บทที่ 9

ภาษาที่ใช้เขียนโปรแกรม

Programming Languages

วัตถุประสงค์

หลังจากเรียนจบบทที่ 9 แล้วนักศึกษาต้องสามารถ:

- เข้าใจวิวัฒนาการของภาษาคอมพิวเตอร์
- อธิบายความแตกต่างระหว่างภาษาเครื่อง ภาษาแอสแซมบลี และภาษาระดับสูง
- เข้าใจกระบวนการสร้างและรันโปรแกรม
- อธิบายความแตกต่างระหว่างประเภทของภาษา: procedural, object-oriented, functional, declarative, and special.
- เข้าใจภาษา C เบื้องต้น

9.1

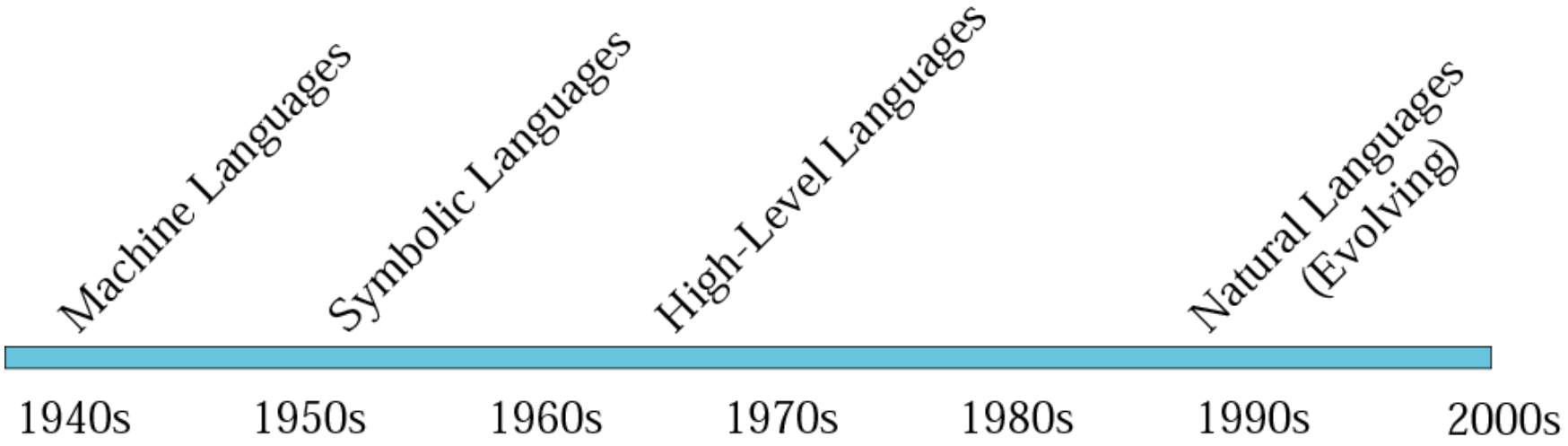
วิวัฒนาการของภาษาคอมพิวเตอร์

ในการเขียนโปรแกรมเพื่อสั่งงานให้คอมพิวเตอร์ทำงานตามที่ต้องการนั้น นักเขียนโปรแกรมจะต้องใช้ภาษาคอมพิวเตอร์ ซึ่งเป็นชุดของคำสั่งที่จัดเรียงอย่างมีระบบและถูกต้องตามหลักไวยากรณ์ของภาษา ตลอดระยะเวลาที่ผ่านมา ภาษาคอมพิวเตอร์ได้มีวิวัฒนาการมาอย่างต่อเนื่องจาก ภาษาเครื่อง (machine language) ไปสู่ภาษาธรรมชาติ (natural language) ตามรายละเอียดในรูปที่ 9.1

9.1.1 ภาษาเครื่อง (machine languages): ในระยะเริ่มแรกของการพัฒนาคอมพิวเตอร์ ภาษาเดียวที่มีคือภาษาเครื่อง คอมพิวเตอร์แต่ละเครื่องจะมีภาษาของตนเองซึ่งประ กอบขึ้นด้วยเลข 0 และ 1 ตัวอย่างโปรแกรม 9.1 เป็นโปรแกรมที่ทำการคูณเลข 2 จำนวนแล้วทำการพิมพ์ผลลัพธ์ คำสั่งของภาษาเครื่องประกอบด้วย 0 กับ 1 เพราะวงจรภายในของคอมพิวเตอร์ทำจาก สวิทช์ ทรานซิสเตอร์ และ อุปกรณ์อิเล็กทรอนิคส์อื่นๆที่มีสถานะเป็น ‘on’ กับ ‘off’ โดยที่ ‘off’ แทนด้วย 0 และ ‘on’ แทนด้วย 1

รูปที่ 9-1

วิวัฒนาการของภาษาคอมพิวเตอร์



Program 9.1 Program in machine language

```
1   00000000 00000100 0000000000000000
2   01011110 00001100 11000010 00000000 00000010
3                     11101111 00010110 00000000 00000101
4                     11101111 10011110 00000000 00001011
5   11111000 10101101           11011111 00000000 00010010
6                     01100010 11011111 00000000 00010101
7   11101111 00000010           11111011 00000000 00010111
8   11110100 10101101 11011111  00000000 00011110
9   00000011 10100010           11011111 00000000 00100001
10  11101111 00000010           11111011 00000000 00100100
11  01111110 11110100 10101101
12  11111000 10101110 11000101  00000000 00101011
13  00000110 10100010           11111011 00000000 00110001
14  11101111 00000010           11111011 00000000 00110100
15                              00000100 0000000000111101
16                              00000100 0000000000111101
```

The only language understood by a computer is machine language.

9.1.2 ภาษาสัญลักษณ์ (Symbolic Languages): เนื่องจากภาษาเครื่องมีความยุ่งยากในการเขียนและการใช้ จึงทำให้ในต้นปี ค.ศ. 1950 (พ.ศ. 2493) นักคณิตศาสตร์ชาวอเมริกันได้พัฒนาภาษาใหม่ที่ยึดแนวมาจากภาษาเครื่อง โดยใช้ symbols (สัญลักษณ์) หรือ mnemonics (สิ่งช่วยความจำ) แทนคำสั่งภาษาเครื่อง เนื่องจากภาษาใหม่ใช้ symbols แทนคำสั่ง ภาษาใหม่จึงถูกเรียกว่า symbolic languages โปรแกรม 9.2 ทำการคูณเลข 2 จำนวนแล้วพิมพ์ผลลัพธ์ ถ้าเปรียบเทียบกับโปรแกรมภาษาเครื่องในโปรกรม 9.1 จะเห็นว่าโปรแกรมใหม่พอที่จะอ่านเข้าใจได้ง่ายกว่า

โปรแกรมเฉพาะที่เรียกว่า assembler ใช้สำหรับแปล symbolic language ให้เป็นภาษาเครื่อง และเนื่องจาก symbolic language จะต้องแปลและรวบรวมเข้าด้วยกัน (assemble) เป็นภาษาเครื่อง จึงทำให้ต่อมาได้ชื่อว่า assembly languages และชื่อนี้ก็ยังใช้มาจนถึงปัจจุบัน

Program 9.2 Program in symbolic language

```
1   Entry    main,    ^m<r2>
2   subl2    #12,sp
3   jsb      C$MAIN_ARGS
4   movab    $CHAR_STRING_CON
5
6   pushal   -8(fp)
7   pushal   (r2)
8   calls    #2,read
9   pushal   -12(fp)
10  pushal   3(r2)
11  calls    #2,read
12  mull3    -8(fp),-12(fp),-
13  pushal   6(r2)
14  calls    #2,print
15  clrl     r0
16  ret
```

