

G

T

G

T

5

E

5

N

O

N

O

o

S

o

G

T

G

T

5

E

5

N

O

N

O

o

s

o

G

T

G

T

5

E

5

Syntax

&

Semantic

Introduction

- 1) syntax โครงสร้างภาษาเชิงนัย
- 2) semantics ความหมายของโปรแกรม

ex. syntax → DD / DD / DDDD

01/02/2001 → US Jan 2, 2001
other Feb 1, 2001

syntax เดิมกัน คนละ semantics

Don't forget

compiler ทำให้ที่

ตรวจสอบความผิดพลาดของ

syntax (syntax error)

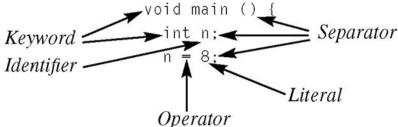
language description

- 1) แบบเรียน (tutorials) เช่นรู้ syntax และ semantics ผ่านการทดลองต่อไป
- 2) คู่มืออ้างอิง (reference manuals) คิมพิช ใจไทย ในรูปแบบ context-free grammar
- 3) นิยามแบบเป็นทางการ (formal definition) ยิ่งขึ้นรูปแบบสัญลักษณ์ ใช้เพื่อวินิจฉัย ผู้เขียนช่วย

Describing syntax

- sentence ประกอบที่นี่จาก set ต่อไปนี้
- language set ของ sentence
- lexeme หน่วยย่อยของ syntax
- token lexeme รวมกันเป็น token

Comment → // compute result = the nth Fibonacci number



ถ้า lexeme 12 unit = token(keyword , Identifier , Operator , Literal , Separator)

Syntactic Elements

- character set
- identifiers ตัวแปร
- Operator symbols
- keywords / Reserved word keyword เท่าที่ฝึกอบรมมา ไม่ต้องจำรู้
- Reserved word นามเต็มที่ใช้
- comments
- Separator & Brackets (), , ;
- Expression
- statements

Formal approach ເກົ່າວ່ອນບາຍ syntax

- Recognizers ອີ່ຈຸນ compilers
- Generators ອີ່ generate sentence
- Context - Free Grammars (CFG) ປະກອບ 4 ສ່ວນ

- P (production) ສ່ວນ string ໃນກາຍາ
- T (terminal symbol) ສັນລົກຂົມລືນສຸດ ບໍລິສັດຮຽດແຕກເປັນ ລົກສັດໃຫ້ອີ່
- N (nonterminal symbol) ຂົງລືນລົກສຸດ ສາຂາຮັດແຕກເປັນ ລົກສັດໃຫ້ອີ່
- S (start symbol) ສັນຄໍາທີ່ໄໝຕົ້ນຂອງກາຍານີ້ ຖ.

ex. production

$$A \rightarrow w \rightarrow \text{nonterminal symbol} / \text{terminal symbol}$$

↑

nonterminal symbol

	non	non	non
<ປະໂຍດ>	->	<ປະຄານ><ກວຽາ><ກວມ>	
<ປະຄານ>	->	ຈັນ ເຂອ ເຈາ	
<ກວຽາ>	->	ກິນ ຕີ	
<ກວມ>	->	ຫ້າວ ຜູນ້າ	
<ປະໂຍດ>	->	<ປະຄານ><ກວຽາ><ກວມ>	
	ຈັນ	ກິນ	ຫ້າວ
	ເຂອ	ຕີ	ຜູນ້າ

• Backus - Naur Form (BNF) ກົດ CFG

→ ປະກອບຕົ້ນ, "!" ນິວ

$$A \rightarrow w \rightarrow \text{! } T \text{ ! } N \text{ ! }$$

↓

nonterminal symbol ແລ້ວ S ດັກກຳນົດຢູ່ນ production ພຶກຕົ້ນ

↳ production (nonter)

binaryDigit \rightarrow 0 \rightarrow terminal symbol

binaryDigit \rightarrow 1

ex.

$$\bullet \text{ binary Digit } \rightarrow 011$$

integer \rightarrow digit | integer digit

digit \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<integer> ::= <digit> | <integer> <digit>

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

ຫົວໆ

integer ::= digit | integer digit

digit ::= "0" ... "9"

