บทที่ 8

ภาษาโปรแกรม

8.1 หลักของภาษาโปรแกรม

8.2 รูปแบบของภาษาโปรแกรม

8.3 ประวัติความเป็นมา

8.4 เอกสารอ้างอิงและเว็บไซต์ที่ควรรู้

วัตถุประสงค์

- แนะนำภาษาโปรแกรม
- แนวคิดและที่มาของภาษาโปรแกรม
- ภาษาโปรแกรมในรูปแบบต่างๆ

วิธีการหรือเครื่องมือที่ใช้ในการติดต่อสื่อสารระหว่างมนุษย์ก็คือภาษาธรรมชาติ คอมพิวเตอร์ก็มีลักษณะคล้ายกัน คือ นอกจากจะใช้ระบบเลขฐานสองหรือภาษาเครื่องในการติดต่อสื่อสารระหว่างและภายในคอมพิวเตอร์แล้ว ยังมีภาษาโปรแกรม (Programming language) เป็นสื่อกลางในการสื่อสารกันระหว่างโปรแกรมเมอร์ ภาษาโปรแกรมมีข้อแตกต่างจากภาษาธรรมชาติที่สำคัญอยู่ 2 ประการ ประการแรก ภาษาโปรแกรมสามารถใช้ในการสื่อสารระหว่างมนุษย์กับเครื่องคอมพิวเตอร์ได้ ประการที่สอง ภาษาโปรแกรมมีขอบเขตที่ใช้ในการแสดงความหมายแคบกว่าภาษาธรรมชาติ เพราะเพียงต้องการช่วยให้สื่อสารแนวความคิดทางคอมพิวเตอร์ทำได้ง่าย ดังนั้น ภาษาโปรแกรมจึงมีข้อกำหนดที่แตกต่างจากภาษาธรรมชาติ ซึ่งเราจะศึกษาเกี่ยวกับข้อกำหนดเหล่านี้ในรายวิชานี้นั่นเอง

นักศึกษาสาขาเทคโนโลยีสารสนเทศจำเป็นต้องศึกษาเกี่ยวกับภาษาโปรแกรม ในฐานะเป็นผู้ใช้ภาษาของวันนี้และภาษาโปรแกรมใหม่ๆ ที่จะเกิดขึ้นในอนาคต สำหรับนักศึกษาสาขาวิทยาการคอมพิวเตอร์นั้นจำเป็นต้องรู้ลึกซึ้งในแง่ของการออกแบบภาษา ดังนั้น จึงจำเป็นต้องมีความรู้ความเข้าใจภาษาโปรแกรมอย่างกระจ่าง รู้ถึงลักษณะของภาษา ข้อดีและข้อเสียของภาษาโปรแกรมหลากหลายรูปแบบ รวมถึงการนำไปประยุกต์ใช้ให้เหมาะสมกับงานได้ อีกทั้งยังเป็นพื้นฐานในการเรียนรู้ภาษาโปรแกรมใหม่ๆ ได้ง่ายขึ้น และเป็นพื้นฐานที่สำคัญในการสร้างภาษาโปแกรมภาษาใหม่ได้ การศึกษาภาษาโปรแกรมและการนำไปใช้เพียงภาษาเดียวจึงไม่สามารถก่อให้เกิดความรู้ความเข้าใจได้อย่างถ่องแท้ ในรายวิชานี้จะกล่าวถึงหลักการทั่วไปของภาษาโปรแกรม และกล่าวถึงภาษาโปรแกรมหลายๆ รูปแบบ

8.1 หลักของภาษาโปรแกรม (Principles)

ภาษาโปรแกรมเป็นเครื่องมือที่สั่งให้คอมพิวเตอร์ทำงานได้ตามที่ต้องการ และเป็นเครื่องมือหรือวิธีการที่โปรแกรมเมอร์ใช้ในการสื่อสารกันด้วย นอกจากนี้ยังอาจมองได้ว่าภาษาโปรแกรมเป็นสัญลักษณ์ที่ใช้สำหรับขั้นตอนการแก้ปัญหา (Notation for algorithms) ซึ่งเป็นวิธีการแสดงความสัมพันธ์ระหว่างแนวคิด เพื่อใช้ในการควบคุมอุปกรณ์คอมพิวเตอร์

