บทที่ 3

การออกแบบภาษา

บทนี้จะอธิบายถึงลักษณะของภาษาที่ได้ออกแบบขึ้น โดยอธิบายไวยากรณ์ของภาษาต้นแบบในรูปของแผนภาพไวยากรณ์ (syntax diagram) และอธิบายความหมาย (semantics) ของภาษาซึ่งมีรายละเอียดดังนี้

3.1 ลักษณะโดยทั่วไปของภาษา

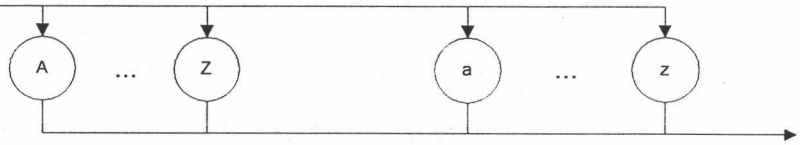
ภาษาที่ใช้เป็นภาษาต้นแบบ จะเป็นภาษาที่มีความสามารถในการทำงานแบบประมวลผลพร้อมกัน โดยวากยสัมพันธ์ (syntax) ที่กำหนดขึ้นจะใช้เลียนแบบ วากยสัมพันธ์ ของภาษาซี (C language) เป็นส่วนใหญ่ ทั้งนี้เนื่องจากภาษาซีเป็นภาษาที่มีรู้จักกันดี และผู้ใช้กันอย่างแพร่หลาย

ตัวแปลภาษาจะแบ่งการทำงานออกเป็นสองขั้นตอนได้แก่ขั้นตอนคอมไพเลอร์ (compiler) และขั้นตอนอินเตอร์พรีเตอร์ (interpreter) โดยในขั้นตอนคอมไพเลอร์จะทำการแปลคำสั่งจากภาษาในระดับสูง อันได้แก่ภาษาต้นแบบ เพื่อสร้างเป็นรหัสกลาง (intermediate code) โดยรหัสกลางที่ได้จะเป็นรหัสกลางซึ่งถูกบรรจุอยู่ในรูปของรหัสไบท์ (byte code) ขั้นตอนที่สองจะเป็นการสร้างอินเตอร์พรีเตอร์ ที่นำเอารหัสกลางที่ได้ไปปฏิบัติการ ซึ่งรหัสกลางนี้สามารถนำไปประมวลผลพร้อมกัน

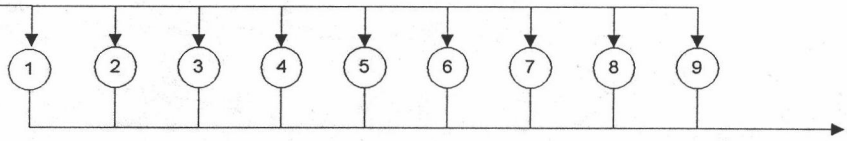
3.2 แผนภาพไวยากรณ์ (Syntax Diagram)

