งให้คอมไพเลอร์ทราบว่าจุดเริ่มต้นของโปรแกรมอยู่ที่ใดหากสั่ง main ( ) หลายครั้งคอมไพเลอร์จะไม่ทราบว่าจะเริ่มต้นโปรแกรมที่ใดแน่ วงเล็บ ( ) หลังชื่อ main หรือชื่อ printf เป็นเครื่องบ่งชี้ว่านี่คือฟังก์ชัน วงเล็บปีกกาเปิดใต้ main ( ) แสดงว่าตัวโปรแกรมหรือฟังก์ชันเริ่มต้นเที่นี้เรื่อย ๆ ไปจนถึงวงเล็บปีกกาปิด (ขอให้สังเกตว่าวงเล็บปีกกาเปิดและปิดต้องอยู่ตรงกันในแนวดิ่ง หากมีฟังก์ชันอื่น ๆ ซ้อน ๆ กันอยู่ภายในให้ใช้วงเล็บปีกกาเปิดปิดของแต่ละฟังก์ชันเป็นเครื่องบ่งชี้จุดเริ่มต้นและจุดจบหรือจุดปลายของฟังก์ชันนั้นด้วยวงเล็บปีกกาที่ตรงกันในแนวดิ่ง แต่ควรย่อหน้าคนละจุดมิเช่นนั้นจะตรวจแก้โปรแกรมยาก ย่อหน้าหนึ่งก็คือ-modul หรือ block หนึ่ง) แปลว่าจบโปรแกรม ขอให้สังเกตด้วยว่าภาษา C ไม่มี line number วงเล็บปีกกาเปิดใต้ main( ) จึงเป็นเครื่องบ่งชี้จุดเริ่มต้นของโปรแกรม ขณะที่ภาษาเบสิคใช้ line number ที่มีค่าต่ำสุดเป็นเครื่องบ่งชี้ และใช้วงเล็บปีกกาปิด(หรือ exit(0))ที่ตรงกับวงเล็บปีกกาเปิดใต้ฟังก์ชัน main( )(ที่จริงไม่ต้องตรงกันก็ได้) เป็น

[1/] สามารถเขียนเป็น main( ) {printf(" this is my first 'program .\ n")}

เครื่องบ่งบอกการจบโปรแกรมขณะที่ภาษาเบสิคในคำสั่ง END บ่งบอกการจบโปรแกรม อนึ่ง วงเล็บปีกกาเปิดปิดในภาษา C ยังมีข้อควรทราบอีกด้วยว่า วงเล็บปีกกาเปิดใต้ main ( ) แสดงจุดเริ่มต้นโปรแกรม วงเล็บปีกกาเปิดใต้ฟังก์ชันใด ๆ แสดงจุดเริ่มต้นของฟังก์ชันนั้น ๆ วงเล็บปีกกาของฟังก์ชัน main ( ) แสดงจุดปลายของโปรแกรมหลัก วงเล็บปีกกาปิดใต้ฟังก์ชันใด ๆ (หรือของฟังก์ชันใด ๆ) แสดงจุดปลายของฟังก์ชันนั้น แล้วส่งการควบคุม (Control) ต่อไปยัง main ( ) หรือฟังก์ชันที่เรียกมา คำสั่ง return ในภาษา C จึงไม่ใช่สิ่งจำเป็นเพราะเป็น automatic return อยู่แล้วคือ ในทันทีที่พบวงเล็บปีกกาปิด

