Data Structures

โครงสร้างข้อมูล

Introduction

- คอมพิวเตอร์ มีหน้าที่
  - รับข้อมูล (Input)
  - ประมวลผล (Process)
  - แสดงผลลัพธ์ (Output)

- ทุกอย่างจะเกี่ยวข้องกับ “ข้อมูล (Data)”
- ข้อมูล ที่ถูกประมวลผลโดยคอมพิวเตอร์ ต้องมีการจัดวางในรูปแบบโครงสร้างที่เหมาะสม

ความหมายของโครงสร้างข้อมูล

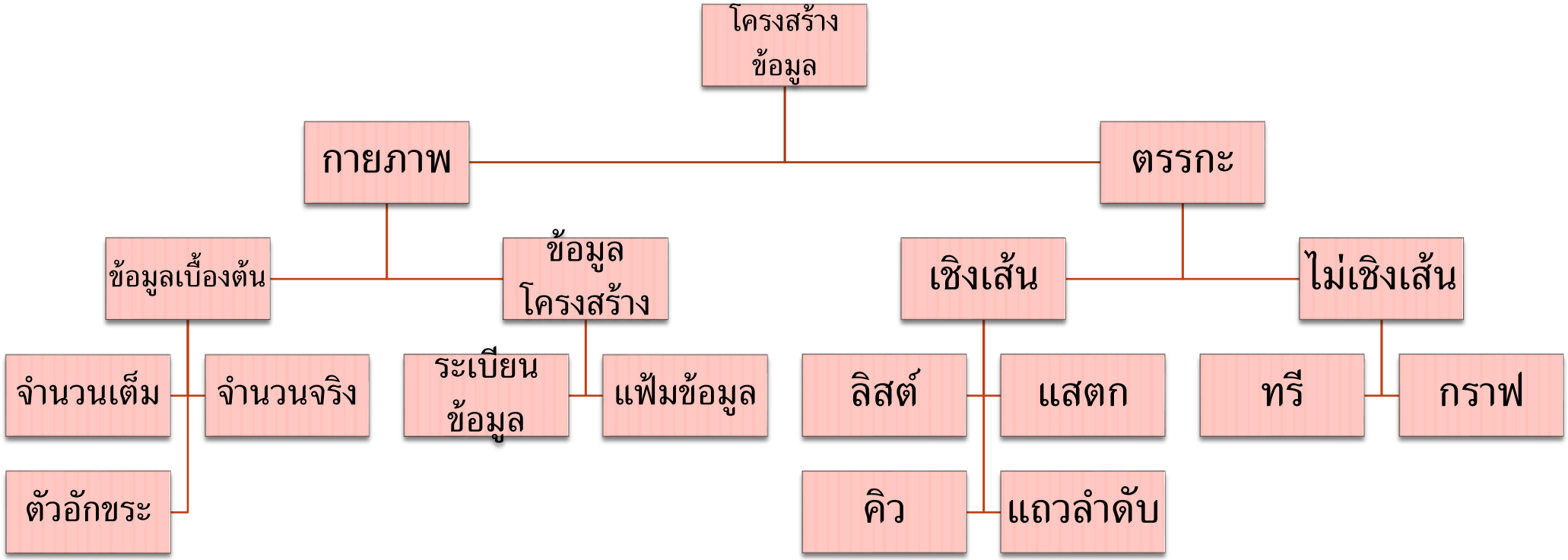
- ข้อมูล (Data) คือ ข้อเท็จจริงต่าง ๆ ซึ่งอาจเป็นตัวเลขหรือไม่ก็ได้
- โครงสร้าง (Structure) คือ ความสัมพันธ์ของสมาชิกในกลุ่ม

- โครงสร้างข้อมูล (Data Structures) คือ
  - ความสัมพันธ์ระหว่างข้อมูลที่อยู่ในโครงสร้างนั้น ๆ
  - รวมทั้งกระบวนการในการจัดการข้อมูลในโครงสร้าง
  - เพื่อให้สามารถใช้งานข้อมูลได้อย่างมีประสิทธิภาพ

ความหมายของโครงสร้างข้อมูล (ต่อ)

- สามารถสร้างโครงสร้างข้อมูลใหม่
  - ประยุกต์จากโครงสร้างข้อมูลเดิมที่มีอยู่แล้ว
- เพื่อใช้ร่วมกันกับโปรแกรมที่เราออกแบบ
- ภาษาคอมพิวเตอร์ปัจจุบันรองรับโครงสร้างข้อมูลหลากหลาย