วากยะสัมพันธ์ของภาษาสามารถอธิบายได้โดยแผนภาพไวยากรณ์ (syntax diagram) ดังนี้

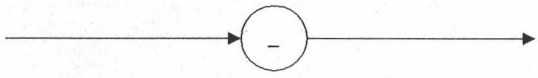
letter



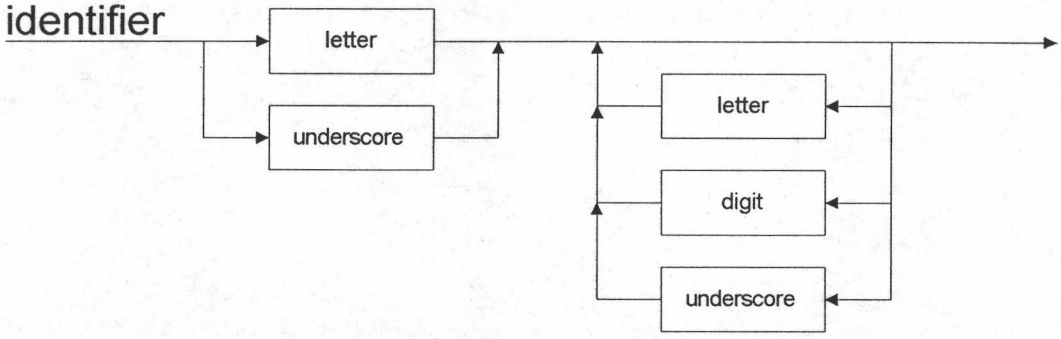
digit



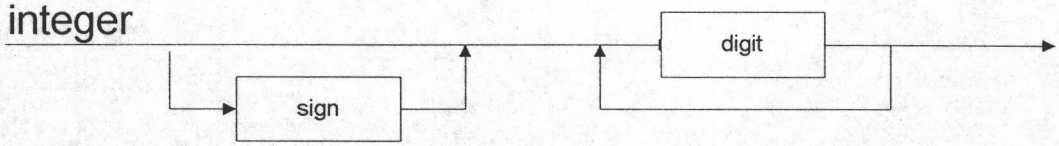
underscore



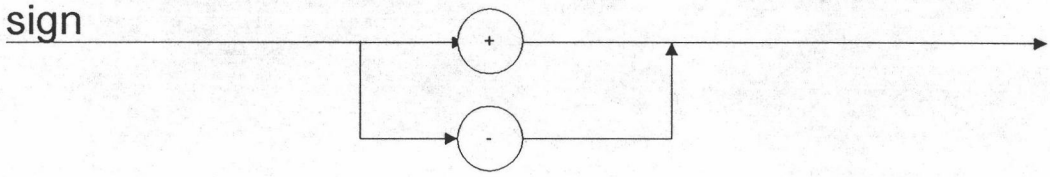
identifier



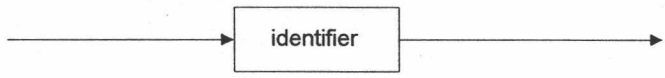
integer



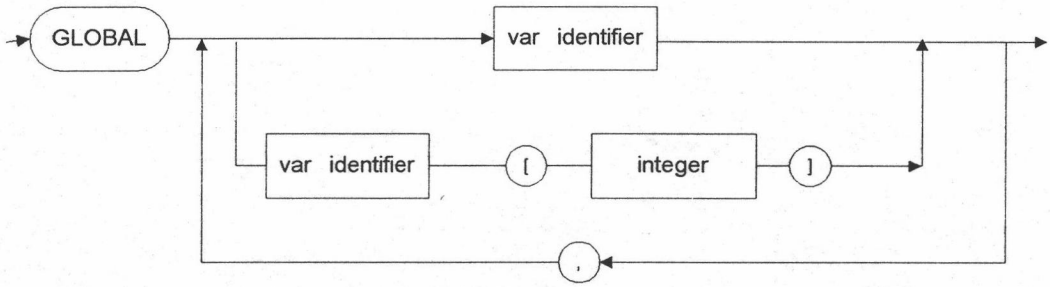
sign



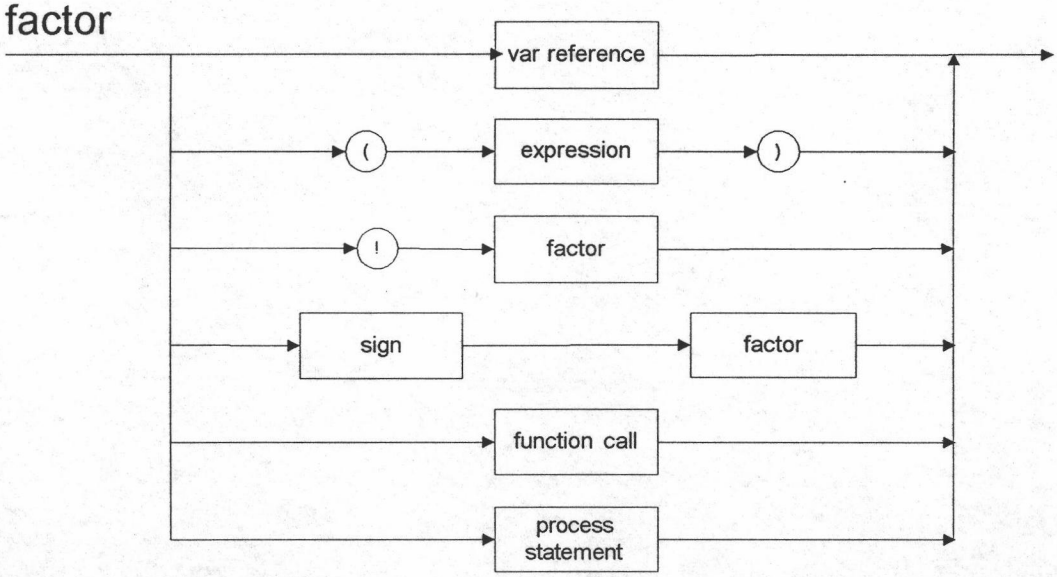
variable identifier
process identifier
function identifier