9.1.3 ภาษาระดับสูง (High-Level Languages): ถึงแม้ว่าภาษาสัญลักษณ์จะช่วยให้การเขียนโปรแกรมง่ายขึ้นระดับหนึ่ง แต่นักเขียนโปรแกรมก็ยังจะต้องใช้คำสั่งที่เกี่ยวข้องกับฮาร์ดแวร์อยู่ นอกจากนี้การเขียนภาษาสัญลักษณ์ยังจะต้องเข้ารหัสทุกๆคำสั่งอย่างละเอียด ทำให้การเขียนโปรแกรมเป็นงานที่ยุ่งยากและน่าเบื่อหน่าย นักเขียนโปรแกรมควรไปใช้เวลากับการแก้ปัญหาโดยตรงมากกว่าที่จะต้องเสียเวลาในการเขียนโปรแกรม ด้วยเหตุนี้จึงทำให้มีการพัฒนาภาษาระดับสูงขึ้น

ภาษาระดับสูงสามารถใช้กับเครื่องคอมพิวเตอร์ที่แตกต่างกันได้ ทำให้นักเขียนโปรแกรมมีเวลาให้กับการแก้ปัญหามากขึ้น ไม่ต้องมากังวลกับการติดต่อกับเครื่องคอมพิวเตอร์มากนัก ทำให้นักเขียนโปรแกรมไม่ต้องลงไปในรายละเอียดในการเขียนโปรแกรมภาษาสัญลักษณ์อีก แต่ภาษาระดับสูงก็ยังต้องถูกแปลไปเป็นภาษาเครื่องอยู่ดี ตัวแปลภาษาดังกล่าวอาจเป็น คอมไพเลอร์ (compiler) หรือ อินเตอร์พรีตเตอร์(interpreter)

Program 9.3 Program in C++ language

```
1   /*     This program reads two integer numbers from the
2          keyboard and prints their product.
3   */
4   #include <iostream.h>
5
6   int main (void)
7   {
8   //     Local Declarations
9          int number1;
10         int number2;
11         int result;
12  //      Statements
13         cin >> number1;
14         cin >> number2;
15         result = number1 * number2;
16         cout << result;
17         return 0;
18  }      // main
```

9.1.4 ภาษาธรรมชาติ (Natural Languages): คอมพิวเตอร์ในอุดมคติของมนุษย์คือคอมพิวเตอร์ที่เราสามารถใช้ภาษาธรรมชาติ หรือภาษาที่มนุษย์ใช้สื่อสารกันเพื่อสื่อสารกับคอมพิวเตอร์ได้ คอมพิวเตอร์ต้องเข้าใจภาษามนุษย์และต้องตอบสนองทันที ภาษาธรรมชาติเช่นภาษาไทย ภาษาอังกฤษ ภาษาจีน ภาษาญี่ปุ่น เป็นต้น ถึงแม้ว่าจะเป็นเรื่องที่ไม่น่าจะเป็นไปได้ แต่ในปัจจุบันได้มีการวิจัย ทดลอง และปฏิบัติการกันอยู่ในห้องปฏิบัติการ มีการนำผลการวิจัยมาใช้บ้าง แต่ยังอยู่ในขอบเขตจำกัด เชื่อว่าอีกไม่ช้าไม่นาน เราอาจจะมีคอมพิวเตอร์ในอุดมคติได้ใช้กัน

9.2

การสร้างโปรแกรม

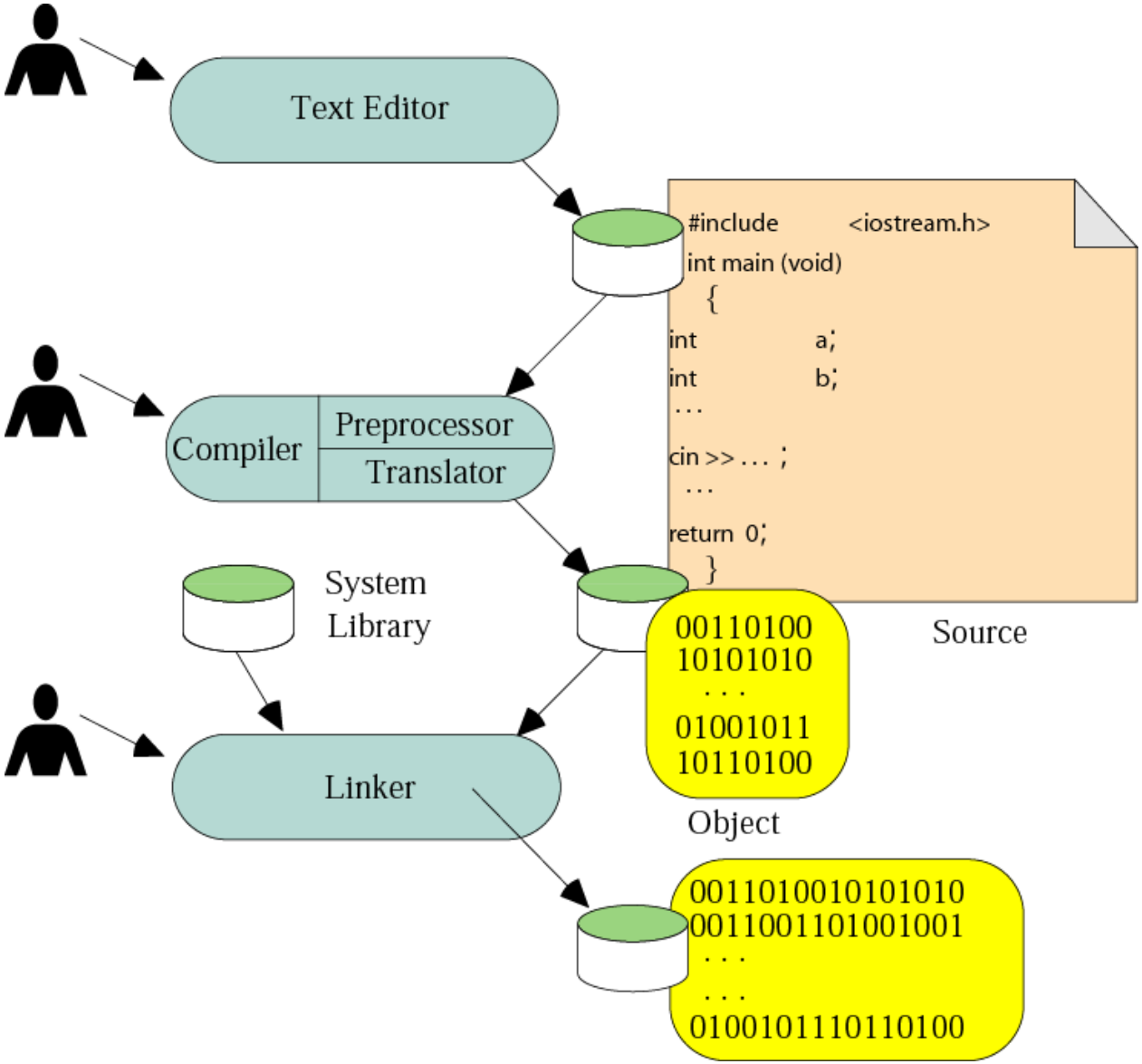
เราทราบจากตอนต้นว่าคอมพิวเตอร์จะเข้าใจโปรแกรมเฉพาะโปรแกรมภาษาเครื่องเท่านั้น ในหัวข้อนี้จะอธิบายถึงวิธีการในการแปลโปรแกรมภาษาระดับสูงไปเป็นภาษาเครื่อง กระบวนการนี้เป็นหน้าที่ของนักเขียนโปรแกรมที่จะต้องเขียนโปรแกรมแล้วเปลี่ยนให้เป็น **executable (ภาษาเครื่อง)** file ซึ่งมี 3 ขั้นตอนคือ