อย่างไรก็ตาม เราอาจสงสัยว่าฟังก์ชันอื่น ๆ เช่น printf ( ), putchar ( ), getchar ( ) เริ่มทำงานเมื่อใด มีสัญญาณหรือรหัสใดแจ้งให้เริ่ม และหยุดทำงาน ที่จริงแล้วฟังก์ชันเหล่านี้เป็นฟังก์ชันใน standard library หรือ compiler library ซึ่งได้จัดเตรียมไว้ให้ผู้ใช้เรียบร้อยแล้วในคอมไพเลอร์ (ซึ่งเป็นการอำนวยความสะดวกให้ผู้ใช้เป็นอย่างมาก แต่คอมไพเลอร์ในภาษา C นั้น เขียนขึ้นหลายรุ่น บางรุ่นมีฟังก์ชันน้อยแปลว่าผู้เขียนต้องเขียนฟังก์ชันเองมาก บางรุ่นมีฟังก์ชันมากครบถ้วน - (full feature) ซึ่งก็จะอำนวยความสะดวกแก่ผู้ใช้มาก และเราอาจจำเป็นต้องเขียนฟังก์ชันที่ต้องการเพิ่มเติมอีกบ้างเล็กน้อย) เมื่อคอมไพเลอร์อ่านโปรแกรม พบฟังก์ชันดังกล่าวก็จะตรวจดูว่ามีฟังก์ชันนั้นใน library หรือไม่ เมื่อตรวจพบก็จะนำรหัสในคอมไพเลอร์ของฟังก์ชันนั้นบรรจุลงในโปรแกรม (เรียกวิธีปฏิบัติเช่นนั้นว่า linker) เรื่องฟังก์ชันนี้จะกล่าวถึงอีกครั้งหนึ่งในบทที่ 4 และ 6 และขอให้สังเกตว่าโดยปกติฟังก์ชันจะไม่ทำงานใด ๆ เลย เว้นแต่เราให้ข้อมูลหรือข้อสนเทศที่เหมาะสมส่งเข้าสู่ฟังก์ชันนั้น ข้อมูลหรือข้อสนเทศนั้นเรียกว่าอาร์กิวเมนต์ (argument) หากมีมากกว่าหนึ่งอาร์กิวเมนต์เรียกว่า argument list เช่นใน printf ( ); อาร์กิวเมนต์คือ " this is my first program\n" หรือ volume (l, h, w) คือฟังก์ชันคำนวณปริมาตร l (ความ

ยาว), h (ความสูง) และ w (ความกว้าง) คือ อาร์กิวเมนต์เรียก l, h, w ว่า argument list ขอให้สังเกตว่าทุกคำสั่งในภาษา C ต้องมีเครื่องหมายอัฒภาค (;) ปิดท้ายเพื่อแสดงการจบคำสั่งนั้น ๆ เสมอ

1.3 สไตล์การโปรแกรม

ภาษา C เป็นภาษาที่ไม่มีหมายเลขบรรทัด(line number) และไม่บังคับว่าแต่ละบรรทัดจะต้องย่อหน้าเท่ากัน วงเล็บปีกกาไม่จำเป็นต้องอยู่ตรงกันเพราะคอมไพเลอร์จะอ่านโปรแกรมเมื่อพบวงเล็กปีกกาเปิดก็จะทราบว่าที่นี่คือจุดเริ่มต้นของโปรแกรมหรือฟังก์ชัน และเมื่ออ่านพบวงเล็บปีกกาปิดคอมไพเลอร์จะทราบว่าที่นี่คือ จุดจบปลายของโปรแกรมหรือฟังก์ชัน ทั้งนี้ภาษา C นิยมเขียนตัวอักษรตัวเล็ก และสงวนอักษรตัวใหญ่ไว้สำหรับ - Symbolic constant และ string แต่เนื่องจากการเขียนคำสั่งที่อิสระเกินไปจะเป็นเหตุให้ตรวจแก้ไขโปรแกรมได้ยากวิธีที่นิยมปฏิบัติในเรื่องวงเล็บปีกกาเปิด-ปิดก็คือ นิยมให้วงเล็บปีกกาเปิด-ปิดของแต่ละฟังก์ชัน หรือ loop อยู่ในแนวตรงกัน ถ้าโปรแกรมใดมีฟังก์ชันและ loop มากก็ให้วงเล็บปีกกาเปิด-ปิด ของแต่ละฟังก์ชัน หรือ loop อยู่ในย่อหน้าที่ต่างกัน หรือจะเขียนข้อความทั้งปวงลงในบรรทัดเดียวกันก็ได้ (ดูเชิงอรรถ 1 ตอน 1.2) ซึ่งไม่ผิดกติกาและคอมไพเลอร์ก็สามารถอ่านได้แต่คงไม่สะดวกในการตรวจแก้