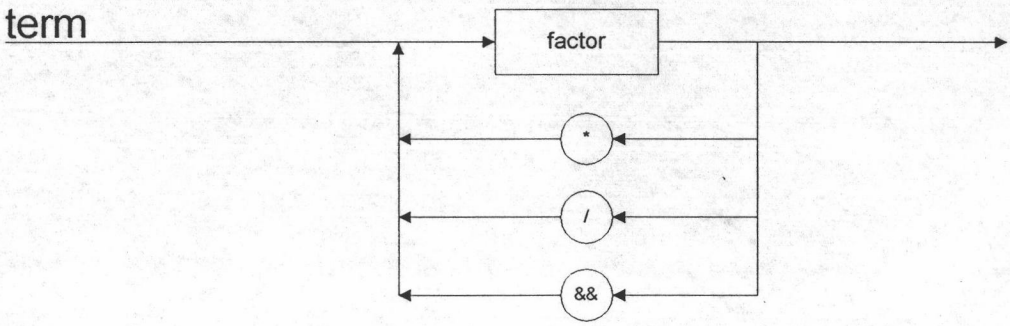
variable declaration



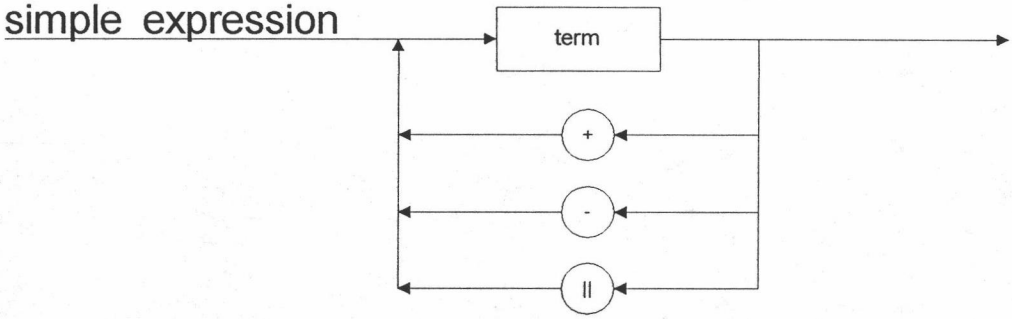
factor



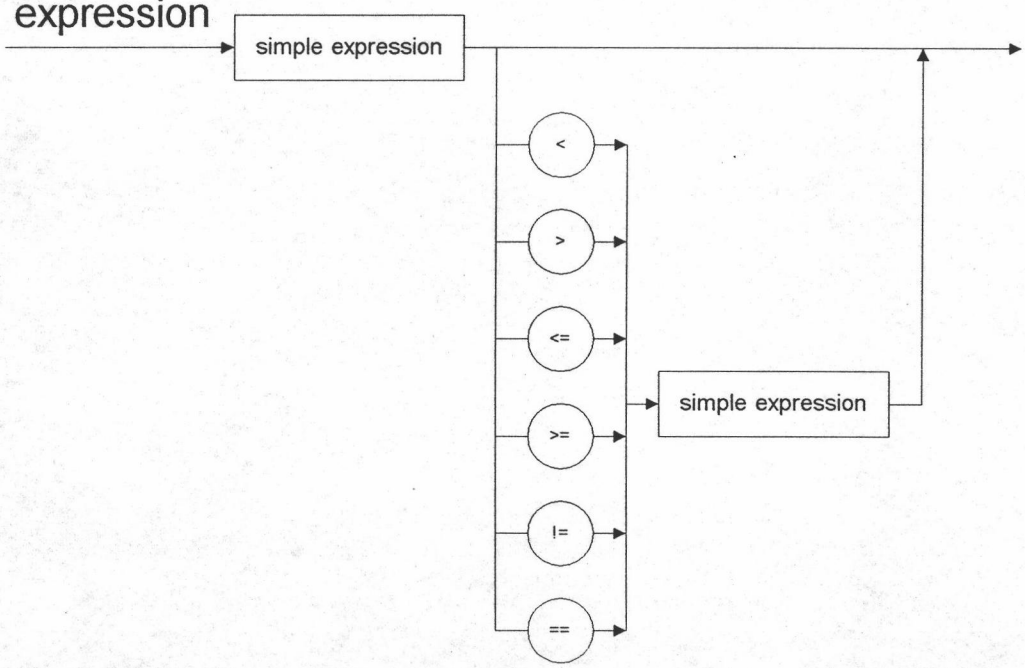
term



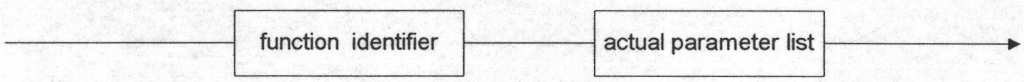
simple expression



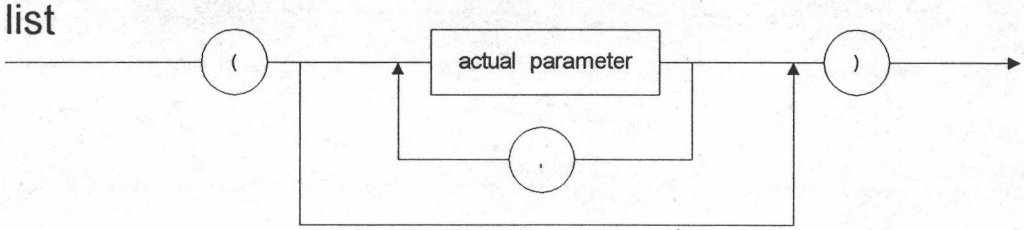
expression

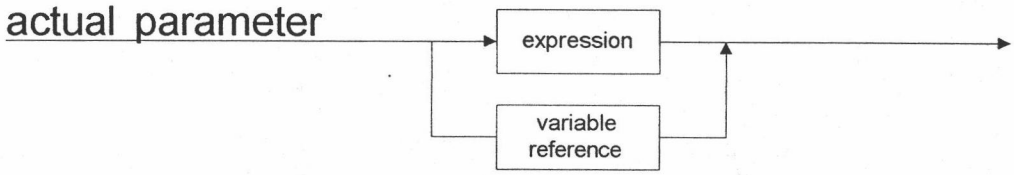


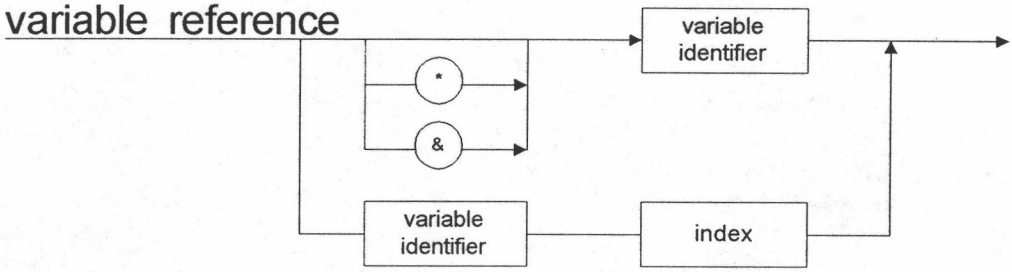
function statement

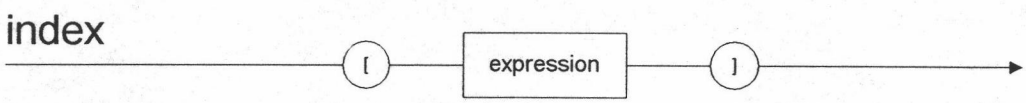


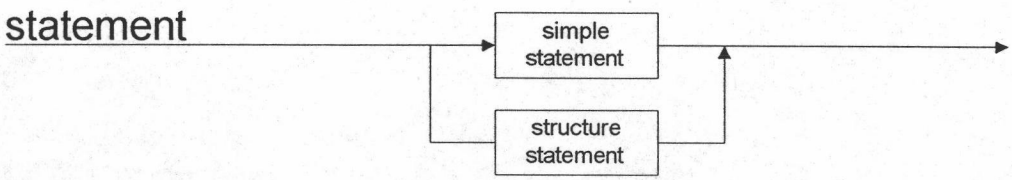
actual parameter list

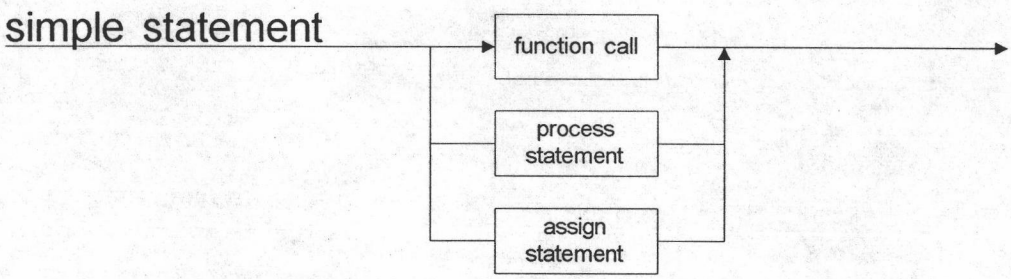


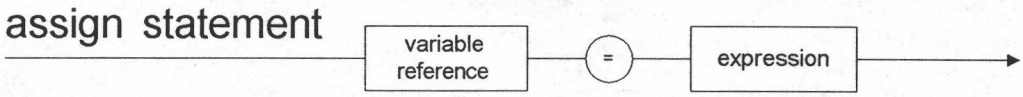

actual parameter
expression
variable reference


variable reference
variable identifier
*
&
variable identifier
index

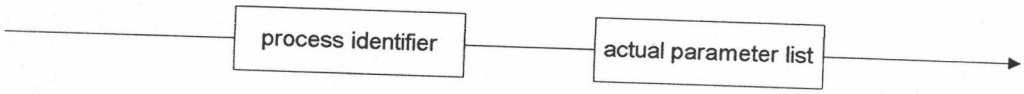

index
[
expression
]


