

# บทที่ 1

## แนะนำโปรแกรมภาษา C

### (Introduction to C programming)

ภาษา C เป็นภาษาที่มีโครงสร้างแบบง่าย ไม่มีข้อจำกัดมากมายในการเขียนโปรแกรม รวมทั้งเป็นภาษาที่มีรูปแบบอิสระ (free-format) หมายความว่า เราสามารถเริ่มต้นเขียน โปรแกรม บนตำแหน่งใด ๆ ในแต่ละบรรทัดได้อย่างอิสระ ในบทนี้จะกล่าวถึงการเขียนโปรแกรมภาษา C ในแบบง่าย ๆ รวมทั้งแสดงโครงสร้างของภาษา C ด้วย

**ตัวอย่างโปรแกรมที่ 1.1** เป็นโปรแกรมที่ใช้ฟังก์ชัน printf พิมพ์ข้อความว่า welcome to C! ออกสู่จอภาพ

```
1  /* A first program in C */
2  #include <stdio.h>
3
4  /* function main begins program execution */
5  int main ()
6  {
7      printf ( "Welcome to C!\n" );
8
9      return 0; /* indicate that program ended successfully */
10
11 } /* end function main */
```

Welcome to C!

**หมายเหตุ** ในการพิมพ์โปรแกรมไม่ต้องมีหมายเลขหน้า แต่ในที่นี้จะใช้ในการอ้างอิง

จากบรรทัดที่ 1 /\* A first program in C \*/ จะมีเครื่องหมาย /\* นำหน้า และเครื่องหมาย \*/ อยู่หลังข้อความ ซึ่งเครื่องหมายดังกล่าวในภาษา C จะเรียกว่า หมายเหตุ (comment) ซึ่งจะมีประโยชน์คือ ช่วยอธิบายรายละเอียดของโปรแกรม เช่น โปรแกรมนี้ใครเป็นคนเขียน เขียนเมื่อไร

มีจุดประสงค์อะไร เป็นต้น ข้อความที่อยู่ระหว่างเครื่องหมาย /\* และ \*/ คอมไพเลอร์ จะไม่สนใจ นำมาแปลความหมาย และไม่มีการเปลี่ยนเป็นภาษาเครื่อง

บรรทัดที่ 2 `#include <stdio.h>` เป็นตัวชี้ทาง (directive) ไปยัง 프리โปรเซสเซอร์ ของ C (C preprocessor) ซึ่งจะมีเครื่องหมาย # นำหน้า ซึ่งจะเป็นสัญลักษณ์บอกให้คอมไพเลอร์รู้ว่า จะต้องประมวลผลส่วนนี้ก่อนที่โปรแกรมจะถูกแปลโดยการไปนำเนื้อหาของส่วนหัวมาตรฐาน นำข้อมูลเข้า/ออก (stdio.h) เข้ามารวมกันในโปรแกรม ซึ่งส่วนหัวนี้ จะบรรจุรายละเอียดที่ถูกใช้โดย คอมไพเลอร์ เมื่อมีการแปลใน โปรแกรมนี้มีการเรียกใช้ฟังก์ชัน printf จึงต้องมีการระบุ `#include <stdio.h>` ซึ่งเป็นส่วนที่อยู่ในไลบรารี (library)

บรรทัดที่ 4 เป็นหมายเหตุ (comment)

บรรทัดที่ 5 `int main()` ในภาษา C จะถือว่าเป็นฟังก์ชันหลักของโปรแกรม กล่าวคือ ทุก ๆ โปรแกรมในภาษา C จะต้องมีฟังก์ชัน main() เสมอ โดยปกติโปรแกรมภาษา C อาจจะมี 1 ฟังก์ชันหรือมากกว่า 1 ฟังก์ชันก็ได้ แต่ทุกโปรแกรมต้องมีฟังก์ชัน main() ซึ่งจะมีการกระทำการ (execute) ที่ฟังก์ชัน main ก่อนเสมอ

บรรทัดที่ 6 { เป็นเครื่องหมายที่ระบุจุดเริ่มต้น โปรแกรมและต้องใช้คู่กับเครื่องหมาย } ซึ่งอยู่ในบรรทัดที่ 11

