# บทที่ 4 C++ Fundamentals and Elementary Data Types

## วัตถุประสงค์

1) เพื่อให้นักศึกษาเข้าใจส่วนประกอบของโปรแกรมเบื้องต้น
2) เพื่อให้นักศึกษาเข้าใจหลักการทำงานของตัวแปร
3) เพื่อให้นักศึกษาเข้าใจความแตกต่างและข้อจำกัดของข้อมูลพื้นฐานชนิดต่างๆ และเลือกใช้ข้อมูลพื้นฐานได้อย่างเหมาะสม
4) เพื่อให้นักศึกษารู้จักวิธีการนำเข้าข้อมูล และแสดงผลข้อมูลโดยโปรแกรม
5) เพื่อให้นักศึกษาเขียนโปรแกรม นำเข้าข้อมูล ประมวลผลอย่างง่าย และแสดงผลได้

## 4.1 โครงสร้างของโปรแกรม C++

ในการเขียนโปรแกรมด้วยภาษา C++ นั้น เราเขียนลงในไฟล์ที่มีนามสกุลว่า .cpp เพื่อให้ตัวแปลผลข้อมูลแปลผลของโค้ดได้อย่างถูกต้อง ตัวแปลผลโค้ดจากภาษาคอมพิวเตอร์ชั้นสูงเช่น ภาษา C++ ให้เป็นภาษาที่เครื่องคอมพิวเตอร์สามารถเข้าใจและทำงานได้เรียกว่า คอมไพเลอร์ (compiler) ในวิชานี้เราใช้ GNU C++ เป็นคอมไพเลอร์หลัก รูปที่ 1 แสดงตัวอย่างของโปรแกรมภาษา C++

```
#include <iostream>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    return 0;
}
```

**รูปที่** 1 **ตัวอย่างของโปรแกรมภาษา** C++

โปรแกรมที่เขียนด้วยภาษา C++ นั้นมีโครงสร้างโปรแกรมพื้นฐานและส่วนประกอบต่างๆ ดังนี้

1) โปรแกรมทุกโปรแกรมจะต้องมีฟังก์ชันเมน (main()) คือฟังก์ชันหลัก ซึ่งโปรแกรมจะเริ่มทำงาน (รัน: run) ที่ฟังก์ชันเมนเสมอ

```
int main() {
    return 0;
}
```

2) โปรแกรมจะต้องมีการอ้างอิงคำสั่งจากไลบรารี (library) หรือคลังโปรแกรมรวมคำสั่งต่างๆ เพื่อที่คอมไพเลอร์ทราบว่าจะแปลผลคำสั่งต่างๆ อย่างไร ไลบรารีนั้นมีทั้งไลบรารีมาตรฐาน (standard library) และไลบรารีที่ผู้เขียนโปรแกรมกำหนดขึ้นมา (user-defined library) ซึ่งจะกล่าวโดยละเอียดในเรื่องฟังก์ชัน โดยปกติแล้วโปรแกรมในภาษา C++ จะต้องอ้างอิงไลบรารี iostream เสมอเพื่อให้สามารถรับเข้าข้อมูลและแสดงผลรันได้เป็นอย่างน้อย ดังนั้นในสื่อและเอกสารประกอบการสอนนี้ผู้เขียนจะขอละไว้ในฐานที่เข้าใจว่าทุกโปรแกรมจะต้องมีการอ้างอิงไลบรารีนี้เพื่อประหยัดเนื้อที่

```
#include <iostream>
```

3) เราใช้สัญลักษณ์ // ในการเขียนคำอธิบายโค้ด โดยคอมไพเลอร์จะไม่แปลผลข้อความที่อยู่หลัง // ในบรรทัดนั้น อีกสัญลักษณ์ที่เราใช้คือ /* ... */ เพื่อใช้อธิบายโค้ดในหลายบรรทัดต่อกันโดยไม่ต้องเขียน // ทุกบรรทัด

```
// Hello World Program
```

```
/*
   Kornchawal Chaipah
   198110 Computer Programming
*/
```

4) โปรแกรมประกอบด้วยโค้ดที่เรียกว่า statement หรือคำสั่งให้โปรแกรมทำตาม

```
cout << "Hello World\n";
return 0;
```