statement
simple statement
structure statement

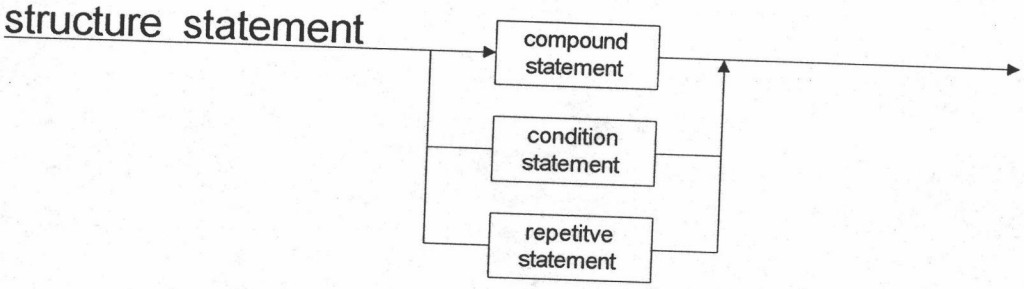

simple statement
function call
process statement
assign statement


assign statement
variable reference
=
expression

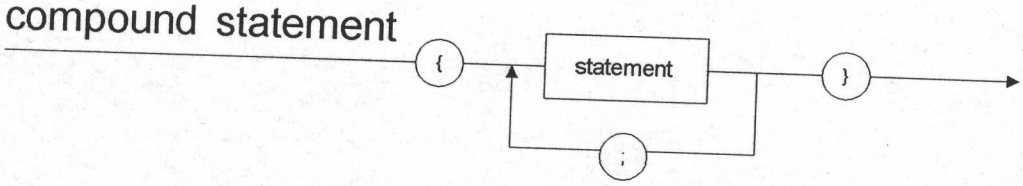
process statement



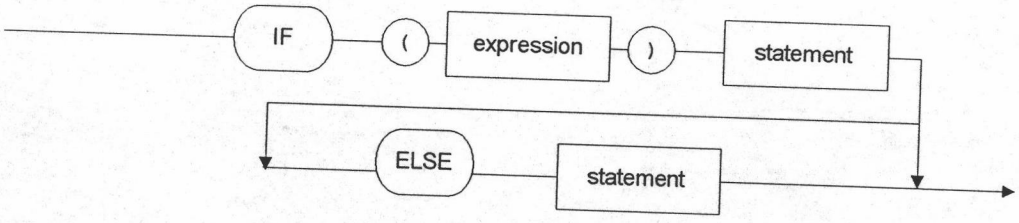
structure statement



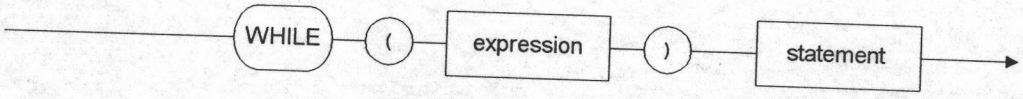
compound statement



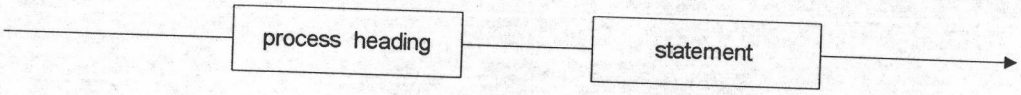
conditional statement



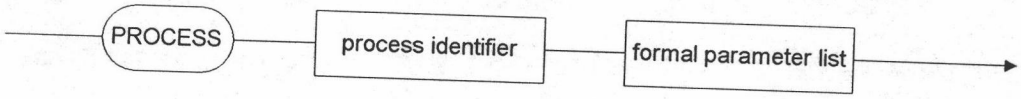
repetitive statement



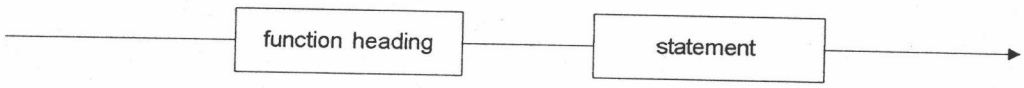
process declaration



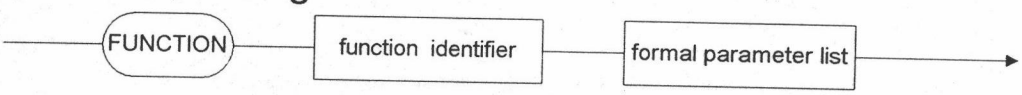
process heading



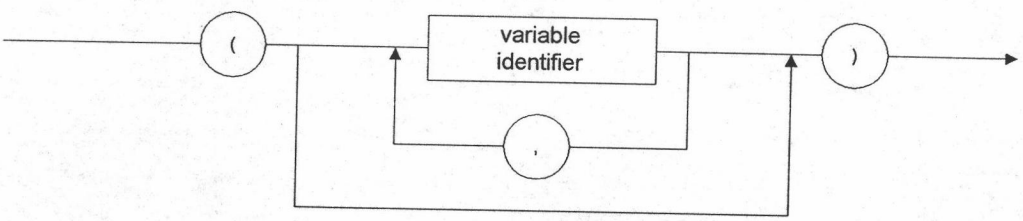
function declaration



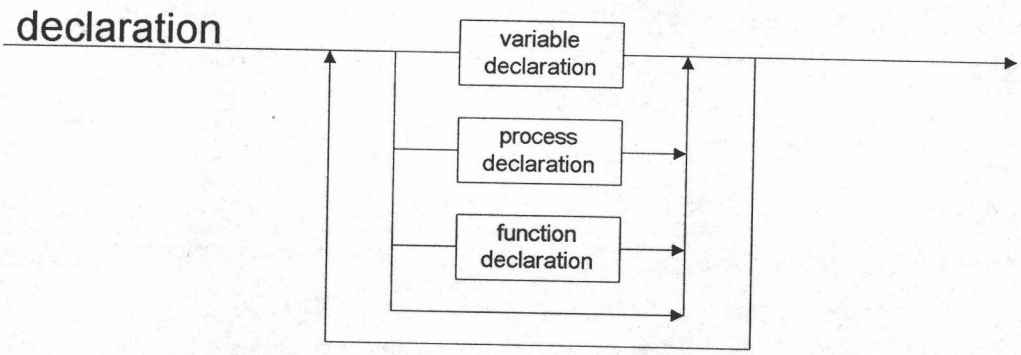
function heading



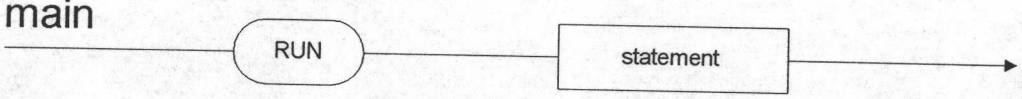
formal parameter list



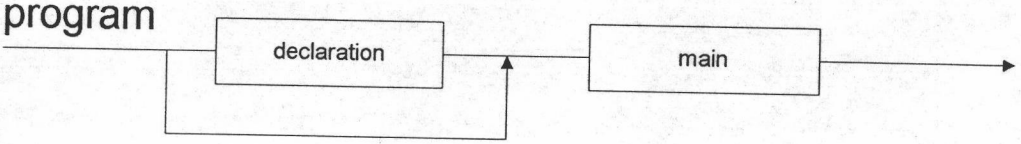
declaration



main



program



3.3 โครงสร้างของภาษา

โครงสร้างของภาษาต้นแบบ ประกอบด้วยส่วนประกอบดังนี้

```
GLOBAL
        global variable declarations ;
FUNCTION function declaration ;
        body of function
PROCESS process declaration ;
        body of process ;
RUN
        main body of program
```

3.3.1 ชนิดของข้อมูล (data type)

