# ภาษาคอมพิวเตอร์และการโปรแกรม


โดย

นาย วทัญญู เกศฉิม รหัสนศ.1661010541270

นาย ภูริวัฒน์ พยัคฆานนท์ รหัสนศ.1661010541273

นาย ฐปนรรฆ์ เจิมจรุง รหัสนศ.1661010541282

นาย ธนสิทธิ์ เข็มทอง รหัสนศ.1661010541285

นางสาว กัญญาวีร์ อ่อนนอม รหัสนศ.1661010541276

นางสาว โชติกา หุ่นรูปหล่อ รหัสนศ.1661010541279

สาขาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์

# คำนำ

รายงานเล่มนี้จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของวิชาภาษาคอมพิวเตอร์และการโปรแกรม ชั้นปีที่ 1 เพื่อให้ได้ศึกษาหาความรู้ในเรื่องพรีโปรเซสเซอร์ไดเรกทีฟและมาโครและได้ศึกษาอย่างเข้าใจเพื่อเป็นประโยชน์กับการเรียน

ผู้จัดทำ

# สารบัญ

**เรื่อง** **หน้า**

# พรีโปรเซสเซอร์ไดเรกทีฟ และมาโคร (Preprocessor Directives & Macro)

ในบทนี้จะกล่าวถึงพรีโปรเซสเซอร์ไดเรกทีฟ ซึ่งเป็นที่ประกาศไว้บนสุดของโปรแกรม โดยคอมไพเลอร์จะประมวลผลพรีโปรเซสเซอร์ไดเรกทีฟเหล่านี้ก่อนแปลผลโปรแกรมรวมทั้งกล่าวถึงมาโครที่มีส่วนสัมพันธ์กับพรีโปรเซสเซอร์ไดเรกทีฟด้วย

## พรีโปรเซสเซอร์ไดเรกทีฟ (Preprocessor Directives)

พรีโปรเซสเซอร์ไดเรกทีฟ คือ ส่วนที่ประกาศไว้บนสุดของโปรแกรม เพื่อระบุให้คอมไพเลอร์กระทำตามใดๆตามที่กำหนด ก่อนทำการแปลผลโดย (pro-processing) แบ่งออกเป็นหลายประเภทดังนี้

| พรีโปรเซสเซอร์ไดเรกทีฟ | ความหมาย |
|---|---|
| #include | - ใช้สำหรับรวมไฟล์ที่ระบุ เข้าไว้ในการแปลโปรแกรม |
| #if | - เป็นคำสั่งเงื่อนไขที่ใช้ตรวจสอบว่าควรทำงานตามคำสั่งที่ระบุไว้ภายใต้เงื่อนไข #if หรือไม่ |
| #ifdef | - เป็นคำสั่งที่ใช้ตรวจสอบว่า มาโครได้ถูกกำหนด (define) ไว้แล้ว ใช่หรือไม่ |
| #ifndef | - เป็นคำสั่งที่ใช้ตรวจสอบว่า มาโครยังไม่ได้ถูกกำหนดไว้ (not define) ใช่หรือไม่ |
| #elif | - หากตรวจสอบเงื่อนไขของ #if, #ifdef, #ifndef แล้วเป็นเท็จ ก็จะทำการตรวจสอบว่าควรทำตามคำสั่งที่ระบุไว้ภายใต้เงื่อนไข #elif หรือไม่ |
| #else | - หากตรวจสอบเงื่อนไขของ #if, #ifdef, #ifndef, #elif แล้วเป็นเท็จ ก็จะทำงานตามคำสั่งที่ระบุไว้ภายใต้เงื่อนไข else นี้ |
| #endif | - ใช้ระบุจุดสิ้นสุดของเงื่อนไข |
| #define | - ใช้สำหรับกำหนด preprocessor macro |
| #undef | - ใช้สำหรับยกเลิก preprocessor macro ที่ได้กำหนด (define) ขึ้น |

ก่อนที่จะแสดงตัวอย่างของพรีโปรเซสเซอร์ไดเรกทีฟแต่ละตัว ผู้เขียนจะขออธิบายถึงมาโครสักเล็กน้อย เพราะการใช้งานพรีโปรเซสเซอร์ไดเรกทีฟ #define จะเกี่ยวข้องกับมาโครโดยตรง

