หลักภาษาโปรแกรม

Principles of Programming Languages

กฤดาภัทร สีหารี

หลักภาษาโปรแกรม

Principles of Programming Languages

กฤดาภัทร สีหารี

คำนำ

การเรียนหลักการจะต้องเรียนรู้เรื่องพื้นฐานต่าง ๆ ที่ถูกใช้ในการออกแบบภาษาโปรแกรม เช่น ตัวแปร การจัดการหน่วยเก็บข้อมูลกับตัวแปร ขอบเขต เวลาชีวิต โปรแกรมย่อย รูปแบบการเขียนโปรแกรมหรือที่เรียกว่ากระบวนทัศน์ หลักการผูกมัด ไวยากรณ์ และหลักการต่าง ๆ ที่ภาษาโปรแกรมเลือกใช้ในการออกแบบภาษาโดยเฉพาะ เช่น หลักการสำหรับการเขียนโปรแกรมเชิงวัตถุ รวมไปถึงปัญหาต่าง ๆ ที่มักเกิดขึ้นในการเขียนโปรแกรม การเรียนเขียนโปรแกรมเพียงภาษาเดียวถูกมองว่ายากแล้ว แต่เมื่อต้องเรียนรู้หลักการที่มีในภาษาต่าง ๆ นั้น ก็อาจจะสร้างความเข้าใจได้ยากขึ้น อย่างไรก็ตาม เมื่อได้รับทราบหลักการ ผู้เรียนจะมองเห็นหลักการที่มีความคล้ายคลึงกันในแต่ละภาษา แล้วก็จะสามารถเรียนรู้ภาษาใหม่ ๆ ได้อย่างรวดเร็ว การเรียนรู้หลักการของภาษาโปรแกรมจะสร้างความรู้ และความเข้าใจในการออกแบบของภาษาต่าง ๆ ที่ถูกนำมาใช้งาน เมื่อมีความเข้าใจแล้วก็จะทำให้เขียนโปรแกรมได้อย่างมีประสิทธิภาพ เช่น มีข้อผิดพลาดน้อยลง มีการประมวลผลที่รวดเร็ว สามารถรับมือกับข้อผิดพลาดที่อาจเกิดขึ้นในการรันโปรแกรมได้ มีการเขียนโปรแกรมที่ดี และง่ายต่อการแก้ไขบำรุงรักษา

ตำราฉบับนี้เรียบเรียง และเขียนขึ้นโดยใช้ตัวอย่างโปรแกรมจากหลาย ๆ ภาษา เพื่อนำมาศึกษาหลักการโดยคำนึงถึงการเปรียบเทียบคุณลักษณะ และหลักการต่าง ๆ ในภาษาโปรแกรม ผู้เขียนหวังเป็นอย่างยิ่งว่าการเรียนรู้หลักการจะทำให้ผู้เรียนเกิดความเข้าใจในการเขียนโปรแกรม มีความสนุกมากขึ้นในการเขียนโปรแกรม และพยายามฝึกฝนทักษะอย่างต่อเนื่องตลอดไป

รศ. ดร. กฤดาภัทร สีหารี

31 ตุลาคม 2565

สารบัญ

ฎ

บทที่ 1
แนะนำสู่รายวิชา

1.1 หลักการ

การเขียนโปรแกรมจะต้องเริ่มจากการเรียนรู้หลักการ เพื่อสร้างความเข้าใจในภาษาที่ใช้สำหรับการเขียนโปรแกรม แล้วฝึกฝนการเขียนโปรแกรมโดยนำปัญหาต่าง ๆ มาเป็นโจทย์ในการแก้ปัญหา เมื่อฝึกฝนบ่อย ๆ ก็จะมีทักษะ เมื่อมีทักษะมากขึ้นก็มีความเชี่ยวชาญ แต่จะเรียนรู้ได้อย่างรวดเร็วได้นั้นจะต้องรู้เรื่องของหลักการในการเขียนโปรแกรมอย่างเข้าใจ

หลักการ (Principle) คือแนวคิดพื้นฐานในการศึกษาเรื่องหนึ่ง ๆ เพื่อนำไปพัฒนา หรือสร้างสิ่งต่าง ๆ ต่อไป

หลักการภาษาโปรแกรม (Principles of Programming Languages) หรือหลักภาษาโปรแกรม คือแนวคิดพื้นฐานในการศึกษาเกี่ยวกับภาษาโปรแกรม หลักการอาจเป็นแนวคิดหรือเป็นกฎที่ถูกนำไปใช้ในการเขียนโปรแกรม ยกตัวอย่างเช่น จะเขียนโปรแกรมภาษาจาวา (Java) อาจจะต้องศึกษาหลักการที่เกี่ยวข้องกับเรื่องต่าง ๆ ดังต่อไปนี้

(1) ตัวแปร (Variable)
(2) ชนิดข้อมูล (Data type)
(3) คำสั่งควบคุม (Control construct)
(4) นิพจน์ (Expression)
(5) หลักการในการโปรแกรมเชิงวัตถุ (Object-Oriented Programming) ซึ่งต้องรู้หลักการในการสร้างคลาส (Class) และเมท็อด (Method) รวมถึงหลักการในการถ่ายทอด (Inheritance) และหลักการความหลากหลายของรูป (Polymorphism)

หากทราบหลักการของภาษาโปรแกรมแล้วก็จะทราบวิธีการเขียนโปรแกรมที่ดีได้อย่างรวดเร็ว สามารถลดข้อผิดพลาดจากการเขียนคำสั่ง และมีตรรกะในการแก้ปัญหาที่ดีได้ แต่ละภาษาโปรแกรมมักจะมีหลักการอ้างอิงมาจากแนวคิดพื้นฐานเดียวกัน แต่อาจมีความแตกต่างในการเลือกใช้หลักการใน

