บทที่ 3

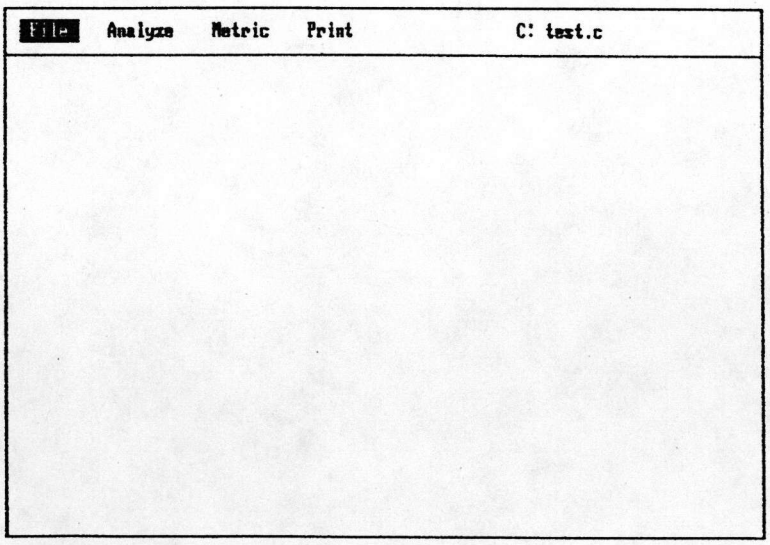
การออกแบบระบบ (System Design)

ในบทนี้จะนำเอาทฤษฎีในบทที่ 2 มาออกแบบเพื่อสร้างเป็นโปรแกรมวัดความซับซ้อนตามวิธีของ ฮอลสตีด เชน แมคคลู และโอวิโด โดยการออกแบบระบบของโปรแกรมตัววัดนี้จะแบ่งเป็น การออกแบบเมนู การรายงานผลบนหน้าจอและเครื่องพิมพ์ การออกแบบการทำงานในส่วนหลักๆของโปรแกรม และการออกแบบโครงสร้างข้อมูล

3.1 ออกแบบหน้าจอ

โปรแกรมวัดความซับซ้อน จะมีการโต้ตอบกับผู้ใช้ในลักษณะของพูลดาวน์เมนูโดยมีหน้าจอต่างๆดังต่อไปนี้

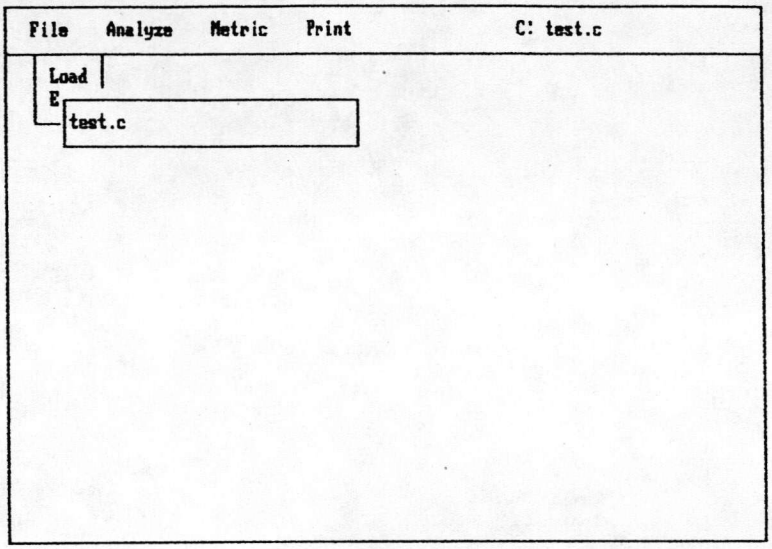
3.1.1 เมนูหลัก (Main menu) ประกอบด้วยตัวเลือกการทำงาน 4 ตัวคือ File Analyze Method Print



รูปที่ 3.1 แสดงหน้าจอของเมนูหลัก

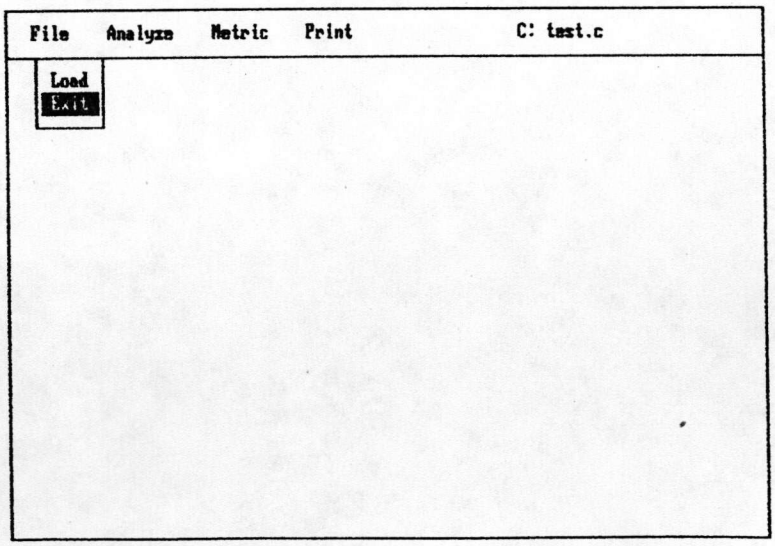
3.1.2 เมนู File ทำหน้าที่ดำเนินงานเกี่ยวกับแฟ้มข้อมูลประกอบด้วยเมนูย่อย 2 ตัวเลือกคือ