## มาโคร (Macro)

มาโคร คือ สิ่งที่ถูกกำหนดขึ้นจากพรีโปรเซสเซอร์ไดเรกทีฟ #define ซึ่งมาโครแต่ละชุดจะมีชื่อเฉพาะของตัวเอง ลักษณะพิเศษของมาโครคือ สามารถกระทำการทางคณิตศาสตร์และการทำงานทางตรรกะเพื่อตัดสินเงื่อนไขต่างๆ โดยมาโครแบ่งออกเป็น 2 ประเภทใหญ่ๆ คือ

- มาโครมาตรฐาน (Standard Predefined Macros)

- มาโครที่ผู้เขียนโปรแกรมกำหนดขึ้นเองโดยใช้พรีโปรเซสเซอร์ไดเรกทีฟ #define

## มาโครมาตรฐาน (Standard Predefined Macros)

เป็นมาโครมาตรฐานที่ภาษาซีได้จัดหาไว้ให้กับผู้เขียนโปรแกรม สามารถเรียกใช้ได้ทันที โดยมาโครมาตรฐานจะเริ่มต้นด้วย underscore 2 ตัว(_ _) ตามด้วยชื่อมาโครมาตรฐานและปิดท้ายด้วย underscore 2 ตัวอีกครั้ง ซึ่งการเขียน underscore 2 ตัว ให้เขียนติดกันห้ามเว้นวรรคซึ่งเมื่อเขียนติดกัน จะทำให้เห็นเป็นเส้นเดียว (__) โดยมาโครมาตรฐานที่ใช้งานอยู่บ่อยครั้งมีดังนี้

- __FILE__ เป็นมาโครมาตรฐานที่ใช้สำหรับแสดงชื่อไฟล์ปัจจุบันที่เรียกใช้งานอยู่ในรูปแบบของค่าคงที่สตริง
- __LINE__ เป็นมาโครมาตรฐานที่ใช้สำหรับแสดงบรรทัดปัจจุบันของไฟล์ที่เรียกใช้งานพรีโปรเซสเซอร์มาโคร ในรูปแบบของเลขจำนวนเต็มฐานสิบ
- __DATE__ เป็นมาโครมาตรฐานที่ใช้สำหรับแสดงวันที่ที่เริ่มทำการรันพรีโปรเซสเซอร์มาโคร ในรูปแบบของค่าคงที่สตริง 11 ตัวอักษร เช่น “Jan 01 2009” เป็นต้น
- __TIME__ เป็นมาโครมาตรฐานที่ใช้สำหรับแสดงเวลาที่เริ่มทำการรันพรีโปรเซสเซอร์มาโคร ในรูปแบบของค่าคงที่สตริง 8 ตัวอักษร เช่น “16:54:55” เป็นต้น
- __STDC__ เป็นมาโครมาตรฐานที่ใช้สำหรับแสดงว่าคอมไพเลอร์ภาษาซีที่ใช้เป็น ISO/ANSIC ใช่หรือไม่ ในรูปแบบของเลขจำนวนเต็ม โดยถ้าคิด __STDC__ = 0 คือไม่เป็น ISO/ANSIC แต่ถ้าค่า __STDC__ = 1 คือ เป็น ISO/ANSI C

| NOTE : โดยส่วนใหญ่แล้วมาโครมาตรฐาน __FILE__ และ __LINE__ จะมีประโยชน์กับแสดง error messageต่างๆ เพื่อแสดงชื่อไฟล์และหมายเลขบรรทัดที่เกิดความผิดพลาดขึ้น |
|---|

**ตัวอย่างที่ 13.1** แสดงการทำงานของมาโครมาตรฐาน __FILE__ , __LINE__ และ __DATE__

```
1  #include <stdio.h>
2
3  #define WHERE_MACRO1 printf("in file %s , use WHERE_MACRO1 at line
                   %d\n",__FILE__, __LINE__);
4  #define WHERE_MACRO2 printf("in file %s , use WHERE_MACRO2 at line
                   %d\n",__FILE__, __LINE__);
5
6  int main()
7  {
8      WHERE_MACRO1;
9      WHERE_MACRO2;
10     printf("Print this result in %s\n", __DATE__);
11     return 0;
12 }
```

**ผลลัพธ์ของโปรแกรม**

```
in file tempCodeRunnerFile.c , use WHERE_MACRO1 at line 9
in file tempCodeRunnerFile.c , use WHERE_MACRO2 at line 10
Print this result in Jan  3 2024
```

**อธิบายโปรแกรม**

**บรรทัดที่ 1 :** เป็นการนำเข้าไลบรารี stdio.h ซึ่งจำเป็นสำหรับฟังก์ชัน printf()

**บรรทัดที่ 3-4 :** กำหนดมาโคร WHERE_MACRO1 และ WHERE_MACRO2 โดยนิยามให้มีค่าเท่ากับฟังก์ชัน printf() พร้อมใส่ค่าพารามิเตอร์ __FILE__ และ __LINE__ เข้าไปด้วย