1. เขียนและแก้ไขโปรแกรม
2. ทำการคอมไพล์โปรแกรม
3. เชื่อมโยง (link) โปรแกรมกับโมดูลอื่นๆที่จำเป็น

รูปที่ 9.2 แสดงการทำงานในขั้นตอนทั้งสาม

รูปที่ 9-2

การสร้างโปรแกรม



9.2.1 การเขียนและการแก้ไขโปรแกรม: ซอฟท์แวร์ที่ใช้เขียน

โปรแกรมมีชื่อว่า text editor ซอฟท์แวร์ text editor ช่วยให้นักเขียนโปรแกรมสามารถสร้าง แก้ไข และเก็บโปรแกรมและข้อมูลได้โดยสะดวก บางครั้งเราอาจใช้ text editor ในการเขียนจดหมาย สร้างรายงาน หรือเขียนโปรแกรม หลัง จากที่เขียนโปรแกรมเสร็จเราจะเก็บโปรแกรมไว้ในแผ่นจานแม่เหล็กในรูปของแฟ้มข้อมูล ซึ่งแฟ้มข้อมูลนี้จะเป็นอินพุทของคอมไพเลอร์ในโอกาสต่อไป แฟ้มข้อมูลนี้เรียกว่า source file

9.2.2 การคอมไพล์โปรแกรม: โปรแกรมใน source file จะต้องถูกแปลไปเป็นภาษาเครื่องเพื่อที่จะให้คอมพิวเตอร์เข้าใจ โปรแกรมส่วนใหญ่จะใช้คอมไพเลอร์ในการแปล คอมไพเลอร์เองก็เป็นโปรแกรมที่ประกอบด้วย 2 โปรแกรมย่อยคือ preprocessor และ translator

9.2.3 การเชื่อมโยงโปรแกรม (Linking Program): เนื่องจากโปรแกรมหนึ่งๆประกอบด้วยโปรแกรมย่อยๆหลายโปรแกรม โปรแกรมย่อยเหล่านี้บางส่วนเราเขียนเอง บางส่วนก็นำมาจากที่อื่นเช่นโปรแกรมย่อยที่ทำหน้าที่เกี่ยวกับ I/O และโปรแกรมย่อยที่ทำหน้าที่ด้านการคำนวณ โปรแกรมต่างๆเหล่านี้จะต้องเชื่อมต่อเป็นโปรแกรมเดียวกันโดยซอฟท์แวร์ตัวหนึ่งชื่อ **Linker** เมื่อเชื่อมต่อกันแล้วจะได้โปรแกรมที่พร้อมที่ทำงานได้ เรียกโปรแกรมที่ได้นี้ว่า “**Executable Program**”