ภาษาโปรแกรมแต่ละภาษาสามารถพิจารณาว่า เป็นเซตของข้อกำหนดอย่างเป็นทางการของ Syntax (วากยสัมพันธ์) ซึ่งเป็นส่วนของคำศัพท์พื้นฐานที่ใช้ในโครงสร้างของภาษา และ Semantic ซึ่งก็คือความหมายของคำศัพท์เหล่านั้นทั้งในแง่ของคำและในแง่ของการทำงานในทางปฏิบัติ ซึ่งข้อกำหนดเหล่านี้จะช่วยในเข้าใจการทำงานของภาษาได้เป็นอย่างดี ข้อกำหนดเหล่านี้มักรวมถึง

- ข้อมูลและโครงสร้างข้อมูล
- คำสั่งและลำดับการทำงาน
- ปรัชญาในการออกแบบ
- สถาปัตยกรรมของภาษา

หลักการของการออกแบบภาษาโปรแกรม อาจแบ่งออกเป็น 3 ประเด็นหลัก ได้แก่

- วากยสัมพันธ์ (syntax)
- ชื่อและชนิด (names and types)
- ความหมาย (semantics)

แนวคิดของหลายๆ อย่างของหลักการออกแบบนี้ถูกหยิบยกมาจากภาษาศาสตร์และคณิตศาสตร์ ซึ่งจะกล่าวโดยย่อในบทนี้ และกล่าวโดยละเอียดในบทถัดๆ ไป

วากยสัมพันธ์

วากยสัมพันธ์ของภาษาเป็นสิ่งที่ใช้อธิบายส่วนประกอบของโครงสร้างโปรแกรมที่ถูกต้อง ซึ่งสามารถตอบคำถามได้ในหลายๆ เรื่อง เช่น รูปแบบของไวยากรณ์ที่ใช้ในการเขียนโปรแกรมในภาษานั้นๆ เป็นอย่างไร กลุ่มคำและสัญลักษณ์พื้นฐานที่โปรแกรมเมอร์ใช้ในการเขียนโปรแกรมที่โครงสร้างถูกต้องมีอะไรบ้าง

โครงสร้างที่เกี่ยวข้องกับวากยสัมพันธ์ของภาษาในยุคปัจจุบันจะกำหนดโดยใช้รูปแบบทางภาษาศาสตร์ที่เรียกว่า *context-free grammar* ซึ่งกล่าวถึงในบทที่ 9

ชื่อและชนิด

คำศัพท์ของภาษาโปรแกรมจะรวมถึงกฎของการตั้งชื่อต่างๆ ด้วย เช่น ตัวแปร ฟังก์ชั่น คลาส พารามิเตอร์ ชื่อต่างๆ เหล่านี้ยังเกี่ยวเนื่องถึงคุณสมบัติต่างๆ ตลอดช่วงชีวิตของโปรแกรมอีกด้วย เช่น ขอบเขต (scope) การมองเห็น (visibility) และการเชื่อมโยง (binding)

ชนิดในที่นี้หมายถึงชนิดของข้อมูลหรือค่าที่โปรแกรมสามารถจัดการได้ เช่น ชนิดข้อมูลพื้นฐาน ชนิดข้อมูลแบบมีโครงสร้าง และชนิดข้อมูลที่มีโครงสร้างซับซ้อน ชนิดข้อมูลพื้นฐานได้แก่ จำนวนเต็ม จำนวนจริง ตัวอักษณ ค่าบูลลีน ชนิดข้อมูลแบบมีโครงสร้างประกอบด้วยสตริง ลิสต์ ทรี และตารางแฮช โครงสร้างข้อมูลที่ซับซ้อนประกอบด้วยฟังก์ชันและคลาส ซึ่งเราจะศึกษาเกี่ยวชื่อและชนิดข้อมูลนี้ในบทที่ 10