1.4 ตัวแปร

ตัวแปรคือปริมาณที่มีค่าเปลี่ยนไปตามค่าที่กำหนดให้ ในภาษา C นั้น เราจะต้องกำหนดชนิด(type) ของตัวแปรก่อน ตัวแปรในภาษา C มีหลายชนิด เช่น int,char, float, long, double และอื่น ๆ ซึ่งอาจเป็นภาระยุ่งยากอยู่บ้างตอนเริ่มหัดใช้แต่จะ

มีประโยชน์มากในภายหลัง ตัวแปรอาจประกอบไปด้วยตัวอักษร ตัวเลข ขีด (คือ _) ตัวแปรหนึ่งจะมีกี่อักขระได้คือเราพอใจจะใช้ชื่อยาว-สั้นเท่าไรก็ได้ ขอแต่เพียงให้อักขระแรกเป็นตัวอักษร หรือขีด (_) อย่างไรก็ตาม คอมไพเลอร์โดยทั่วไปจะสนใจเฉพาะ 8 อักขระแรก อักขระตัวที่9 เป็นต้นไปไม่มีความหมายอะไร ทั้งนี้จะต้องใช้อักษรตัวเล็กเท่านั้น (อักษรตัวใหญ่สงวนไว้สำหรับ symbolic constant) ส่วนเครื่องหมายขีด (_) เราจะใช้หรือไม่ใช้ก็ได้ โดยมากนิยมใช้เพื่อช่วยขยายชื่อตัวแปรให้สื่อความหมายได้มากขึ้น เช่น is_male, is_female, let_count, delta_let ทั้งยังสามารถช่วยป้องกันมิให้เราใช้ชื่อตัวแปรซ้ำกับชื่อฟังก์ชันใน compiler-library และ keyword ต่อไปนี้อีกด้วย

auto	double	if	static	break	else	int
struct	case	entry	long	switch	char	return
register	typedef	continue	float	return	union	default
for	sizeof	unsigned	do	goto	short	while

แสดงว่าชื่อ auto_type หรือ double_xใช้เป็นชื่อตัวแปรได้เพราะมีเครื่องหมายขีด (_) จำแนกให้เห็นถึงความแตกต่างกับคำว่า auto และ double ไว้แล้ว

1.5 ตัวอย่างโปรแกรม

ขอให้พิจารณาโปรแกรมบวกเลข 2 จำนวนต่อไปนี้

```
/* add two number and print result */
main ( )
    {
     int sum, x, y;
     x = 20;
     y = 30;
     sum = x+y;
     printf (" the sum Of% d and % d = % d", x, y, sum);
    }
```

จากโปรแกรมจะพบว่า

1) /* add two number and print result */ คือหมายเหตุ - (remark หรือ comment) แจ้งให้ผู้อ่านโปรแกรมทราบว่า โปรแกรมนี้เป็นโปรแกรมที่สั่งให้บวกเลข 2 จำนวนเข้าด้วยกัน แล้วพิมพ์ผลบวกออกมา

2) int sum, x, y; เป็นการกำหนดชนิดของตัวแปรชื่อ sum, x และ y ให้เป็นตัวแปรแบบ integer คำสั่งนี้อาจแยกเขียนเป็น

```
int sum;
int x;
int y;
```

การเขียนแบบนี้เรียกว่า screen-oriented text editor ส่วนวิธีแรกคือ int sum, x, y; เรียกว่า line-oriented text editor เราสามารถเลือกใช้วิธีใดก็ได้ขอให้สังเกตว่าเราจะกำหนดชนิด (type) ของตัวแปรไว้ตั้งแต่เริ่มต้นโปรแกรม การกำหนดชนิดของตัวแปรจะมีผลให้มีการจองที่เป็นจำนวนไบท์ไว้ในส่วนความจำตามตัวอย่างนี้ตัวแปรชื่อ sum, x และ y ใช้ที่ 2 ไบท์เท่ากัน