บางส่วนเพื่อสร้างจุดเด่นให้กับภาษา ดังนั้นหากสามารถเขียนโปรแกรมภาษาหนึ่ง ๆ ได้ จะสังเกตได้ว่าในการเรียนรู้ภาษาถัดไปก็มักจะทำได้รวดเร็วขึ้น

1.2 ประโยชน์ในการเรียนหลักการของภาษาโปรแกรม

สิ่งที่จะได้เมื่อทราบหลักการของภาษาโปรแกรม อาทิ

- ทำให้ผู้เรียนสามารถแสดงแนวคิดในการแก้ปัญหาได้อย่างหลากหลายวิธี และสามารถนำความรู้นั้นมาประยุกต์ในการแก้ไขปัญหาได้
- ทำให้ผู้เรียนสามารถเลือกใช้ภาษาสำหรับการเขียนโปรแกรมที่เหมาะสมในการแก้ไขปัญหาหรือพัฒนาระบบงานได้
- ทำให้ผู้เรียนเรียนรู้ภาษาสำหรับการเขียนโปรแกรมใหม่ ๆ ได้ง่ายขึ้น สามารถเรียนรู้ด้วยความเข้าใจโดยใช้เวลาที่สั้นลง
- ทำให้ผู้เรียนมีความเข้าใจที่ดีขึ้นในการเขียนโปรแกรม เช่น ทราบลำดับ และวิธีในการประมวลผลคำสั่งในโปรแกรมที่ถูกต้อง
- ทำให้ผู้เรียนสามารถหาจุดบกพร่องหรือบัก (bug) ของโปรแกรมได้เร็วขึ้น และสามารถแก้ไขคำสั่งให้ถูกต้องได้
- ทำให้ผู้เรียนมีความรู้ และความเข้าใจมากขึ้นในภาษาที่เคยเรียนไปแล้ว
- ทำให้ผู้เรียนสามารถเขียนโปรแกรมที่มีการประมวลผลที่มีประสิทธิภาพดีขึ้น โดยสามารถเลือกใช้คำสั่งที่เหมาะสมในการประมวลผล

เมื่อผู้เรียนมีความเข้าใจในเรื่องหลักการแล้วก็จะสามารถเขียนโปรแกรมที่ดีได้ โดยคุณลักษณะของโปรแกรมที่ดี อาจพิจารณาได้จากข้อต่อไปนี้

- โปรแกรมมีจุดบกพร่องน้อย
- ไม่ต้องแก้ไขโปรแกรมบ่อย
- สามารถใช้งานโปรแกรมที่พัฒนาได้ในระยะนาน
- โปรแกรมทำงานได้อย่างถูกต้อง เช่น การคำนวณต่าง ๆ มีความถูกต้อง
- โปรแกรมไม่มีข้อผิดพลาดในเวลารัน เช่น ไม่ใช้หน่วยความจำสิ้นเปลืองจนกระทั่งประมวลผลต่อไม่ได้
- โค้ดในโปรแกรมอ่านง่าย ซึ่งจะเป็นประโยชน์ในการปรับปรุงโปรแกรมในระหว่างการดูแลรักษาโปรแกรมในระยะยาว

กิจกรรมที่ 1.1

กำหนดโค้ดกิจกรรมที่ 1.1 ซึ่งเขียนด้วยภาษาซี

(1) ให้วิจารณ์โปรแกรมโดยเปรียบเทียบกับคุณลักษณะของโปรแกรมที่ดี

(2) ให้ทดลองรันโปรแกรมเพื่อค้นหาว่าโปรแกรมมีข้อผิดพลาดหรือไม่

การเปรียบเทียบคุณลักษณะกับโปรแกรมที่ดี อาจพิจารณาดังต่อไปนี้

- โอกาสเกิดจุดบกพร่องเป็นอย่างไร ? มีโอกาสเกิดมากหรือน้อยเพราะสาเหตุใด ?
- การแก้ไขโค้ดเป็นอย่างไร ? สามารถทำได้ง่ายหรือยาก ?
- ความถูกต้องของโค้ดน่าจะตรวจได้ง่ายหรือยาก ?
- โค้ดในโปรแกรมอ่านง่ายหรือไม่ ? หรือเมื่อดูโค้ดแล้วทราบลำดับการทำงานที่ชัดเจนหรือไม่ ?

```
#include <stdio.h>
int main(void) {
const int SIZE = 10;int peopleage[10] = {60,18,53,51,42,56,61,34,78,26};
int gen_z=0,gen_millenial=0,gen_x=0,gen_babyboomer=0,gen_silent=0;
int i;
if (SIZE>0) for(i=0;i<SIZE;i++)
if (peopleage[i] <=25) {printf("This person %d is in gen Z.\n",i+1);gen_z++;}else if
(peopleage[i] >26 && peopleage[i]<=40){
printf("This person %d is in gen millenial.\n",i+1);gen_millenial++;}else if
(peopleage[i] >41 && peopleage[i]<=55){printf("This person %d is in gen
X.\n",i+1);gen_x++;}else if
(peopleage[i] >56 &&
peopleage[i]<=75){ printf("This person %d is in gen baby
boomers.\n",i+1);gen_babyboomer++;}else
{printf("This person %d is in gen silent generation.\n",i+1);gen_silent++;}else
printf("Please specify people's ages.");
return 0; }
```