บรรทัดที่ 7 `printf( "Welcome to C!\n" );` เป็นคำสั่งที่บอกให้คอมไพเลอร์พิมพ์ ข้อความที่ถูกปิดล้อมด้วยเครื่องหมาย “ ” คือ Welcome to C! ออกสู่จอภาพ แต่เครื่องหมาย `\n` จะไม่ถูกพิมพ์ออกมา เนื่องจากเป็นเครื่องหมายที่บอกว่าจะขึ้นบรรทัดใหม่ (newline) ซึ่งเครื่องหมาย `\n` นี้ เราจะเรียกว่าลำดับ หลีก (escape sequence) และท้ายคำสั่ง printf จะมีเครื่องหมาย semicolon (;) ซึ่งเป็นเครื่องหมายที่ใช้ระบุว่าเป็นจุดสิ้นสุดของข้อความสั่ง (statement)

บรรทัดที่ 9 `return 0; /* indicate that program ended successfully */` จะเป็นข้อความสั่ง สุดท้ายในฟังก์ชัน main ที่จะบอกว่า สิ้นสุดโปรแกรมหลัก และค่าเลข 0 หมายความว่า โปรแกรมนี้มีการประมวลผลโดยไม่มีข้อผิดพลาด

บรรทัดที่ 11 } เป็นเครื่องหมายที่ระบุว่าเป็นจุดสิ้นสุดของโปรแกรม

จากตัวอย่างโปรแกรมที่ 1.1 สามารถสรุปได้ดังนี้

#include <stdio.h>	จะบอกให้คอมไพเลอร์นำเพิ่มมาตรฐานนำข้อมูลเข้า/ออก (stdio.h) เข้ามารวมกับโปรแกรม
main	จะเป็นเครื่องหมายบอกว่าจะต้องเริ่มการทำงาน (execute) ที่ฟังก์ชันนี้ก่อน
()	จะต้องอยู่หลัง main เสมอ
/* */	เป็นหมายเหตุที่ช่วยให้เราหรือผู้อ่านโปรแกรมเราได้ชัดเจนยิ่งขึ้น
;	ทุก ๆ ข้อความสั่ง (statement) ในภาษา C จะต้องมีเครื่องหมาย ; ต่อท้าย
{ }	เป็นวงเล็บปีกกาที่ต้องมีในโปรแกรม C เพื่อระบุจุดเริ่มต้น และจุดสิ้นสุดคำสั่งของโปรแกรม

**ตัวอย่างโปรแกรมที่ 1.2** เป็นโปรแกรมที่พิมพ์ข้อความ Welcome to C! ออกสู่จอภาพ เหมือนกับตัวอย่างโปรแกรมที่ 1.1 เพียงแต่จะใช้คำสั่ง printf 2 ครั้ง

```

1  /* Printing on one line with two printf statements */
2  #include <stdio.h>
3
4  /* function main begins program execution */
5  int main ()
6  {
7      printf ( "Welcome" );
8      printf ( "to C!\n" );
9
10 return 0; /* indicate that program ended successfully */
11
12 } /* end function main */

```

Welcome to C!

จากตัวอย่างนี้ บรรทัดที่ 7 จะพิมพ์ข้อความ Welcome แล้วตามด้วยอักขระว่าง แต่เครื่องหมายเคอร์เซอร์ยังอยู่ที่ตำแหน่งท้ายของอักขระว่าง

บรรทัดที่ 8 จะพิมพ์ข้อความ to C! ต่อจากอักขระว่างในบรรทัดเดียวกัน แล้วขึ้นบรรทัดใหม่ โดยเคอร์เซอร์จะอยู่ที่ตำแหน่งแรกของบรรทัดใหม่

**ตัวอย่างโปรแกรมที่ 1.3** เป็นโปรแกรมการพิมพ์ข้อความหลายบรรทัด แต่ใช้คำสั่ง printf คำสั่งเดียว ซึ่งอาศัยลำดับหลักมาช่วย

```

1  /* Printing multiple lines with a single printf */
2  #include <stdio.h>
3
4  /* function main begins program execution */
5  int main ()
6  {
7      printf ( “Welcome\nto\nC!\n” );
8
9      return 0; /* indicate that program ended successfully */
10
11 } /* end function main */