< > ແກນຕົ້ນ nonterminal ຕ້ອງກຳນົດແກນຕົ້ນ terminal

::= ແກນ \rightarrow

- Derivation ໃຊ້ສຳນັກວ່າມາດົດຕໍ່ string ເຈັນຄູກຕອນ syntax ພ້ອມໄວ້ ສີ 2 ຮູບແບບ ຄື້ນ
 - ການແປລະຫຼາຍສູງ (leftmost derivation) ແປລະ nonterminal symbol ເປັນ terminal symbol
ແລະ ທີ່ໄປເປັນ ຈຸນທຳສູງ
 - ການແປລະຫຼາຍລູດ (rightmost derivation) ເພື່ອອນ leftmost ແກ້ວກຳດາກຈວາ

Ex. 352 ເປັນ integer ພ້ອມໄວ້

$$\begin{aligned}
 <\text{integer}> &\Rightarrow <\text{integer}><\text{digit}> \\
 &\Rightarrow <\text{integer}> 2 \\
 &\Rightarrow <\text{integer}><\text{digit}> 2 \\
 &\Rightarrow <\text{integer}> 52 \\
 &\Rightarrow <\text{digit}> 52 \\
 &\Rightarrow 352 \quad (\text{rightmost})
 \end{aligned}$$

$$\begin{aligned}
 <\text{integer}> &\Rightarrow <\text{integer}><\text{digit}> \\
 &\Rightarrow <\text{integer}><\text{digit}><\text{digit}> \\
 &\Rightarrow <\text{digit}><\text{digit}><\text{digit}> \\
 &\Rightarrow 3 <\text{digit}><\text{digit}> \\
 &\Rightarrow 35 <\text{digit}> \\
 &\Rightarrow 352 \quad (\text{leftmost})
 \end{aligned}$$

Ex. $A = B^*(A+C)$ ເປັນ assign ພ້ອມໄວ້

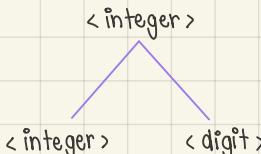
```

<assign> ::= <id> = <expr>
<id>   ::= A | B | C
<expr> ::= <id> + <expr>
          | <id> * <expr>
          | ( <expr> )
          | <id>
  
```

$$\begin{aligned}
 <\text{assign}> &\Rightarrow <\text{id}> = <\text{expr}> \\
 &\Rightarrow A = <\text{expr}> \\
 &\Rightarrow A = <\text{id}> * <\text{expr}> \\
 &\Rightarrow A = B^* <\text{expr}> \\
 &\Rightarrow A = B^* (<\text{expr}>) \\
 &\Rightarrow A = B^* (<\text{id}> + <\text{expr}>) \\
 &\Rightarrow A = B^* (A + <\text{expr}>) \\
 &\Rightarrow A = B^* (A + <\text{id}>) \\
 &\Rightarrow A = B^* (A + C) \\
 \therefore A = B^* (A + C) \quad \text{ເປັນ assign } \#
 \end{aligned}$$

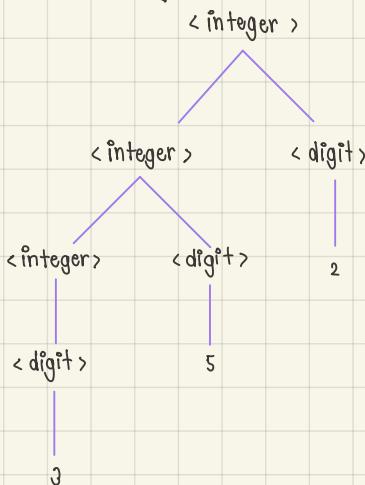
- Parse tree ເນື້ອເສດອອຳ string ຊີ້ວ່າ, ເຈັນຄູກຕອນ ຄວາມໂຂຍາກສົນ BNF

$$<\text{integer}> \Rightarrow <\text{integer}><\text{digit}>$$

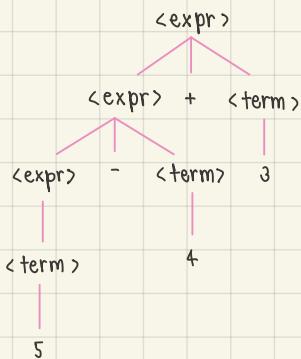


ສຳກັນອອນຕົ້ນໄຟ້ຢ່ອງໃນ parse tree ຄື້ນ node ທີ່ເຈັນແທນ left-hand side ນີ້ອອນ nonterminal ສຳກັນແທນ node ຢື່ອງ → ພກ