**บรรทัดที่ 8-9 :** เรียกใช้มาโคร WHERE_MACRO1 และ WHERE_MACRO2 แต่ละครั้งจะแสดงข้อความ in file %s , use WHERE_MACRO1 at line %d\n หรือ in file %s , use WHERE_MACRO2 at line %d\n โดยแทนที่ %s ด้วยชื่อไฟล์ปัจจุบัน และ %d ด้วยหมายเลขบรรทัดปัจจุบัน

**บรรทัดที่ 9 :** เรียกใช้ฟังก์ชัน printf() เพื่อแสดงข้อความ Print this result in %s\n โดยแทนที่ %s ด้วยวันที่ปัจจุบัน

**ตัวอย่างที่ 13.2** แสดงการทำงานของมาโครมาตรฐาน __FILE__ , __TIME__ และ __STDC__

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int ret;
6       printf("This is %s program\n", __FILE__);
7       printf("Time to run this program is %s\n", __TIME__);
8       ret = __STDC__;
9
10      if (ret == 1){
11          printf("RUN BY STANDARD ISO/ANSI C\n");
12      }
13      else{
14          printf("DON'T RUN BY STANDARD ISO/ANSI C\n");
15      }
16  }
```

**ผลลัพธ์ของโปรแกรม**

```
This is testMacro2.c program
Time to run this program is 15:12:10
RUN BY STANDARD ISO/ANSI C
```

**อธิบายโปรแกรม**

**บรรทัดที่ 7 :** แสดงเวลาที่ผู้เขียนได้ทำการรันโปรแกรม โดยผู้เขียนรันโปรแกรม ณ เวลา 15:12:10 ซึ่งหากผู้อ่านรันโปรแกรมแล้ว ผลที่ได้อาจแตกต่างจากของผู้เขียน เนื่องมาจากการรันโปรแกรมในเวลาที่ต่างกัน

**บรรทัดที่ 8 :** นำมาโครมาตรฐาน __STDC__ มาตรวจสอบคอมไพเลอร์ ว่าเป็นคอมไพเลอร์ชนิด ISO/ANSI C หรือไม่ ซึ่งผลแสดงให้เห็นว่าคอมไพเลอร์ที่ผู้เขียนโปรแกรมใช้งานอยู่เป็นคอมไพเลอร์ชนิด ISO/ANSI C

## มาโครที่เขียนโปรแกรมกำหนดขึ้นเองโดยใช้ #define

เป็นมาโครที่ผู้เขียนโปรแกรมกำหนดขึ้นเพื่อให้ทำงานต่างๆตามที่ต้องการ ซึ่งแบ่งออกเป็น 3 ประเภท คือมาโครอย่างง่าย , มาโครที่สามารถรับพารามิเตอร์ได้และมาโครแบบมีเงื่อนไข

### มาโครอย่างง่าย (Simple macro)

รูปแบบของมาโครประเภทนี้ คือ

| รูปแบบ : #define ชื่อมาโคร ค่าที่ต้องการกำหนดให้กับมาโคร |
|---|

เช่น