```

Welcome to C!
---------------------

บรรทัดที่ 7 จะพิมพ์ข้อความ Welcome แล้วขึ้นบรรทัดใหม่ แล้วพิมพ์ข้อความ to แล้วขึ้นบรรทัดใหม่ แล้วพิมพ์ข้อความ C! แล้วขึ้นบรรทัดใหม่

**ตัวอย่างโปรแกรมที่ 1.4** เป็นโปรแกรมแสดงการบวกจำนวนเต็ม 2 จำนวน โดยให้ผู้ใช้ป้อนข้อมูลทางแป้นพิมพ์ โดยใช้ฟังก์ชันมาตรฐาน scanf แล้วนำมาบวกกัน และนำผลบวกที่ได้แสดงผลที่จอภาพ โดยใช้คำสั่ง printf ซึ่งในโปรแกรมนี้นี้จะต้องมีการใช้ตัวแปรจำนวน 3 ตัว คือ integer1, integer2, และ sum เป็นชนิดจำนวนเต็ม

```

1  /* Addition program */
2  #include <stdio.h>
3
4  /* function main begins program execution */
5  int main ()
6  {
7      int integer1 ; /* first number to be input by user */
8      int integer2 ; /* second number to be input by user */
9      int sum      /* variable in which sum will be stored */
10
11     printf ( "Enter first integer\n" ); /* prompt */
12     scanf ( "%d" , &integer1 );      /* read an integer */
13
14     printf ( "Enter second integer\n" ); /* prompt */
15     scanf ( "%d" , &integer2 );      /* read an integer */
16
17     sum = integer1 + integer2; /* assign total to sum */
18
19     printf ( "Sum is %d\n" , sum ); /* print sum */
20     return 0; /* indicate that program ended successfully */
21
22 } /* end function main */

```

```

Enter first integer
45
Enter second integer
72
Sum is 117

```

ตัวอย่างนี้ บรรทัดที่ 7-9

```

int integer1; /* first number to be input by user */

int integer2; /* second number to be input by user */

int sum;      /* variable in which sum will be stored */

```

จะเป็นการนิยามชื่อ integer1, integer2 และ sum ซึ่งเราจะเรียกว่าเป็นการประกาศตัวแปรให้ integer1, integer2 และ sum เป็นตัวแปรที่มีชนิดข้อมูลแบบจำนวนเต็ม เพื่อบอกให้คอมพิวเตอร์ รู้ล่วงหน้าว่าในโปรแกรมของเราจะมีการใช้ตัวแปรดังกล่าวให้เตรียมของเนื้อที่ให้กับตัวแปร 3 ตัวนี้ด้วย

จริง ๆ แล้วการประกาศตัวแปรในบรรทัดที่ 7-9 เราสามารถเขียนในบรรทัดเดียวกันได้เลย เช่น `int integer1, integer2, sum;` เพียงแต่การประกาศในบรรทัดที่ 7-9 จะทำให้เราสามารถใส่หมายเหตุให้กับแต่ละตัวแปรได้เท่านั้นเอง ส่วนในรายละเอียดการตั้งชื่อตัวแปร และการประกาศชนิดของตัวแปร จะกล่าวในบทถัดไป

บรรทัดที่ 11 `printf(“Enter first integer\n”); /* prompt */` จะเป็นการพิมพ์ข้อความ `Enter first integer` ออกสู่จอภาพ และมีเคอร์เซอร์อยู่ตำแหน่งแรกของบรรทัด ถัดไป ข้อความดังกล่าวเราจะเรียกว่า ข้อความพร้อมรับ (prompt) ซึ่งจะเป็นตัวระบุให้ผู้ใช้กระทำบางอย่าง