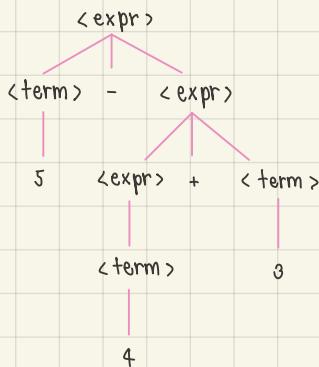
ex. parse tree ของ 352 เป็นตัวสกุลที่ร่างเป็น Integer แล้ว



ex. $(5 - 4) + 3$



ex. $5 - (4+3)$



- Extended BNF (EBNF) คือ BNF ที่มีกฎเพิ่มเติม

$\langle \text{expr} \rangle ::= \langle \text{expr} \rangle + \langle \text{term} \rangle$
 $\quad\quad\quad | \langle \text{expr} \rangle - \langle \text{term} \rangle$
 $\quad\quad\quad | \langle \text{term} \rangle$

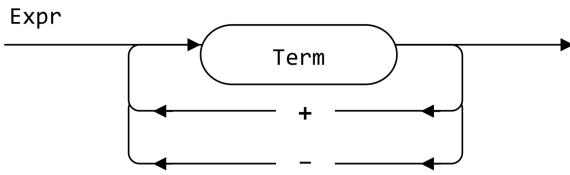
เงื่อนไขของ EBNF คือ $\langle \text{expr} \rangle ::= \langle \text{expr} \rangle \{ (+|-) \langle \text{term} \rangle \}$

ถ้า $\langle \text{if-statement} \rangle ::= \text{if}(\langle \text{expr} \rangle) \langle \text{statement} \rangle$
 $\quad\quad\quad | \text{if}(\langle \text{expr} \rangle) \langle \text{statement} \rangle \text{ else } \langle \text{statement} \rangle$

เงื่อนไขของ EBNF คือ $\langle \text{if-statement} \rangle ::= \text{if}(\langle \text{expr} \rangle) \langle \text{statement} \rangle [\text{else } \langle \text{statement} \rangle]$

- Syntax Diagram เจํานิรูปแบบ EBNF

Ex:



* ຕົວທຳເປັນ EBNF ກ່ອນ *

Names

Scope

Bindings

เนื้อหาคือการเชื่อมโยง (Binding)

ชื่อของตัวแปร function datatype class ต้องสื่อกำกับกับคุณสมบัติ เว่งกว่า binding binding ฝี 2 ลักษณะ ได้แก่

- static จะเกิดขึ้นก่อนรันโปรแกรม (before run-time) } ms binding เป็นฝีที่ 2 แบบ
- dynamic จะเกิดขึ้นขณะรันโปรแกรม (at run-time)

static scoping ต้องสื่อกำประพฤติของตัวแปร

ชื่อ ตัวแปร และ คุณสมบัติของตัวแปร

ชื่อ (Names) ลักษณะที่ต้องพิจารณาในกรณีที่ต้องเขียนอยู่ใน syntax ของภาษา program คือ

- A. ขอนอนชื่อ C C# Java ไม่จำเป็นต้องมี
- ตัวอักษรคิวเตช C ใช้ underscore ('_')
- C. แต่ต้องระวังตัวอักษรเล็กกับใหญ่ ex. ในภาษา C Hello hello ต้องไม่ใช้ชื่อเดียวกัน
- คำศัพท์ เนื่องจากมีสำหรับภาษาต่างๆ ต้องเป็นคำสงวน (Reserved word) หรือ คำสำคัญ (Keyword) keyword ยกเว้นภาษา C แต่ Reserved word ไม่สามารถใช้ภาษา C ได้

ตัวแปร (Variables) ต้องตั้งให้ลึกหรือกลุ่มตัวแปร ไว้ใน RAM ตัวแปร เป็นการ binding ซึ่งกับ ตำแหน่ง (address) datatype, ค่า (value), อายุการใช้งาน (lifetime)

หน้างานของค.จ. (Address) ตัวแปรเป็นค่าหน้างานของค.จ. ต้องสื่อกำหนดบุคคล (address) ของตัวแปร ฝีลักษณะ