**#define PI 3.14** กำหนดให้มาโคร PI มีค่าเท่ากับค่าคงที่ 3.14

**#define TIMES 2** กำหนดให้มาโคร TIMES มีค่าเท่ากับค่าคงที่ 2

**#define ROUNDS 3*TIMES** กำหนดให้มาโคร ROUNDS มีค่าเท่ากับค่าคงที่ 3 คูณด้วยค่าของมาโครTIMES (จะพบว่านอกจากจะกำหนดค่าคงที่ให้กับค่าคงที่ให้กับมาโครแล้ว ยังสามารถใช้ค่าของมาโครหนึ่งกำหนดให้กับอีกมาโครหนึ่งได้ด้วย) ซึ่งค่าของมาโคร ROUNDS จะมีค่าเท่ากับ 3*TIMES = 3*2 = 6

**#define ADD 2+1+a** กำหนดให้มาโคร ADD มีค่าเท่ากับค่าคงที่ 2+1+ค่าของตัวแปร a (จะพบว่านอกจากจะกำหนดค่าคงที่ให้กับมาโครแล้ว ยังสามารถใช้ตัวแปรกำหนดค่าให้กับมาโครได้ด้วย)

NOTE : มีข้อที่ควรระมัดระวัง คือ หากกำหนดให้มาโครกระทำการทางคณิตศาสตร์ ให้คำนึงถึงลำดับของโอเปอเรเตอร์เป็นสำคัญด้วยเช่น กำหนด #define ADD 2+1+a และกำหนดให้ a=3 ต้องการหาค่าของ 5*ADD โดยหากพิจารณาแล้วจะพบว่าค่า ADDมีค่าเท่ากับ 6 ผู้อ่านอาจคิดว่าผลลัพธ์ที่ได้ควรจะเป็น 5*6 คือ 30 แต่ในความเป็นจริงแล้วกลับไม่เป็นเช่นนั้นเพราะมีเรื่องราวเกี่ยวกับลำดับของโอเปอเรเตอร์มาเกี่ยวข้องดังนี้

เมื่อระบุ 5*ADD จะหมายถึง 5*2+1+a ซึ่งหากพิจารณาลำดับของโอเปอเรเตอร์ * กับ + แล้วจะพบว่าโอเปอเรเตอร์ * มีลำดับความสำคัญสูงกว่าโอเปอเรเตอร์ + ดังนั้น จึงทำ 5*2=10 ก่อนจากนั้นจึงนำ 10+1+a ได้ 14 ไม่ใช่ 30 อย่างที่ต้องการ ดังนั้นเพื่อป้องกันข้อมูลผิดพลาดผู้เขียนแนะนำให้ผู้อ่านใส่วงเล็บให้กับค่าที่ต้องการกำหนดให้กับมาโครทุกครั้งเพื่อป้องกันข้อผิดพลาด ดังนี้

เช่น

#define PI (3.14)

#define TIMES (2)

#define ROUNDS (3*TIMES)

#define ADD (2+1+a) เป็นต้น

แต่สำหรับกรณีของการกำหนดค่าคงที่ให้กับมาโคร เช่น ในบรรทัดที่ 1 และ 2 อาจไม่จำเป็นต้องใส่วงเล็บให้กับค่าที่กำหนดให้กับมาโครก็ได้ เพราะว่าคงจะไม่เกิดปัญหากับมาโครอย่างแน่นอน ถึงตอนนี้ลองมาพิจารณากันต่อว่า หากใส่วงเล็บให้กับการทำงานของมาโครแล้ว ผลลัพธ์ที่ได้จะเป็นเช่นไร ดังนี้ 5*ADD = 5*(2+1+a) = 5*(2+1+3)หากพิจารณาแล้วพบว่า () มีลำดับความสำคัญของโอเปอเรเตอร์สูงกว่า * ดังนั้นจึงทำงานในวงเล็บก่อน ได้ 5*(6) ผลลัพธ์ที่ได้ คือ 30 เป็นผลลัพธ์ที่ถูกต้องตรงตามที่ต้องการ

**ตัวอย่างที่ 13.3** แสดงการทำงานของมาโครอย่างง่าย (Simple macro)