บรรทัดที่ 12 `scanf(“%d”, &integer1); /* read an integer */` จะเป็นการรับข้อมูลทางแป้นพิมพ์ โดยใช้ฟังก์ชัน `scanf` ซึ่งฟังก์ชัน `scanf` จะมีอาร์กิวเมนต์ 2 ตัว คือ “%d” และ `&integer1` โดยอาร์กิวเมนต์ตัวแรกเป็นส่วนที่เรียกว่ารูปแบบควบคุมสายอักขระ (format control string) ซึ่งใช้ระบุชนิดของข้อมูลที่ให้ผู้ใช้ป้อนเข้ามา ส่วน %d เป็นตัวระบุการแปลงผันจำนวนเต็ม (conversion specifier) เพื่อใช้ระบุว่า ข้อมูลที่ป้อนเข้ามาต้องเป็นชนิดจำนวนเต็ม โดยตัวอักษร d ใช้แทนคำว่า decimal integer (จำนวนเต็มฐานสิบ) ส่วนอาร์กิวเมนต์ตัวที่ 2 ของฟังก์ชัน `scanf` จะเริ่มต้นเครื่องหมาย ampersand (&) ซึ่งเราจะเรียกเครื่องหมายนี้ว่าตัวดำเนินการที่อยู่ (address operator) และตามด้วยตัวแปรซึ่งมีการนำเครื่องหมาย & กับตัวแปรมารวมกัน จะเป็นการบอกให้ฟังก์ชัน `scanf` ได้ระบุไว้ ส่วนรายละเอียดในการรับข้อมูลเข้า จะกล่าวในภายหลังเมื่อโปรแกรมประมวลผลมาถึงคำสั่ง `scanf` ก็จะหยุดรอให้ผู้ใช้ป้อนข้อมูลให้กับตัวแปร `integer1` เมื่อผู้ใช้ป้อนข้อมูลจำนวนเต็มแล้ว กด return ก็จะเป็นส่งจำนวนเต็มไปให้คอมพิวเตอร์และคอมพิวเตอร์ก็จะกำหนดค่าให้กับตัวแปร `integer1`

บรรทัดที่ 14-15 ก็ทำในทำนองเดียวกันกับบรรทัดที่ 11-12

บรรทัดที่ 17 `sum = integer1 + integer2; /* assign total to sum */` จะเป็นการนำค่าของจำนวน `integer1` และ `integer2` มารวมกัน แล้วนำผลลัพธ์ที่ได้ไปไว้ในตัวแปร `sum` โดยการใช้เครื่องหมาย = ซึ่งเราจะเรียกว่า ตัวกำหนดค่า (assignment operator) ในที่นี้ ตัวดำเนินการ = และ + เราจะเรียกว่าตัวดำเนินการทวิภาค (binary operators) เพราะว่ามีตัวถูกดำเนินการ 2 ตัว เช่น ตัวดำเนินการ + จะมีตัวถูกดำเนินการ 2 ตัว คือ `integer1` และ `integer2` ส่วนตัวดำเนินการ = จะมีตัวถูกดำเนินการ 2 ตัว คือ `sum` และค่าของนิพจน์ `integer1 + integer2`

บรรทัดที่ 19 `printf( "Sum is %d\n", sum ); /* print sum */` จะเป็นการนำค่าของตัวแปร `sum` มาแสดงที่จอภาพ แต่เนื่องจากตัวแปร `sum` มีชนิดข้อมูลเป็นแบบจำนวนเต็ม ดังนั้นในส่วนของรูปแบบควบคุมสายอักขระจะต้องกำหนดตัวระบุการแปลงผัน จำนวนเต็ม (%d) ให้สอดคล้องกับตัวแปร `sum` ด้วย และจากบรรทัดที่ 19 เราสามารถนำนิพจน์ `integer1 + integer2` มาเป็นอาร์กิวเมนต์ตัวที่ 2 และเขียนได้ดังนี้ `printf( "Sum is %d\n", integer1 + integer2 );`

บรรทัดที่ 20 `return 0; /* indicate that program ended successfully */` จะส่งค่า 0 ไปให้ระบบปฏิบัติการ ถ้าโปรแกรมมีการประมวลผลแล้วไม่มีข้อผิดพลาดเกิดขึ้น