3.1.2.1 Load มีหน้าที่รับแฟ้มข้อมูลภาษาซีที่ต้องการวัด



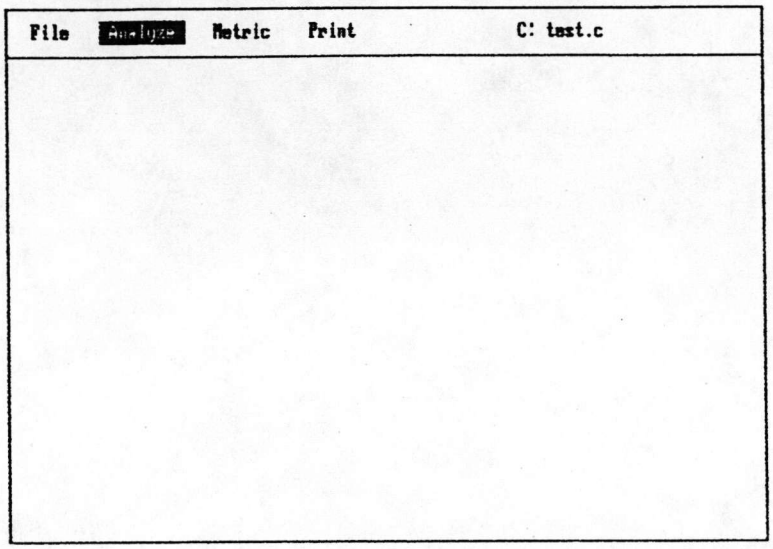
รูปที่ 3.2 แสดงหน้าจอการ load file

3.1.2.2 exit ออกจากโปรแกรม



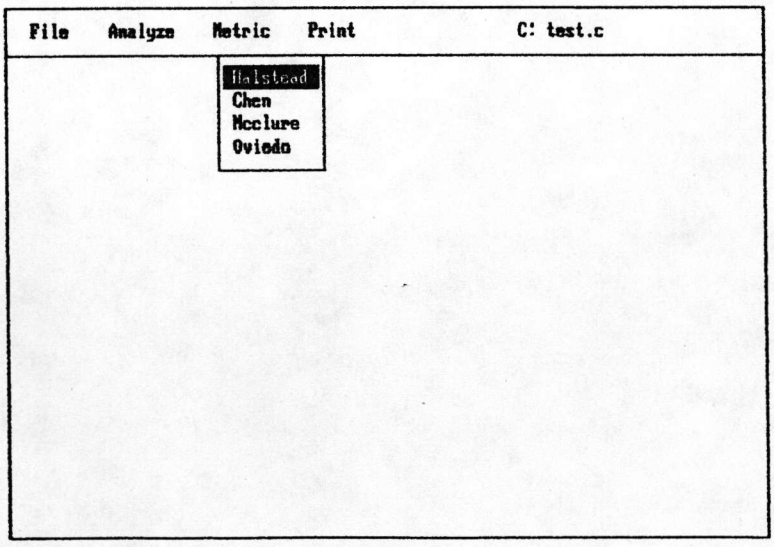
รูปที่ 3.3 แสดงหน้าจอการออกจากโปรแกรม

3.1.3 เมนู Analyze ทำหน้าที่ในส่วนของการวิเคราะห์ข้อมูลโดยจะแบ่งโปรแกรมที่นำมาวัดความซับซ้อนเป็นโมดูล แล้วนำแต่ละโมดูลนั้นมาสร้างเป็นโปรแกรมกราฟ และหาค่าของพารามิเตอร์ของวิธีวัดทั้งสี่แบบ



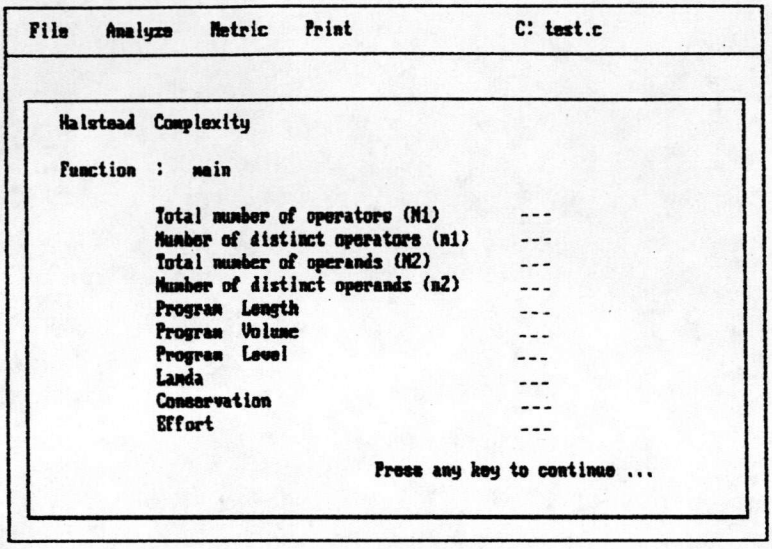
รูปที่ 3.4 แสดงหน้าจอของเมนู Analyze

3.1.3 เมนู Method ทำหน้าที่เป็นตัวเลือกวิธีการวัดค่าความซับซ้อนแล้วรายงานผลตามวิธีนั้น หน้าจอของเมนู Method นี้แสดงในรูปที่ 3.5 ซึ่งจะประกอบด้วยเมนูย่อย 4 ตัวคือ