```
1  #include <stdio.h>
2
3  #define PT 3.14
4  #define TIMES 2
5  #define ROUNDS 3*TIMES
6  #define ADD (2 + 1 + a)
7  #define ADD1 2 + 1 + a
8
9  int main()
10 {
11     int a = 3, result, result1;
12     printf("Value of Macro PT is %f\n", PT);
13     printf("Value of Macro TIMES is %d\n", TIMES);
14     printf("Value of Macro ROUNDS is %d\n", ROUNDS);
15     printf("Value of Macro ADD (2+1+%d) is %d\n", a, ADD);
16     result = 5 * ADD;
17     result1 = 5 * ADD1;
18     printf("Result is %d\n", result);
```

```
19      printf("Result1 is %d\n", result1);
20  }
```

**ผลลัพธ์ของโปรแกรม**

```
Value of Macro PT is 3.140000
Value of Macro TIMES is 2
Value of Macro ROUNDS is 6
Value of Macro ADD (2+1+3) is 6
Result is 30
Result1 is 14
```

## มาโครที่สามารถรับพารามิเตอร์ได้ (Macro with paramrter)

รูปแบบของมาโครประเภทนี้ คือ

| รูปแบบ : #define ชื่อมาโคร (อาร์กิวเมนต์) สิ่งที่ต้องการกำหนดให้มาโครทำงาน |
|---|

เช่น

**#define SQUARE(A) (A*A)** กำหนดอาร์กิวเมนต์ A เพื่อรับพารามิเตอร์เข้ามาในมาโคร SQUARE และมาโคร SQUARE ก็จะคำนวณหาค่ายกกำลังสองของค่าพารามิเตอร์นั้นออกมาคือ ทำ A*A

**#define ADD(B) (B+B)** กำหนดอาร์กิวเมนต์ B เพื่อรับพารามิเตอร์เข้ามาในมาโคร ADD และ มาโคร ADD ก็จะคำนวณหาค่าของผลบวก B+B ออกมา

อย่าลืมว่า A*A และ B+B เป็นการสั่งให้มาโครทำงานแบบคณิตศาตร์ ดังนั้น อย่าลืมใส่วงเล็บเพื่อป้องกันความผิดพลาดด้วย

**ตัวอย่างที่ 13.4** แสดงการทำงานของมาโครที่สามารถรับพารามิเตอร์ได้ (Macro with parameter)

```
1   #include <stdio.h>
2
3   #define SQUARE(A) (A * A)
```

```
4   #define ADD(B) (B + B)
5
6   int main()
7   {
8       int param = 3, result = 0;
9       printf("Value of param is %d\n", param);
10      result = SQUARE(5);
11      printf("Result of SQUARE(5) is %d\n", result);
12      result = SQUARE(param);
13      printf("Result of SQUARE (param) is %d\n", result);
14      result = ADD(param);
15      printf("Result of ADD(param) is %d\n", result);  }
```

**ผลลัพธ์ของโปรแกรม**

```
Value of param is 3
Result of SQUARE(5) is 25
Result of SQUARE (param) is 9
Result of ADD(param) is 6
```

## มาโครแบบมีเงื่อนไข (Macro with condition)

รูปแบบของมาโครประเภทนี้ คือ

| รูปแบบ : #define ชื่อมาโคร (อาร์กิวเมนต์ตัวที่ 1 , … , อาร์กิวเมนต์ตัวที่ n) เงื่อนไข |
|---|

เช่น

#define MAX(x,y) x>y ? x : y เป็นการกำหนดให้มาโครทำงานตามเงื่อนไข โดยนำคำสั่ง if แบบย่อมาใช้งาน โดยกำหนดมาโคร MAX ตรวจสอบเงื่อนไข ถ้า x>y จริง มาโคร MAX จะให้ค่าเท่ากับ ค่าของ x แต่หากตรวจสอบเงื่อนไขแล้วเป็นเท็จ มาโคร MAX จะให้ค่าเท่ากับค่าของ y ซึ่งลักษณะนี้เป็น มาโครอย่างย่อ ที่สามารถกำหนดให้อยู่ในบรรทัดเดียวกันได้ แต่หากต้องการกำหนดการทำงานให้กับ มาโครในรายละเอียดที่มากขึ้น โดยไม่สามารถเขียนมาโครให้อยู่ในบรรทัดเดียวกันได้สามารถทำงานได้ดังนี้

```
#define ชื่อมาโคร (อาร์กิวเมนต์)          {          \
                                  สิ่งที่ต้องการสั่งให้มาโครทำงาน   \
                                      . . .                         \
                                      . . .                         \
                          }
```

จะสังเกตเห็นว่ามาโครที่มีขั้นตอนการทำงานหลายๆบรรทัด เมื่อจบการทำงานใน 1 บรรทัดแล้ว จะต้องนำเครื่องหมาย \ ปิดท้ายไว้ที่แต่ละบรรทัดด้วย ยกเว้น บรรทัดสุดท้ายของมาโคร ที่เป็นเช่นนี้เพราะเครื่องหมาย \ แสดงถึงว่า การทำงานของมาโครยังคงมีต่อไป ยังไม่สิ้นสุดการทำงานของมาโครซึ่งในบรรทัดสุดท้ายของมาโครแสดงถึงจุดสิ้นสุดของมาโครแล้ว จึงไม่ต้องมีเครื่องหมาย \ ปิดท้ายอีก

NOTE : การเขียนโปรแกรมแบบหลายบรรทัด สิ่งสำคัญ คือ ห้ามลืมนำเครื่องหมาย “ \ “ ปิดท้ายไว้ทุกบรรทัดด้วย(ยกเว้นบรรทัดสุดท้ายที่ไม่ต้องปิดท้ายด้วยเครื่องหมายนี้)เพื่อเป็นการบ่งบอกว่าการทำงานของ

## เปรียบเทียบมาโครกับฟังก์ชัน

หากผู้อ่านพิจารณามาโครที่มีการรับพารามิเตอร์เข้ามาในมาโครแล้ว อาจจะคิดว่ามาโครเหมือนกับฟังก์ชัน แต่หากพิจารณาโดยละเอียดแล้วจะพบว่ามาโครแตกต่างกับฟังก์ชันโดยสิ้นเชิง ดังนี้