ความหมาย

ความหมายของโปรแกรมจะกำหนดโดย semantics นั่นคือ เมื่อรันโปรแกรม ผลกระทบของแต่ละคำสั่งที่เกิดกับค่าของตัวแปรในโปแรกรมจะถูกกำหนดโดยความหมายของภาษา ดังนั้น เมื่อเราเขียนโปรแกรม เราจะต้องเข้าใจแนวคิดพื้นฐานก่อน เช่นว่าเมื่อเราใช้คำสั่งในการกำหนดค่าจะเกิดผลอะไรขึ้นกับตัวแปร ถ้าเรามี semantic model ที่ไม่ขึ้นกับแพลตฟอร์มใดๆ เราก็จะสามารถนำ model นี้ไปใช้ได้กับเครื่องที่แตกต่างกันได้ การศึกษาในหัวข้อนี้จะทำให้ทราบถึงการทำงานของตัวแปลภาษา การเชื่อมโยงระหว่างตัวแปลภาษากับข้อกำหนดทาง semantic ของภาษา ซึ่งการศึกษาในหัวข้อนี้โดยในหัวข้อนี้เราจะไม่ลงลึกถึงรายละเอียด จึงจะไม่ขอกล่าวไว้ในเอกสารนี้

8.2 รูปแบบของภาษาโปรแกรม (Paradigms)

คำว่า Paradigm โดยทั่วไปจะหมายถึงรูปแบบของความคิดที่ลักษณะเป็นไปในแนวทางเดียวกัน ดังนั้น paradigm ในทางการโปรแกรมนั้นจึงหมายถึง รูปแบบของวิธีการคิดในการแก้ปัญหาภายใต้ลักษณะเฉพาะของโปรแกรมและภาษา รูปแบบของภาษาโปรแกรมสามารถแบ่งออกได้เป็น 4 รูปแบบหลัก คือ

- ภาษาเชิงคำสั่ง (Imperative programming)
- ภาษาเชิงวัตถุ (Object-oriented programming - OOP)

- ภาษาเชิงฟังก์ชัน (Functional programming)
- ภาษาเชิงตรรกะ (Logic programming)

ภาษาโปรแกรมบางภาษาออกแบบมาเพื่อรองรับมากกว่าหนึ่งรูปแบบ ตัวอย่างเช่น ภาษา C++ ซึ่งเป็นทั้งภาษาเชิงคำสั่งและภาษาเชิงวัตถุ หรือภาษาเชิงทดลองอย่างเช่น ภาษา Leda ก็ออกแบบมาให้รองรับทุกรูปแบบ ในขณะที่ภาษาโปรแกรมในยุคแรกๆ เช่น PL/I หรือ Algol 68 หรือ Ada มักออกแบบโดยรองรับเพียงรูปแบบเดียวเพื่อการใช้งานทั่วไป

ภาษาเชิงคำสั่ง

ภาษาเชิงคำสั่งเป็นรูปแบบที่เก่าแก่ที่สุด เนื่องจากมีรากฐานมาจากคอมพิวเตอร์ที่มีสถาปัตยกรรมแบบ Von Neumann ที่ออกแบบโดยมีลักษณะที่ต้องเก็บโปรแกรมและตัวแปรไว้ในหน่วยความจำ และโปรแกรมจะประกอบด้วยชุดคำสั่งที่ทำการคำนวณ กำหนดค่าให้กับตัวแปล รับข้อมูลเข้า แสดงผล รวมทั้งการควบคุมลำดับการทำงานของชุดคำสั่ง

การสร้างกลุ่มคำสั่ง (block) โดยใช้ Procedural abstraction เป็นสิ่งจำเป็นสำหรับภาษาเชิงคำสั่ง ซึ่งประกอบด้วย

- คำสั่งกำหนดค่า (Assignment)
- คำสั่งวนซ้ำ (Loop)
- คำสั่งทำงานตามลำดับ (Sequence)
- คำสั่งเงื่อนไข (Conditional statement) และ
- คำสั่งในการจัดการความผิดปกติของโปรแกรม (Exception handling หรือสิ่งที่ไม่คาดว่าจะเกิดขึ้นในโปรแกรม)

ตัวอย่างภาษาเชิงคำสั่งที่เด่นๆ ได้แก่ ภาษาโคบอล (Cobol) ฟอร์แทรน (Fortran) ซี (C) เอดา (Ada) และเพิร์ล (Perl)

ภาษาเชิงวัตถุ

ภาษาเชิงวัตถุจะกำหนดรูปแบบของโปรแกรมในลักษณะของกลุ่มของวัตถุที่มีการติดต่อสื่อสารกันโดยการส่ง message เพื่อเปลี่ยนสถานะ การส่ง message นี้ทำให้ข้อมูลของวัตถุมีลักษณะเป็นแบบ active แทนที่จะเป็นแบบ passive ลักษณะเช่นนี้เป็นสิ่งที่ทำให้ภาษาเชิงวัตถุแตกต่างจากภาษาเชิงคำสั่งอย่างชัดเจน การสร้างกลุ่มคำสั่งจะอาศัยหลักการพื้นฐานดังต่อไปนี้