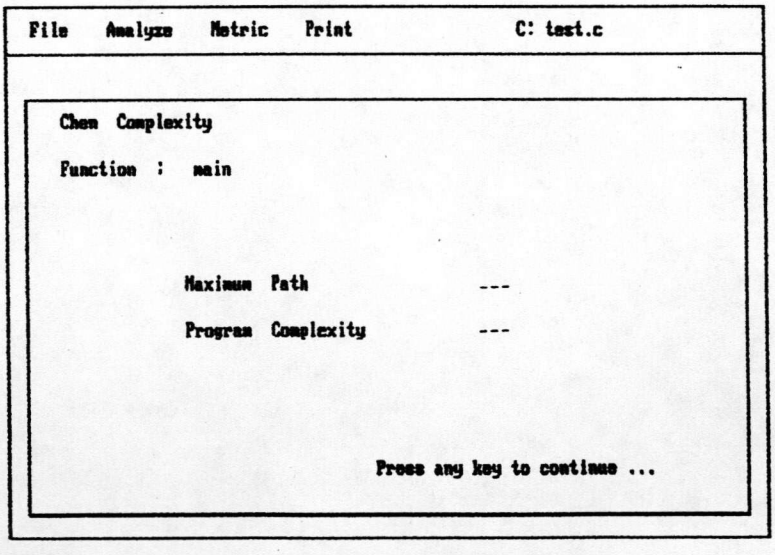
รูปที่ 3.5 แสดงหน้าจอของเมนู Method

3.1.3.1 Halstead คำนวณและรายงานค่าความซับซ้อนตามวิธีของฮอลสตีด



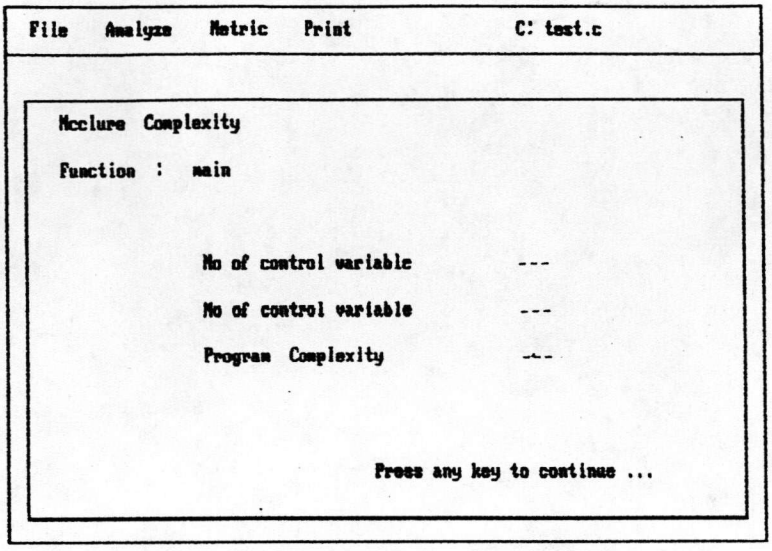
รูปที่ 3.6 แสดงหน้าจอผลการวัดตามวิธีของฮอลสตีด

3.1.3.2 Chen คำนวณและรายงานค่าความซับซ้อนตามวิธีของเชน



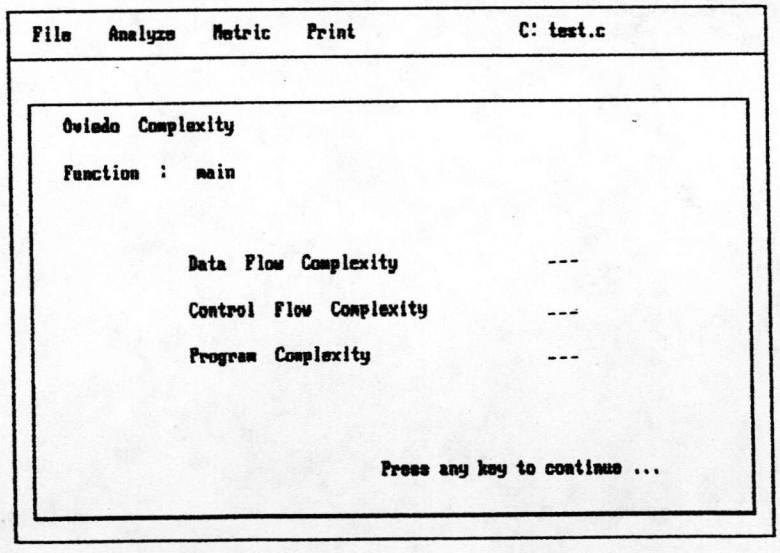
รูปที่ 3.7 แสดงหน้าจอผลการวัดตามวิธีของเชน