5) Statement หรือคำสั่งต่างๆ ต้องจบด้วยเครื่องหมายเซมิโคลอน (;) เสมอ ไม่เช่นนั้นคอมไพเลอร์จะไม่แปลผลให้
6) Statement หรือคำสั่งต่างๆ ต้องอยู่ภายในขอบเขตของปีกกาคู่ ( { ... } ) ของ main() ฟังก์ชัน
7) คำสั่งต่างๆ ในภาษา C++ มีลักษณะเป็น case sensitive หรืออักษรตัวใหญ่ตัวเล็กนั้นถูกแปลผลว่าแตกต่างกัน เช่นคำว่า cout และ Cout ไม่ใช่คำสั่งเดียวกัน ตัวแปร x และ X ไม่ใช่ตัวแปรเดียวกัน
8) เมื่อจบฟังก์ชันเมน โปรแกรมจะจบการทำงานด้วยคำสั่ง return 0; หรือ exit(1); โปรแกรมจะไม่ทำคำสั่งใดๆ หลังจากสองคำสั่งนี้

## 4.2 การส่งออกข้อมูล

จากที่เราได้กล่าวในบทเรียนเรื่องอัลกอริทึมนั้น เรารู้ว่าอัลกอริทึมจะต้องมีอย่างน้อย 1 เอาต์พุตเสมอ ในการเขียนโปรแกรมเพื่อแก้ปัญหาต่างๆ ก็เช่นกัน เราจะต้องมีการส่งออกข้อมูล (outputing) เสมอ การส่งออกข้อมูลในวิชานี้มี 2 แบบคือ การส่งออกข้อมูลเพื่อแสดงทางหน้าจอ และการส่งออกข้อมูลเพื่อบันทึกลงในไฟล์ สำหรับบทนี้เราจะเรียนการส่งออกข้อมูลเพื่อแสดงทางหน้าจอก่อน

ในภาษา C++ การส่งออกข้อมูลเพื่อแสดงผลทางหน้าจอนั้น เราจะใช้คำสั่ง cout แล้วตามด้วยเครื่องหมาย << จากนั้นเราจึงใส่สิ่งที่เราต้องการแสดงออกทางหน้าจอ ไม่ว่าจะเป็นค่าตัวเลข ตัวแปร อักขระ อักขระพิเศษหรือข้อความต่างๆ ถ้าเราต้องการแสดงผลหลายๆ ผลออกในคราวเดียวกัน เราก็คั่นสิ่งที่เราต้องการแสดงด้วยเครื่องหมาย << หลังจากนั้น เราจะปิดคำสั่งด้วยเครื่องหมาย ; เป็นสิ่งสุดท้าย ดังแสดงในรูปที่ 2 cout คือคำสั่งให้แสดงผลออกทางหน้าจอ ส่วน exp (expression) หมายถึงนิพจน์ ในที่นี้คือค่าต่างๆ ที่เราต้องการแสดงออกทางหน้าจอ

```
cout << exp1 << exp2 << ... << expn;
```

**รูปที่** 2 **รูปแบบการใช้คำสั่ง** cout

ถ้าหลังคำสั่ง cout นั้นมีการแสดงผลหลายนิพจน์ ลำดับการแสดงผลออกจากทางหน้าจอนั้นจะแสดงจากซ้ายไปขวาเสมอ (นั่นคือตามลำดับการเขียน) โดยจะไม่มีเว้นวรรคคั่น ถ้าเราต้องการให้มีเว้นวรรคหรือการเว้นบรรทัดคั่นระหว่างค่าต่างๆ เราต้องเขียนอักขระว่าง (space) หรืออักขระพิเศษ เช่นแท็บ (‘\t’) หรือตัวเคาะบรรทัดใหม่ (‘\n’) เข้าไป หรือเราจะเขียนคำสั่ง endl (end line) แทรกเข้าไปก็ได้ เช่นเดียวกันกับการที่เราเคาะบรรทัดหรือเปลี่ยนบรรทัดใหม่ในโปรแกรมจะไม่ทำให้มีการขึ้นบรรทัดใหม่ในการแสดงผลด้วย ถ้าเราต้องการให้มีการเริ่มบรรทัดใหม่เราต้องใส่อักขระพิเศษ หรือคำสั่ง endl เข้าไปในคำสั่ง cout ด้วยเสมอ อย่างเช่น

โค้ด:

```
1    cout << 2 << "Hello" << "World" << endl;
2    cout << 2 << " " << "Hello" << " " << "World";
3    cout << 2 << " " << "Hello World" << endl;
```