- ตัวแปรชื่อเดียวกัน ฝี ตาม ต่างกันได้ ex. program ของชื่อ sub1, sub2 ฝี ตัวแปรแบบ local ชื่อ sum เนื่องกัน 2 ตัวแปร ไม่เกี่ยงชื่อกัน
- ตัวแปรเดียวกัน ฝี ตาม ที่ต่างกัน sub program ฝี ตัวแปรแบบ local เมื่อมาใช้งาน sub program แต่ต้องร้อง กากะบุคุณ ฉะต่างกัน
- ตัวแปร 2 ตัวเข้าด้วยกัน ฝี หน้างานค.จ.เดียวกันได้ จะเรียกว่า Alias

ค่าของตัวแปร (Value) เป็นค่าที่เก็บอยู่ในหน้างานค.จ.

ex. $x = y + 1;$

↑ ค่าที่เก็บในหน้างานค.จ. เก็บค่า $y+1$

explicit dereferencing คือการใช้ระบุการถือ ตัวแปร ว่า เป็น การอ้างถึง ค่า

ex. ในกลุ่มภาษา C ใช้ * (pointer)

$\text{int } x=1, y=2;$

$\text{int* } p;$

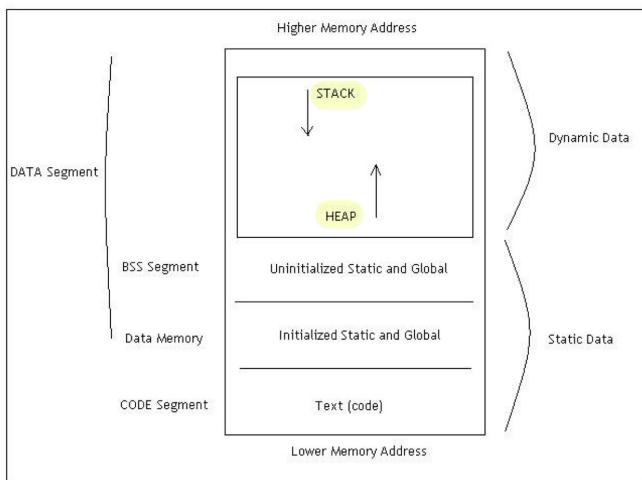
$p = \&x;$

$y = *p;$

ชนิดคือมูล (Type) ชนิดของตัวแปร กำหนดช่วงค่าของตัวแปร (range of value) การเชื่อมโยงชนิดคือมูล (type binding) แบ่งเป็น 2 ประเภท ได้แก่

- static type binding เกิดก่อน runtime ฉะ จดหมาย หน้างานค.จ. เมื่อมาใช้งาน ไม่ต้องแล้ว ฉะอยู่ด้วยกัน

- dynamic type binding ไม่สืการระบุชนิดของล็อกการประมวลผลการประมวลผลต่อไป เช่นไปใช้กับชนิดของมูลเพื่อฝึกการกำหนดค่าในคำสั่งกำหนดค่า สีคณิตเรียกว่าใน การเขียนโปรแกรม ex. เมื่อต้องการเรียงลำดับเรียงชื่อนาม (sort) ใน array ก็จะมีระบุ type ของตัวเป็น dynamic type ได้แก่ JavaScript และ PHP เช่นบ่งชื่อยูนิตแบบ dynamic
- อายุ (Lifetime) ช่วงเวลาของตัวที่ต้องเปรียบเทียบไปกับพ.ก. ในหน่วยค.วิชา ตามๆ นี้เป็นต้น ตัวแรกคือการเรียงลำดับ กับพ.ก. ในหน่วยค.วิชาที่จัดสร้างขึ้น และลิ้นสูตรเชือกเลิกการเรียงลำดับ กับพ.ก. ในหน่วยค.วิชา อุตสาหกรรม 3 ประเภท คือ
- static เก็บพ.ก. ในการเก็บโปรแกรมที่รันอยู่ (code segment) ชื่อยูนิตแบบ read-only
 - heap พ.ก. ในการจัดเก็บที่จัดสร้างขึ้น program สำหรับขอ และคืนพ.ก. เพื่อฝึกคำสั่งนี้ทำต่อๆ กัน algorithm ในการจัดการพ.ก.
 - stack จัดสร้างพ.ก. หน่วยค.วิชา เมื่อฝึกเขียนใช้ sub program เพื่อโปรแกรมป้องกันแล้ว จะคืนพ.ก. เมื่อใดๆ sub program อื่นๆ ต่อไป แต่ระบบจะสร้าง stack frame ส่วนล่างสุดของ stack stack frame ประกอบด้วย พ.ก. สำหรับจัดเก็บ parameter ต้องเปรียบแบบ local และพ.ก. สำหรับจัดเก็บค่าของ register ต่อๆ กัน เป็นต้นต่อๆ กัน