ชนิดข้อมูลของภาษาต้นแบบ จะเป็นแบบไม่มีชนิดข้อมูล (untype) ซึ่งข้อมูลที่ใช้จะเป็นข้อมูลที่มีความยาวคงที่ (fix length) 16 บิต ทั้งนี้เพื่อความคล่องตัวในการใช้งาน อีกทั้งระบบนี้เป็นระบบสำหรับศึกษาค้นคว้า ซึ่งมิได้ให้ความสำคัญกับระบบชนิดของข้อมูล (type system) จุดสนใจของงานวิจัยนี้จะเป็นเรื่องของรหัสกลาง (intermediate code) ที่สามารถทำงานแบบประมวลผลพร้อมกัน จึงเลือกใช้โครงสร้างภาษาที่ไม่มีชนิดข้อมูล ซึ่งมีความยุ่งยากน้อยกว่า

3.3.2 ตัวแปรและค่าคงที่

ข้อมูลแต่ละชุดในโปรแกรมหนึ่งๆ อาจอยู่ในรูปของตัวคงที่ (constant) หรือ ตัวแปร (variable) ซึ่งตัวคงที่ ที่บรรจุอยู่ในข้อมูลแต่ละตัว จะเป็นข้อมูลแบบไม่มีชนิดข้อมูล (untype) ที่มีขนาดคงที่ 16 บิต

3.3.3 นิพจน์ (expression)

นิพจน์ในภาษาอาจสร้างขึ้นได้จากตัวคงที่ ตัวแปร หรือจากฟังก์ชัน ตามกฎที่คล้ายคลึงกับกฎทางพีชคณิต เช่น หากกำหนดตัวแปร A,B และ C สามารถเขียนเป็นนิพจน์

$$A + B * C$$

โดยที่ สัญลักษณ์ + และ * แทนการดำเนินการ (operation) การบวกและการคูณตามลำดับ

3.3.4 อันดับความสำคัญของตัวดำเนินการ (precedence of operators)

ตัวดำเนินการแต่ละตัวจะมีอันดับความสำคัญในการดำเนินการไม่เท่ากัน เมื่อมีการคำนวณในนิพจน์ อันดับความสำคัญของตัวดำเนินการจะเป็นเครื่องกำหนดว่าตัวดำเนินการตัวใดจะได้ดำเนินการก่อน

operator	precedence	categories
& !	4	unary operators
* / &&	3	multiplying operators
+ - ||	2	adding operators
== != < <= > >=	1	relational operators