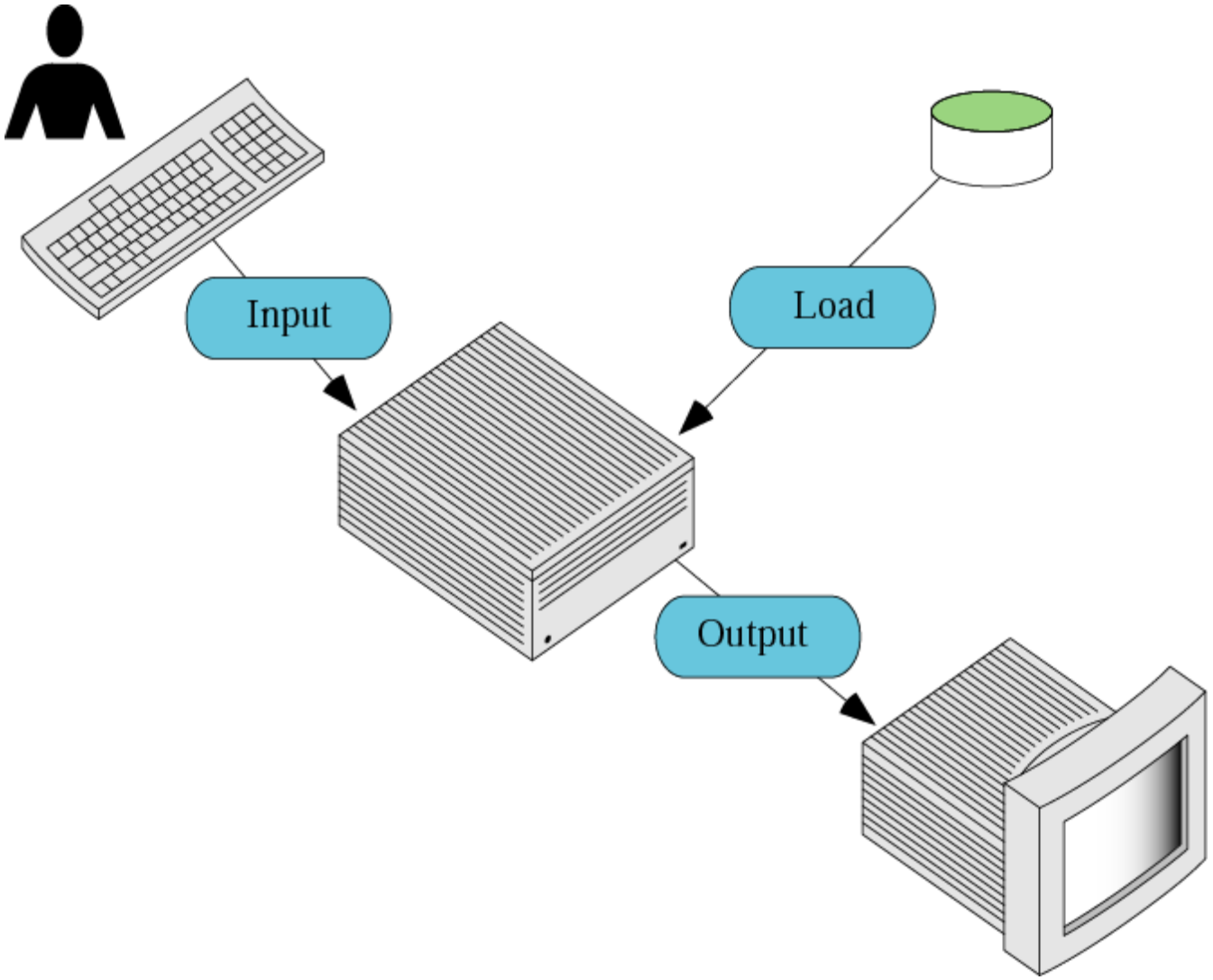
9.3

การทำงานของโปรแกรม

PROGRAM EXECUTION

รูปที่ 9-3

Program execution



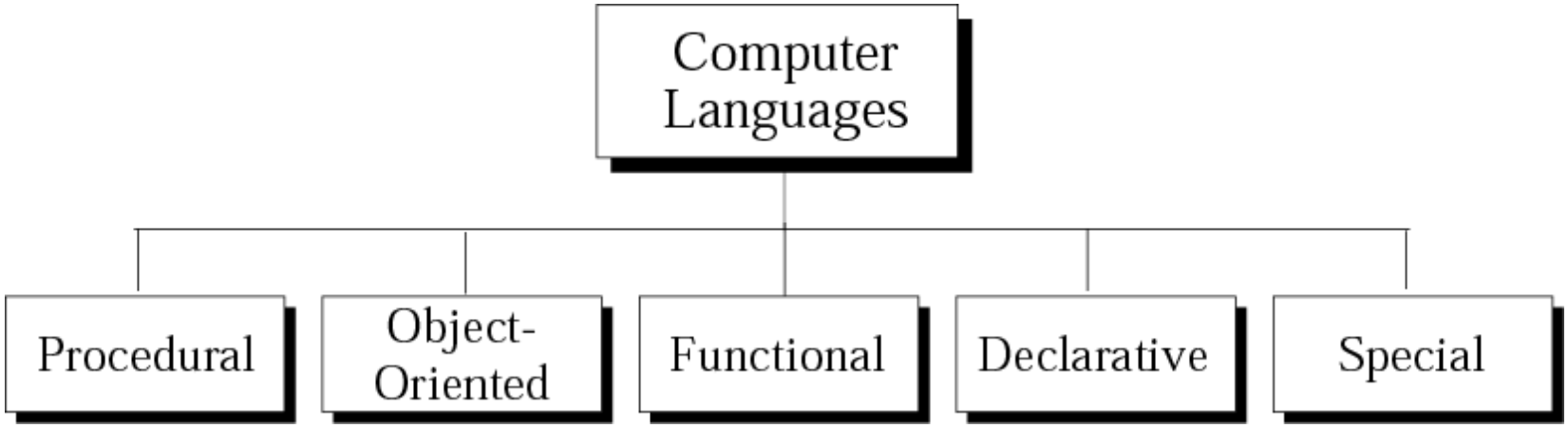
9.4

ประเภทของภาษาคอมพิวเตอร์

CATEGORIES OF LANGUAGES

รูปที่ 9-4

ประเภทของภาษาคอมพิวเตอร์



Functional Languages

- ในโปรแกรมประเภท functional programming นั้นโปรแกรมเปรียบเสมือนฟังก์ชันทางคณิตศาสตร์ คำว่าฟังก์ชันมองเหมือนกล่องดำที่รับอินพุท ทำการประมวลผล แล้วส่งผลลัพธ์ออกไป เช่นฟังก์ชันชื่อ summation มีอินพุทจำนวน n ตัว ทำการบวกเลขทั้งหมด แล้วสร้างผลลัพธ์ 1 ตัว เป็นต้น

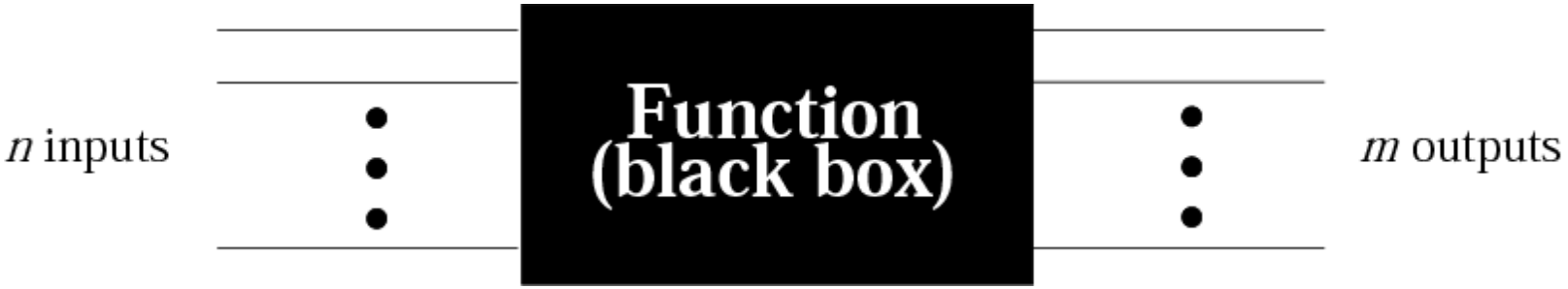
  ตัวอย่างภาษาประเภทนี้ได้แก่ ภาษา LISP รูปแบบการทำงานเช่น

  (car 6 7 8 9 3 5 7) = 6

  (cdr 6 7 8 9 3 5 7) = (7 8 9 3 5 7)

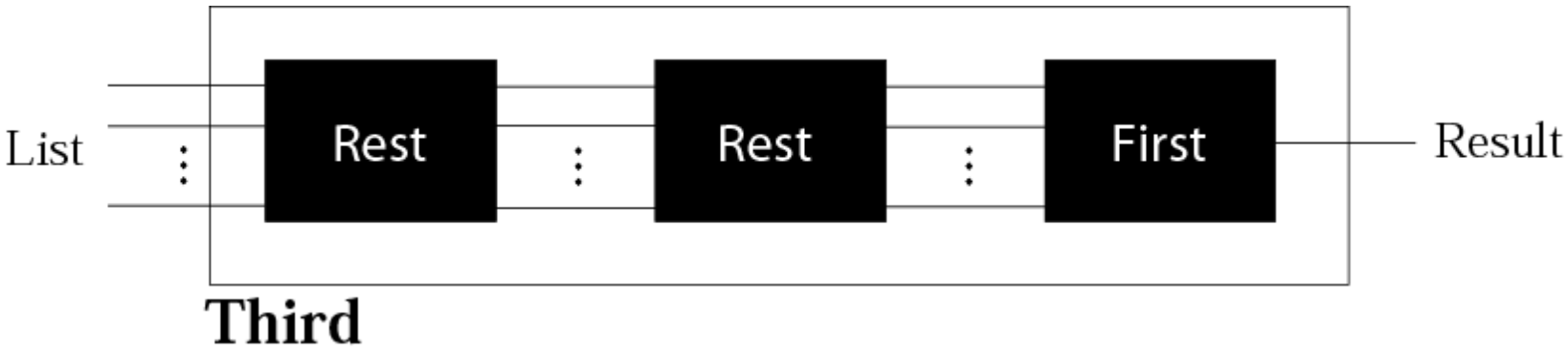
รูปที่ 9-5

Function in a functional language



รูปที่ 9-6

Extracting the third element of a list



Special Languages

- ภาษาพิเศษเป็นภาษาที่เกิดใหม่ที่ไม่สามารถจัดกลุ่มอยู่ในใดกลุ่มหนึ่งได้ เป็นภาษาที่สร้างขึ้นมาเพื่อประโยชน์ใช้สอยค่อนข้างเฉพาะด้าน ตัวอย่างภาษากลุ่มนี้เช่น

  * **HTML** (Hypertext Markup Language) เป็นภาษาที่ใช้งานด้านการกำหนดรูปแบบ (format) ของเอกสารโดยเฉพาะเอกสารบนอินเตอร์เนต

  * **PERL** (Practical Extraction and Report Language) เป็นภาษาระดับสูงที่มีไวยากรณ์คล้ายภาษา C แต่มีประสิทธิภาพกว่า

  * **SQL** (Structured Query Language) เป็นภาษาที่ใช้ค้นหาข้อมูลในฐานข้อมูลเพื่อตอบคำถามที่ต้องการทราบเกี่ยวกับข้อมูล