บรรทัดที่ 22 `}` เป็นเครื่องหมายบอกจุดสิ้นสุดของฟังก์ชัน `main`

แนวคิดเกี่ยวกับหน่วยความจำ (Memory Concepts) ชื่อของตัวแปร เช่น `integer1`, `integer2` และ `sum` แท้จริงแล้วมันจะสมนัยกับตำแหน่งที่อยู่ในหน่วยความจำของคอมพิวเตอร์ โดยตัวแปรจะต้องประกอบไปด้วยชื่อ, ชนิด และค่าจากบรรทัดที่ 12 เมื่อพบคำสั่ง `scanf` ก็จะรอให้ผู้ใช้ป้อนข้อมูลชนิดจำนวนเต็มเข้าไปและค่านั้นจะถูกเก็บในตำแหน่งที่ตัวแปร `integer1` ชื่ออยู่ สมมติว่าผู้ใช้ป้อนตัวเลข 45 เข้าไป คอมพิวเตอร์ก็จะนำค่า 45 ไปไว้ในตำแหน่งที่ตัวแปร `integer1` ชื่ออยู่ ดังรูป

integer1	45
----------	----

โดยค่าที่อยู่ในหน่วยความจำก่อนหน้านี้ จะถูกแทนที่ด้วยตัวเลข 45 ในทำนองเดียวกัน บรรทัดที่ 15 เมื่อพบคำสั่ง `scanf` ก็จะรอให้ผู้ใช้ป้อนข้อมูลชนิดจำนวนเต็ม สมมติว่าผู้ใช้ป้อนตัวเลข 72 เข้าไป คอมพิวเตอร์ก็จะนำค่า 72 ไปไว้ในตำแหน่งที่ตัวแปร `integer2` ชื่ออยู่ โดยข้อมูลที่อยู่ในหน่วยความจำก่อนหน้านี้ที่จะถูกแทนที่ด้วยตัวเลข 72 โดยตำแหน่งที่อยู่ของตัวแปรทั้งสองนี้ไม่จำเป็นต้องอยู่ติดกัน ดังรูป

integer1	45
Integer2	72

ในทำนองเดียวกัน บรรทัดที่ 17 ตัวแปร `sum` ที่ชื่ออยู่ในตำแหน่งในหน่วยความจำ จะมีค่าเท่ากับ 117 ดังรูป

integer1	45
Integer2	72
sum	117

**ตัวอย่างโปรแกรมที่ 1.5** จะเป็น โปรแกรมที่แสดงข้อความ My first program ที่จอภาพ แต่เขียนใน 3 แบบ แต่ให้ผลลัพธ์เช่นเดียวกัน เนื่องจากโปรแกรมภาษา C จะมีลักษณะเป็น free format แต่จะพบว่า การเขียนแบบที่ 1 จะดูดีที่สุดใน และมีลักษณะเป็นโปรแกรม block

```

แบบที่ 1    #include <stdio.h>
              main ()
              {
                printf ( "My first program" );
              }

```

```

แบบที่ 2    #include <stdio.h>
              main () { printf ( "My first program" ); }

```

```

แบบที่ 3    #include <stdio.h>
              main
              (
              )
              {
                printf
                (
                "My first program"
                )
              ;
              }

```

**ข้อควรระวัง** ในการเขียน โปรแกรม ถ้าเราพิมพ์โปรแกรมส่วนของพรีโปรเซสเซอร์ ไดรอกทีฟ ดังนี้

```
#include <stdio.h> ถูก แต่ถ้าเขียนเป็น
```

```
#include
```



<stdio.h> จะไม่ถูกต้อง แต่เราสามารถทำให้ถูกต้องได้โดยใส่เครื่องหมาย \ โดยเขียนดังนี้

```
#include \ (เติมเครื่องหมาย \ ไว้ด้านหลังตัวอักษรสุดท้าย)
```

```
<stdio.h> (ส่วนข้อความนี้จะพิมพ์ในบรรทัดที่สอง แต่จะเริ่มต้นที่ตำแหน่งใด ๆ ก็ได้ในบรรทัดนี้)
```

แต่ถ้าเราต้องการพิมพ์ค่าคงที่สายอักขระ โดยใช้คำสั่ง printf ดังนี้

```
printf ("A PROGRAM THAT COMPUTE CITY INCOME TAX.");
```

จะถือว่าเป็นคำสั่งที่ถูกต้อง แต่ถ้าเขียนคำสั่ง printf ดังนี้

```
printf ("A PROGRAM THAT COMPUTE  
CITY INCOME TAX.");
```

จะถือว่าผิดไวยากรณ์ (Syntax error)

ถ้าต้องการพิมพ์ในลักษณะเช่นนี้ ให้ใส่เครื่องหมาย \ ดังนี้

```
printf ("A PROGRAM THAT COMPUTE \ (เติม \ ท้าย)
```

```
CITY INCOME TAX."); (ส่วนข้อความนี้ต้องพิมพ์ที่ตำแหน่งแรกของบรรทัดที่สอง
```