3.1.3.3 McClure คำนวณและรายงานค่าความซับซ้อนตามวิธีของแมคคลู



รูปที่ 3.8 แสดงหน้าจอผลการวัดตามวิธีของแมคคลู

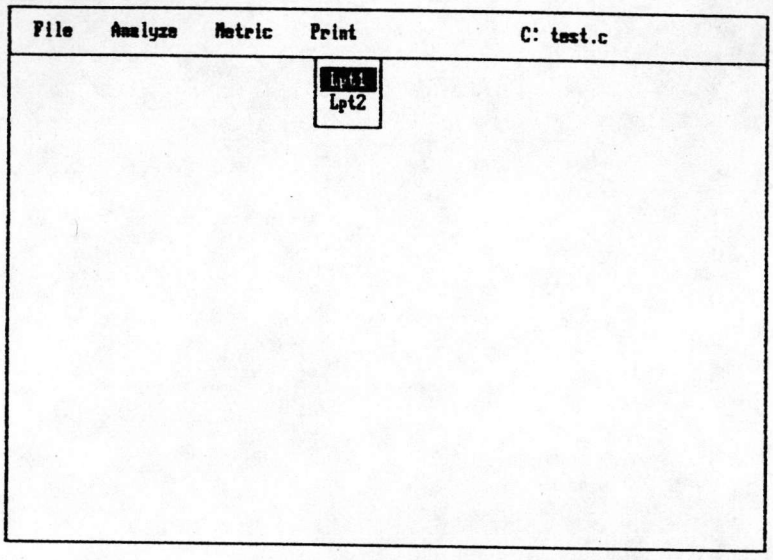
3.1.3.4 Oviedo คำนวณและรายงานค่าความซับซ้อนตามวิธีของโอวิโด



รูปที่ 3.9 แสดงหน้าจอผลการวัดตามวิธีของโอวิโด

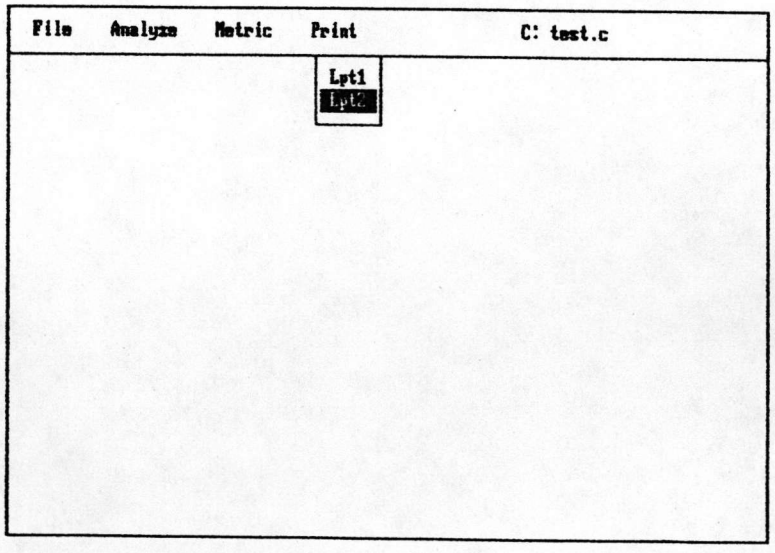
3.1.4 เมนู Print ทำหน้าที่พิมพ์ผลการวัดความซับซ้อนประกอบด้วยเมนูย่อย 2 ตัวเลือกคือ

3.1.4.1 Lpt1 พิมพ์รายงานผลการวัดออกทางพอร์ต lpt1



รูปที่ 3.10 แสดงหน้าจอการพิมพ์ทาง lpt1

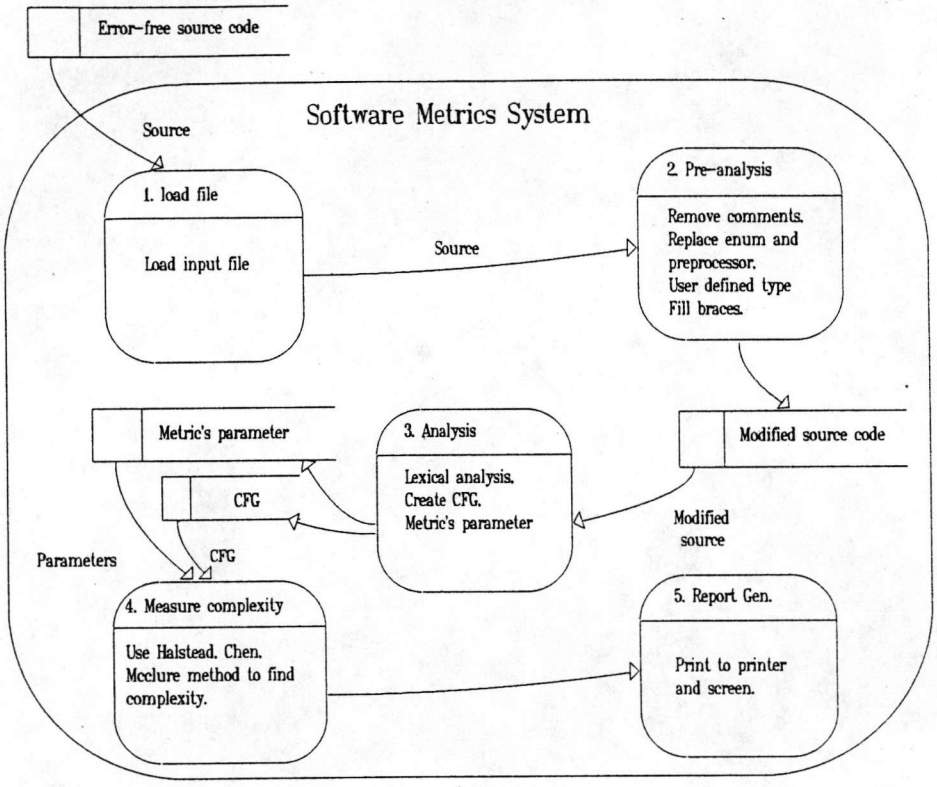
3.1.4.2 Lpt2 พิมพ์รายงานผลการวัดออกทางพอร์ต lpt2