3) x = 20; และ y = 30; และ sum = x+y; เครื่องหมายเท่ากับ (=) เรียกว่า assignment operator x = 20; หมายถึงใส่มูลค่าเท่ากับ 20 ลงในที่ขนาด 2 ไบท์ ชื่อ x y = 30 หมายถึงใส่มูลค่าเท่ากับ 30 ลงในที่ขนาด 2 ไบท์ชื่อ y และ sum = x+y หมายถึงใส่ผลบวกของมูลค่าในที่ชื่อ x และ y ลงในที่ชื่อ sum อนึ่งจากโปรแกรมจะพบว่า x=20 และ y=30 ก็คือการกำหนดค่าเริ่มต้น(initialization) ให้แก่ตัวแปร x และ y นั่นเอง ซึ่งหากเราไม่กำหนดค่าเริ่มต้นไว้เช่นนี้ x และ y จะมีค่าเริ่มต้นเท่ากับ 0 ซึ่งเป็น default ของเครื่อง ขอให้สังเกตว่าเรามิได้กำหนดค่าเริ่มต้นให้แก่ตัวแปรชื่อ sum แสดงว่าเราต้องการอาศัย default

4) ฟังก์ชัน printf( ) ตามตัวอย่างนี้มีรูปที่ละเอียดขึ้นเพราะนอกจากมีอาร์กิวเมนต์ 3 รายการคือ x, y และ sum แล้วยังมี control string รูปไวยากรณ์ของ printf( ) ปรากฏดังนี้คือ

printf (" control string", argument 1, argument 2, ...)

control string จะต้องอยู่ในเครื่องหมายคำพูด แสดงว่า control string อาจหมายถึงข้อความที่เราต้องการให้พิมพ์เช่นต้องการให้พิมพ์ว่า sum of x and y isเราก็ใช้ control string เป็น " sum of x and y is " แต่รูปแบบของ Control string นั้นโดยมากเราจะมีไว้เพื่อกำหนดรูปการพิมพ์ (เหมือน PRINT USING ##...# ในภาษาเบสิค) รูปแบบการพิมพ์ของแต่ละตัวแปรจะปรากฏดังนี้คือ)

%d พิมพ์เป็นจำนวนในฐาน 10

%o พิมพ์เป็นจำนวนในฐาน 8

%x พิมพ์เป็นจำนวนในฐาน 16

%u พิมพ์เป็นจำนวนในฐาน 10 ที่เป็นบวกเสมอ (unsigned)