มิฉะนั้นจะมีอักขระว่างเกิดขึ้น)

## ตัวอย่างโปรแกรมที่ 1.6 จะเป็น โปรแกรมที่แสดงส่วนสำคัญ (element) ในภาษา C

```

1  /*
2  * Converts distance in miles to kilometers.
3  */
4  #include <stdio.h>
5  #define KMS_PER_MILE 1.609
6
7  int
8  main (void)
9  {
10     double miles;
11     kms;
12
13     /* Get the distance in miles. */
14     printf ( "Enter the distance in miles> " );
15     scanf ( "%lf", &miles );
16
17     /* Convert the distance to kilometers. */
18     kms = KMS_PER_MILE * miles;
19     /* Display the distance in kilometers. */
20     printf ( "That equals %f kilometers. \n", kms );
21
22     return ( 0 );
23 }

```

Diagram annotations for the code above:
 

- เพิ่มมาตรฐานส่วนหัว (standard header file) points to line 1.
- หมายเหตุ (comment) points to lines 2-3 and 13-14.
- หมายเหตุ (comment) points to lines 17-18.
- หมายเหตุ (comment) points to lines 19-20.
- หมายเหตุ (comment) points to line 21.
- หมายเหตุ (comment) points to line 22.
- หมายเหตุ (comment) points to line 23.
- หมายเหตุ (comment) points to line 24.
- พรีโปรเซสเซอร์ไดเรกทีฟ (preprocessor directive) points to lines 4 and 5.
- ค่าคงตัว (constant) points to line 5.
- คำสงวน (reserved word) points to line 7.
- ตัวแปร (variable) points to line 10.
- ตัวแปร (variable) points to line 11.
- ชื่อมาตรฐาน (standard identifier) points to lines 14 and 15.
- สัญลักษณ์พิเศษ (special symbol) points to line 18.
- เครื่องหมายวรรคตอน (punctuation) points to line 23.
- สัญลักษณ์พิเศษ (special symbol) points to line 24.

```

Enter the distance in miles> 10.00
That equals 16.090000 kilometers.

```

ตัวชี้ทาง (directive) จะเป็นส่วนที่ถูกประมวลผลก่อน โดยจะมีการนำข้อมูลจากเพิ่มมาตรฐานส่วนหัวที่อยู่ในไลบรารี นำมารวมกับโปรแกรมก่อนที่จะมีการแปล เนื่องจากโปรแกรมนี้อาศัยการใช้ฟังก์ชัน `scanf` และ `printf` จึงต้องมีตัวชี้ทาง (directive) เขียนไว้ที่ส่วนต้นของโปรแกรกดังนี้ `#include <stdio.h>` ส่วน `#define KMS_PER_MILE 1.609` ก็เป็น preprocessor directive อีกแบบหนึ่งที่ประกอบด้วยค่าคงตัวมาโคร (constant macro) ซึ่งตัวแปร `KMS_PER_MILE` จะมีค่า 1.609