รูปที่ 3.11 แสดงหน้าจอของการพิมพ์ทาง lpt2

3.2 ออกแบบการทำงานส่วนต่างๆของโปรแกรม

โปรแกรมต้วัดนี้ประกอบฟังก์ชัน 5 ส่วนหลักๆ มีรายละเอียดดังนี้



รูปที่. 3.12 แสดงส่วนประกอบของโปรแกรมวัดความซับซ้อน

3.2.1 **ส่วนรับข้อมูลเข้า** โปรแกรมต้วัดนี้จะทำการวัดค่า และแสดงผลออกในแต่ละโมดูลของโปรแกรม ดังนั้นโปรแกรมที่นำมาวัดจึงไม่จำเป็นต้องมีครบทุกโมดูล อาจนำมาวัดเพียงโมดูลใดโมดูลหนึ่งก็ได้

3.2.2 **ส่วนเตรียมข้อมูลก่อนทำการวิเคราะห์** เนื่องจากโปรแกรมภาษาซีที่นำมาวัดอาจต้องมีการเพิ่มเติม เปลี่ยนแปลง แก้ไขก่อนเพื่อให้มีความถูกต้อง และสะดวกต่อการหาค่าความซับซ้อน ซึ่งได้แก่

ก. **หมายเหตุ** (comment) คือข้อความที่อยู่ภายในเครื่องหมาย '/*' และ '*/' ซึ่งผู้เขียนโปรแกรมใส่ไว้เพื่ออธิบายถึงการทำงานของคำสั่งนั้น โปรแกรมต้วัดจะทำการตัดส่วนที่เป็นหมายเหตุทั้งหมดนี้ทิ้งไป เพื่อประหยัดหน่วยความจำและสะดวกต่อการทำงาน

ข. ประโยคคำสั่งพรีโปรเซสเซอร์ คือส่วนที่จะถูกประมวลผล (process) ก่อนที่ตัวแปรภาษาจะทำการแปลโปรแกรม ซึ่งจะประกอบด้วยประโยคต่างๆ ที่ขึ้นต้นด้วย '#' เช่น

1. ประโยค #include เป็นการดึงเอาแฟ้มโปรแกรมอื่นๆ มาแปลร่วมกันกับแฟ้มนั้น

2. ประโยค #define เนื่องจากภาษาซีอนุญาตให้ผู้เขียนโปรแกรมกำหนดค่าคงที่ไว้ล่วงหน้าโดยใช้แมคโคร เพื่อความสะดวกและการสื่อความหมายที่ดี เมื่อจะทำการแปลจึงนำเอาค่าคงที่เหล่านี้มาแทนที่แมคโครนี้ เช่นเดียวกันโปรแกรมตัววัดนี้ก่อนทำการวัดค่าความซับซ้อน ต้องนำเอาค่าคงที่เหล่านี้มาแทนที่ทุกๆจุดภายในโปรแกรมที่มีการเรียกใช้ นอกจากนี้ เราสามารถกำหนดให้แมคโครประกอบด้วยอาร์กิวเมนต์ได้เวลามีประโยคไหนอ้างอิงถึงแมคโครเหล่านี้ ก็จะมีการแทนค่าของอาร์กิวเมนต์อย่างเหมาะสม เช่น ให้มีการกำหนดแมคโครชื่อ MAX ดังนี้

```
#define  MAX(A,B) ((A)>(B)?(A):(B))
```

และในโปรแกรมมีประโยค

```
x = MAX(x,y);
```

โปรแกรมตัววัดจะทำการแทนที่คำสั่งนี้เป็น

```
x = ((x)>(y)?(x):(y));
```

3. ประโยคที่แปลอย่างมีเงื่อนไข ได้แก่ #if #ifdef #endif #else #elif และ #ifndef เป็นการสร้างข้อกำหนดให้โปรแกรมเพื่อสามารถแปล โดยให้มีบางส่วนของโปรแกรมแตกต่างกันไปตามความต้องการ