การเรียงลำดับต้องไปรับ กับพ.ก. ในหน่วยค.วิชา เผื่อง่ายๆ 4 แบบ

- static คำสั่ง binding ก่อน execution
- stack - dynamic คำสั่ง binding เมื่อฝึกประมวลผลต่อไป (runtime binding)
- explicit heap - dynamic ฝึกการกำหนดพ.ก. / cell ในหน่วยค.วิชา หลังคืนพ.ก. ในหน่วยค.วิชา (deallocation) โดยใช้คำสั่ง delete
- implicit heap - dynamic ฝึก allocation หรือ deallocation โดยใช้คำสั่งกำหนดค่า

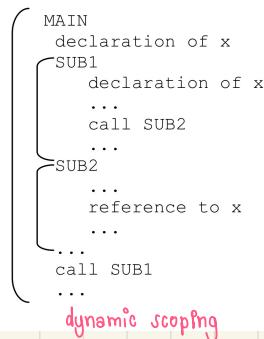
ขอบเขต (Scope) กฎของ การกำหนดขอบเขต (scope rule) เป็น 2 ประเภท

- **static scoping** ทำการ binding ซึ่งกับส่วนของโปรแกรมตามตอนที่ปรากฏใน source program ทำวิเคราะห์ compile ทีละชิ้น

```

1 void sort (float a[ ], int size) {
2     int i,j;
3     for (i=0; i<size; i++)
4         for (j=i; j<size; j++)
5             if (a[j] < a[i] ) {
6                 float t;
7                 t = a[i];
8                 a[i] = a[j];
9                 a[j] = t;
10            }
11 }
```

- **Dynamic scoping** การ binding ซึ่งนี้มีตัวแปร ขึ้นอยู่กับการเรียกใช้ในโปรแกรม ไม่ใช่ในตอนที่ประกค คืนนาสามารถคำนึงถึงการเรียกใช้ต่อจัดล้อมจน run program only!



สภาพแวดล้อมในการอ้างอิง

(referencing environment)

คือ กลุ่มของตัวแปรที่มีค่าสัมภาระของเนื้อหา ถ้าภาษาเป็น static scoping referencing environment จะเป็นตัวแปร local ที่หนึ่ง ต้องรู้ referencing environment ขณะ compile ทำให้สิ่ง code และโครงสร้างข้อมูล เพื่ออ้างอิงตัวแปร ขณะ runtime

dynamic scoping หมายความว่าตัวแปรแบบ local ที่หนึ่ง

รวมกับตัวแปรในโปรแกรมป้องกัน ถ้าสถานะ active ตัวแปรทางตัวสถานะ ไม่ active อยู่ ควบคุมอยู่ใน referencing environment ในกรณีอ้างอิงของเนื้อหา

ค่าคงที่

ตัวแปรภาษาหนึ่งที่มีการเชื่อมโยง กับค่าเพียงแค่ครั้งเดียว ตลอดทั้งโปรแกรม การสิ่งค่าคงที่ เกาะจะไม่มีสัมภาระ เมื่อขึ้นแปลงค่าของมันให้อ้า สัมภาระจะคงตัวแปร พื้นที่กับกำหนดค่าเมื่อต้นไปพร้อมๆ กันได้