```
มาโคร      : #define  MAX(x , y)  x>y  ?  x : y
ฟังก์ชัน   : int  findMax(int  x , int y)  {
                          ...
             }
```

ข้อแตกต่างระหว่างมาโครและฟังก์ชัน คือ

- มาโครถูกกำหนดขึ้นด้วยพรีโปรเซสเซอร์ไดเรกทีฟ #define แต่ฟังก์ชันไม่ได้เป็นเช่นนั้น
- การกำหนดอาร์กิวเมนต์เพื่อรับพารามิเตอร์เข้ามาในมาโครนั้น จะระบุเพียงแต่ชื่อตัวแปรเท่านั้นในขณะที่ฟังก์ชันต้องกำหนดชนิดของข้อมูลให้กับตัวแปรด้วย

**ตัวอย่างที่ 13.5** แสดงการทำงานของมาโครกับเงื่อนไข (Macro with condition)

```
1  #include <stdio.h>
2
```

```
3   #define MAX(x, y) ((x) > (y) ? (x) : (y))
4   #define MAX_MIN(x, y) {                         \
5       if (x > y)                                  \
6           printf("MAX : %d , MIN : %d\n", x, y); \
7       else                                        \
8           printf("MAX : %d , MIN : %d\n", y ,x); \
9   }
10
11  int main()
12  {
13      int x = 5, y = 8, max;
14      max = MAX(x, y);
15      printf("max value from macro MAX is %d\n", max);
16      MAX_MIN(x, y);
17  }
```

**ผลลัพธ์ของโปรแกรม**

```
max value from macro MAX is 8
MAX : 8 , MIN : 5
```

## สรุปส่งท้ายมาโคร

การทำงานของมาโครนั้นเป็นไปในลักษณะของการแทนที่ (substitution) คือ ก่อนการแปลผลโปรแกรม คอมไพเลอร์จะตรวจสอบว่าที่ใดของโปรแกรมมีการเรียกใช้งานมาโครบ้าง และเมื่อตรวจสอบพบแล้วก็จะทำการแทนที่ค่าของมาโครให้กับตัวที่เรียกใช้งานมาโครนั้นๆ เช่น

```
#define MAX_MIN(x, y) {                         \
    if (x > y)                                  \
        printf("MAX : %d , MIN : %d\n", x, y); \
    else                                        \
        printf("MAX : %d , MIN : %d\n", y ,x); \
}
main() {
```

```
        MAX_MIN (5, 2);
}
```

เมื่อรันโปรแกรม บรรทัดที่เรียกใช้งานมาโคร คือ MAX_MIN(5,2) จะถูกแทนที่ด้วยกรอบการทำงานของมาโคร ดังนี้

```
main() {
    If (5 > 2)
            printf(“MAX : %d , MIN : %d\n”, 5 , 2);
    else
            Printf((“MAX : %d , MIN : %d\n”, 2 , 5);
}
```

## ตัวอย่างแสดงการทำงานของพรีโปรเซสเซอร์ไดเรกทีฟ

ต่อไปจะเป็นตัวอย่างแสดงการทำงานของพรีโปรเซสเซอร์ไดเรกทีฟแต่ละตัว ซึ่งสามารถใช้งานได้ในรูปแบบที่แตกต่างออกไป

**ตัวอย่างที่ 13.6** แสดงการทำงานของพรีโปรเซสเซอร์ไดเรกทีฟ (ตัวอย่างที่ 1)

```
1   #include <stdio.h>
2   #include <string.h>
3
4   #define A 5
5
6   int main()
7   {
8       #ifdef A
9           #undef A
10          #define A printf("This is Macro A\n");
11          A;
```

```
12      #endif
13      #ifndef B
14          #define B "HELLO!!!"
15          char greeting[9];
16          strcpy(greeting, B);
17      #else
18          #define A printf("This is Macro A\n")
19          A;
20          #define B printf("This is Macro B\n")
21          B;
22      #endif
23
24      printf("%s\n", greeting);
25      printf("End processing Macro\n");
26      return 0;
27  }
```

**ผลลัพธ์ของโปรแกรม**