ค. ประโยค enum เช่นเดียวกับประโยค #define ประโยค enum นี้ใช้ในการกำหนดค่าคงที่ที่เป็นตัวเลขไว้ล่วงหน้า แต่อาจมีการกำหนดค่าเริ่มต้นหรือไม่ก็ได้หรืออาจมีการกำหนดเพียงบางค่าก็ได้ โปรแกรมตัววัดนี้จะทำการตรวจหาจุดที่เรียกใช้แล้วนำเอาค่าคงที่ที่เป็นตัวเลขไปแทนที่ในโปรแกรม เช่นมีการประกาศ

```
#define MAXSIZE 20
enum size {TINY, SMALL=10, MEDIUM, BIG=-1, LARGE=MAXSIZE};
enum object {CIRCLE, TRIANGLE=SMALL, RECTANGLE};
        .
        .
x = (SMALL + BIG) * y[CIRCLE] + z[LARGE];
```

โปรแกรมตัววัดจะแปลงส่วนของโปรแกรมนี้เป็น

```
enum size {TINY, SMALL=10, MEDIUM, BIG=-1, LARGE=20};
enum object {CIRCLE, TRIANGLE=10,RECTANGLE};
        .
        .
x = (10 - 1) * y[0] + z[20];
```

ง. ประโยค typedef มีไว้เพื่อสร้างประเภทของข้อมูลขึ้นใหม่เพื่อแทนประเภทของข้อมูลเดิม โดยจะถูกดำเนินการในขั้นตอนของการคอมไพล์ และอาจกำหนดได้ทั้งในและนอกฟังก์ชัน การทำงานของโปรแกรมตัววัดนี้ จะเก็บประเภทที่กำหนดขึ้นใหม่นี้ไว้ในรายการเชื่อมโยง (linked list) และตัวแปรที่ถูกกำหนดประเภทขึ้นใหม่จะมีประเภทของโทเกนเป็น USRDEFTYPE

จ. วงเล็บปีกกา คำสั่งควบคุมสายงาน (control flow) ในภาษาซีได้แก่ if-else for while do switch เราสามารถเขียนวงเล็บปีกกา { และ } ครอบเพื่อรวบรวมการประกาศ และคำสั่งเข้าด้วยกันเป็นคำสั่งประกอบ (compound statement) เป็นการบอกขอบเขตของฟังก์ชันหรือคำสั่งควบคุมสายงานหรือที่เราเรียกว่าบล็อก (block)

สำหรับการทำงานของโปรแกรมตัววัดนี้ เพื่อให้สะดวกต่อการเขียนโปรแกรมหาขอบเขตของคำสั่งควบคุมสายงาน จึงได้เพิ่มโปรแกรมส่วนที่ทำหน้าที่เดิมวงเล็บปีกกา โดยมีการทำงานในลักษณะการเรียกซ้ำ (recursive) โดย

1. สามารถหาขอบเขตบล็อกจากส่วน else ที่อาจละในคำสั่ง if-else ได้ เนื่องจากอาจเกิดความกำกวมขึ้นเมื่อละส่วน else จากลำดับ if ที่ซ้อนกัน

(nest if) โดยภาษาซีกำหนดให้ส่วน else จับคู่กับ if ที่ใกล้ที่สุด

2. สามารถหาขอบเขตบล็อก จากกลุ่มคำสั่งควบคุมสายงานที่ซ้อนและต่อเนื่องกัน และในกรณีมีคำสั่งควบคุมสายงานเพียงคำสั่งเดียวอาจเขียนวงเล็บปีกกาหรือไม่เขียนก็ได้ หรืออาจเขียนบ้างไม่เขียนบ้างก็ได้ ตัวอย่างการทำงานของโปรแกรมตัววัดจะได้ผลดังนี้ เช่น ส่วนของโปรแกรมที่นำเข้าเป็น

```
if (x == 1)
    while (y == 2)
        do
            a[i] = 0;
        while (z == 3);
else
    for (j =0; j<10; j++)
        a[j] = 1;
```

เมื่อผ่านส่วนการเตรียมข้อมูลก่อนทำการวิเคราะห์ จะแปลงโปรแกรมข้างต้นเป็น

```
if (x == 1) {
    while (y == 2) {
        do {
            a [i] = 0;
        } while (z == 3);
    }
} else {
    for (j =0; j<10; j++) {
        a[j] = 1;
    }
}
```

ฉ. ฟังก์ชันต้นแบบ (function prototype) โปรแกรมภาษาซีอาจมีการประกาศชื่อของฟังก์ชันไว้ต้นโปรแกรม ซึ่งโปรแกรมตัววัดต้องสามารถแยกแยะความแตกต่างว่าโทเกนนั้นเป็นฟังก์ชันต้นแบบ หรือเป็นตัวแปรโกลบอล หรือเป็นตัวฟังก์ชันจริง

3.2.3 ส่วนวิเคราะห์ ทำหน้าที่ดังนี้

ก. **เก็บตัวแปรและประเภทของตัวแปร** ประเภทของตัวแปรพื้นฐานในภาษาซีได้แก่ char int float double นอกจากนี้ยังมีตัวขยาย (qualifier) ซึ่งใช้ขยายประเภทข้อมูลพื้นฐานดังกล่าว ได้แก่ short long unsigned เช่น long int x; นอกจากนี้ยังมีการกำหนดขอบเขตของตัวแปร เป็น static register extern auto เช่น extern int x; โปรแกรมตัววัดนี้แบ่งจะเก็บประเภทตัวแปรพร้อมกับตัวแปรนั้นโดยแบ่งการเก็บเป็น