1.3 รูปแบบการเขียนโปรแกรมที่ดี

การเขียนโปรแกรมที่ดีควรคำนึงถึงการกำหนดบล็อกคำสั่ง และการเยื้องหน้า

(1) การกำหนดบล็อก (block) เป็นการกำหนดขอบเขตของคำสั่ง เพื่อให้เห็นกลุ่มคำสั่งที่ทำงานร่วมกันภายในขอบเขตที่กำหนด ในภาษาโปรแกรมหลาย ๆ ภาษาเลือกใช้เครื่องหมายปีกกา {...} เพื่อกำหนดขอบเขตคำสั่ง เช่น ภาษาซี และภาษาจาวา ซึ่งทำให้การอ่านโค้ดง่ายขึ้น ในการกำหนดขอบเขตคำสั่งนั้นมักมีข้อกำหนดตามมา เช่น จะสามารถกำหนดคำสั่งภายในขอบเขตได้กี่คำสั่ง ซึ่งภาษาส่วนใหญ่กำหนดให้มีได้หลายคำสั่ง ดังนั้นหากต้องการกำหนดเพียงหนึ่งคำสั่งก็อาจไม่จำเป็นต้องใช้เครื่องหมายกำหนดบล็อกคำสั่งก็ได้

(2) การเยื้องหน้า (Indentation) เพื่อให้อ่านง่าย เมื่อมีการกำหนดขอบเขตของคำสั่งแล้วการเยื้องหน้าก็ควรทำให้สอดคล้องกันเพื่อให้เห็นขอบเขตของคำสั่งที่ชัดเจนขึ้น

ตัวอย่างการกำหนดบล็อกคำสั่งของคำสั่ง for ในภาษาซี

```
for (m = 0; m <n; m++){
        result = a[m] + 10;
}
```

หรือคำสั่ง if-else เช่น

```
if( a < 20 ) {
        printf("a is less than 20\n" );
}
else {
        printf("a is not less than 20\n" );
}
```

ซึ่งภาษาซีนั้นอาจเขียน if-else โดยไม่ใช้เครื่องหมายปีกกาก็ได้ แต่จะต้องมีเพียง 1 ประโยคคำสั่งในส่วน if และ else เท่านั้น เช่น

```
if( a < 20 )
        printf("a is less than 20\n" );
else
        printf("a is not less than 20\n" );
```

และจำเป็นต้องใช้ปีกกาเสมอเมื่อมีหลายประโยคคำสั่งที่ถูกดำเนินการ เช่น

```
if( a < 20 ) {
        printf("a is less than 20\n" );
        printf("a is less than 20\n" );
}
else {
        printf("a is not less than 20\n" );
        printf("a is not less than 20\n" );
}
```

ตัวอย่างการเยื้องย่อหน้าในภาษาซี ต่อไปนี้

```
for (i = 0; i <j; i++){
        for (m = 0; m <n; m++){
                result = a[m] + 10;
        }
}
```

อนึ่ง ในภาษาซี หรือภาษาจาวา การเยื้องย่อหน้าเป็นการกำหนดความเป็นระเบียบของโค้ด เพื่อให้อ่านง่ายเท่านั้น ส่วนการเยื้องย่อหน้าในภาษาไพธอน เป็นการกำหนดกลุ่มคำสั่งที่อยู่บล็อกเดียวกัน ดังนั้นการจัดระเบียบโค้ดจะต้องระมัดระวังเป็นพิเศษ

กิจกรรมที่ 1.2

จากกิจกรรมที่ 1.1 ให้จัดระเบียบโค้ดใหม่โดยใช้การเยื้องย่อหน้า และการกำหนดเครื่องหมายปีกกา {...} กำกับบล็อกคำสั่ง

กิจกรรมที่ 1.3

กำหนดโค้ดภาษาไพธอนที่เขียนเกมทายเลขเป็นเกมที่คอมพิวเตอร์กำหนดค่าเริ่มต้นของตัวเลขที่ทาย (ตัวแปร guess) คือ 5 และผลลัพธ์ (ตัวแปร number) คือ 10 ถ้าผู้เล่นทายตัวเลขสูงกว่า 10 โปรแกรมจะแจ้งว่าทายสูงไปแล้วให้ป้อนตัวเลขใหม่ และถ้าทายต่ำกว่า 10 โปรแกรมจะแจ้งว่าทายต่ำไปแล้วให้ป้อนตัวเลขใหม่ และถ้าเท่ากับ 10 โปรแกรมจะแจ้งว่าทายถูก

โค้ดที่กำหนดเป็นคำสั่งที่ยังไม่ได้เรียบเรียงลำดับ ให้นำคำสั่งไปจัดเรียงใหม่แล้วรันในเครื่องมือเขียนโปรแกรมภาษาไพธอนใดก็ได้ เพื่อให้ทำงานได้ตามความต้องการข้างต้น

คำแนะนำ

ไพธอนเป็นภาษาที่ไม่มีเครื่องหมายกำหนดบล็อกคำสั่ง แต่ใช้การเยื้องหน้าเป็นการกำหนดบล็อกคำสั่งแทน ดังนั้นในการเรียงคำสั่งควรพิจารณาการกำหนดบล็อกคำสั่ง

```
number = 10
while guess != number:
guess = 5
if guess > number:
print(guess, "is too high.")
guess = int(input("Guess again: "))
elif guess < number:
print(guess, " is too low.")
print(guess, "is the correct number.")
```

1.4 โดเมนกับภาษาโปรแกรม

ภาษาโปรแกรมมักถูกนำไปใช้ในการแก้ปัญหาในโดเมนหนึ่ง ๆ โดเมน (Domain) คือสิ่ง หรือเรื่องที่สนใจซึ่งมีความเฉพาะเจาะจง ภาษาโปรแกรมที่พัฒนาขึ้นมานั้นมักมีความสัมพันธ์กับโดเมนในการแก้ปัญหาหรือโดเมนของระบบงาน จึงอาจจำแนกภาษาโปรแกรมตามการนำไปใช้ในโดเมนต่าง ๆ ออกเป็นหลายลักษณะดังตัวอย่างต่อไปนี้