ประเภทของโครงสร้างข้อมูล



โครงสร้างข้อมูลทางกายภาพ

ใช้โดยทั่วไปในภาษาคอมพิวเตอร์

1. ข้อมูลเบื้องต้น (Primitive Data Types)
   - จำนวนเต็ม (Integer)
   - จำนวนจริง (Real)
   - ตัวอักขระ (Character)
2. ข้อมูลโครงสร้าง (Structured Data Types)
   - ระเบียนข้อมูล (Record)
   - แฟ้มข้อมูล (File)

ข้อมูลเบื้องต้น (Primitive Data Types)

- ข้อมูลจำนวนเต็ม (Integer)

ชนิดข้อมูล	ขนาด	ช่วงของข้อมูลที่เก็บไว้
Byte	8 bits	-128 to 127 (i.e., -2^7 to $2^7 - 1$)
Short	16 bits	-32768 to 32767 (i.e., -2^{15} to $2^{15} - 1$)
Int	32 bits	-2147483648 to 2147483647 (i.e., -2^{31} to $2^{31} - 1$)

ตัวเลข 1,023,004 ควรใช้ตัวแปรประเภทใด

ข้อมูลเบื้องต้น (Primitive Data Types)

- ข้อมูลจำนวนจริง (Real)

ชนิด	ขนาด	ตัวอย่าง	ช่วงของข้อมูลที่เก็บไว้
long	64 bits	11.25, -1.5E4, 2.5e-02	9223372036854775808 to 9223372036854775807 (i.e., -2^{63} to $2^{63} - 1$)
float	32 bits	11.25, -1.5E4, 2.5e-02	3.4E-38...3.4E+38
double	64 bits	11.25, -1.5E4, 2.5e-02	1.7E-308...1.7E+308

ข้อมูลเบื้องต้น (Primitive Data Types)

ตัวอย่าง

- 437.875 สามารถเขียนได้เป็น $0.437875x10^3$
-
- โดย .437875 คือ mantissa และ 3 คือ exponent.

- หรือสามารถเขียนเป็น 0.437875E+3 หรือ +0.437875E3

ข้อมูลเบื้องต้น (Primitive Data Types)

- ตัวอักขระ
- เป็นการนำตัวอักขระมาเรียงต่อกันเป็นข้อความตั้งแต่หนึ่งตัวเป็นต้นไป สามารถเก็บตัวอักขระได้ 255 ตัว โดยตัวอักขระจะต้องอยู่ในเครื่องหมาย “ ”
- ภาษาซี มีการเติมตัวอักษรว่าง Null (\0) เป็นตัวสุดท้ายของสตริง

โครงสร้างข้อมูลทางตรรกะ

เกิดจากจินตนาการของผู้ใช้เพื่อแก้ปัญหาในโปรแกรมที่สร้างขึ้น

1. **โครงสร้างข้อมูลเชิงเส้น** (Linear Lists)
   - ข้อมูลมีความสัมพันธ์เรียงต่อเนื่องกัน
   - Ex. Array, Linked List, Stack, Queue

2. **โครงสร้างข้อมูลไม่เชิงเส้น** (Non-Linear Lists)
   - ข้อมูลแต่ละตัวมีความสัมพันธ์แบบไม่เรียงต่อเนื่อง
   - Ex. Tree, Graph

การเลือกใช้โครงสร้างข้อมูล

1. โครงสร้างข้อมูลนั้นสามารถสร้างความสัมพันธ์ให้กับข้อมูลชุดนั้น ได้อย่างสมบูรณ์ที่สุด
2. โครงสร้างนั้นต้องง่ายต่อการดำเนินการในระบบงาน

การแทนที่ข้อมูลในหน่วยความจำหลัก

- การแทนที่ข้อมูลแบบ สแตติก (Static Memory Representation)
- การแทนที่ข้อมูลแบบไดนามิก (Dynamic Memory Representation)

การแทนที่ข้อมูลในหน่วยความจำหลัก (ต่อ)

การแทนที่ข้อมูลแบบสแตติก

- เป็นการแทนที่ข้อมูลที่จองเนื้อที่แบบคงที่แน่นอน
- ต้องมีการกำหนดขนาดก่อนใช้งาน

ข้อดี : ง่ายต่อการใช้งาน

ข้อเสีย : ไม่สามารถปรับขนาดให้เพิ่มขึ้นหรือลดลงได้

ตัวอย่าง : แถวลำดับ (Array)

การแทนที่ข้อมูลในหน่วยความจำหลัก (ต่อ)

การแทนที่ข้อมูลแบบไดนามิก

- เป็นการแทนที่ข้อมูลแบบไม่จองเนื้อที่ ยืดหยุ่นได้ตามความต้องการของผู้ใช้
- หน่วยความจำที่ไม่ใช้สามารถส่งคืนเพื่อนำกลับมาใช้ใหม่ได้อีก