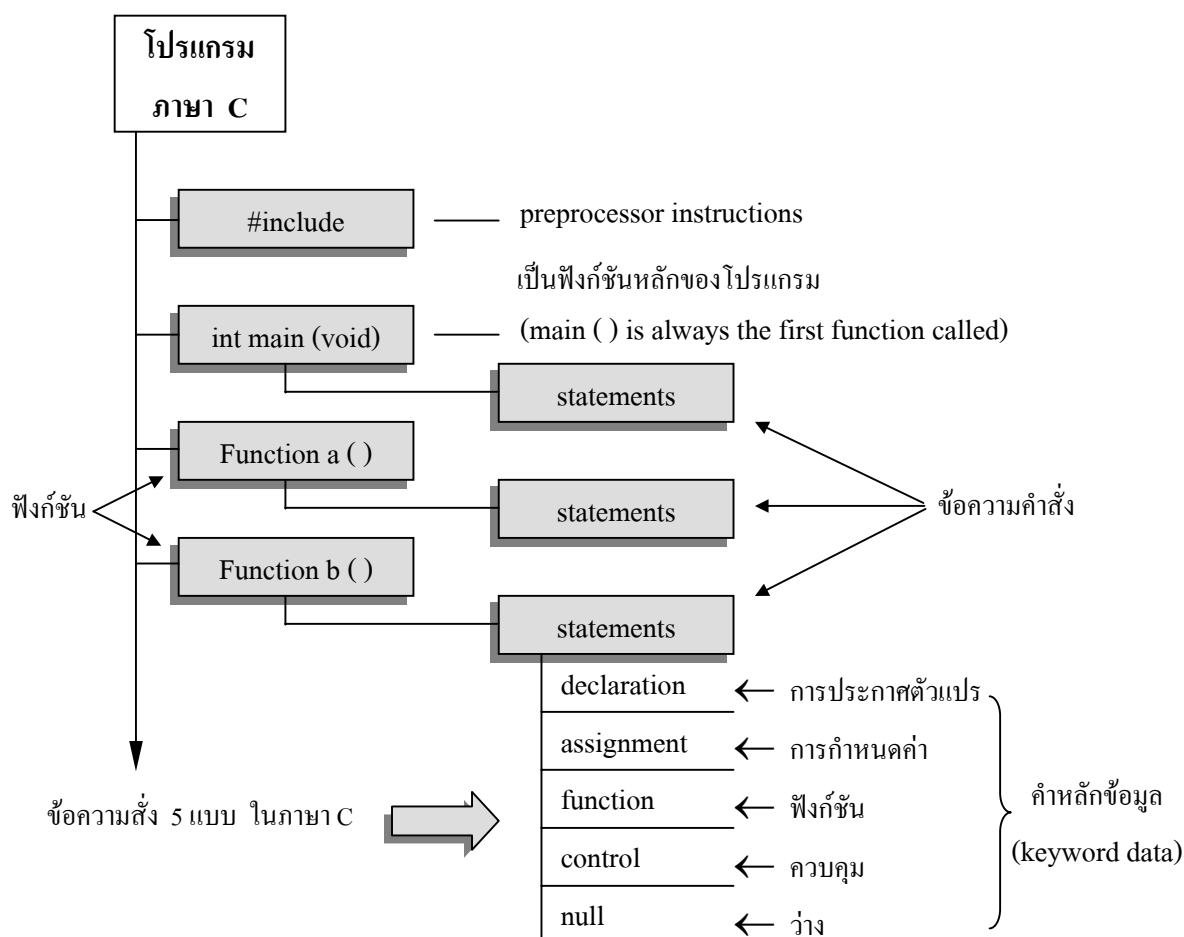
โดยจะมีการแทนค่าตัวแปร `KMS_PER_MILE` ด้วยค่า 1.609 ในโปรแกรมทั้งหมดก่อนที่จะมีการ  
 แปล เช่น บรรทัดที่ 18 `kms = KMS_PER_MILE * miles ;`

จะถูกแทนค่าเป็น `kms = 1.609 * miles ;`

หมายเหตุ อาร์กิวเมนต์ของ `main` เป็น `void` หมายความว่า ไม่ต้องมีอาร์กิวเมนต์ ส่วนนี้ไม่ต้อง  
 เขียนก็ได้ จากตัวอย่างโปรแกรมต่าง ๆ เราสามารถนำมาเขียนเป็นรูปแบบทั่วไปของ  
 โปรแกรมภาษา C ได้ดังนี้