Table 9.1 Common tags

Beginning Tag	*Ending Tag*	*Meaning*
<HTML>	</HTML>	document
<HEAD>	</HEAD>	document head
<BODY>	</BODY>	document body
<TITLE>	</TITLE>	document title
<Hi>	</Hi>	different header levels
<B>	</B>	boldface
<I>	</I>	Italic
<U>	</U>	underlined
_		subscript
[	]	superscript
<CENTER>	</CENTER>	centered
 		line break
<OL>	</OL>	ordered list
<UL>	</UL>	unordered list
<LI>	</LI>	an item in the list
<IMG>		an image
<A>	</A>	an address (hyperlink)

Program 9.4 HTML Program

```
<HTML>
        <HEAD>
                <TITLE> Sample Document </TITLE>
        </HEAD>
        <BODY>
                This is the picture of a book:
                <IMG SRC="Pictures/book1.gif"   ALIGN=MIDDLE>
        </BODY>
</HTML>
```

9.5

PROCEDURAL LANGUAGE: C

องค์ประกอบของภาษา C

- ภาษา C เป็นภาษาที่ได้รับความนิยมอย่างสูงในวงการคอมพิวเตอร์ เป็นภาษาที่มีประสิทธิภาพสูง สามารถเข้าไปถึงรายละเอียดการifieกระทำระดับบิตในหน่วยความจำรีจิสเตอร์ (register) องค์ประกอบที่สำคัญของภาษา C มีดังนี้

  * IDENTIFIERS
  * DATA TYPES : integer, char, float
  * VARIABLES
  * CONSTANTS

รูปที่ 9-7

ตัวแปร

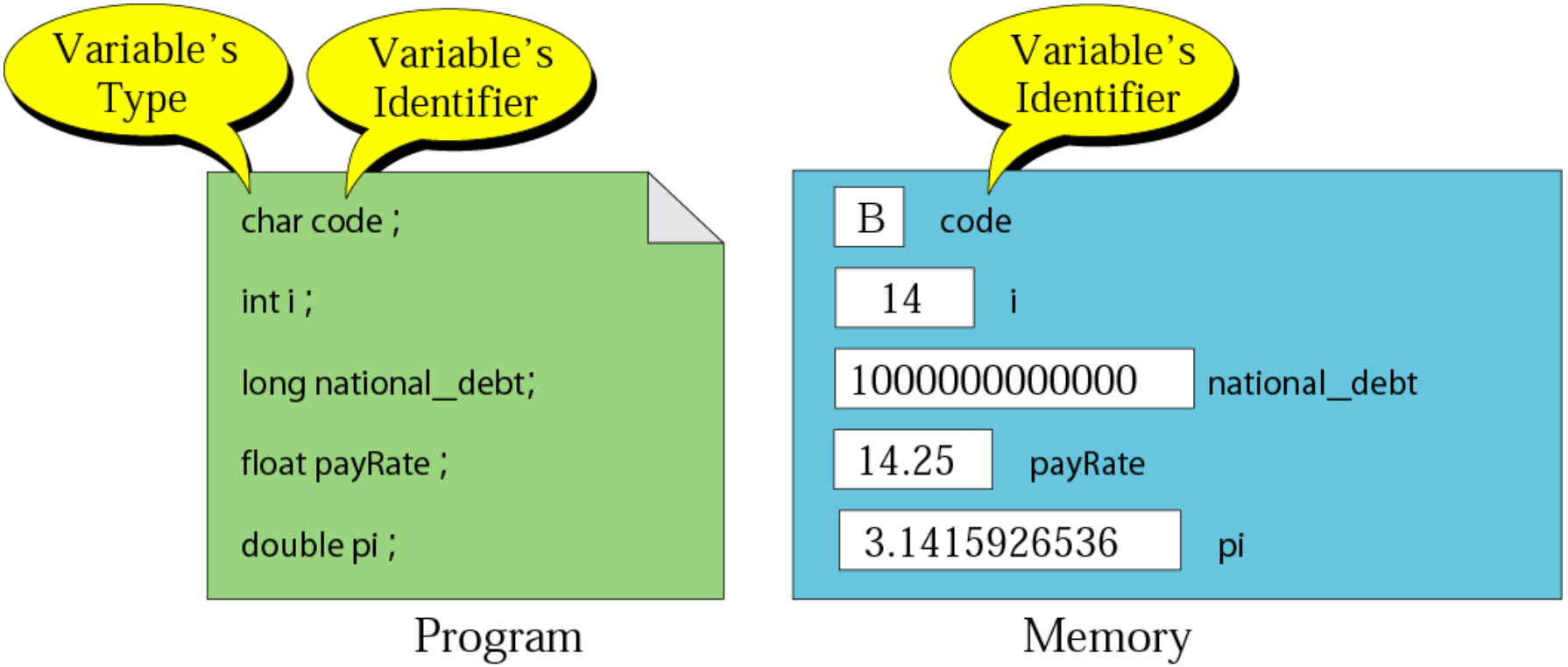


Table 9.2 Arithmetic operators

Operator	*Definition*	*Example*
+	**Addition**	3 + 5
–	**Subtraction**	2 – 4
*	**Multiplication**	Num * 5
/	**Division (quotient)**	Sum / Count
%	**Division (remainder)**	Count % 4
++	**Increment**	Count ++
––	**Decrement**	Count ––

Table 9.3 Relational operators

Operator	*Definition*	*Example*
<	Less than	Num1 < 5
<=	Less than or equal to	Num1 <= 5
>	Greater than	Num2 > 3
>=	Greater than or equal to	Num2 >= 3
==	Equal to	Num1 == Num2
!=	Not equal to	Num1 != Num2

Table 9.4 Logical operators

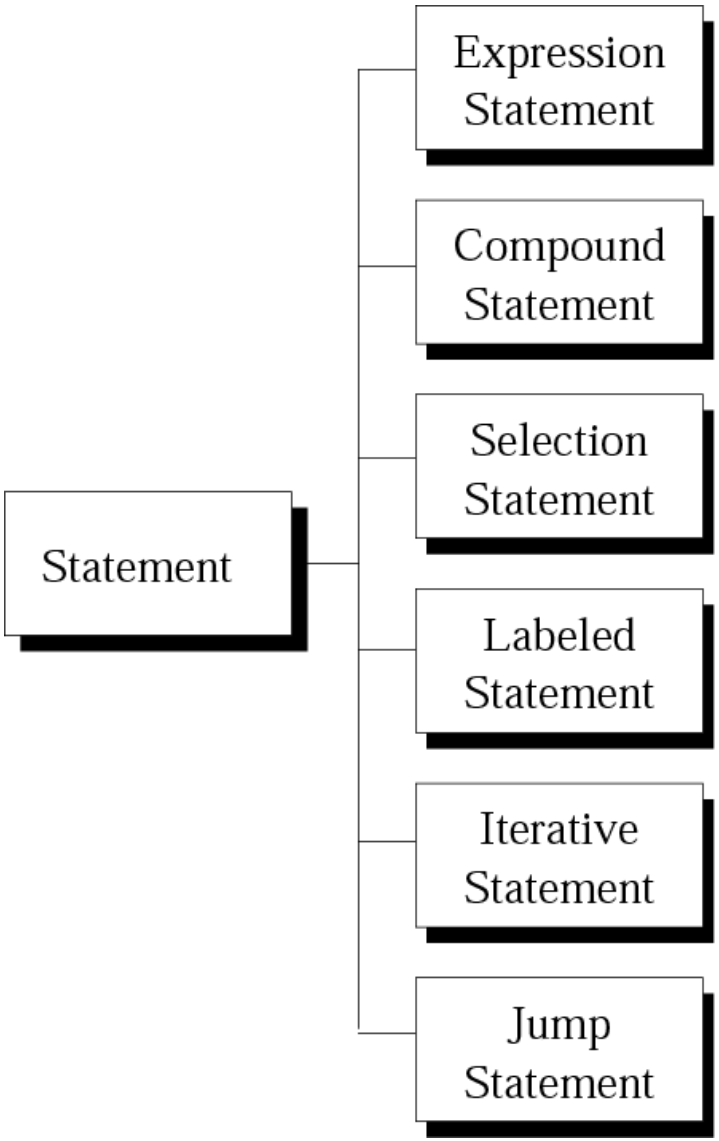
Operator	*Definition*	*Example*
!	**NOT**	! (Num1 < Num2)
&&	**AND**	(Num1 < 5) && (Num2 > 10)
‖	**OR**	(Num1 < 5) ‖ (Num2 > 10)