1. ตัวแปรโกลบอล เนื่องจากตัวแปรโกลบอลเป็นตัวแปรส่วนกลางที่ทุกโมดูลรู้จักและอาจถูกเรียกใช้ได้ในทุกโมดูล จึงต้องเก็บตัวแปรเหล่านี้ไว้ตลอดการทำงานของโปรแกรมตัววัด

2. เก็บตัวแปรโลคอล เก็บตัวแปรโลคอลที่ได้ประกาศไว้ภายในโมดูลไว้ในรายการเชื่อมโยง และจะทำการ free ข้อมูลในรายการเชื่อมโยงนี้ทั้งหมดเมื่อวัดค่าความซับซ้อนของโมดูลนั้นเสร็จแล้ว

ข. เก็บอาร์กิวเมนต์ที่ส่งค่าระหว่างฟังก์ชัน โดยเก็บไว้ในรายการเชื่อมโยงเดียวกับตัวแปรโลคอล การส่งค่าอาร์กิวเมนต์ในภาษาซีอาจทำได้สองลักษณะเป็นแบบเก่าและแบบใหม่ดังนี้

Old style	Modern style

int sum (x, y)
int x, y;
{
}
	int sum (int x, int y)
	{
	}

ค. **แบ่งโปรแกรมเป็นโมดูล** โปรแกรมตัววัดจะต้องทำการตรวจหา (scan) ตำแหน่งเริ่มต้นรวมทั้งชื่อของแต่ละโมดูล และเก็บค่าไว้ เพื่อเวลาจะคำนวณหาค่าความซับซ้อนของแต่ละโมดูลจะได้นำค่าเหล่านี้มาใช้

ง. ตัดโปรแกรมเป็นโทเกนย่อย แล้วเก็บค่าโทเกนเหล่านี้ไว้ในรายการเชื่อมโยงเป็นพารามิเตอร์ของฮอลสตีดแต่ละโทเกนแบ่งเป็นประเภทต่างๆกันดังนี้

ประเภทโทเกน	อธิบาย
ALNUM	โทเกนที่ประกอบด้วยอักขระและตัวเลข
ASSIGN	ตัวแปรที่มีการกำหนดค่า
BLOCK	ตัวดำเนินการวงเล็บปีกกา {...} แสดงขอบเขตของกลุ่มคำสั่ง
CHARCONSTANT	ค่าคงที่ที่เป็นอักขระ เช่น 'x', '\077' ,'\\', '\r'
CLOSEPAREN	ตัวดำเนินการวงเล็บปิด ')'
COLON	ตัวดำเนินการ ':'
COMMA	ตัวดำเนินการ ','
COMPARE	ตัวดำเนินการ <, >, <=, >=, ==, !=
DELIMITER	ตัวดำเนินการที่เป็นอักขระคั่น เช่น +, -, *, /, !, ->, &
FINISHED	จุดสิ้นสุดของโมดูล
FUNCTION	ฟังก์ชัน เช่น func (a, b);
KEYWORD	คำหลักของภาษาซีเช่น if,else, while, int, char
NUMBER	ค่าคงที่ที่เป็นตัวเลข
OPENPAREN	ตัวดำเนินการวงเล็บเปิด '('
PREPROCESSOR	เช่น #define, #include, #if, #else
SEMICOLON	ตัวดำเนินการ ; แสดงจุดสิ้นสุดของคำสั่ง
STDLIB	ฟังก์ชันมาตรฐาน เช่น strcpy (x, y);, printf ("x");
STRING	ค่าคงที่ที่เป็นอักขระ
USRDEFTYPE	ประเภทของข้อมูลที่ผู้เขียนโปรแกรมกำหนดจากคำสั่ง typedef
VARIABLE	ตัวแปรต่างๆ

ตารางที่ 3.1 แสดงประเภทของโทเกนในโปรแกรมตัววัด

จ. สร้างโปรแกรมกราฟ นำเอาโทเกนที่ได้มาประมวลผลผลสร้างเป็นโปรแกรมกราฟ ที่ประกอบด้วยโหนดและเส้นเชื่อม และเก็บค่าตัวแปรทุกตัวในแต่ละโหนดไว้ด้วยกัน โดยแยกตามประเภทของตัวแปรว่ามีค่าใช้ได้เฉพาะถิ่น (locally avaliable variable) หรือเป็นตัวแปรที่ได้รับการตีแผ่เฉพาะถิ่น (locally exposed variable)

3.2.4 ส่วนการหาค่าความซับซ้อน เป็นส่วนที่นำเอาข้อมูลที่ได้จากการวิเคราะห์มาคำนวณหาค่าความซับซ้อนตามทฤษฎีของ ฮอลสตีด, เชน, แมคคลูและโอวิโด

3.2.5 ส่วนรายงานผล ทำการรายงานค่าที่วัดได้ออกทางจอภาพและทางเครื่องพิมพ์

3.3 การออกแบบโครงสร้างข้อมูล

เนื่องจากโปรแกรมการวัดความซับซ้อนนี้ ต้องทำการวัดความซับซ้อนจากทฤษฎีตัววัด 4 แบบ ซึ่งแบ่งเป็นตัววัดแบบตัววัดขนาด, ตัววัดโครงสร้างข้อมูล และตัววัดโครงสร้างตรรกะ จึงต้องเก็บทั้งรายละเอียดของพารามิเตอร์ในลักษณะที่แตกต่างกัน รวมทั้งต้องเก็บโครงสร้างของโปรแกรมในลักษณะของโปรแกรมกราฟ ดังนั้นจึงแบ่งการเก็บข้อมูลเป็น 2 แบบดังนี้