- การแบ่งแยกวัตถุ (Object classification)
- การสืบทอดคุณสมบัติ (Inheritance)
- การส่งผ่านข้อความ (Message passing)

ตัวอย่างภาษาเชิงวัตถุ ได้แก่ ภาษาสมอลทอล์ก (Smalltalk) ซีพลัสพลัส (C++) จาวา (Java) และซีชาร์ป (C#)

ภาษาเชิงฟังก์ชัน

เป็นภาษาที่จำลองปัญหาทางคอมพิวเตอร์ในรูปของฟังก์ชันทางคณิตศาสตร์ ซึ่งประกอบด้วยข้อมูลเข้า (หรือ domain) และผลลัพธ์ (หรือ range) ภาษาโปรแกรมในกลุ่มนี้แตกต่างจากภาษาที่ใช้คำสั่งกำหนดค่า ตัวอย่างเช่น คำสั่งกำหนดค่า x = x+1 เป็นคำสั่งที่ไม่สมเหตุสมผลทางคณิตศาสตร์และในภาษาเชิงฟังก์ชันด้วย ฟังก์ชันจะมีการสัมพันธ์และรวมกับฟังก์ชันอื่นโดยการใช้

- การประกอบฟังก์ชัน (functional composition)
- การใช้เงื่อนไข (conditional)
- การวนซ้ำ (recursion)

ตัวอย่างภาษาเชิงฟังก์ชันที่สำคัญ ได้แก่ ภาษาลิสพ์ (Lisp) สคีม (Scheme) แฮสเคล (Haskell) และภาษาเอ็มแอล (ML)

ภาษาเชิงตรรกะ

เป็นภาษาเชิงประกาศ (declarative language) ที่แก้ปัญหาโดยประกาศผลของโปรแกรมที่น่าจะได้รับแทนที่จะระบุว่าทำอย่างไรจึงจะได้ผลลัพธ์ บางครั้งภาษากลุ่มนี้อาจเรียกว่าเป็นภาษา rule-based language เนื่องจากโปรแกรมมีลักษณะคล้ายกับเซตของกฎ หรือข้อบังคับของปัญหาแทนที่จะเป็นลำดับการทำงานของคำสั่ง

การแปลความการประกาศของโปรแกรมเชิงตรรกะ จะสร้างเซตของการแก้ปัญหาที่เป็นไปได้ทั้งหมดตามที่โปรแกรมระบุไว้ ภาษาเชิงตรรกะเหมาะสำหรับการแก้ปัญหาที่มีรายละเอียดไม่สมบูรณ์

ตัวอย่างภาษาเชิงตรรกะที่สำคัญ ได้แก่ ภาษาโปรล็อก (Prolog)

8.3 ประวัติความเป็นมา

คอมพิวเตอร์ในยุคแรกควบคุมแบบ Hardwired ซึ่งเป็นการควบคุมการทำงานของฮาร์ดแวร์โดยใช้ฮาร์ดแวร์ ทำให้ไม่มีความยืดหยุ่นในการใช้งาน ดังนั้นจึงมีแนวคิดในการสร้างคอมพิวเตอร์โดยใช้ชุดของคำสั่งในการควบคุมการทำงาน จึงเกิดการพัฒนาภาษาโปรแกรมในยุคแรก คือ ภาษาเครื่องและภาษาแอสเซมบลีที่เริ่มต้นใช้เมื่อประมาณทศวรรษที่ 1940 นับตั้งแต่นั้นเป็นต้นมา ก็มีการพัฒนาภาษาโปรแกรมหลายร้อยภาษารวมถึงภาษาย่อย (dialect) ต่างๆ ด้วย ภาษาโดยมากจะมีช่วงอายุการใช้งานที่จำกัด แต่ก็ยังมีบางภาษาที่ยังคงได้รับความนิยมแพร่หลายมาจนทุกวันนี้ หลายภาษาเป็นส่วนสำคัญที่มีอิทธิพลต่อการออกแบบภาษาใหม่ในอนาคต สิ่งที่ช่วยกระตุ้นให้เกิดการพัฒนาภาษาโปรแกรมในช่วงหลายทศวรรษที่ผ่านมานี้ก็คือวิวัฒนาการของคอมพิวเตอร์ที่มีประสิทธิภาพดีขึ้น และการประยุกต์ใช้โดยผู้ใช้งานในหลากหลายสาขา ซึ่งสาขาที่มีอิทธิพลอย่างมากต่อภาษาโปรแกรมมีดังนี้