DESSERT NOTE

ขอบเขตตัวแปร คือ ส่วนของ

โปรแกรมที่สามารถมองเห็นตัวแปร

ที่กำหนด และเข้าใช้งานได้



Data
type

ຮະບບນີ້ຈົມລ

ຄືວິຈຳກຳນົດທີ່ເກີ່ວຂອງກັນ data type ໃຫ້ 2 ສ່ວນ

- ກລິໄກໃນການກຳນົດ data type ກົບໂຄຮ່ວງຂອ່ມູນ ໂຄງສຽງຂອ່ມູນຄື່ອ ໂຄງລວງທີ່ມີຄຳນໍ້າສຳພາດວ່າງົດວ່າງົດ
ທີ່ມີກໍາ ex. ດົກກໍາທີ່ ຂໍ້ຕ່າຍກໍາທີ່ ຕ້ວແປ ງົດຕີໃນ record parameter ຮອມໄປຄື່ອ ຢຸບພົບ
- ກູ້ເກີ່ວຂອງກຳນົດ ທີ່ໃຊ້ໃນ ກາສຂອງລອບການໃຫ້ຈົນຂອ່ມູນໃນຖຸກຫອງ ເພື່ອໄວ້ໃໝ່ ເກີ່ວຂອ່ມູນຄື່ອພົບຄະດີ
 - ຄ. ເກີ່ວເທີມ ກຳນົດຂອງນີ້ຂອ່ມູນ (type equivalence) ຕරອສອບວ່າ data type ດີສອງຄ່າເໜີລືອນກຳນົດໄໝວ່າ
 - ຄ. ເກີ່ວໄຟ້ຂອງນີ້ຂອ່ມູນ (type compatibility) ຕຽບລອບຄ່າຂອງ data type ເຊັ່ນໃນບັນບັນນີ້ໄດ້ນີ້
 - ກາຮອນຫຼຸກຂອ່ມູນ (type inference) ກຳນົດ data type ໃນນີ້ພຽນ ຜິດຄາດາກບົບກອບຈຳກັນ

ເງື່ອສຳຮຽນແປ່ງກາຍ໌ໄວ້ 2 ປະເທດ ອື່ນ

- ກາຍາທີ່ມີຄື່ອມູນແບບຄົກທີ່ (Statically typed language) ເກີດກາສເຊື່ອມໂບຈະນີ້ສຶ່ນງລົກທີ່ຕ້ວແປໄວ່ໃນ
ຊ່ວງເລາ compile ກາສປະກສຕ້ວແປ ຕາດຮະບຸນີ້ຄື່ອມູນ ໂຄງ ຂໍ້ຕ່າຍເຈົ້າ (explicit declaration) ບໍ່ໄວ້
ໂຄບຮູບ (implicit declaration)
- ກາຍາທີ່ມີຄື່ອມູນແບບ ພລວ່າຕົ້ນ (Dynamically typed language) ເຊັ່ນໂຈງ data type ກົບຕ້ວແປ
ໃນຊ່ວງທີ່ໄປແກຣມທຳກຳ ຕ້ວແປ ເປັນແລ້ນແປ່ນ data type ໄດ້ຕົດອອຄະດາທີ່ທຳກຳກຳນົດ

ກາສຕ້າງສອບນີ້ຄື່ອມູນ (Type checking)

- static type checking ເກີດຈົນໃນຊ່ວງເລາ compile ໂຄງຕຽບສອບ ວ່າຮົດຂອງຕ້ວແປທີ່ກຳນົດໄວ້ຖຸກຕົ້ນເປັນ
ບຽງກົດ 3 ໃໝ່ສຳພາດຄົນອາຄ່າ int ກົບ String ໄວ້ ກາສຕ້າງສອບ data type
ແບບ static ລະ ໃໝ່ສື່ໂອກສເກີ່ວຂອ່ມູນຄື່ອມູນຄະດີທີ່ໄປແກຣມທຳກຳ
- Dynamic type checking ເກີດຈົນໃນຊ່ວງເລາກຳນົດ program ຕ້ວແປຈະໄຟ້ຄາຕຽບຈຸນກ່າຊາງ ມີໂປ່ງແກຣມ
ກາຍາທີ່ມີຮະບບນີ້ຄື່ອມູນແບບເສັງອາດ (strong typed language) ex. ຈົວວັດ, pascal ໄປແກຣມທີ່
ໃໝ່ສື່ອັດພລາດ ລັກາຍາທີ່ມີ int + char ໄວ້ ອື່ອ ກາຍາທີ່ມີຮະບບນີ້ຄື່ອມູນແບບອ່ອນແວ (weak typed language)
ການແປ່ນແປ່ນຂອ່ມູນ (Type conversion) ການແປ່ນແປ່ນຂອ່ມູນແປ່ນເປົ້າໃໝ່ 2 ປະເທດ ອື່ນ
 - ການແປ່ນແປ່ນຂອ່ມູນໂຄບຮູບ (implicit type conversion) ເນັ້ນກ່າວ່າ coercion ເປັນການແປ່ນແປ່ນ data type
ອື່ນນີ້ມີກົດໂຄບ compile ຕາມກູ້ການເປັນແປ່ນຂອ່ມູນ ລະ ເປັນແປ່ນ data type ທີ່ນີ້ສຳຄັນສູງກ່າວ່າ
ex. int + float int ລັກູ້ເປັນແປ່ນ data type ເປັນ float ແລ້ວ + ກ່າວ່າ
 - ການແປ່ນແປ່ນຂອ່ມູນແບບຕ່າແຈ້ງ (explicit type conversion) ເນັ້ນກ່າວ່າ casting ບໍ່ໄວ້ cast ເປັນ
data type ໂຄງຜູ້ເປົ້າ