%e พิมพ์เป็นจำนวนเลขตามแบบ double หรือ float ในรูปแบบ EXP เช่น 1.12E10

%f พิมพ์เป็นจำนวนเลขตามแบบ double หรือ float ในรูปจำนวนเลขฐาน 10

%g พิมพ์ตามแบบ %e หรือ %f ก็ได้ขึ้นอยู่กับว่าแบบใดสั้นกว่า

%c เป็น string คือให้พิมพ์เป็นอักขระเพียง 1 อักขระ

%s เป็น string คือให้พิมพ์เป็นอักขระตั้งแต่ 1 อักขระเป็นต้นไป

ดังนั้น prirtf (" the sum of %d and %d = %d ", x, y, sum) ; จะพิมพ์ดังนี้

สมมุติว่า x=20 และ y=30 จะพิมพ์เป็น

```
the sum of 20 and 30 = 50
```

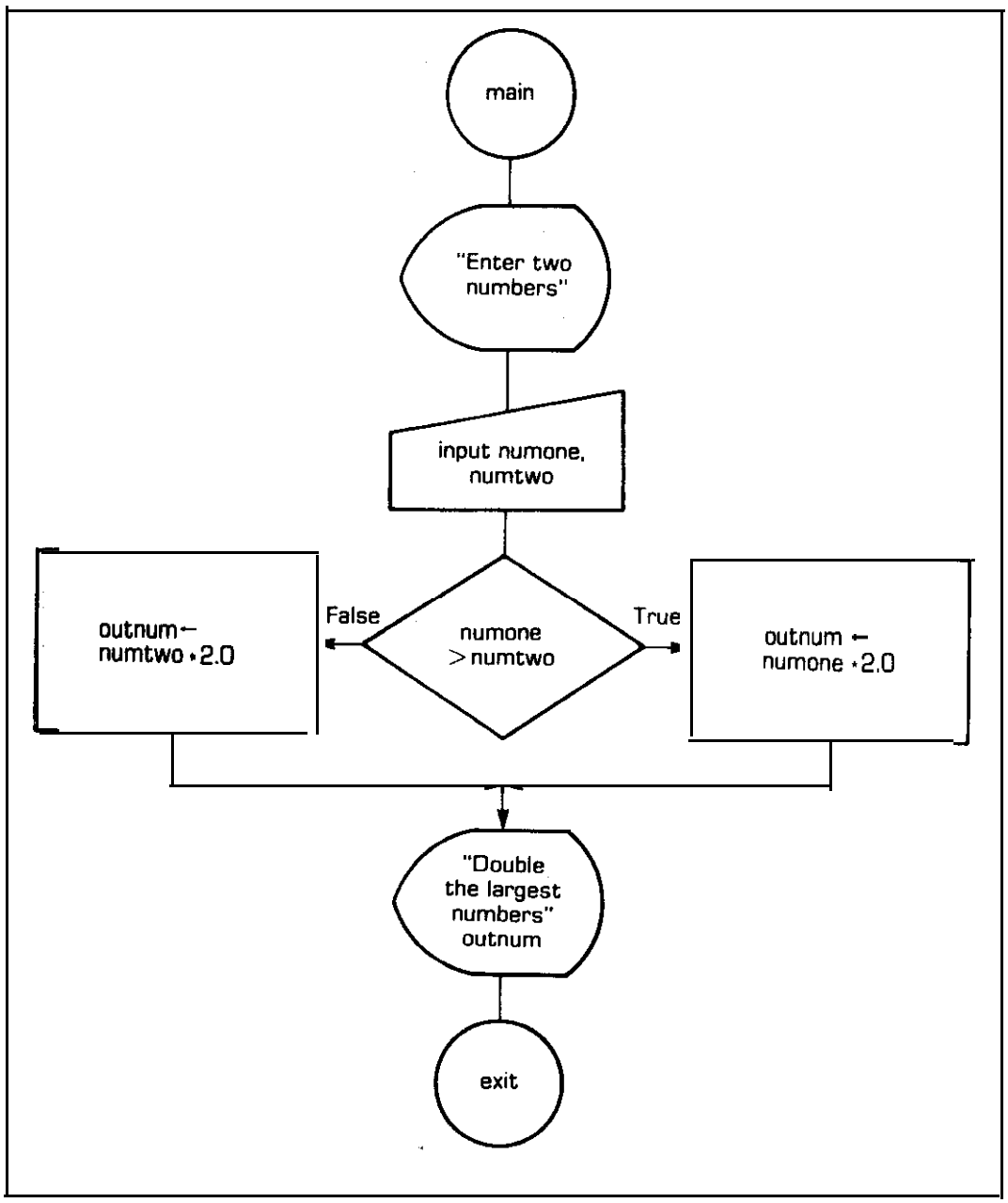
%d เป็นการระบุ format ของ x, y และ sum ว่าเป็นจำนวนในฐาน 10

ลองพิจารณาตัวอย่างโปรแกรมอีกโปรแกรมหนึ่งดังต่อไปนี้ โปรแกรมนี้เป็นโปรแกรมที่ใช้สั่งให้คำเนินการเปรียบเทียบเลข 2 จำนวนแล้วเอา 2 คูณเข้ากับจำนวนที่มีค่าสูงกว่า ดังนี้ ขอให้เป็นที่สังเกตไว้ด้วยว่าภาษา C นั้นเรายึดหลักเกณฑ์ของการโปรแกรมโครงสร้าง (structured programming) เป็นเครื่องมือในการเขียนผังควบคุมและโปรแกรม การเขียนโปรแกรมใช้หลักเกณฑ์ของการทำงานเป็นโมดุลเรียกว่า modular approach ที่เชื่อมโยงโมดุล (ในภาษา C โมดุลก็คือฟังก์ชัน) เข้าด้วยกันด้วยอาร์กิวเมนต์ (argument) ตัวแปรภายนอก (external variable) และตัวแปร pointer โมดุลหรือฟังก์ชันอาจเป็น definition function หรือ compiler library function ก็ได้ ตามตัวอย่างต่อไปนี้เป็นการใช้ฟังก์ชันในคอมไพเลอร์คือ printf ( ) และ scanf ( ) ขอให้สังเกตคำสั่ง if-else ซึ่งเป็นคำสั่งแบบโปรแกรมโครงสร้าง และจะพบว่าเราต้องกำหนดลักษณะตัวแปร (declaration) ไว้เสมอเพื่อเป็นการเตรียมที่ (จำนวนไบท์) ในส่วนความจำให้แก่ตัวแปรนั้นๆเพื่อว่าคอมไพเลอร์จะได้ตามไปอ่านหรือเข้าถึง (access) ณ แอดเดรสที่ข้อมูลปรากฏอยู่ในจำนวนไบท์ที่กำหนด

```
main( )
/* output twice the larger number entered */
{
        float numone,numtwo,outnum;
        printf("enter two number");
        scanf(" %f%f,& numone,& numtwo);
        if(numone > numtwo)
              outnum=outnum*2;
        else
              outnum=numtwo*2;
        printf("numble the largest number is %f,outnum");
        exit(0);
}
```