- กลุ่มภาษาโปรแกรมที่ใช้พัฒนาระบบงานทางวิทยาศาสตร์ (Scientific Application) ภาษาโปรแกรมถูกนำไปใช้เขียนโปรแกรมเพื่อแก้ปัญหาที่เกี่ยวข้องกับการคำนวณทางคณิตศาสตร์

เป็นหลัก ตัวอย่างภาษาโปรแกรมในกลุ่มนี้ อาทิ ภาษาฟอร์แทรน (Fortran) และภาษาอัลกอล์ 60 (ALGOL 60)

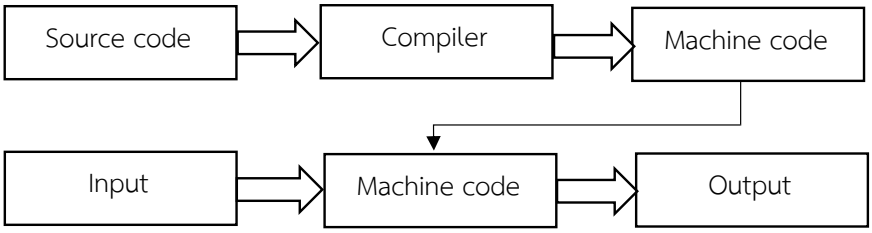
- กลุ่มภาษาโปรแกรมที่ใช้พัฒนาระบบงานทางธุรกิจ (Business Application) ภาษาโปรแกรมถูกนำไปใช้ในการแก้ไขปัญหาที่เกี่ยวข้องกับการคำนวณเลขทศนิยมฐานสิบ และการออกรายงานต่าง ๆ ภาษาโปรแกรมในกลุ่มนี้ อาทิ ภาษาโคบอล (COBOL) และภาษาอาร์พีจี (RPG)
- กลุ่มภาษาโปรแกรมที่ใช้พัฒนาระบบงานทางปัญญาประดิษฐ์ (Artificial Intelligence Application) ภาษาโปรแกรมถูกนำไปใช้ในการแก้ปัญหาทางปัญญาประดิษฐ์ โดยไม่เน้นการแก้ปัญหาที่เกี่ยวข้องกับการประมวลผลตัวเลข การเขียนโปรแกรมมีการกำหนดรูปประโยคคำสั่งแบบตัวอักษรหรือข้อความ ได้แก่ ภาษาลิสป์ (LISP) โปรล็อก (Prolog) และภาษาสคีม (Scheme)
- กลุ่มภาษาโปรแกรมสำหรับระบบ (Systems Programming) ภาษาโปรแกรมในกลุ่มนี้ถูกพัฒนาขึ้นเพื่อแก้ปัญหาหรือเพื่อทำงานกับระบบฮาร์ดแวร์หนึ่ง ๆ หรือระบบปฏิบัติการหนึ่ง ๆ ภาษาโปรแกรมที่ออกแบบ และพัฒนาเพื่อให้ทำงานกับระบบต่าง ๆ นั้นมักจะถูกออกแบบโดยผูกติดกับสถาปัตยกรรมเครื่องคอมพิวเตอร์หรือระบบปฏิบัติการ อาทิ สถาปัตยกรรมระบบไอบีเอ็ม มีภาษาพีแอลเอส (PL/S) เป็นภาษาพื้นฐานในการเขียนโปรแกรม สถาปัตยกรรมระบบ เดคพีดีพี-10 (DEC PDP-10) มีภาษาบลิสส์ (BLISS) เป็นภาษาพื้นฐานในการเขียนโปรแกรม และระบบปฏิบัติการยูนิกซ์มีภาษาซีเป็นภาษาพื้นฐานในการเขียนโปรแกรม
- กลุ่มภาษาโปรแกรมสำหรับเว็บ (Web Software) ภาษาโปรแกรมในกลุ่มนี้เป็นภาษาเพื่อการแสดงผลบนระบบเว็บ หรือเพื่อการพัฒนาโปรแกรมประยุกต์บนระบบเว็บ เช่น ภาษาจาวา ภาษาพีเอชพี (PHP) และภาษาจาวาสคริปต์ (JavaScript)

อนึ่ง บางภาษาโปรแกรมอาจไม่สามารถระบุโดเมนหลักได้ เพราะมีการนำไปใช้ในการแก้ปัญหาในหลายโดเมน ยกตัวอย่างเช่น ภาษาไพธอน

1.5 การนำโปรแกรมไปใช้

การนำโปรแกรมไปใช้ (Program Implementation) คือวิธีการในการนำโปรแกรมไปใช้งาน เมื่อเขียนโปรแกรมเสร็จก็ต้องมีวิธีการนำไปใช้ สามารถแบ่งออกเป็น 3 วิธีการนำโปรแกรมไปใช้

1.5.1 โปรแกรมที่จะต้องคอมไพล์ก่อนนำไปใช้ (Compilation Program)