3.3.1 รายการเชื่อมโยง เนื่องจากความยืดหยุ่นในการเพิ่มเติมหรือลดข้อมูลได้ดีกว่าใช้ตัวแปรชุด และกรณีที่ยังไม่ทราบจำนวนของข้อมูลล่วงหน้า การขยายขนาดของรายการก็สามารถทำได้สะดวกกว่า โดยใช้เก็บข้อมูลเกี่ยวกับ

ก. พารามิเตอร์ของตัววัดฮอลสตีด ได้แก่จำนวนชนิดของตัวดำเนินการ จำนวนชนิดของตัวถูกดำเนินการ จำนวนของตัวดำเนินการทั้งหมดในโมดูล และจำนวนของตัวถูกดำเนินการทั้งหมดในโมดูล โดยมีการนิยามในโปรแกรม ดังนี้

```
typedef struct Plist {
    char item [NAME_LEN];
    int  count;
    struct Plist *next;
} PLIST;
```

ข. ชื่อและตำแหน่งของฟังก์ชัน ใช้ในการเข้าถึงตำแหน่งของฟังก์ชันนั้นๆ มีรูปแบบการนิยาม

```
typedef struct Funclist {
        char name [MAX_LEN];
        char *loc;
        struct Funclist *next;
    } FUNC_LIST;
```

ค. ตัวแปรโกลบอลและโลคอล มีรูปแบบการนิยาม

```
typedef struct List {
        char item [MAX_LEN];
        struct List *next;
    } LIST;
```

3.3.2 กราฟ เป็นโครงสร้างข้อมูลแบบหนึ่งที่ประกอบด้วยโหนดและเส้นเชื่อม จึงเหมาะกับการแทนโครงสร้างของโปรแกรมในรูปของโปรแกรมกราฟหรือ CFG มากที่สุด โครงสร้างข้อมูลแบบกราฟ ยังแบ่งได้เป็นกราฟแบบมีทิศทาง และแบบไม่มีทิศทาง และอาจแทนได้ทั้งในรูปของ ตัวแปรชุดและในรูปของรายการเชื่อมโยง สำหรับงานวิจัยนี้เลือกใช้ กราฟแบบมีทิศทางโดยจะแทนโหนดด้วยตัวแปรชุด และแทนเส้นเชื่อมกับตัวแปรในรูปของรายการเชื่อมโยง ซึ่งมีข้อดีคือใช้หน่วยความจำน้อยกว่าเก็บเป็นตัวแปรชุด และทำให้มีจำนวนเส้นเชื่อมกับตัวแปรได้ไม่จำกัด

ตัววัดที่ต้องใช้โครงสร้างข้อมูลแบบกราฟ ในการคำนวณหาค่าความซับซ้อน คือ ตัววัดเชน แมคคลู และโอวิโด โดยมีรูปแบบของการนิยามดังนี้

```
typedef struct Edge {
        int    nodeptr;
        struct Edge *next;
    } EDGE;
```

EDGE เป็นนิยามของเส้นเชื่อมระหว่างโหนด โดยเก็บเลขที่ของโหนดใดๆ ที่โหนดปัจจุบันเชื่อมกับโหนดนั้นไว้ในตัวแปร nodeptr

```
typedef struct List {
        char item [MAX_LEN];
        struct List *next;
      } LIST;
```

LIST เป็นนิยามของโครงสร้างที่ใช้เก็บชื่อตัวแปรที่ปรากฏภายในโหนดทั้งหมด โดยเก็บไว้ในตัวแปร item ที่มีขนาด MAX_LEN ซึ่งเป็นค่าที่ได้ทำการนิยามไว้ก่อน (predefine) โดยให้มีค่าเป็น 32

```
typedef struct {
        int node [MAX_NODES];
        EDGE *edge [MAX_NODES];
        LIST *define [MAX_NODES];
        LIST *expose [MAX_NODES];
      } GRAPH ;
```

GRAPH เป็นนิยามของโครงสร้างข้อมูลที่ใช้แทนโปรแกรมกราฟ โดยมีตัวแปร node เก็บหมายเลขประจำโหนดนั้น ตัวแปร edge เป็นตัวชี้ไปยังหมายเลขของโหนดที่เชื่อมกับโหนดปัจจุบัน ตัวแปร define กับ expose เป็นตัวชี้ไปยังตัวแปรมีค่าใช้ได้เฉพาะถิ่นและตัวแปรที่ได้รับการตีแผ่เฉพาะถิ่น (locally exposed variable) ทั้งหมดภายในโหนดนั้น ซึ่งเป็นพารามิเตอร์ที่ใช้หาค่าความซับซ้อนของตัววัดโอวิโด และยังใช้ในการหาตัวแปรควบคุมซึ่งใช้หาค่าความซับซ้อนตามวิธีของแมคคลูได้อีกด้วย