สัปดาห์ที่ 3 Control Flow

เรียบเรียงโดย ชาคริต วัชโรภาส

วิชา 01418113 Computer Programming

0.1 Programming Daradigms

ภายใต้ Imperative Programming Paradigms:

- Statements execute in a statically declared sequence.
  - Procedural - Simple flow-control, typically using goto's
  - Structured - Instead of goto's, uses for/loop/do/while/etc. (Top-Down)
  - Object Oriented - Encapsulated polymorphic behavior via inheritance
  - Event-Driven - Code in handlers which get triggered by events
- ภาษา C ใช้ Structured Programming Paradigm
- *อ้างอิงจาก* [What is the difference between structured and procedural languages? Is C a structured programming or procedural language? (https://www.quora.com/What-is-the-difference-between-structured-and-procedural-languages-Is-C-a-structured-programming-or-procedural-language)]

0.2 ลองพิจารณาเรื่องเหล่านี้

- ในคอมพิวเตอร์ ถ้า a, b, และ c เป็นจำนวนจริง (float) แล้ว (a + b) + c จะมีค่าเท่ากับ a + (b + c) เสมอไปหรือไม่ (associative property)
- เขียนโค้ดที่แปลงอักขระในสตริงจากอักษรตัวใหญ่ให้เป็นตัวอักษรตัวเล็ก
- ทำความเข้าใจโค้ดนี้

```
char str[20];
int i, str_size = 20;
for (i=0; i < str_size-1 && (c=getchar()) != '\n' && c != EOF; i++)
   str[i] = c;
str[i] = 0;
```

---

1. Statements and Blocks

- expression กลายเป็น statement เมื่อถูกปิดท้ายด้วยเครื่องหมาย semi-colon ;
- x = 0 กลายเป็น x = 0;
- เครื่องหมายปีกกา {} ใช้รวมส่วนของ declarations และ statements เข้าด้วยกันเป็น block
- ฟังก์ชัน, if, else, while, for มักใช้ block ในการรวม statement หลายอันเข้าด้วยกัน
- Null statement คือ statement ที่ปราศจากส่วนของ expression โดยเหลือเพียงเครื่องหมาย semi-colon ; เท่านั้น

---

2. If-Else

```
if (expression)
   statement1
else
   statement2
```

- เรามักจะเขียนในรูปนี้ if (expression) แทนที่จะเขียน if (expression != 0)
- ต.ย. (else นี้จะคู่กับ if ที่อยู่ใกล้ที่สุด)

```
if (n > 0)
if (a > b)
   z = a;
else
   z = b;
```

---

3. Else-If

```
if (expression1)
   statement1
else if (expression2)
   statement2
else if (expression3)
   statement3
else if (expression4)
   statement4
else
   statement5
```

- ถ้า expression ตัวไหนมีค่าเป็นจริง statement ที่เกี่ยวข้องก็จะถูกทำงาน

---

4. Switch

- switch ใช้ในการตรวจสอบเงื่อนไขในลักษณะที่ดูว่าค่าของ expression นั้น match เข้ากับค่าคงที่ (ที่เป็นตัวเลขจำนวนเต็ม) ตัวใด หลังจากนั้นโปรแกรมจะกระโดดมาทำงาน statement ภายใต้ค่าคงที่นั้นลงมาเรื่อยๆ จนกว่าจะเจอ break แล้วจึงหลุดออกจาก switch

```
switch (expression) {
   case const-expr:
      statements
   case const-expr:
      statements
   default:
      statements
}
```

- default เป็น optional คือจะมีหรือไม่มีก็ได้ แต่ถ้ามีแล้ว คำสั่ง statements ที่อยู่ภายใต้ default จะถูกทำงานเมื่อค่าของ expression ไม่ match กับ case ใดเลย
- ต.ย.

```
int n_digits = 0;
int n_non_digits = 0;
int ch;

while ((ch=getchar()) != EOF) {
   switch (ch) {
      case '0': case '1': case '2': case '3':
      case '4': case '5': case '6': case '7':
      case '8': case '9':
         n_digits++;
         break;
      default:
         n_non_digits++;
         break;
   }
}
printf("จำนวนตัวเลข: %d, ไม่ใช่ตัวเลข: %d\n", n_digits, n_non_digits);
```