ภาพที่ 1.1 โปรแกรมที่จะต้องคอมไพล์ก่อนนำไปใช้

ในกลุ่มนี้ต้องคอมไพล์เพื่อขจัดจุดบกพร่องก่อนทำการสร้างรหัสเครื่อง (Machine code) โดยโค้ดที่ได้จากภาษาโปรแกรมในกลุ่มนี้มักจะยึดติดกับสถาปัตยกรรมที่ใช้งาน ภาษาโปรแกรมในกลุ่มนี้ได้แก่ ภาษาซี ภาษาซีพลัสพลัส (C++) ภาษาปาสคาล (Pascal) ภาษาโคบอล และภาษาฟอร์แทรน คอมไพเลอร์ (Compiler) เป็นคำที่ใช้เรียกโปรแกรมที่ทำหน้าที่คอมไพล์โปรแกรมที่นักพัฒนาโปรแกรมเขียนขึ้น และทำหน้าที่ในการสร้างรหัสเครื่องซึ่งเป็นไฟล์นามสกุลแตกต่างกันไปในแต่ละภาษา เช่น ภาษาซีมีคอมไพเลอร์สร้างไฟล์นามสกุลอีเอ็กซ์อี (EXE) เป็นรหัสเครื่องที่สามารถรันได้กับสถาปัตยกรรมหนึ่ง ๆ ภาพที่ 1.1 แสดงกระบวนการในการคอมไพล์ซึ่งซอร์สโค้ดจะถูกแปลงเป็นรหัสเครื่องโดยคอมไพเลอร์ เมื่อนำโปรแกรมไปใช้งานรหัสเครื่องจะรับอินพุตซึ่งอาจเป็นข้อมูลต่าง ๆ ที่ป้อนผ่านคีย์บอร์ดแล้วทำการสร้างเอาต์พุตไปยังหน่วยเอาต์พุต (เช่น หน้าจอ)

1.5.2 โปรแกรมที่จะต้องตีความก่อนนำไปใช้ (Interpretation Program)

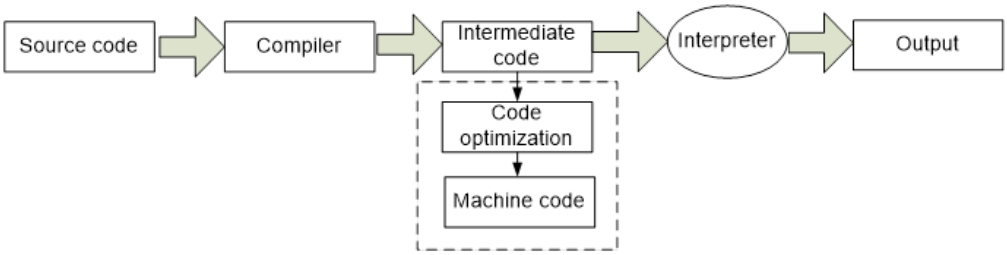
ในกลุ่มนี้ไม่มีกระบวนการคอมไพล์ และไม่มีการสร้างรหัสเครื่อง แต่มีกระบวนการตีความ (Interpretation) เพื่อแสดงผลลัพธ์จากการรันโปรแกรมโดยมีตัวตีความ (Interpreter) ทำหน้าที่ในการตีความเพื่อแสดงเอาต์พุตออกมา (ภาพที่ 1.2) ภาษาโปรแกรมที่อยู่ในกลุ่มนี้ ได้แก่ ภาษาพีเอชพี ภาษาเอชทีเอ็มแอล (HTML) ภาษาเอ็กซ์เอสแอลที (XSLT) ภาษาไพธอน ภาษาเพิร์ล (Perl) และภาษาจาวาสคริปต์

ภาพที่ 1.2 โปรแกรมที่จะต้องตีความก่อนนำไปใช้

1.5.3 โปรแกรมที่จะต้องมีการคอมไพล์และตีความก่อนนำไปใช้ (Hybrid Implementation Program)

ในกลุ่มนี้มีกระบวนการนำไปใช้แบบผสม ซึ่งต้องมีกระบวนการคอมไพล์และกระบวนการตีความ จะมีการสร้างโค้ดระหว่างกลาง และอาจมีหรือไม่มีการสร้างรหัสเครื่องก็ได้ เช่น การใช้งานภาษาจาวาโดยปกติแล้วไม่มีการสร้างรหัสเครื่อง หลังจากการคอมไพล์จะได้โค้ดระหว่างกลางหรืออาจเรียกว่า ไบต์โค้ด (Bytecode) ซึ่งเป็นไฟล์นามสกุล .class ไบต์โค้ดจะถูกประมวลผลด้วยตัวตีความที่เรียกว่า เจวีเอ็ม (Java Virtual Machine: JVM) การเขียนโปรแกรมในบางเฟรมเวิร์กสามารถสร้างรหัสเครื่องได้ เช่น การเขียนโปรแกรมด้วยภาษาจาวาในลักษณะ JIT (just-in-time) การเขียนโปรแกรมในลักษณะ JIT นี้มักนำไปใช้ในการเขียนโปรแกรมกับฮาร์ดแวร์พิเศษที่สามารถประมวลผลโค้ดภาษาจาวาได้

ภาพที่ 1.3 แสดงองค์ประกอบต่าง ๆ ในภาษาโปรแกรมที่มีการนำไปใช้แบบผสม ซึ่งมีกระบวนการเพิ่มประสิทธิภาพโค้ด (Code optimization) เพื่อปรับปรุงโค้ดระหว่างกลางให้มีประสิทธิภาพมากขึ้น แล้วนำโค้ดนั้นไปสร้างเป็นรหัสเครื่องต่อไป การเขียนโปรแกรมสำหรับแพลตฟอร์มแอนดรอยด์ (Android) ในโทรศัพท์มือถือที่รองรับระบบปฏิบัติการแอนดรอยด์บางเวอร์ชันมีการเรียกใช้ JIT Compiler เพื่อสร้างรหัสเครื่อง (machine code) ที่สามารถรันได้บนโทรศัพท์มือถือในบางรุ่นโดยเฉพาะ