Table 9.5 Assignment operators

Operator	*Example*	Meaning
==	**Num = 5**	Store 5 in Num
+=	**Num += 5**	Num = Num + 5
–=	**Num –= 5**	Num = Num – 5
*=	**Num *= 5**	Num = Num * 5
/=	**Num /= 5**	Num = Num / 5
%=	**Num %= 5**	Num = Num % 5

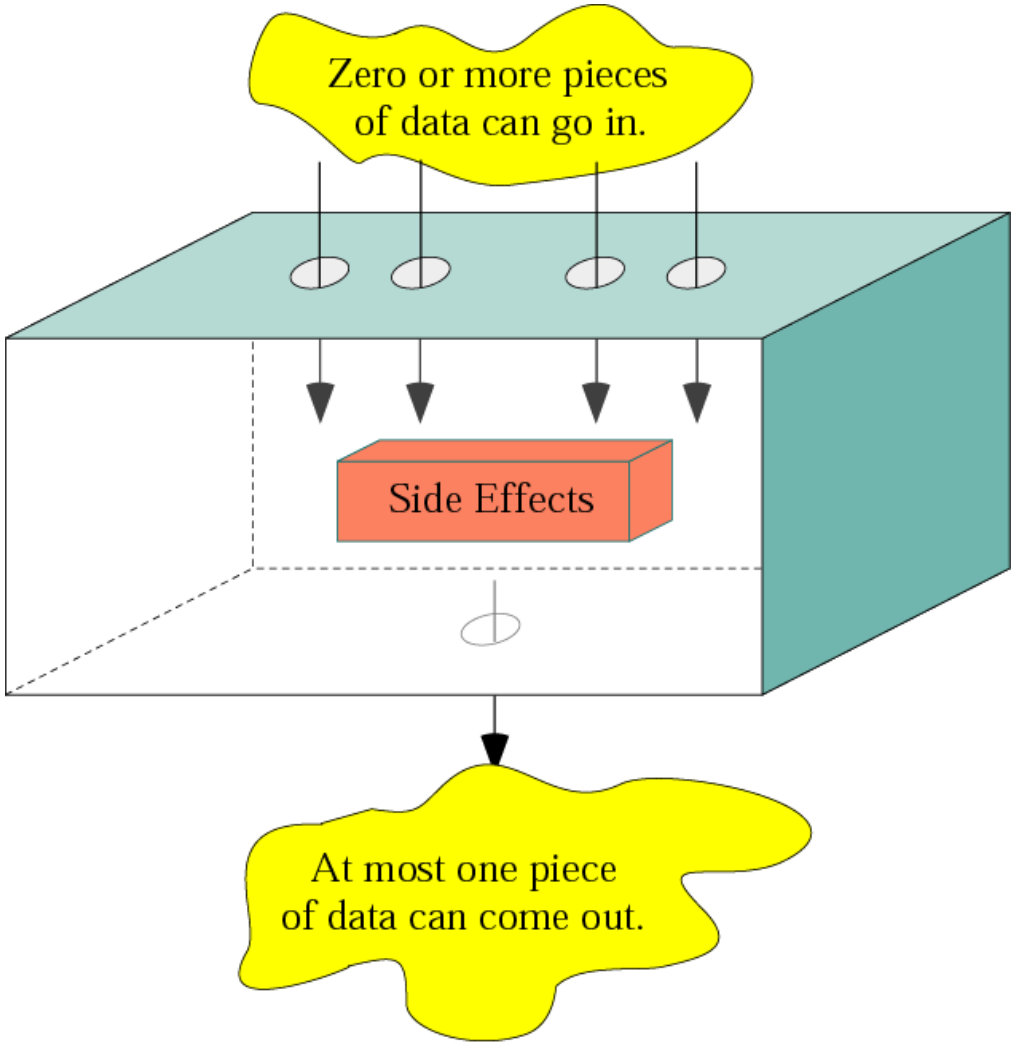
รูปที่ 9-8

Statements



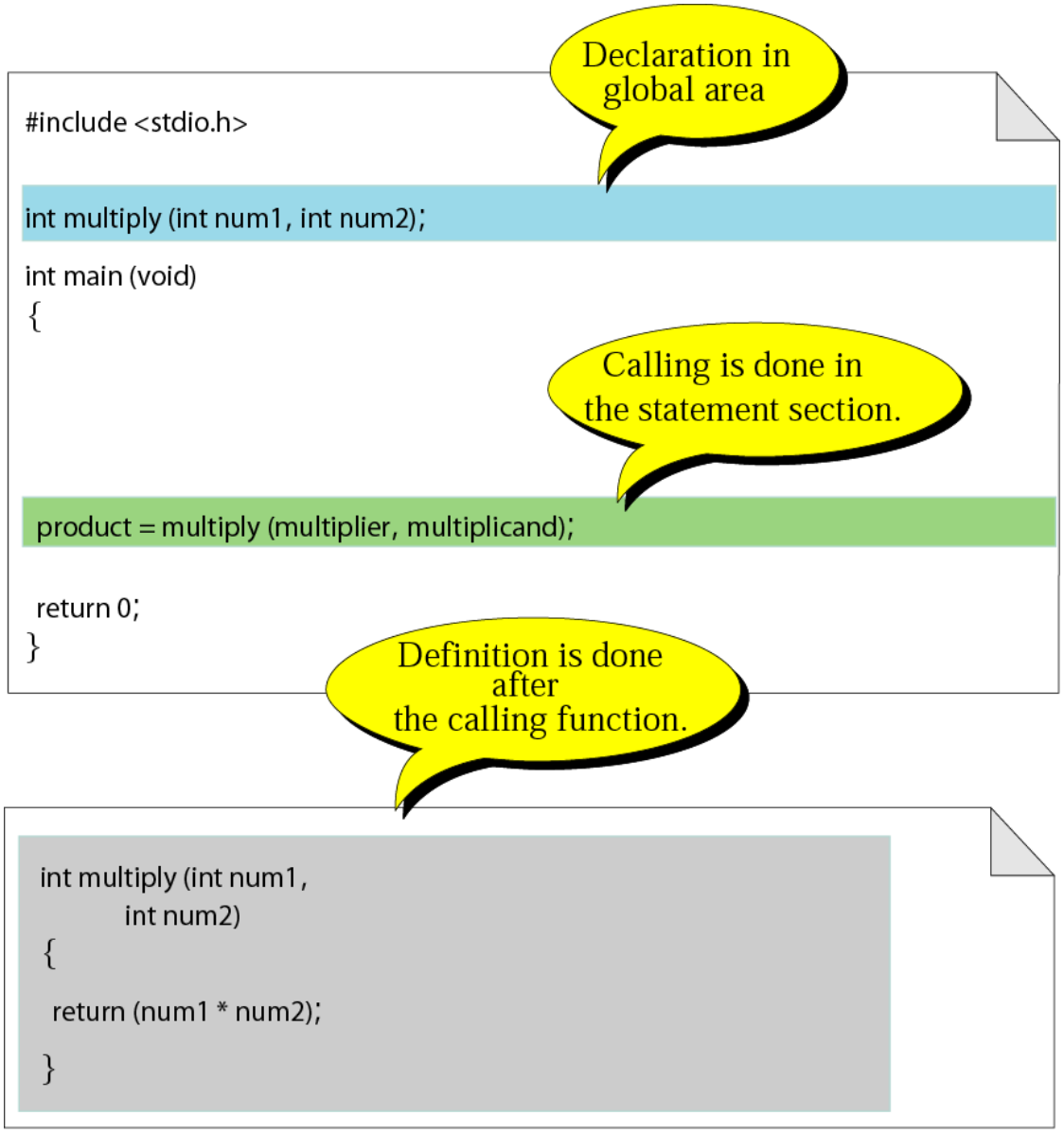
รูปที่ 9-9

Side effect of a function



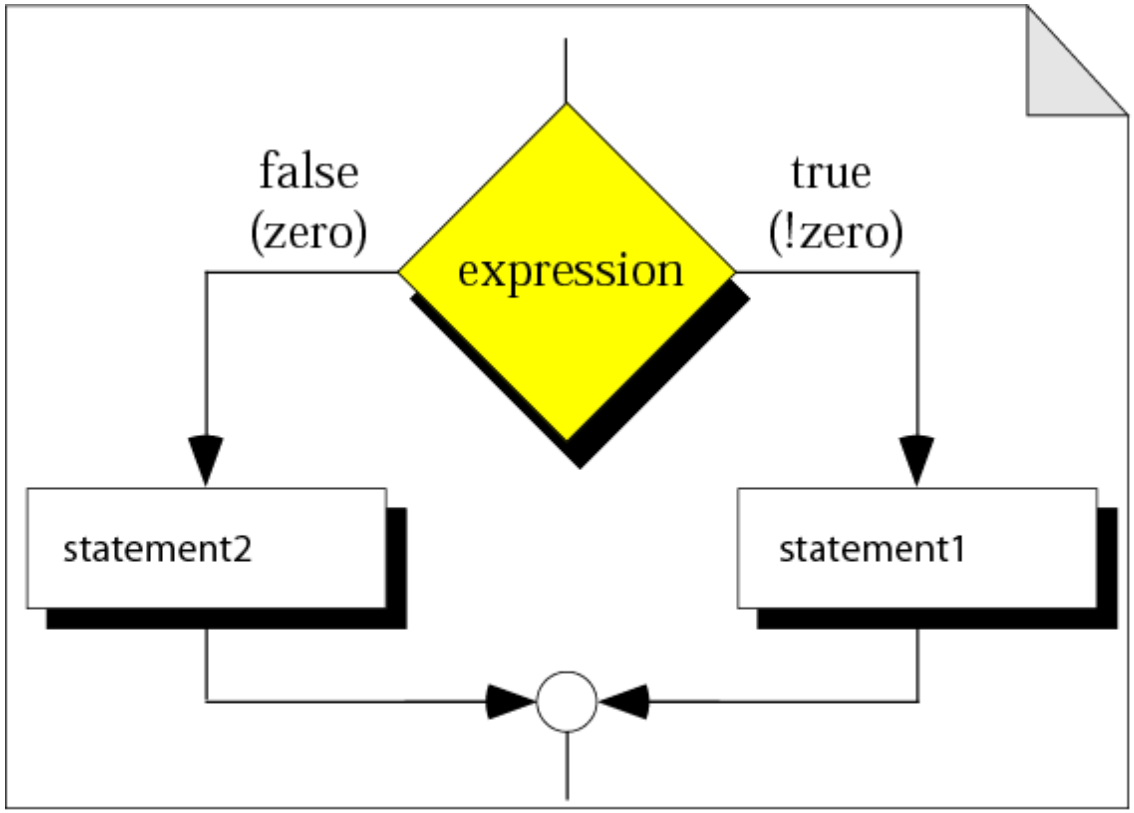
รูปที่ 9-10

Function declaration



รูปที่ 9-11

if-else statement