- ปัญญาประดิษฐ์
- การศึกษา
- วิทยาศาสตร์และวิศวกรรมศาสตร์
- เทคโนโลยีสารสนเทศ
- ระบบและเครือข่าย
- เวิร์ลไวด์เว็บ

ภาษาโปรแกรมอาจจำแนกออกตามลักษณะหรือยุคของการพัฒนาภาษาได้เป็น 3 ยุคหลัก ดังต่อไปนี้

ภาษาเครื่อง (Machine language)

จัดเป็นภาษาในยุคที่ 1 ซึ่งถือกำเนิดมาพร้อมกับคอมพิวเตอร์ เป็นภาษาที่เครื่องสามารถเข้าใจได้โดยตรง ภาษาเครื่องนี้ยังเป็นที่มาของคำว่า *โค้ด (code)* ในปัจจุบันอีกด้วย ซึ่งหมายถึงข้อความที่ประกอบขึ้นเป็นโปรแกรมในภาษาใดๆ ภาษาเครื่องจัดเป็นภาษาระดับต่ำ (Low level language) เนื่องจากทำความเข้าใจได้ยาก จึงไม่เป็นที่นิยมใช้ในปัจจุบัน แต่มีข้อดีคือทำงานได้เร็ว

ภาษาสัญลักษณ์ (Symbolic language)

เนื่องจากภาษาเครื่องเข้าใจได้ยาก และภาษาอังกฤษใช้คำมากเกินกว่าจะที่ใช้ในการแสดงความหมาย ภาษาแอสเซมบลีจึงได้ถือกำเนิดขึ้น โดยใช้สัญลักษณ์และชื่อแทนตัวเลข ทำให้เข้าใจได้ง่ายขึ้นภาษาเครื่องจัดเป็นภาษาระดับต่ำ แต่ยังมีข้อเสียคือเป็นภาษาที่ขึ้นอยู่กับเครื่อง การใช้งานโปรแกรมจะต้องทำการแปลจากภาษาสัญลักษณ์ให้เป็นภาษาเครื่องก่อน โดยใช้ตัวแปลภาษา คือ แอสเซมเบอร์

ภาษาระดับสูง (High level language)

เป็นภาษาที่ใช้คำและสัญลักษณ์ที่คุ้นเคย อ่านแล้วเข้าใจได้ง่าย มีความใกล้เคียงกับภาษาอังกฤษมากขึ้น มีจุดเด่นคือ ไม่ขึ้นอยู่กับเครื่อง นั่นคือเราสามารถนำโปรแกรมไปรันบนเครื่องที่ต่างกันได้โดยทำการแก้ไขโปรแกรมเพียงเล็กน้อยหรือไม่ต้องแก้ไขเลย ผู้เขียนโปรแกรมไม่จำเป็นต้องเข้าการทำงานภายในคอมพิวเตอร์มากนัก โดยมากภาษาระดับสูงนี้จะมีชุดคำสั่งพื้นฐานให้ใช้งาน (Program libraries) ทำให้สะดวกในการเขียนโปรแกรม การใช้งานโปรแกรมจะต้องทำการแปลให้เป็นภาษาเครื่องก่อน โดยใช้ตัวแปลภาษาคือ คอมไพเลอร์ (Compiler) หรืออินเทอร์พลีเตอร์ (Interpreter)