ตารางที่ 3.1 precedence of operators

กฎที่ใช้ในการกำหนดอันดับการดำเนินการ จะพิจารณาดังนี้

- ตัวดำเนินการที่มีอันดับความสำคัญสูงกว่าตัวดำเนินการตัวอื่น จะถูกดำเนินการก่อน
- หากตัวดำเนินการสองตัวมีอันดับความสำคัญเท่ากันจะดำเนินการจากด้านซ้ายของนิพจน์ก่อน

เช่นนิพจน์ 7 + 3 * 5 จะดำเนินการ 3 คูณ 5 ก่อน แล้วจึงนำผลลัพธ์ที่ได้มาบวกกับค่า 7 ซึ่งจะได้คำตอบเป็น 22 เป็นต้น

3.3.5 สเตตเมนต์ (statement)

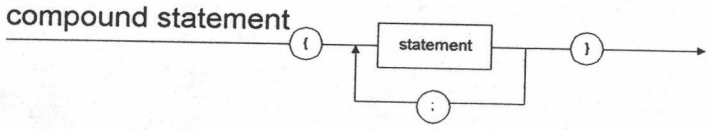
สเตตเมนต์แบ่งออกได้เป็นดังนี้

- สเตตเมนต์กำหนดค่า (Assignment statement)

ตัวแปรตัวหนึ่งจะถูกกำหนดค่าได้โดยสเตตเมนต์กำหนดค่า ซึ่งประกอบด้วยตัวแปรตามด้วยเครื่องหมายที่ใช้ในการกำหนดค่า (=) และนิพจน์ซึ่งแสดงการคำนวณตัวอย่างเช่น A = 15 หมายถึงสเตตเมนต์กำหนดค่าที่กำหนดค่า 15 ให้แก่ตัวแปร A หรือกล่าวอีกในหนึ่งก็คือนำค่า 15 ไปเก็บไว้ที่ตัวแปร A ดังนั้นสเตตเมนต์กำหนดค่า มิได้หมายถึงการเท่ากันเชิงคณิตศาสตร์

- สเตตเมนต์ผสม (compound statement)

ในการแก้ปัญหาต่างๆ ปัญหาส่วนใหญ่ ไม่สามารถเขียนเป็นอัลกอริทึมด้วยสเตตเมนต์เดียว การรวมสเตตเมนต์หลายๆสเตตเมนต์เข้าด้วยกัน จะเป็นสเตตเมนต์ผสม โดยจะปรากฏอยู่ในเครื่องหมาย { และ } ซึ่งแต่ละสเตตเมนต์จะแยกกันด้วยเครื่องหมายอัฒภาค(semicolon) ซึ่งอธิบายได้โดยแผนภาพไวยากรณ์ดังนี้

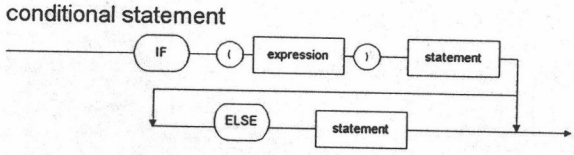


รูปที่ 3.1 ไวยากรณ์ของ สเตตเมนต์ผสม

- สเตตเมนต์เงื่อนไข (conditional statement)

โปรแกรมที่มีความสามารถในการทำงานในการปฏิบัติตามคำสั่งเรียงตามลำดับเท่านั้น ก็ยังไม่อาจถือว่าเป็นโปรแกรมที่แก้ปัญหาส่วนใหญ่ได้ โดยเฉพาะในปัญหาที่มีความสลับซับซ้อนอาจต้องมีการกำหนดทางเลือกในการแก้ปัญหาอย่างใดอย่างหนึ่งจากทางเลือกหลายทาง

ในภาษาต้นแบบมีโครงสร้างที่ใช้ในการแก้ปัญหาทางเลือกทางใดทางหนึ่งจากทางเลือกสองทาง โดยใช้สเตตเมนต์เงื่อนไข ซึ่งมีรูปแบบดังนี้



รูปที่ 3.2 ไวยากรณ์ของ สเตตเมนต์เงื่อนไข

จากไดอะแกรมข้างต้น สามารถแบ่งได้เป็นสองกรณีคือ

กรณีที่หนึ่ง

IF (expression)
statement

ในกรณีแรกจะตรวจสอบนิพจน์ (expression) ซึ่งเป็นนิพจน์เงื่อนไขในการทำงานสเตตเมนต์ต่อไป หากนิพจน์เงื่อนไขมีค่าเป็นศูนย์จะถือว่านิพจน์นี้เป็นเท็จ และจะไม่ปฏิบัติคำสั่งที่ปรากฏในสเตตเมนต์ถัดมา หากนิพจน์เงื่อนไขมีค่าเป็นค่าอื่นๆ ที่ไม่เท่ากับศูนย์ จะปฏิบัติตามสเตตเมนต์ที่กำหนด

ค่านิพจน์เงื่อนไข	ค่าความจริง
เท่ากับศูนย์	เท็จ
ไม่เท่ากับศูนย์	จริง

ตารางที่ 2 ค่าความจริงของนิพจน์

กรณีที่สอง

```
IF (expression)
        statement                    // True action
ELSE
        statement                    // False action
```

ในกรณีนี้จะตรวจสอบนิพจน์เงื่อนไขหากมีค่าเป็นจริงคือมีค่าไม่เท่ากับศูนย์ จะดำเนินการในสเตตเมนต์ ที่เป็น true action มิฉะนั้นจะดำเนินการตามสเตตเมนต์ภายหลังคำสงวน ELSE ซึ่งเป็นสเตตเมนต์ false action

ในสเตตเมนต์เงื่อนไขที่มีสองทางเลือกจะพบว่าสเตตเมนต์ที่ต้องดำเนินการภายหลังทางเลือกจะเป็นสเตตเมนต์สเตตเมนต์เดียวเท่านั้น หากไม่สามารถนิยามการทำงานภายในสเตตเมนต์เดียวได้ ก็สามารถใช้สเตตเมนต์ผสมก็ได้