a. Logical Flow

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

b. Code

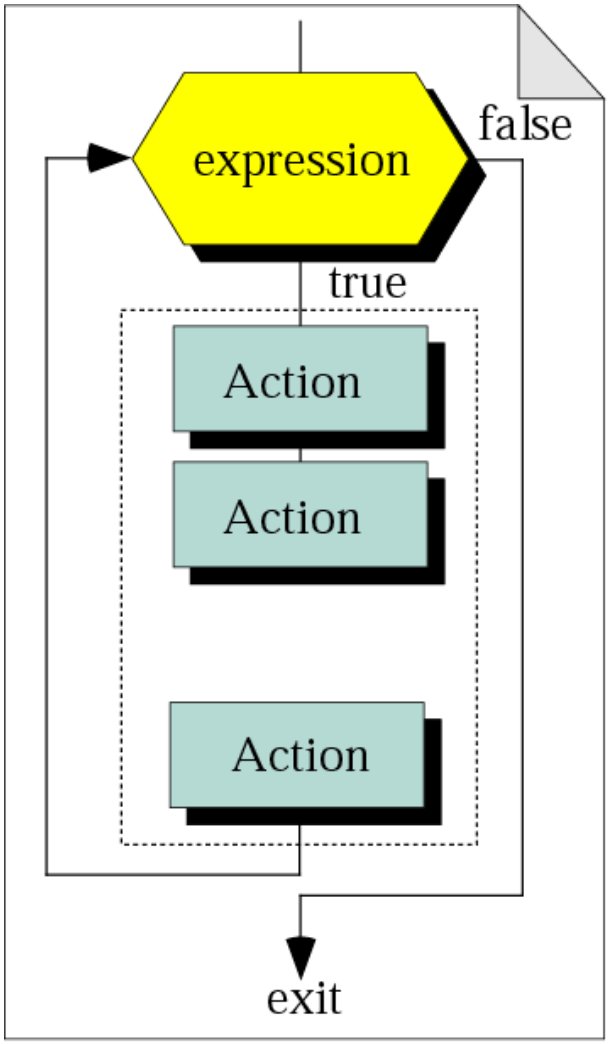
รูปที่ 9-12

switch statement

```
switch  ( expression )
 {
  case  constant-1 :          statement

                              statement

  case  constant-2 :          statement

                              statement

  case  constant-n  :         statement

                              statement

  default           :         statement

                              statement
 }
```