ภาพที่ 1.3 โปรแกรมที่จะต้องคอมไพล์หรือตีความก่อนนำไปใช้

1.6 ข้อดีและข้อเสียของแต่ละวิธีการนำไปใช้

ข้อดีและข้อเสียของวิธีการนำไปใช้ของภาษาสำหรับการเขียนโปรแกรมทั้งสามรูปแบบ ข้อดีแสดงในตารางที่ 1.1 และข้อเสียแสดงในตารางที่ 1.2 จะสังเกตเห็นได้ว่า หากภาษาโปรแกรมเสียเวลาในการคอมไพล์ก็มักจะทำให้สามารถรันโปรแกรมได้รวดเร็ว ทั้งนี้ เป็นเพราะว่าในช่วงเวลาคอมไพล์นั้นมีการขจัดข้อผิดพลาดออกไปจากโปรแกรมแล้ว ดังนั้นเมื่อนำไปรันก็สามารถประมวลผลได้รวดเร็ว ส่วนภาษาโปรแกรมที่ไม่มีการคอมไพล์จะไม่เสียเวลาในการคอมไพล์ แต่เสียเวลาในการรันโปรแกรม เนื่องจากในระหว่างการรันนั้นเกิดกระบวนการตรวจสอบข้อผิดพลาด และหากมีการแก้ไขก็จะต้องจัดการให้เสร็จก่อนรันใหม่เสมอ ๆ นอกจากนี้ภาษาโปรแกรมที่มีการสร้างรหัสเครื่องมักผูกติดกับสถาปัตยกรรมหนึ่ง ๆ กล่าวคือ หากใช้ภาษาโปรแกรมที่พัฒนาขึ้นเพื่อสนับสนุนสถาปัตยกรรมนั้น ๆ ก็จะทำงานได้ ดังนั้นจึงอาจไม่สามารถรองรับหลายสถาปัตยกรรม และไม่สะดวกในการเคลื่อนย้ายโปรแกรม

ตารางที่ 1.1 ข้อดีของวิธีการนำไปใช้

โปรแกรมที่จะต้องคอมไพล์ก่อนนำไปใช้	โปรแกรมที่จะต้องตีความก่อนนำไปใช้	โปรแกรมที่จะต้องมีการคอมไพล์และตีความก่อนนำไปใช้
● โค้ดที่นำไปใช้ในเวลารันมีขนาดเล็ก ● โค้ดที่นำไปใช้ในเวลารันสามารถประมวลผลได้เร็ว	● การพัฒนาโปรแกรมทำได้เร็ว ● โปรแกรมมีความยืดหยุ่นสามารถนำไปใช้งานได้หลายระบบปฏิบัติการ เช่น วินโดว์ ลินุกซ์ ● การนำไปใช้มีลักษณะไดนามิก เช่น โค้ดของโปรแกรมถูกพัฒนาหรือนำไปใช้งานในสภาวะต่าง ๆ ได้	● โค้ดที่นำไปใช้ในเวลารันมีขนาดเล็ก ● โค้ดที่นำไปใช้ในเวลารันสามารถประมวลผลได้เร็ว ● โปรแกรมมีความยืดหยุ่นสามารถนำไปใช้งานได้หลายระบบปฏิบัติการ

ตารางที่ 1.2 ข้อเสียของวิธีการนำไปใช้

โปรแกรมที่จะต้องคอมไพล์ก่อนนำไปใช้	โปรแกรมที่จะต้องตีความก่อนนำไปใช้	โปรแกรมที่จะต้องมีการคอมไพล์และตีความก่อนนำไปใช้
● โค้ดที่ผ่านการคอมไพล์จะนำไปรันหรือใช้งานบนเครื่องคอมพิวเตอร์ที่มีสถาปัตยกรรมเฉพาะ ● เสียเวลาในการคอมไพล์นาน	● เวลารันทำงานช้า เพราะเสียเวลาในการตรวจสอบและแก้ไขข้อผิดพลาด	● โค้ดที่ผ่านการคอมไพล์เพื่อสร้างรหัสเครื่องจะนำไปรันหรือใช้งานบนเครื่องคอมพิวเตอร์ที่มีสถาปัตยกรรมเฉพาะ ● เสียเวลาในการคอมไพล์นาน

1.7 สิ่งที่มีอิทธิพลต่อการใช้งานภาษาโปรแกรม

ภาษาสำหรับการโปรแกรมมีมากมายหลายภาษา สิ่งที่มีอิทธิพลต่อการใช้งานภาษาโปรแกรมมีอยู่ 3 ปัจจัย อาทิ

(1) ธุรกิจหรือแวดวงอุตสาหกรรมซอฟต์แวร์ เช่น เมื่อบริษัทใหญ่ ๆ ที่มีอิทธิพลในแวดวงการพัฒนาซอฟต์แวร์แนะนำภาษาใหม่ ๆ นักพัฒนาโปรแกรมก็มักจะให้ความสนใจเสมอ ซึ่งบริษัทที่มีอิทธิพลในแวดวงอุตสาหกรรม เช่น บริษัทไอบีเอ็ม ไมโครซอฟท์ ออราเคิล กูเกิล แอปเปิล และเฟซบุ๊ก ให้ความสนใจภาษาใดทุกคนก็มักจะสนใจ และทดลองใช้ตาม หรือการนำเสนอระบบปฏิบัติการใหม่ก็นำมาซึ่งภาษาที่สามารถสนับสนุนระบบปฏิบัติการนั้น ๆ เช่น ระบบปฏิบัติการแอนดรอยด์ใช้ภาษาจาวา หรือระบบปฏิบัติการไอโอเอส ใช้ภาษาสวิฟต์ (Swift) ในการพัฒนาโปรแกรม การแนะนำภาษาขึ้นมาใหม่นั้น มักแสดงถึงความเข้ากันได้ของภาษา และระบบปฏิบัติการ