```
This is Macro A
HELLO!!!
End processing Macro
```

**อธิบายโปรแกรม**

(ตัวอย่างนี้เป็นการกำหนดให้มาโครอยู่ในลักษณะของค่าคงที่เลขจำนวนเต็ม)

**บรรทัดที่ 2 :** โหลด header file string.h ซึ่งมีฟังก์ชั่นสำหรับการจัดการ string

**บรรทัดที่ 8 :** ใช้ #ifdef เพื่อ เช็คว่า Macro A ถูกกำหนดไว้หรือไม่ (ถูก define ไว้หรือไม่)

**บรรทัดที่ 9 :** ใช้ #undef ยกเลิกการกำหนดค่า Macro A

**บรรทัดที่ 10 :** กำหนดค่า Macro A ใหม่เป็นฟังก์ชัน printf ที่พิมพ์ข้อความ "This is Macro A\n" แล้วเรียกใช้ Macro A ที่กำหนดใหม่ ในบรรทีดที่ 11

**บรรทัดที่ 13 :** ใช้ #ifndef เช็คว่า Macro B ไม่ได้ถูกกำหนดไว้หรือไม่ (ไม่ถูก define ไว้หรือไม่)

**บรรทัดที่ 14 :** กำหนดค่า Macro B เป็นสตริง "HELLO!!!"

**บรรทัดที่ 15 :** ประกาศตัวแปร greeting ประเภท char ที่เป็นอาร์เรย์ขนาด 9 สำหรับเก็บข้อความ

**บรรทัดที่ 16 :** ทำการคัดลอกข้อความจาก Macro B ("HELLO!!!") ไปยังตัวแปร greeting

**บรรทัดที่ 17 :** จะถูกทำงานถ้าเงื่อนไขใน #ifndef B เป็นเท็จ

**บรรทัดที่ 24 :** พิมพ์ค่าของตัวแปร greeting ที่ถูกกำหนดค่าด้วย Macro B

**ตัวอย่างที่ 13.7** แสดงการทำงานของพรีโปรเซสเซอร์ไดเรกทีฟ (ตัวอย่างที่ 2)

```
1   #include <stdio.h>
2
3   #define TEST1 2
4   #define TEST2 3*TEST1
5   int times = 11, result;
6
7   int main()
8   {
9       #if TEST1
10          result = TEST1*times;
11          printf("Value of Macro TEST1 is %d ===> %d x %d times\n", TEST1, TEST1,
        times);
12      #elif TEST2
13          result = TEST2*times;
14          printf("Value of Macro TEST2 is %d ===> %d x %d times\n", TEST2, TEST2,
        times);
15      #else
16          result = 0;
17      #endif
18
19      if(result == 0) {
20          printf("NOT DEFINE MACRO TEST1 & TEST2\n");
```

```
21      }
22      else {
23          printf("Result in %d\n", result);
24      }
25      return 0;
26  }
```

**ผลลัพธ์ของโปรแกรม**

```
Value of Macro TEST1 is 2 ===> 2 x 11 times
Result in 22
```

**อธิบายโปรแกรม**

(ตัวอย่างนี้เป็นการกำหนดมาโครด้วยค่าของมาโครอื่น)

**บรรทัดที่ 3 :** ใช้ #define กำหนดให้มาโคร TEST1 มีค่าเป็นค่าคงที่เท่ากับ 2

**บรรทัดที่ 4 :** ใช้ #define กำหนดให้มาโคร TEST2 มีค่าเป็นค่าคงที่ 3 คูณด้วยค่าของมาโครTEST1 คือ 3*TEST1 = 3*2 = 6

**บรรทัดที่ 5 :** ประกาศตัวแปร times ชนิด int กำหนดค่าเริ่มต้นเป็น 11 และประกาศตัวแปร result ชนิด int เพื่อเก็บผลลัพธ์การคำนวณ

**บรรทัดที่ 9 :** ใช้ #if ทำการตรวจสอบว่า หากค่าของมาโคร TEST1 เป็นจริงแล้ว ให้ทำงานในบรรทัดที่ 10-11 แต่หากมาโครของ TEST1 เป็นเท็จ ก็จะเข้าสู่เงื่อนไข #elif ในบรรทัดที่ 12 เพื่อตรวจสอบว่าค่าของมาโคร TEST2 เป็นจริงหรือไม่หากค่าของมาโคร TEST2 เป็นจริง จะทำงานตามบรรทัดที่ 13-14 แต่หากเป็นเท็จก็จะเข้าสู่กรณี #else ในบรรทัดที่ 15 และทำงานในบรรทัดที่ 16

**บรรทัดที่ 10 :** กำหนดให้ตัวแปร result มีค่าเท่ากับค่าของมาโคร TEST1 คูณกับค่าของตัวแปร times

**บรรทัดที่ 13 :** ทำงานคล้ายกับ บรรทัดที่ 10 จะเปลี่ยนมาใช้ TEST2 คูณกับค่าของตัวแปร times แทน

**บรรทัดที่ 21-23 :** ตรวจสอบว่า result มีค่าเท่ากับ 0 หรือไม่ ในโปรแกรมเนื่องจาก result มีค่าเท่ากับ 22 (ไม่ใช่ 0) โปรแกรมจะไม่แสดงข้อความ "NOT DEFINE MACRO TEST1 & TEST2"โปรแกรมจะแสดงข้อความ "Result in 22"

**ตัวอย่างที่ 13.8** แสดงการทำงานของพรีโปรเซสเซอร์ไดเรกทีฟ (ตัวอย่างที่ 3)