ex.
 int x = 5;
 int y = 2;
 float z = 2.0;
 float result1 = x/y;
 float result2 = (float)x/y;
 float result3 = x/z;

ค. เท่าเที่ยวกันกับของ data type (Type equivalence)

- ค. เท่าเที่ยวกันโดยโครงสร้าง (structural equivalence) ถ้ามีลักษณะของก่อนหลังเหมือนกัน ex. C, ML
- ค. เท่าเที่ยวกันโดยชื่อ (name equivalence) นิบทั้งสองคำว่ากันในเกติ data type ให้มี ex. Pascal, C#

ค. เท่ากันได้ของ data type

ใช้ data type แทนกันได้ สามารถทำให้ coersion หรือ casting เพื่อกำหนด data type เท่ากันได้

ประเภทของชนิดข้อมูล (Data Type)

แบ่งออกเป็น 2 ประเภท คือ

- ชนิดข้อมูลสเกลาร์ (Scalar Type) มีค่าเดียว ex. ตัวเลข อักษร ค่าตัวราก ชนิดข้อมูลแบบซึ่ง
ex. array string record รวมถึง pointer หมายความคือ address ที่ซึ่งที่เก็บข้อมูล
- ชนิดข้อมูลแบบเชิงประกอบ (Composite Type) ชนิดข้อมูลที่ประกอบกันด้วยข้อมูลสเกลาร์และบุคคลค่า

ชนิดข้อมูลพื้นฐาน (Primitive Data Type)

ผู้ใช้สามารถเลือกใช้ได้ทันที เนื่องจาก built-in data type หรือ predefined data type เป็น data type พื้นฐาน
จะมีคณ. เศรษฐ. คณ. จริง boolean ตัวอักษร

ชนิดข้อมูลเชิงประกอบ (Composite Type)

สร้างจาก type constructor ex. record array set ตามสิ่งที่ต้องการที่มีอยู่แล้วด้านในมี
ex. enum day { Mon, Tue, Wed, Thu, Fri, Sat, Sun };

การแสดงชนิดข้อมูลมากกว่าตัวเดียว รีบดังนี้

- Castesion products ข้อมูลอยู่ในรูปคู่อันดับ ex. record structure

```
ex. struct employee {  
    int id;  
    char name[20];  
}
```

- Disjoint union นำ data type ที่ต้องกันมารวมกัน ผลลัพธ์ที่ได้จะเป็น data type ที่สำคัญ

จะไม่ใช่ชนิดเดียวกัน

```
ex. union myUnion {  
    int i;  
    float r;  
};  
union myUnion u;
```

- Mapping นำ data type ชนิดเดียวกันมารวมกัน โดย ค. สัมพันธ์แบบฟังก์ชัน ผลลัพธ์ที่ได้จะเป็นชนิด
ข้อมูลที่มีค่าของข้อมูลหาก set หนึ่ง ไปถูก set หนึ่ง

- Powersets ดำเนินการกับข้อมูลเพื่อ operation ex. different, subset, union, intersection

ទីផ្សារសំណងជាន់វាំង

Pascal และ Fortran និង integer បានជួយថា real សំណងភាគា C និង int, float

ចំនួល (Numeric type)

ក្នុងភាគាជាន់ដែលត្រូវការគ្រប់គ្រង (integer) និងគ្រប់គ្រងតម្លៃ (Real ឬ floating point) ដែលទទួលបានប្រើបាយ

ex. ចំនួលចិនតរាង (Complex number) និងលេខតម្លៃ (Decimal)