ในบรรทัดแรกคือ main ( ) แสดงให้เห็นว่านีคือ โปรแกรมหลัก (main program หรือ main routine) ที่จะต้องได้รับการทำคำสั่ง (execute) ก่อนโปรแกรมอื่น หรือฟังก์ชันอื่น บรรทัดที่ 2 คือหมายเหตุ (comment หรือ remark) ส่วนนี้คอมไพเลอร์จะไม่แปล บรรทัดที่ 3 คือวงเล็บปีกกา ({) ชี้ให้เห็นว่าตัวโปรแกรมจะเริ่มต้นที่นี้ ตัวโปรแกรมจะเริ่มด้วยการกำหนดตัวแปร (declaration) และให้ทำงาน ดูที่บรรทัดที่ 4 จะเห็นว่าเรากำหนดลักษณะตัวแปร 3 ตัว คือ numone, numtwo และ outnum เป็น float variable (ดูบทที่ 2 เรื่องการกำหนดลักษณะตัวแปร) บรรทัดที่ 5 คือฟังก์ชัน printf ( ) ใช้สำหรับสั่งให้พิมพ์ตัวแปร หรือข้อความในเครื่องหมายคำพูดออกทางจอภาพ บรรทัดที่ 6 คือฟังก์ชัน scanf ( ) ใช้สำหรับอ่านหรือรับค่าที่ส่งเข้าทางแป้นพิมพ์โดยที่ %f จะบอก format ของข้อมูลเข้าในรูปของ float variable & เรียกว่า address operator กำกับหน้าชื่อตัวแปรใดแปลว่าข้อมูลของตัวแปรนั้นคือแอดเดรส (อ่านบทที่ 6) บรรทัดที่ 7 ถึง 10 คำนวณค่าของตัวแปร outnum เป็น 2 แบบ หรือตามผลการเปรียบเทียบค่าของตัวแปร numone กับ numtwo สำหรับคำสั่ง if-else ให้ดูรายละเอียดในบทที่ 3 ในบรรที่ 11 คือ printf ( ) ใช้สั่งพิมพ์ค่าของตัวแปร outnum ออกทางจอภาพ บรรทัดที่ 12 คือ exit (o) แสดงจุดจบของโปรแกรม และบรรทัดสุดท้ายคือ วงเล็บปีกกาปิด (}) ชี้ให้เห็นว่าตัวโปรแกรมได้จบลงที่นี่

ผังควบคุมของโปรแกรมดังกล่าวปรากฏดังนี้


main
"Enter two numbers"
input numone, numtwo
numone > numtwo
False
True
outnum ← numtwo * 2.0
outnum ← numone * 2.0
"Double the largest numbers" outnum
exit

สำหรับการแปล (compile) และเชื่อมโยง (linking) โปรแกรมนี้ให้สามารถทำงานตามคำสั่งได้ (executable) ให้อ่านภาคผนวก โปรแกรมดังกล่าวสามารถเขียนในภาษาอื่นได้แตกต่างกันไป ตามหลักของภาษานั้น ๆ ลองเปรียบเทียบโปรแกรมนี้ ที่เขียนในภาษา BASIC, MBASIC, FORTRAN, PASCAL, PL/I และ COBOL ดังนี้