ข้อดี : ประหยัดหน่วยความจำ
ยืดหยุ่นต่อการใช้งาน

ข้อเสีย : ยากในการจัดการ

- ตัวอย่าง : ตัวชี้ Linked List

ข้อแตกต่างระหว่าง Static Memory และ Dynamic Memory Allocation

S.No	Static Memory Allocation	Dynamic Memory Allocation
1	In the static memory allocation, variables get allocated permanently, till the program executes or function call finishes.	In the Dynamic memory allocation, variables get allocated only if your program unit gets active.
2	Static Memory Allocation is done before program execution.	Dynamic Memory Allocation is done during program execution.
3	It uses stack for managing the static allocation of memory	It uses heap for managing the dynamic allocation of memory
4	It is less efficient	It is more efficient
5	In Static Memory Allocation, there is no memory re-usability	In Dynamic Memory Allocation, there is memory re-usability and memory can be freed when not required

ข้อแตกต่างระหว่าง Static Memory และ Dynamic Memory Allocation

S.No	Static Memory Allocation	Dynamic Memory Allocation
6	In static memory allocation, once the memory is allocated, the memory size can not change.	In dynamic memory allocation, when memory is allocated the memory size can be changed.
7	In this memory allocation scheme, we cannot reuse the unused memory.	This allows reusing the memory. The user can allocate more memory when required. Also, the user can release the memory when the user needs it.
8	In this memory allocation scheme, execution is faster than dynamic memory allocation.	In this memory allocation scheme, execution is slower than static memory allocation.
9	In this memory is allocated at compile time.	In this memory is allocated at run time.
10	In this allocated memory remains from start to end of the program.	In this allocated memory can be released at any time during the program.

ADT

- Abstract Data Type : ชนิดข้อมูลนามธรรม

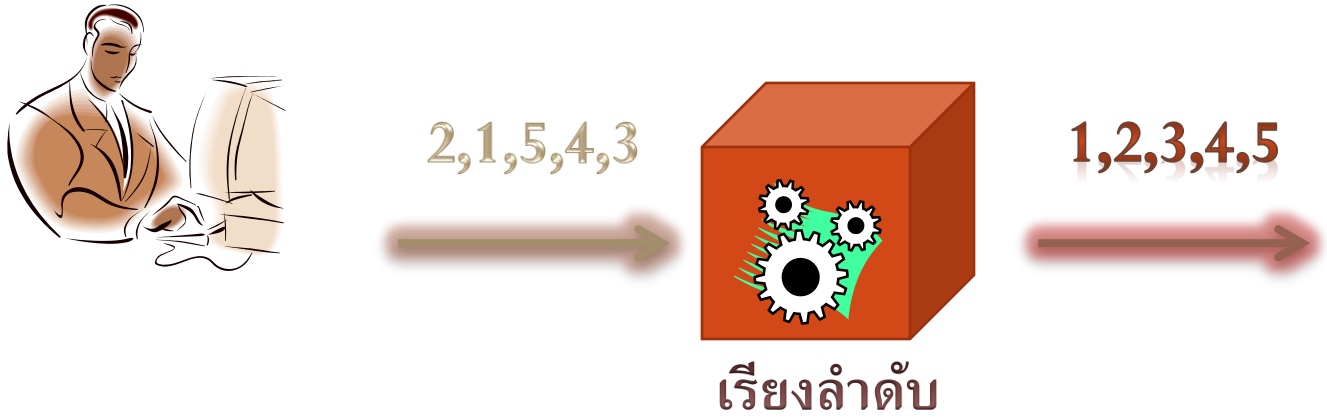
ประกอบด้วย

  - ลักษณะโครงสร้างของข้อมูล
  - ตัวดำเนินการ (operation) ที่สามารถทำงานกับมันได้

- เป็นการประยุกต์ใช้โครงสร้างข้อมูลให้เกิดประโยชน์
- ไม่ผูกติดกับ Hardware หรือ Software

ADT

- ผู้ใช้งานไม่จำเป็นต้องรู้ว่า ADT ดำเนินงานอย่างไร แค่รู้เพียงว่า
  - ทำงานอะไรได้บ้าง
  - ได้ผลลัพธ์อะไรบ้าง



Example ADT

- ADT: AlphaString
- ลักษณะโครงสร้าง
  - สายอักขระ (String) ที่เกิดจากการนำอักขระมาเรียงต่อกัน
- ตัวดำเนินการ
  - เปรียบเทียบ String
  - หาความยาว String
  - ต่อ String
  - แบ่ง String
  - หาตำแหน่งตัวอักษรใน String