ภาษาระดับสูงนี้อาจเป็นออกได้เป็น 3 ยุคย่อยๆ ได้แก่

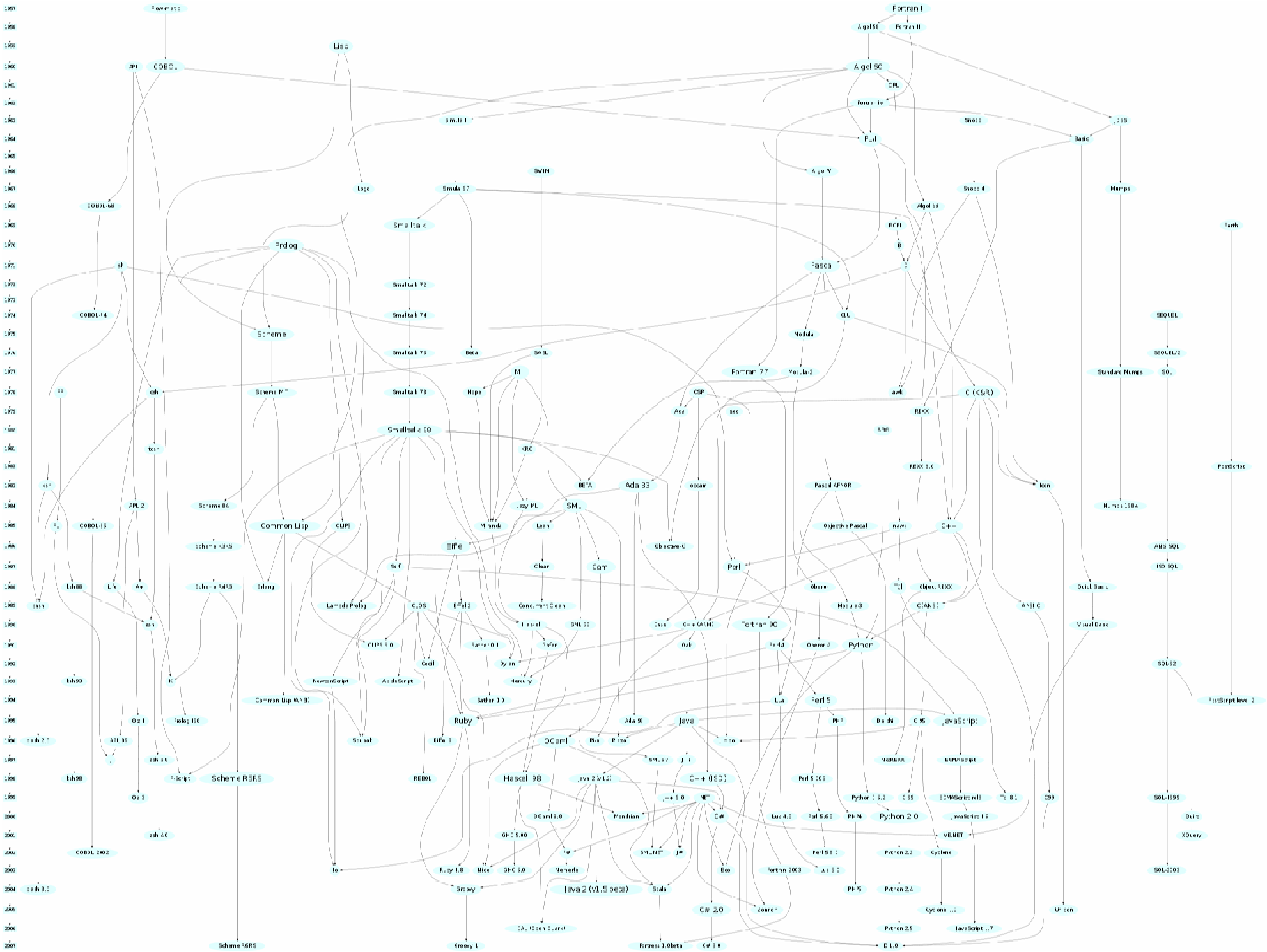
- ภาษายุคที่ 3 (3GL – Third General Language) เป็นภาษาระดับสูงที่มีลักษณะที่ต้องระบุขั้นตอนการทำงานให้แก่คอมพิวเตอร์ ภาษาในกลุ่มนี้ ได้แก่ ภาษาโปรแกรมส่วนใหญ่ในปัจจุบัน
- ภาษายุคที่ 4 (4GL – Fourth General Language) เป็นภาษาที่ไม่จำเป็นต้องระบุวิธีการ ขั้นตอนการทำงานในโปรแกรม แต่บอกว่าต้องการอะไร ทำให้พัฒนาได้เร็ว ใช้งานและบำรุงรักษาง่าย ภาษาในกลุ่มนี้มักจะเป็นโปรแกรมที่รวบรวมการจัดการฐานข้อมูลด้วย
- ภาษายุคที่ 5 (5GL – Fifth General Language) หรือภาษาธรรมชาติ เป็นภาษาที่ใช้ภาษาพูดหรือเขียนของมนุษย์ในการสั่งงาน ใช้ความรู้ทางปัญญาประดิษฐ์ในการสร้างภาษา ภาษาในกลุ่มนี้กำลังได้รับความสนใจและอยู่ในระหว่างการวิจัยและพัฒนา

8.4 เอกสารอ้างอิงและเว็บไซต์ที่ควรรู้

Allen B. Tucker and Robert E. Noonan. Programming Languages – Principles and Paradigms.