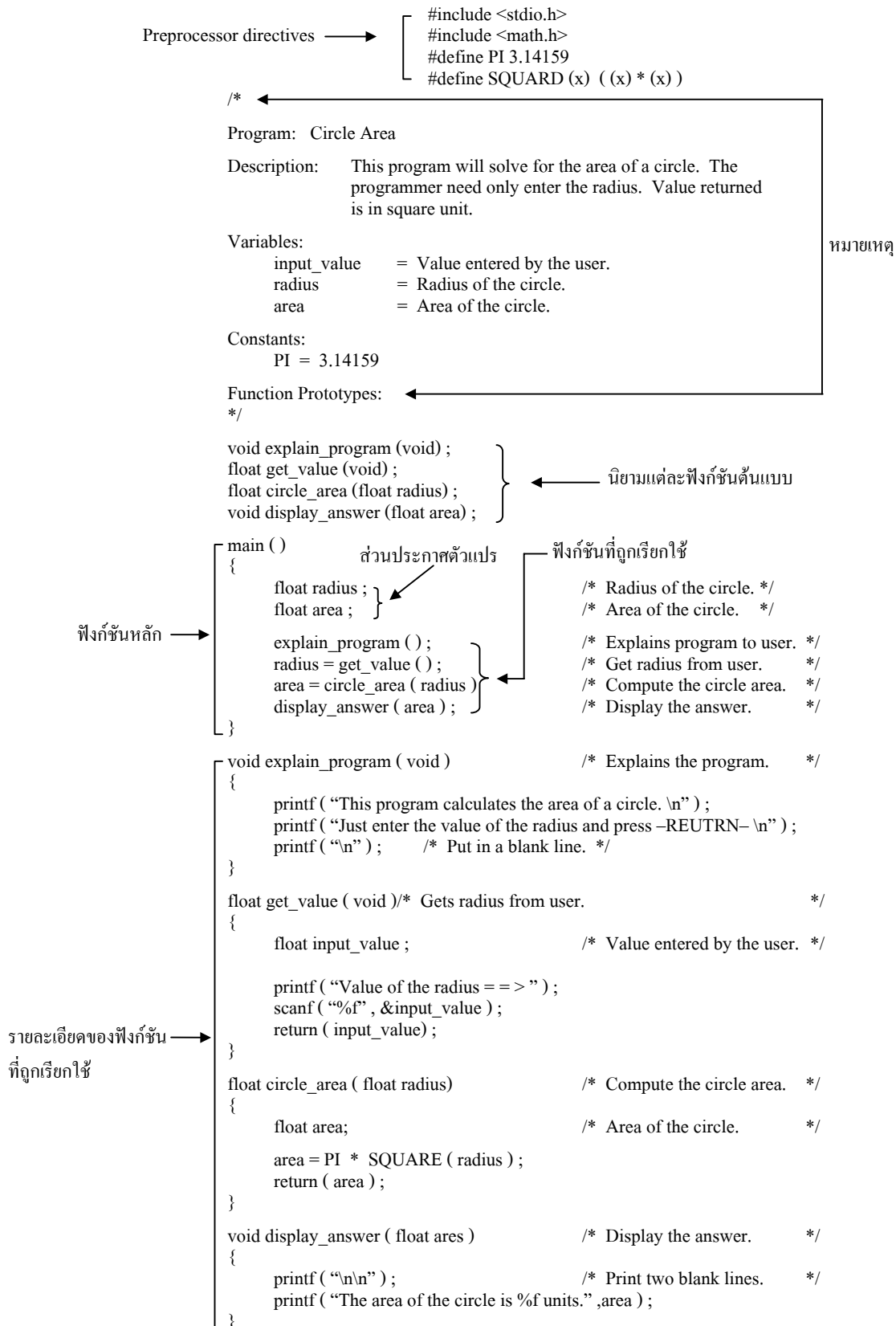
### 1.1 รูปแบบทั่วไปของโปรแกรมภาษา C (General Form to a C Program)

สามารถแสดงได้ดังรูป 1.1



จากรูป 1.1 นี้ เราสามารถกล่าวได้ว่า โปรแกรมของภาษา C จะประกอบด้วย 1 ฟังก์ชัน  
 หรือมากกว่า 1 ฟังก์ชันก็ได้ แต่อย่างน้อยจะต้องประกอบด้วยฟังก์ชัน `main` เป็นฟังก์ชันหลักเสมอ

ตัวอย่างโปรแกรมที่ 1.7 เป็นโปรแกรมที่แสดงโครงสร้างภาษา C ที่มีลักษณะโปรแกรมเป็น block



ถ้าโปรแกรมนั้นมีเฉพาะฟังก์ชัน main เท่านั้น จะมีรูปแบบทั่วไป ดังนี้

```

preprocessor directives
main function heading
{
    declarations ← ส่วนประกาศตัวแปร
    executable statements ← ข้อความสั่งกระทำการ
}
    
```

ตัวอย่างโปรแกรมที่ 1.8 เป็น โปรแกรมที่แสดงส่วนประกอบของโครงสร้างภาษา C แบบง่าย