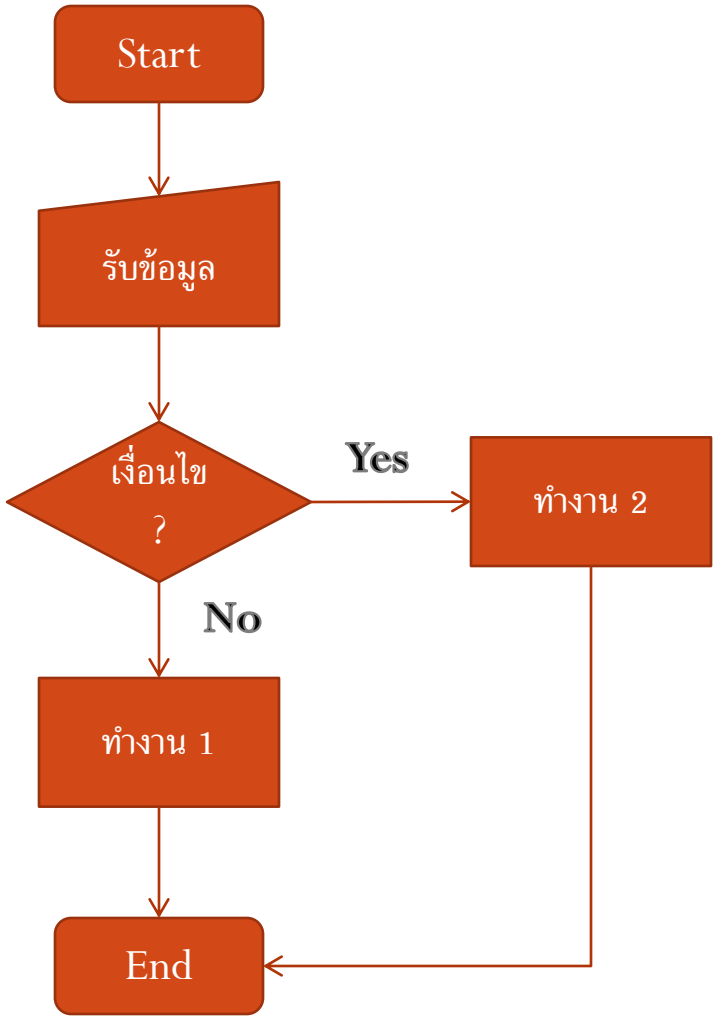
ขั้นตอนวิธี Algorithm

- Algorithm : กลุ่มของคำสั่ง หรือ กฎ ที่สามารถระบุลำดับวิธีการแก้ปัญหาอย่างแม่นยำ
- Original from Muḥammad ibn Mūsā al-Khwārizmī

ทำอย่างไรจึงจะสามารถอธิบายวิธีการแก้ปัญหาได้ ?

- ร้อยแก้ว : Prose
- แผนผัง : Flowchart
- รหัสเทียม : Pseudo Code

Flowchart



Pseudo code

- Mixture of natural language & high-level programming language
- อธิบายแนวทางหลักที่ใช้ในการแก้ปัญหา
- ตัดรายละเอียดที่ไม่จำเป็นออก
- ไม่มีมาตรฐาน

ตัวอย่าง การเขียน Pseudo

- Algorithm 1 ค้นหาตัวเลขที่มีค่ามากที่สุด

Procedure max($a_1, a_2, \ldots, a_n$: integers)

$max := a_1$

for $i := 2$ **to** n

  if $max < a_i$ **then** $max := a_i$

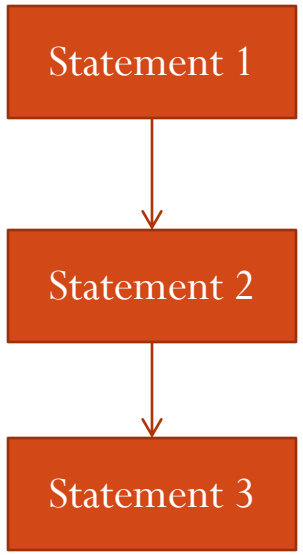
{max คือจำนวนที่มีค่ามากที่สุด}

โครงสร้างการควบคุมพื้นฐาน
(Basic control structure)

- แบบเรียงลำดับ (Sequence)
- แบบเลือกการทำงาน (Selection)
- แบบทำงานซ้ำ (Repetition)

แบบเรียงลำดับ (Sequence)

- ควบคุมแบบเรียงลำดับ
- ตามขั้นตอน
- ต่อเนื่องจากบนลงล่าง



- Ex.