สุจิตรา อดุลย์เกษม. เอกสารประกอบการสอนองค์ประกอบคอมพิวเตอร์และภาษา.

http://dariosantarelli.wordpress.com/2008/06/22/programming-languages-history/


1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
Fortran I
Algol 58
Fortran II
Algol 60
CPL
Fortran IV
Simula I
Snobo
JOSS
PL/I
Basic
Algol W
SWIM
Logo
Simula 67
Snobol4
Mumps
Algol 68
Smalltalk
BCPL
B
C
Prolog
Pascal
Smalltalk 72
Smalltalk 74
CLU
SEQUEL
Scheme
Modula
Smalltalk 76
Beta
SNOBOL4
SEQUEL/2
Standard Numps
SQL
M
Fortran 77
Modula-2
Smalltalk 78
Hope
CSP
awk
C (K&R)
Ada
sed
REXX
Smalltalk 80
ARC
KRC
REXX 3.0
PostScript
BETA
Ada 83
occam
Pascal APHOR
Icon
APL 2
Scheme 84
Common Lisp
CLIPS
Miranda
Lazy ML
SML
Lean
Numps 1984
Objective Pascal
nawk
C++
Eiffel
Objective-C
ANSI SQL
Clear
Self
Coml
Perl
ISO SQL
Erlang
Oberon
Tcl
Object REXX
Quick Basic
Lambda Prolog
CLOS
Eiffel 2
Concurrent Clean
Modula 3
C (ANS)
ANSI C
Haskell
SML 90
Dylan
Gofer
Mercury
Newton Script
AppleScript
Common Lisp (ANSI)
Sather 1.0
Lua
Perl 5
Ruby
Ada 95
Java
Squeak
Eiffel 3
OCaml
Pike
Pizza
Limbo
SML 97
REBOL
Hascell 98
Java 2 (v1.2)
C++ (ISO)
Perl 5.005
Mondrian
Java 6.0
NET
C#
Luz 4.0
Perl 5.6.0
PHP4
Python 2.0
VB.NET
GHC 5.00
SML.NET
J#
Perl 5.8.0
Python 2.2
Cyclone
Ruby 1.8
Nice
GHC 6.0
Nemerle
Boo
Fortran 2003
Lua 5.0
PHP5
Python 2.4
Groovy
Java 2 (v1.5 beta)
Scala
C# 2.0
Zonon
Cyclone 1.0
CAL (Open Quark)
Python 2.5
JavaScript 1.7
Fortress 1.0beta
C# 3.0
D 1.0
Fortran
Forth
Fortran 95
Lisp
API
COBOL
COBOL-68
COBOL-74
COBOL-85
COBOL 2002
sh
csh
tcsh
ksh
bash
ksh88
ksh93
ksh98
bash 2.0
bash 3.0
APL 36
Oz 1
Oz 3
Prolog ISO
F-Script
Scheme R5RS
Scheme R6RS
Scheme M*
Scheme 84
Scheme 4RKS
Scheme 3RKS
Scheme 4RS
Scheme R6RS
Crowy 1
Perl 6
Perl 5.8.8
Perl 5.10.0
Perl 6.0.1
Python 1.5.2
Python 3.0
Python 3.1
Python 3.2
Python 3.3
Python 3.4
Python 3.5
Python 3.6
Python 3.7
Python 3.8
Python 3.9
Python 3.10
Python 3.11
Python 3.12
Python 3.13
Python 3.14
Python 3.15
Python 3.16
Python 3.17
Python 3.18
Python 3.19
Python 3.20
Python 3.21
Python 3.22
Python 3.23
Python 3.24
Python 3.25
Python 3.26
Python 3.27
Python 3.28
Python 3.29
Python