ឈឺតិចខំណួល	C	C#	Java
Integer	short, int, long	sbyte, byte, short, ushort, int, uint, long, ulong	byte, short, int, long
Real	float, double, long	float, double, decimal	float, double
Character	char	char	Char
String		string	String
Boonlean		bool	Boolean

តារាង 4.2 ឈឺតិចខំណួលលេខចាំនាមុនពីរនៃភាគាជាន់

ឈឺតិចខំណួល	ដំណឹងពីការតែងតាំង	ចំនួលដែលបានប្រើបាយ
char	1 byte	-128 ដល់ 127 or 0 ដល់ 255
unsigned char	1 byte	0 ដល់ 255
signed char	1 byte	-128 ដល់ 127
int	2 or 4 bytes	-32,768 ដល់ 32,767 or -2,147,483,648 ដល់ 2,147,483,647
unsigned int	2 or 4 bytes	0 ដល់ 65,535 or 0 ដល់ 4,294,967,295
Short	2 bytes	-32,768 ដល់ 32,767
unsigned short	2 bytes	0 ដល់ 65,535
long	4 bytes	-2,147,483,648 ដល់ 2,147,483,647
unsigned long	4 bytes	0 ដល់ 4,294,967,295

Single precision (32 bits)

s	e = exponent	m = mantissa
---	--------------	--------------

1 bit

8 bits

23 bits

Double precision (64 bits)

s	e = exponent	m = mantissa
---	--------------	--------------

1 bit

11 bits

52 bits

ຕົວອໍານຸຍະ (Charactor)

ເກີ່ນຮັບສ້າງໄລຍ້ໃຫ້ແກນຕົວອໍານຸຍະຕ່າງໆ ໃຫ້ການເຈົ້າໃໝ່ແບບ ASCII ພກ. ອັດເກີ່ນ 1 byte
ex. Java ນໍາໂອ C# ໃຫ້ 2 byte ອັດເກີ່ນໃນຢູ່ Unicode

ຕະຮະກະ (Boolean)

ມີ 2 ຄ່າ ໂດຍເກີ່ນ true , false ໃຫ້ເຫຼືອຄໍາລິ້ນໃຈ ນໍາໂອໃຫ້ແກນຄໍາທີ່ເປັນໄປໄຫ້ເປັນ 2 ຄ່າທີ່ກ່າວນີ້

ຮົດຈອນຂະໜາດ

data type ທີ່ຜູ້ເຊັນໂປຣແກຣມສາມາດກຳນົດໃໝ່ໃນນີ້ໄດ້ ສິນຕະລາງ (structure data type) ປະກອບດ່ວງ data type
 ສິນຫຼາມແລະ ບົຄຮສໄໝຂອ່ມູນ

ຂົດສົ່ວນຸມຸລແດນັ່ນ (Enumeration)

ຜູ້ເຊັນໂປຣແກຣມສາມາດກຳນົດ set ຈອອົບ

ex. enum day {sun, mon, tue, wed, thu, fri, sat};

ຮົດຈອນຸມຸລຂອງຄ່າ (Subrange)

ກຳນົດຂ່ອງຄ່າທີ່ເປັນໄປໄດ້ເປັນ subset ຕ່ອງປາກ Pascal

ex. Type test_score = 0..100;

weekday = mon..fri;

test_score ເປັນ integer ໃຫ້ພກ. ແລ້ວ 1 byte

Array

ຈົດເກີ່ນຂອນຸມຸລນິດເຕີບອົບນີ້ ແຕ່ຟີ່ array ທີ່ເກີ່ນ data type ນັບຮັບໃດໆ ເຊັກຕ່າ Heterogeneous array

- ໄຫວ່າກຮົດໄໝແລ້ວຮົດຈອນຸມຸລກໍໃຫ້ເປັນເລີລີຄົນ

- ການກຳນົດຄ່າເນື່ອຕົ້ນ ແລະການດຳເນີນການ

ex. int list[] = {1, 2, 3, 4}

- ປະເທດຂອງ Array array ສ່ວນໃນຢູ່ຈະເກີດໃໝ່ແບບ static ກາຣີ່ຈ່ອປະເກາຕອນພາກ ແລະ ເວລາໃນການ
 ຈັດສ້ອງ ແບ່ງໄດ້ 5 ປະເທດ