  Statement1

  Statement 2

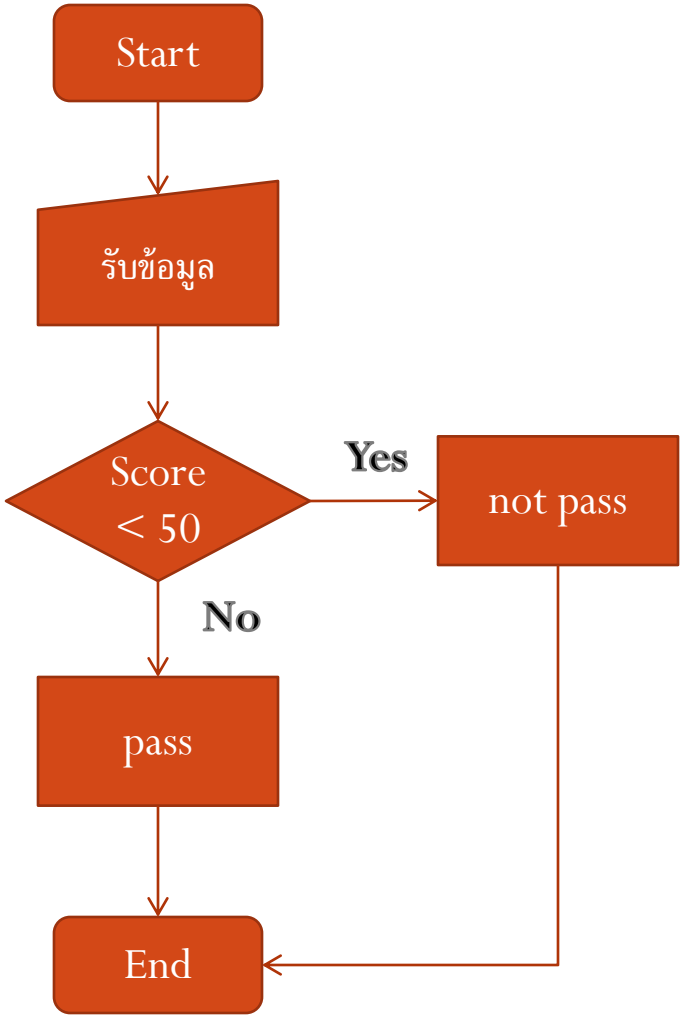
  Statement 3

  :

  :

แบบเลือกการทำงาน (Selection)

- เปรียบเทียบเงื่อนไข
- ถ้าเงื่อนไขเป็นจริงทำกลุ่มงานหนึ่ง
- ถ้าเงื่อนไขเป็นเท็จทำอีกกลุ่มงานหนึ่ง



ตัวอย่าง คำสั่ง if

Ex 1.1 คำสั่ง if อย่างง่าย

```
if score < 50 then
    grade = "not pass"
else
    grade = "pass"
end if
```

Ex 1.2 คำสั่ง if ไม่มีเงื่อนไขเท็จ

```
if score > 50 then
    grade = "pass"
end if
```

ตัวอย่าง คำสั่ง if

Ex 1.3 คำสั่ง if ใช้ operator and/or/not เข้าร่วม

```
If (graduateYear = 3 and gpax < 2.00)  then
   result = "retired"
end if
```

```
If (wrongUser = true or wrongPassword = true)  then
   result = "login failed"
end if
```

```
If not (wrongUser = true and wrongPassword = true)  then
   result = "login system"
end if
```

ตัวอย่าง คำสั่ง if

Ex 1.4 คำสั่ง if .. elseif

```
If (score >= 80)  then
    grade = "A"
else if (score >= 70) then
    grade = "B"
else if (score >= 60) then
    grade = "C"
else if (score >= 50) then
    grade = "D"
else
    grade = "F"
end if
```

แบบทำงานซ้ำ (Repetition)

- กลุ่มคำสั่งอยู่ในภายในลูป (Loop)
- ทำงานซ้ำไปเรื่อย ๆ เมื่อตรงกับเงื่อนไข
- เมื่อเงื่อนไขเป็นเท็จจึงหลุดออกจากลูป

ตัวอย่าง คำสั่ง while

```
while i < 10
    total = total + ai
    i = i + 1
end
```

C++

```
while (i < 10)
{
    count << “position” << i;
     i = i +1
}
```

ตัวอย่าง คำสั่ง for

```
for i = 1 to 10 do
   do statement…
end
```

C++

```
For (i=0 ;  i > 10 ;  i++)
{
    count << “count ” << i;
}
```

End Introduction