BASIC

```
  5 REM OUTPUTS TWICE THE LARGER NUMBER
10 PRINT "ENTER TWO NUMBERS"
20 INPUT N1,N2
30 IF N1>N2 THEN OU=N1*2 : GOTO 50
40 OU=N2*2
50 PRINT "DOUBLE THE LARGEST NUMBER IS",OU
60 END
```

MBASIC

```
  5 REM OUTPUTS TWICE THE LARGER NUMBER
10 PRINT "ENTER TWO NUMBERS"
20 INPUT N1, N2
30 IF N1>N2 THEN OU=N1*2 ELSE OU=N2*2
40 PRINT "DOUBLE THE LARGEST NUMBER IS", OU
50 END
```

FORTRAN

```
C     OUTPUTS TWICE THE LARGER NUMBER
      REAL NUMONE,NUMTWO,
      WRITE(6,50)
50    FORMAT(17HENTER TWO NUMBERS)
      READ (5,100) NUMONE,NUMTWO
100   FORMAT(F10.3,F10.3)
      IF (NUMONE.GT.NUMTWO) GOTO 200
      NUMOUT=NUMTWO*2
      GOTO 300
200   NUMOUT=NUMONE*2
300   WRITE(6,400) NUMOUT)
400   FORMAT(30HDOUBLE THE LARGEST NUMBER IS ,F10.3)
      END
```

PASCAL

```
PROGRAM DOUNUM(INPUT,OUTPUT);
(* OUTPUTS TWICE THE LARGER NUMBER *)
VAR:
  NUMONE,NUMTWO,NUMOUT : REAL;
BEGIN
  WRITE('ENTER TWO NUMBERS');
  READ(NUMONE,NUMTWO);
  IF (NUMONE>NUMTWO) THEN
      NUMOUT: =NUMONE*2
  ELSE
      NUMOUT: =NUMTWO*2;
  WRITE('DOUBLE THE LARGEST NUMBER IS',NUMOUT);
END.
```

PL/I

```
PROCEDURE OPTIONS(MAIN);
/* OUTPUTS TWICE THE LARGER NUMBER */
DECLARE NUMONE,NUMTWO FLOAT;
PUT LIST ('ENTER TWO NUMBERS');
GET LIST (NUMONE,NUMTWO);
  IF (NUMONE>NUMTWO) THEN
  NUMOUT = NUMONE*2;
ELSE
  NUMOUT = NUMTWO*2;
PUT LIST ('DOUBLE THE LARGEST NUMBER IS',NUMOUT);
END;
```

COBOL

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
  DOUNUM.
REMARKS.
  OUTPUTS TWICE THE LARGER NUMBER.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
  SELECT PRINT-FILE ASSIGN TO OUTPUT.
  SELECT READ-FILE ASSIGN TO INPUT.
DATA DIVISION.
FILE SECTION.
FD  PRINT-FILE
    LABEL RECORDS ARE OMITTED.
01  PRINT-LINE PICTURE X (80).
FD  READ-FILE
    LABEL RECORDS ARE OMITTED.
01  IN-REC.
    02 NUMONE   PICTURE 9999999V999.
    02 NUMTWO   PICTURE 9999999V999.
WORKING-STORAGE SECTION.
01  ASK-LINE
    02 FILLER PICTURE X (17) VALUE "ENTER TWO NUMBERS"
01  OUT-LINE.
    02 FILLER PICTURE X(30) VALUE "DOUBLE THE LARGEST NUMBER IS"
    02 NUMOUT PICTURE 999999.999.
PROCEDURE DIVISION.
  OPEN INPUT READ-FILE
    OUTPUT PRINT-FILE.
  WRITE PRINT-LINE FROM ASK-LINE.
  READ READ-FILE.
  IF NUMONE IS GREATER THAN NUMTWO
    COMPUTE NUMOUT = NUMONE*2.
  ELSE
    COMPUTE NUMOUT = NUMTWO*2.
  WRITE PRINT-LINE FROM OUT-LINE.
  CLOSE READ-FILE PRINT-FILE.
  STOP RUN.
```