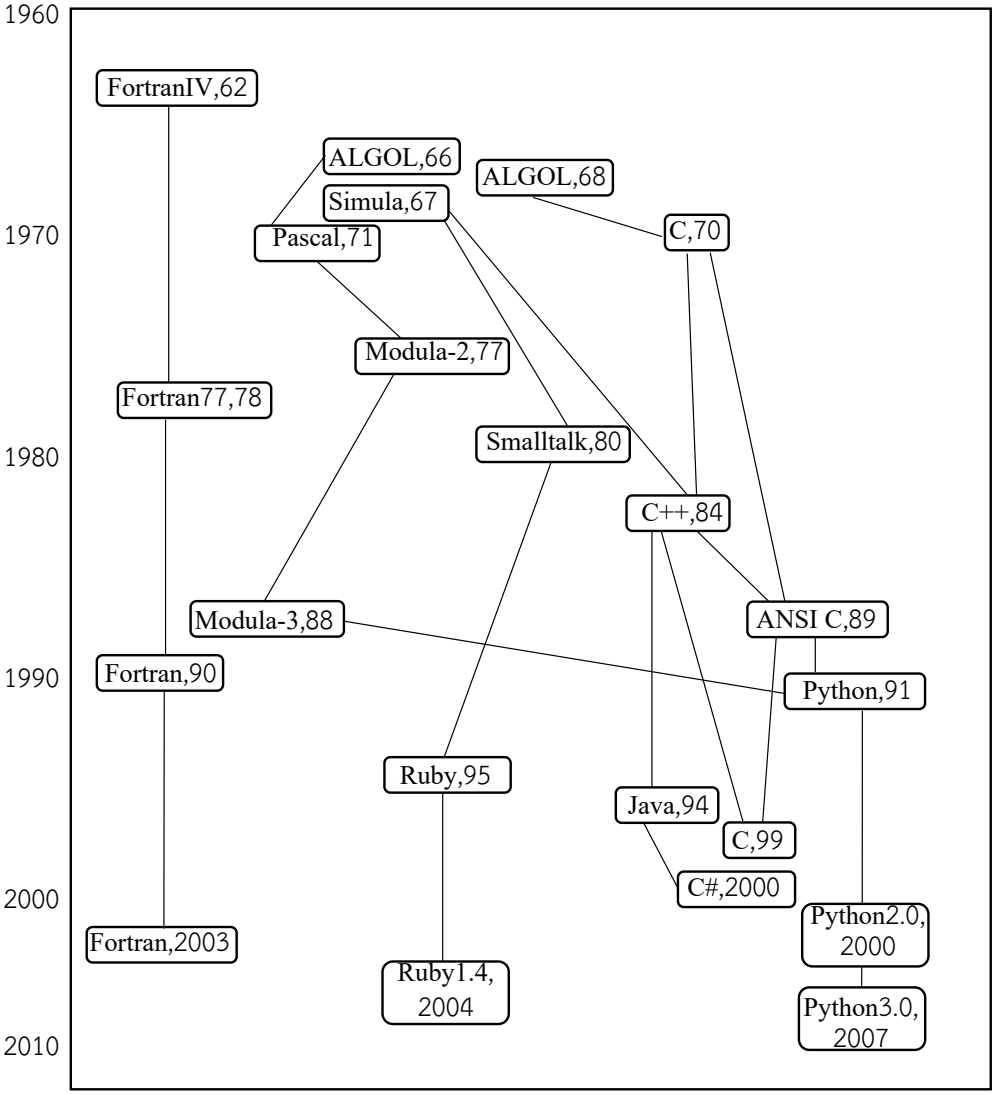
(2) นักวิชาการในแวดวงคอมพิวเตอร์ หากอาจารย์แนะนำให้นักศึกษาได้รู้จัก และใช้ซอฟต์แวร์ภาษาสำหรับการโปรแกรม ก็จะทำให้นักศึกษาที่จบการศึกษาไปได้เลือกใช้ภาษานั้นในการพัฒนาระบบเนื่องจากมีความรู้จัก และคุ้นเคยในการใช้งานมาก่อน และหากเลือกใช้ภาษาที่เป็นโอเพนซอร์ส (Open source) ก็ย่อมทำให้ซอฟต์แวร์ภาษาโปรแกรมของบริษัทผู้ผลิตไม่สามารถ

ขายสินค้าได้ อาจสังเกตเห็นได้ว่าบริษัทที่จำหน่ายซอฟต์แวร์ภาษาโปรแกรมมักติดต่อสถานการศึกษา และสนับสนุนภาษาโปรแกรมของบริษัทตนเองเพื่อสร้างฐานลูกค้าในอนาคต

(3) ความยากหรือง่ายในการนำไปใช้ของภาษาโปรแกรม ภาษาที่ใช้ง่าย เขียนง่าย สามารถแก้ไขปัญหาได้ดี ย่อมเป็นที่นิยม และถูกนำไปใช้แพร่หลาย ดังนั้นภาษาใหม่ ๆ ที่ถูกพัฒนาขึ้นก็มักจะดึงเอาคุณลักษณะของภาษาที่มีการใช้งานอย่างกว้างขวางมาเป็นส่วนหนึ่งในการออกแบบภาษาใหม่ ๆ ซึ่งหากนักพัฒนาโปรแกรมได้เลือกใช้ภาษาใหม่ที่พัฒนาขึ้นก็จะเรียนรู้ได้ง่าย และอาจได้รับความนิยมต่อไป เช่น ภาษาจาวาสคริปต์ เป็นภาษาที่มีคุณลักษณะที่ดีในการทำงานเชิงฟังก์ชัน และสามารถสนับสนุนการเขียนโปรแกรมได้หลายรูปแบบจึงได้รับความนิยมนำไปใช้ในการพัฒนาโปรแกรมบนระบบเว็บอย่างกว้างขวาง จะสังเกตเห็นได้ว่ามีการพัฒนาแพลตฟอร์มหรือไลบรารีเพื่อสนับสนุนการทำงานกับภาษาจาวาสคริปต์ เช่น โนดเจเอส (Node.js) เป็นเฟรมเวิร์กสนับสนุนการทำงานการเขียนโปรแกรมในฝั่งแบ็กเอนด์ และวูเจเอส (Vue.js) เป็นเฟรมเวิร์กสนับสนุนการทำงานการเขียนโปรแกรมในฝั่งฟรอนต์เอนด์ และยังมีไลบรารีอื่น ๆ เช่น ไลบรารีเจคิวรี ไลบรารีเอแจ็กซ์ ที่สนับสนุนการเขียนโปรแกรมกับภาษาจาวาสคริปต์