ในกรณีที่มีทางเลือกหลายทาง เราสามารถนิยามโดยใช้โครงสร้างของสเตตเมนต์ IF ซ้อนกันหลายๆครั้งก็ได้ ทั้งนี้เนื่องจากสเตตเมนต์ภายหลังสเตตเมนต์เงื่อนไขจะเป็นสเตตเมนต์แบบใดก็ได้ เช่น

```
IF  (a != 0)
    IF (a>0)
        P = POS
    ELSE
        P = NEG
```

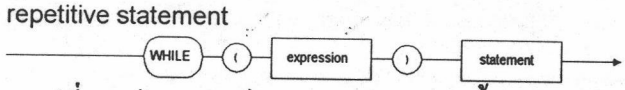
ในตัวอย่างนี้สเตตเมนต์เงื่อนไขได้ปรากฏเป็นสเตตเมนต์ย่อยในสเตตเมนต์เงื่อนไขอีกอันหนึ่ง ดังนั้นเพื่อขจัดความกำกวมของภาษา เราจะกำหนดให้สเตตเมนต์เงื่อนไขที่ปรากฏภายในสุดจะต้องเป็นสเตตเมนต์ที่สมบูรณ์ หรืออาจจะมองเสมือนเป็นสเตตเมนต์ผสมที่อยู่ในวงเล็บดังนี้

```
IF  (a != 0) {
    IF (a>0)
        P = POS
    ELSE
        P = NEG
}
```

ดังนั้น *ELSE* ในที่นี้จะเป็นของ สเตตเมนต์ *IF (a>0)*

- สเตตเมนต์ทำซ้ำ (repetitive statement)

ในการแก้ปัญหาที่ต้องมีการทำงานซ้ำเดิม ภาษาต้นแบบได้ออกแบบให้สามารถทำงานโดยใช้ while statement ซึ่งมีแผนภาพไวยากรณ์ ดังนี้



รูปที่ 3.3 ไวยากรณ์ของ สเตตเมนต์ทำซ้ำ

การทำงานของ while statement จะเริ่มจากการตรวจสอบนิพจน์เงื่อนไข (expression) หากนิพจน์เงื่อนไขมีค่าไม่เท่ากับศูนย์ (ซึ่งถือว่ามีค่าเป็นจริง) จะดำเนินการตามสเตตเมนต์ที่กำหนด จนจบสเตตเมนต์ และจะกลับมาตรวจสอบนิพจน์เงื่อนไขอีกครั้ง ซึ่งสเตตเมนต์นี้จะทำซ้ำจนกว่าจะมีค่าเป็นศูนย์จึงจะจบการทำงานในสเตตเมนต์ทำซ้ำ และจะทำงานตามคำสั่งถัดไป

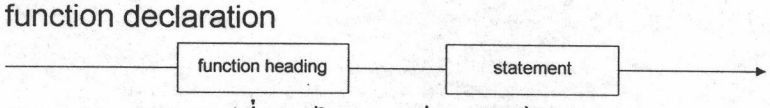
ค่าในนิพจน์เงื่อนไขของ while statement จะเหมือนกับใน conditional statement คือ

ค่านิพจน์เงื่อนไข	ค่าความจริง
เท่ากับศูนย์	เท็จ
ไม่เท่ากับศูนย์	จริง

ตารางที่ 3 ค่าความจริงของนิพจน์

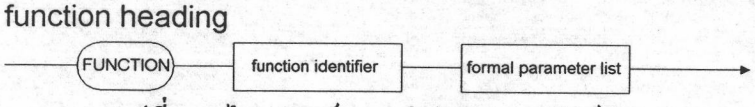
3.3.6 ฟังก์ชัน (function)

ในภาษาต้นแบบ ออกแบบให้สามารถเขียนโปรแกรมย่อยได้โดยใช้โครงสร้างของฟังก์ชัน การประกาศฟังก์ชัน (function declaration) มีองค์ประกอบสองส่วนคือ ส่วนหัวของฟังก์ชัน (function heading) และส่วนสเตตเมนต์ ดังรูป



รูปที่ 3.4 ไวยากรณ์ของฟังก์ชัน

ซึ่งส่วนหัวของฟังก์ชันจะประกอบด้วยชื่อฟังก์ชัน (function identifier) และ formal parameter list



รูปที่ 3.5 ไวยากรณ์ของ ส่วนประกาศฟังก์ชัน

ในการประกาศชื่อของฟังก์ชัน ชื่อที่จะใช้จะต้องไม่ซ้ำกับคำสงวน (reserve word) และจะต้องไม่ซ้ำกับชื่อฟังก์ชัน หรือชื่อตัวแปรที่ประกาศไว้ก่อนแล้ว สำหรับ formal parameter list จะประกอบด้วยตัวแปรที่ต้องการส่งค่ากลับจำนวนไม่เกิน 255 ตัวแปร หรือไม่มีตัวแปรเลยก็ได้

เมื่อคอมไพเลอร์พบตัวแปรในโปรแกรมต้นฉบับ คอมไพเลอร์จะเก็บชื่อตัวแปรไว้ในตารางสัญลักษณ์ (ซึ่งจะอธิบายโดยละเอียดในบทที่ 4) เพื่อใช้ในการอ้างอิงถึงตัวแปรนั้นอีกในภายหลัง

ตัวแปรที่ปรากฏในฟังก์ชันจะถือเป็นตัวแปรเฉพาะที่ (local variable) ทั้งหมด และจะจัดเก็บอยู่ในตารางสัญลักษณ์เฉพาะที่ ตัวแปรเหล่านี้จะปรากฏเฉพาะในการเรียกโปรแกรมย่อยในครั้งหนึ่งๆเท่านั้น และตัวแปรเหล่านี้จะหายไปหลังจากการดำเนินการใช้ฟังก์ชันเหล่านี้เสร็จสิ้นแล้ว