```
1   #include <stdio.h>
2
3   #define A 1
4
5   int main()
6   {
7       #if A == 0
8           char language[4] = "THA";
9       #else
10          char language[4] = "ENG";
11      #endif
12
13      printf("Flag A is %d , Language is %s\n", A, language);
14      return 0;
15  }
```

**ผลลัพธ์ของโปรแกรม**

```
Flag A is 1 , Language is ENG
```

**อธิบายโปรแกรม**

**บรรทัดที่ 7 :** ใช้ #if ตรวจสอบค่าของมาโคร A ว่าเท่ากับ 0 ใช่หรือไม่ ถ้าใช่ ในบรรทัดที่ 8 ก็จะทำการประกาศตัวแปรแบบโกลบอลชนิดสตริงแบบ 4 ช่องขึ้นโดยกำหนดค่าเริ่มต้นเท่ากับ “THA” แต่หากไม่ใช่ ในบรรทัดที่ 10 ก็จะประกาศตัวแปรแบบโกลบอลชนิดสตริงแบบ 4 ช่องเช่นเดียวกัน แต่กำหนดให้มีค่าเป็น “ENG”

**ตัวอย่างที่ 13.9** แสดงการทำงานของพรีโปรเซสเซอร์ไดเรกทีฟ (ตัวอย่างที่ 4)

```
1   /*font1.inc*/
2   #include <stdio.h>
3
4   void zero()
5   {
6       printf("11111111 \n");
7       printf("11       11 \n");
8       printf("11       11 \n");
9       printf("11       11 \n");
10      printf("11       11 \n");
11      printf("11       11 \n");
12      printf("11       11 \n");
13      printf("11111111 \n\n");
14  }
15
16  void one()
17  {
18      printf("   11     \n");
19      printf("   11     \n");
20      printf("   11     \n");
21      printf("   11     \n");
22      printf("   11     \n");
23      printf("   11     \n");
24      printf("   11     \n");
25      printf("   11   \n\n");
26  }
```

```
1   /*font2.inc*/
2   #include <stdio.h>
3
```

```
4   void goodbye()
5   {
6       printf("11111111     11      11  11111111\n");
7       printf("11         11   11      11  11111111\n");
8       printf("11         11   11      11  11          \n");
9       printf("11111111     11      11  11111111\n");
10      printf("11111111      111111   11111111\n");
11      printf("11         11       11       11          \n");
12      printf("11         11       11       11111111\n");
13      printf("11111111         11       11111111\n");
14  }
```

```
1   #include <stdio.h>
2   #include "font1.inc"
3   #include "font2.inc"
4
5   int main()
6   {
7       printf("Welcome to Banner Program !!!\n\n\n");
8       zero();
9       one();
10      goodbye();
11      return 0;
12  }
```

**ผลลัพธ์ของโปรแกรม**

```
Welcome to Banner Program !!!



11111111
11      11
11      11
```

```
11     11
11     11
11     11
11     11
11111111


   11
   11
   11
   11
   11
   11
   11
   11
1111111    11      11  11111111
11      11 11      11  11111111
11      11 11      11  11
11111111   11      11  11111111
11111111     111111  11111111
11      11      11     11
11      11      11     11111111
11111111       11     11111111
```

**อธิบายโปรแกรม**

**บรรทัดที่ 2-3 :** นำไฟล์ font1.inc และ font2.inc

**บรรทัดที่ 7** : พิมพ์ข้อความ "Welcome to Banner Program !!!"

**บรรทัดที่ 8-10** : เรียกใช้ฟังก์ชัน zero() one() และ goodbye() มาใช้ ซึ่งมาจาก ไฟล์ font1.inc และ goodbye() มาจากไพล์ font2.inc