5. Loops - While และ For

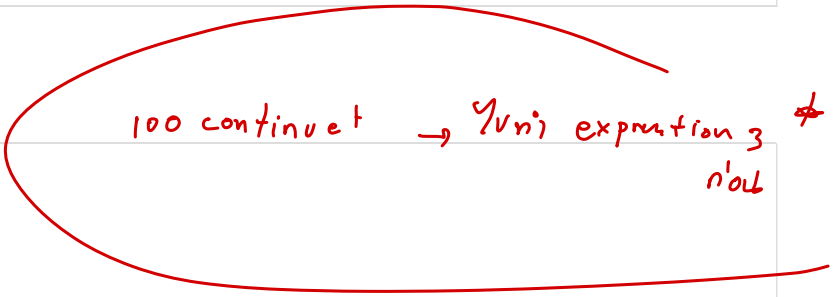
- รูปแบบของ while

```
while (expression)
   statement
```

- ถ้า expression เป็นจริง statement จะถูกทำงาน และถูกวนทำงานจนกว่า expression จะเป็นเท็จ
- รูปแบบของ for

```
for (expr1; expr2; expr3)
   statement
```

มีลักษณะเดียวกับ




```
expr1;
while (expr2) {
   statement
   expr3;
}
```

- โดยทั่วไป expr1 และ expr3 จะเป็นการกำหนดค่าหรือการเรียกใช้ฟังก์ชัน ส่วน expr2 มักจะเป็นการเปรียบเทียบค่า
- แต่ละส่วน expr1, expr2, และ expr3 สามารถถูกละได้ แต่ยังต้องคงเครื่องหมาย ; ไว้

```
for (;;) {
   printf("hello\n");
}
```

ลูปข้างบนมีลักษณะเป็น "infinite" loop โดยจะพิมพ์คำว่า hello อย่างไม่สิ้นสุด

ลองทำโจทย์ "นับจำนวนอักขระในอาเรย์ของอักขระ"

ลองทำโจทย์ "ลบช่องว่างในอาเรย์ของตัวอักขระ"

6. Loops - Do-While

- do-while จะแตกต่างจาก while และ for ในแง่ที่การทดสอบเงื่อนไขจะอยู่ด้านล่างของลูป

```
do
   statement
while (expression);
```

ซึ่งส่วนใหญ่เรามักจะเห็นในรูปนี้

```
do {
   statement
} while (expression);
```

- statement จะถูกทำงาน แล้ว expression จึงถูกหาค่าว่าเป็นจริงหรือเท็จ ซึ่งถ้ามีค่าเป็นจริง statement จะถูกทำงานอีก แล้ว expression จึงถูกหาค่าอีก ซึ่งลูปนี้จะวนทำซ้ำไปเรื่อยๆ จนกว่า expression จะเป็น

เท็จ

---

7. Break และ Continue

- break และ continue จะช่วยโปรแกรมเมอร์ให้สามารถควบคุมการวนซ้ำได้มากขึ้น
- break จะทำให้โปรแกรมหลุดออกมาจากลูป for, while และ do-while โดยทันที ทั้งนี้การหลุดจากลูปจะทำจากลูปวงในสุดที่ break อยู่

> ลองทำโจทย์ "ลบอักขระ whitespace ออกจากด้านหลังของอาเรย์"

- continue จะบังคับการทำงานของลูปให้เกิดการวนซ้ำขึ้นมาใหม่ โดยที่ ในกรณีของ while กับ do-while การทำงานจะกระโดดไปที่การตรวจสอบเงื่อนไข (expression) ส่วนในกรณีของ for นั้น การทำงานจะกระโดดไปที่ expr3 แล้วถึงไปตรวจสอบเงื่อนไข (expr2)

> ลองทำโจทย์ "นับเฉพาะจำนวนเต็มบวก"

---

8. Goto และ Label

- ไม่อยากสอนให้รู้ แต่ถ้าอยากรู้ ก็ถามมาได้
- "Go To Statement Considered Harmful" (https://en.wikipedia.org/wiki/Goto#Criticism)
- ต.ย. spaghetti code

```
#include <stdio.h>

int main()
{
   int routine = 1;
   goto subroutine;

return_label_1:
   printf("routine 1\n");
   routine++;
   goto subroutine;

return_label_2:
   printf("routine 2\n");
   routine++;
   goto subroutine;

return_label_3:
   printf("routine 3\n");
   routine++;

subroutine:
   printf("subroutine reached\n");
   if (routine == 1) goto return_label_1;
   if (routine == 2) goto return_label_2;
   if (routine == 3) goto return_label_3;
}
```