ในกรณีที่มีการประกาศชื่อตัวแปรซ้ำกับระหว่างตัวแปรเฉพาะที่ และตัวแปรส่วนกลาง (global variable) โปรแกรมจะดำเนินการโดยยึดถือค่าจากตัวแปรเฉพาะที่เป็นสำคัญ

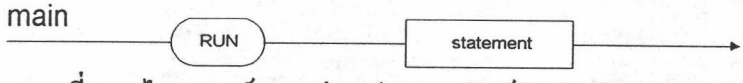
หากในฟังก์ชันมีการอ้างถึงตัวแปรที่ไม่ปรากฏในตารางตัวแปรส่วนกลางหรือในตารางตัวแปรเฉพาะที่เลย โปรแกรมจะดำเนินการเพิ่มตัวแปรดังกล่าวในตารางตัวแปรเฉพาะที่ทั้งหมด หรืออาจจะกล่าวได้ว่าตัวแปรที่ปรากฏครั้งแรกในฟังก์ชัน จะเป็นตัวแปรเฉพาะที่ทั้งหมด และในการดำเนินการ (execute) โปรแกรมจะค้นหาตัวแปรจากตัวแปรเฉพาะที่ก่อนที่จะค้นหาในตารางตัวแปรส่วนกลาง

กฎในการจำแนกตัวแปรเฉพาะที่และตัวแปรส่วนกลาง

- ตัวแปรที่ปรากฏอยู่ภายหลังคำสงวน GLOBAL ทุกตัวจะเป็นตัวแปรส่วนกลาง โดยที่ตัวแปรส่วนกลางเป็นตัวแปรที่ใช้ร่วมกันในทุกกระบวนการ ซึ่งจะอธิบายโดยละเอียดในหัวข้อกระบวนการ
- ตัวแปรนอกจากข้อหนึ่ง จะเป็นตัวแปรเฉพาะที่ทั้งหมด

3.3.7 โปรแกรมหลัก ฟังก์ชันดำเนินการ (run)

ในภาษาต้นแบบ กำหนดให้ทุกๆโปรแกรมย่อยจะต้องเป็นฟังก์ชัน (รวมทั้งกระบวนการก็ถือว่าเป็นโปรแกรมย่อยที่ดำเนินการในรันไทม์เอนวิรอนเมนต์ที่แตกต่างกัน) และกำหนดให้ฟังก์ชันดำเนินการ (function run) จะเป็นฟังก์ชันพิเศษที่จะต้องมีในทุกๆโปรแกรม และถูกกำหนดให้เป็นฟังก์ชันที่ถูกดำเนินการก่อนฟังก์ชันใดๆเสมอ ฟังก์ชันรันจะเป็นโปรแกรมหลักในทุกๆรันไทม์เอนวิรอนเมนต์ ฟังก์ชันดำเนินการ จะมีโครงสร้างดังนี้ คือจะต้องอยู่ในส่วนท้ายสุดของการประกาศใดๆ

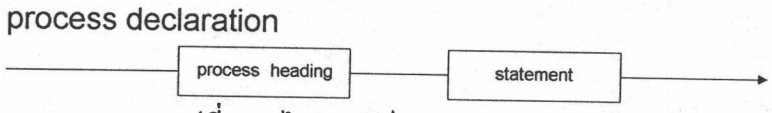


ที่ 3.6 ไวยากรณ์ของ ส่วนประกาศฟังก์ชันดำเนินการ

3.3.8 กระบวนการ (process)

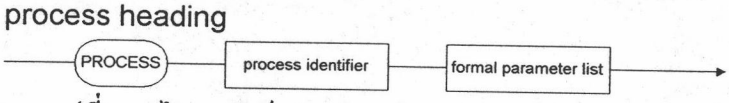
กระบวนการคือส่วนของโปรแกรมย่อยเช่นเดียวกับฟังก์ชัน แต่จะมีความแตกต่างกันคือ กระบวนการจะทำงานในรันไทม์เอนวิรอนเมนต์เฉพาะของตนเอง เสมือนว่ากระบวนการจะประมวลผลเป็นอิสระจากกระบวนการอื่น

การประกาศกระบวนการ (process declaration) มีองค์ประกอบสองส่วนคือ ส่วนหัวของกระบวนการ (process heading) และส่วนสเตตเมนต์ ดังรูป



รูปที่ 3.7 ไวยากรณ์ของ กระบวนการ

ซึ่งส่วนหัวของกระบวนการจะประกอบด้วยชื่อกระบวนการ (process identifier) และ formal parameter list



รูปที่ 3.8 ไวยากรณ์ของ ส่วนประกาศของกระบวนการ

ตัวแปรเฉพาะที่โปรแกรม จะปรากฏอยู่ในรันไทม์เอนวิรอนเมนต์ของกระบวนการทุกกระบวนการด้วย ซึ่งกระบวนการจะมองเห็นตัวแปรเหล่านี้เป็นอิสระจากกัน แต่ในกรณีที่ต้องการให้กระบวนการต่างๆใช้ตัวแปรร่วมกัน ต้องกำหนดตัวแปรส่วนกลาง (global memory) ซึ่งเป็นตัวแปรที่ประกาศตามหลังคำสงวน Global ตัวแปรนี้จะเป็นตัวแปรที่กระบวนการต่างๆใช้ร่วมกัน

3.3.9 คอมเมนต์ (comment)

เมื่อคอมไพเลอร์พบเครื่องหมาย // ข้อความที่อยู่หลังเครื่องหมายนี้เป็นคอมเมนต์ คอมไพเลอร์จะไม่ทำการแปลโปรกรมที่พบหลังเครื่องหมายนี้ทั้งบรรทัด และจะอ่านข้ามบรรทัดนั้นไป