รูปที่ 9-13

while loop



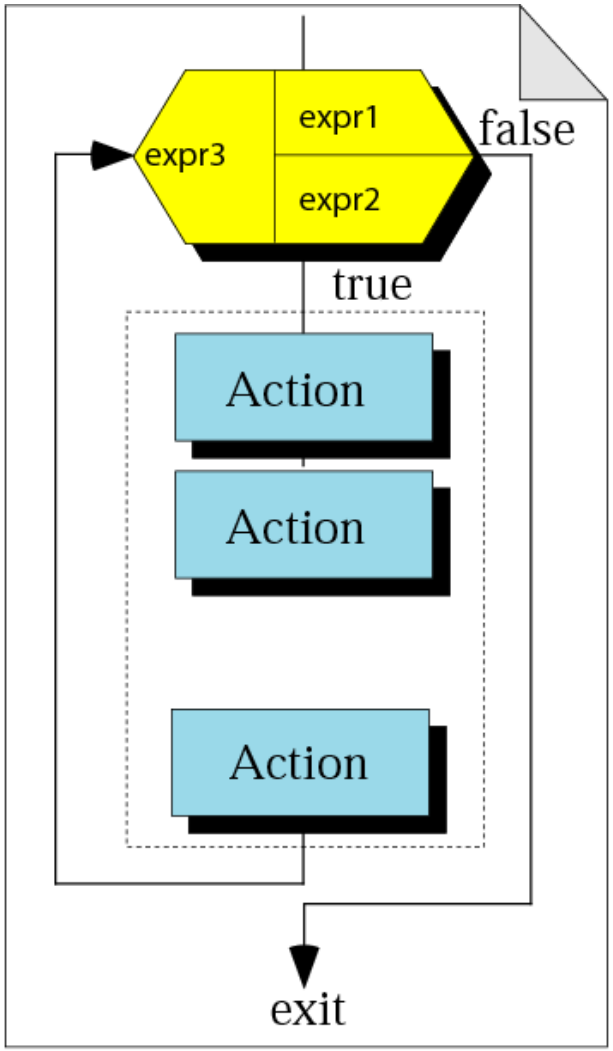
a. Flowchart

```
while ( expression )
{
    Action
    Action

    Action
}
```

b. C Language

รูปที่ 9-14

for loop



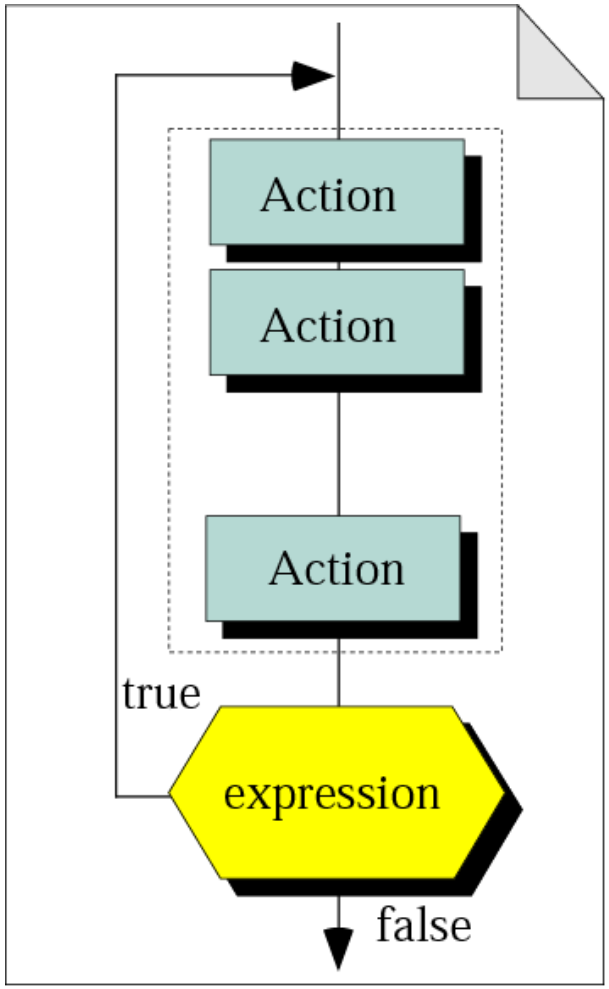
a. Flowchart

```
for (    expr1 ;
         expr2 ;
         expr3 )
{
    Action
    Action

    Action
}
```

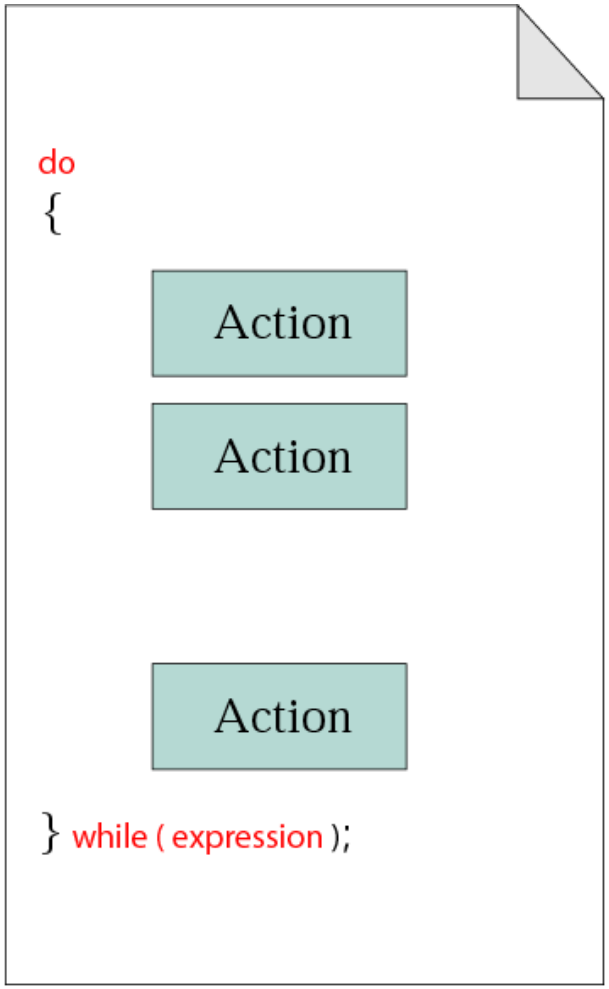
b. C Language

รูปที่ 9-15

do-while loop



a. Flowchart



b. C Language

PROCEDURAL LANGUAGES

- **Procedural Language เป็นภาษาที่ execute คำสั่งทีละคำสั่งตามลำดับที่เรียงไว้ การทำงานเป็นไปแบบทีละขั้นตอน (step by step) ตามอัลกอริธึมที่กำหนดโดยมีโครงสร้างทั้งสามโครงสร้างเป็นตัวกำหนด ภาษาที่สำคัญในกลุ่มนี้ได้แก่**

  *** FORTRAN (FORmula TRANslation)**

  *** COBOL (COmmon Business Oriented Language)**

  *** Pascal : 1971 by Niklaus Wirth in Zurich, Swiss.**

  *** C : 1970 by Dennis Ritchie at Bell Laboratory**

  *** Ada (Augusta Ada Byron) for DoD**

Object-Oriented Language

- Object-oriented language เป็นภาษาเชิงวัตถุซึ่งกำลังเป็นที่นิยมใช้กันมากในวงการนักคอมพิวเตอร์ในปัจจุบัน ภาษาที่ใช้กันมาในปัจจุบันเช่น

  *** C++ : by Bjarne Stroustrup at Bell Laboratory เพื่อปรับปรุงภาษา C ให้มีประสิทธิภาพมากขึ้น**

  *** Java : by บริษัท Sun Microsystem, Inc. โดยอาศัยพื้นฐานจากภาษา C และ C++**

คำที่สำคัญ