1.8 วิวัฒนาการของภาษา

การพัฒนาภาษาสำหรับการโปรแกรมมีมาอย่างยาวนาน บางภาษามีการพัฒนาเสร็จสิ้นในอดีต แต่เพิ่งถูกเลือกใช้ในการพัฒนาซอฟต์แวร์อย่างแพร่หลายในเวลาต่อมา ภาษาโปรแกรมหนึ่ง ๆ มักมีความสัมพันธ์กับภาษาโปรแกรมอื่น ๆ เนื่องจากในการออกแบบภาษานั้นอาจนำข้อดีจากภาษาโปรแกรมที่สำเร็จก่อนหน้ามาออกแบบในภาษาใหม่ หรือทำการปรับปรุงให้มีการใช้งานที่หลากหลายขึ้น



ภาพที่ 1.4 วิวัฒนาการของภาษาจาก [7]

ภาพที่ 1.4 แสดงผังเวลา และการกำเนิดของภาษาต่าง ๆ ยกตัวอย่างเช่น ภาษาซีพลัสพลัส (ค.ศ. 1984) มีต้นกำเนิดมาจากภาษาซิมิวลา และภาษาซี ภาษาซีพลัสพลัส และภาษาจาวา เป็นต้น กำเนิดของภาษาซีชาร์ป และภาษาแอนไซซี (ANSI C) (ค.ศ. 1989) และมอดุลา-3 (Modula-3) เป็นต้น กำเนิดของภาษาไพธอน ภาษาฟอร์แทรนเป็นภาษาสำหรับการคำนวณทางวิทยาศาสตร์ที่มีการพัฒนามาอย่างยาวนานตั้งแต่ก่อน ค.ศ. 1960 จนกระทั่งถึง ค.ศ. 2003

1.9 สรุป

ภาษาโปรแกรมมักมีการออกแบบด้วยหลักการที่คล้ายคลึงกัน ดังนั้นการเรียนรู้หลักการจะทำให้ผู้เรียนสามารถเปรียบเทียบ และเห็นความแตกต่างของภาษาโปรแกรมได้ และทำให้สามารถแก้ปัญหาด้วยวิธีการที่หลากหลาย อีกทั้งทำให้การศึกษาภาษาใหม่ ๆ ทำได้รวดเร็วขึ้น

การเขียนโปรแกรมที่ดีจะต้องพิจารณาการกำหนดบล็อกคำสั่ง และเยื้องหน้า แต่ละภาษาอาจมีข้อกำหนด และสัญลักษณ์ที่เลือกใช้ไม่เหมือนกัน การเขียนโปรแกรมที่ดีจะทำให้อ่านโค้ดได้ง่าย และสามารถแก้ไขโปรแกรมได้อย่างมีประสิทธิภาพ ภาษาโปรแกรมที่ถูกออกแบบมักมีโดเมนหลัก แต่บางภาษาอาจสนับสนุนการทำงานในหลายโดเมน การนำโปรแกรมไปใช้สามารถพิจารณากลุ่มภาษาตามการนำไปใช้ได้สามประเภท คือภาษาที่ต้องคอมไพล์ ภาษาที่ต้องตีความ และภาษาที่ต้องคอมไพล์ และ/หรือตีความซึ่งมีความสามารถสอดคล้องสองกลุ่มแรก ภาษาที่ต้องคอมไพล์มีการขจัดข้อผิดพลาดในเวลาคอมไพล์มักจะรันได้เร็ว ส่วนภาษาที่ต้องตีความไม่มีขั้นตอนของการคอมไพล์ ดังนั้นการตรวจหาข้อผิดพลาดจึงเกิดในเวลารันจึงทำให้รันโปรแกรมได้ช้า ในกระบวนการตีความอาจมีการสร้างภาษาระหว่างกลางซึ่งตัวตีความนำไปใช้ในการประมวลผล และแสดงผล ภาษาจาวาจัดว่าอยู่ในกลุ่มของภาษาที่ต้องคอมไพล์ และ/หรือตีความ เช่น การใช้เฟรมเวิร์ก JIT มีการสร้างรหัสเครื่อง ส่วนการรันไบต์โค้ด (.class) ในเจวีเอ็ม คือการตีความภาษาสื่อกลางซึ่งก็ คือไบต์โค้ดของภาษาจาวา

ภาษาโปรแกรมมักถูกออกแบบโดยนำคุณลักษณะของภาษาที่ถูกพัฒนา และถูกนำไปใช้ก่อนหน้ามาทำการปรับปรุงหรือเพิ่มเติมความสามารถในภาษาใหม่ที่พัฒนาให้โดดเด่นขึ้น ดังนั้นแต่ละภาษาล้วนมีความเกี่ยวข้องกัน และหลักการต่าง ๆ ในแต่ละภาษาจึงมีความคล้ายคลึงกัน