ผลการทำงาน:

```
2HelloWorld
2 Hello World2 Hello World
```

เราจะเห็นได้ว่าในบรรทัดที่ 1 เราไม่มีการใส่อักขระเพื่อเว้นวรรคระหว่างค่า 2, "Hello" และ "World" ทำให้เวลาแสดงผลจะไม่มีการเว้นวรรคระหว่างค่าและข้อความดังกล่าว ต่างกับบรรทัดที่ 2 และ 3 เราได้ใส่เว้นวรรคระหว่างค่า ทำให้เวลาแสดงผลจะมีการเว้นวรรคเกิดขึ้น

ในท้ายบรรทัดที่ 1 เราได้ใส่ endl เพื่อขึ้นบรรทัดใหม่ ในการแสดงผลทำให้มีการเคาะบรรทัดหลังจากโปรแกรมได้แสดงข้อความ `2HelloWorld` ตามโค้ดบรรทัดแรกแล้ว แต่ในโค้ดบรรทัดที่ 2 เราไม่ได้ใส่ endl ก่อนแสดงบรรทัดที่ 3 เพราะฉะนั้นข้อความของโค้ดบรรทัดที่ 2 และ 3 จึงถูกแสดงในบรรทัดเดียวกันบนหน้าจอ

## 4.3 ตัวแปรและชนิดของข้อมูล

### ตัวแปร (Variable)

ตัวแปรคือชื่อที่อ้างอิงถึงค่าต่างๆ ของข้อมูลชนิดใดๆ ที่ถูกบรรจุอยู่ในหน่วยความจำ (memory) ของคอมพิวเตอร์ ตัวแปรหนึ่งตัวจะ "ชี้" หรืออ้างอิงไปที่ "กล่อง" ของหน่วยความจำ 1 ตำแหน่ง ค่าของตัวแปรคือค่าที่ถูกเก็บอยู่ในหน่วยความจำที่ตัวแปรนั้นอ้างอิงถึง ส่วนชนิดของตัวแปรคือชนิดของข้อมูลที่ถูกเก็บอยู่ในหน่วยความจำนั้นๆ

ในการเขียนโปรแกรมนั้น เรานิยมใช้ตัวแปรในการเก็บค่าต่างๆ การสร้างตัวแปรในภาษา C++ นั้นมีกฎดังต่อไปนี้

1) ชื่อตัวแปร ต้องเริ่มด้วยอักษรหรือเครื่องหมาย underscore (_) เท่านั้น ดังนั้นเราไม่สามารถขึ้นต้นชื่อตัวแปรด้วยตัวเลขหรือเครื่องหมายอื่นๆ ได้ หลังจากอักขระตัวแรกของชื่อแล้วเราสามารถใส่อักษร

หรือเลข หรือเครื่องหมาย underscore ในชื่อได้ เช่น เราสามารถสร้างตัวแปรชื่อ X, x, x2, G2000, _myclass, หรือ hello_2worlD ได้ แต่เราไม่สามารถสร้างตัวแปรที่ชื่อ 2xyz, 3000 หรือ 44aa ได้ เพราะขึ้นต้นด้วยตัวเลข

นอกจากนี้ เรายังไม่สามารถตั้งชื่อซ้ำกับ reserved words หรือคำที่กันไว้ใช้พิเศษ คำสั่งมาตรฐาน และ identifiers ในภาษา C++ ได้ เช่น include, return, endl, if, switch, while, main, cout, int และ string

2) การประกาศตัวแปร ต้องเริ่มด้วยชนิดของตัวแปรเสมอ แล้วตามด้วยชื่อตัวแปร เราจะอธิบายเรื่องชนิดของตัวแปรในส่วนถัดไป แต่ให้ทราบคร่าวๆ ว่าชนิดของตัวแปรที่เราใช้ในวิชานี้จะมี char, short และ int สำหรับเลขจำนวนเต็ม float และ double สำหรับเลขจำนวนจริง char สำหรับอักขระหนึ่งตัว string สำหรับข้อความ และ bool สำหรับค่าบูลีน

```
Type varName;
```

3) เราต้องประกาศตัวแปร**<u>ก่อน</u>**นำตัวแปรไปใช้เสมอ
4) การให้ค่าตัวแปร ต้องวางตัวแปรไว้ทางซ้ายมือของเครื่องหมายเท่ากับ (=) เสมอ ส่วนค่าที่เราจะนำมาเก็บไว้ในตัวแปรให้อยู่ทางขวามือ

```
varName = exp;
```

5) เราสามารถให้ประกาศตัวแปรและให้ค่าตัวแปรไปพร้อมกันได้เลย เช่น int x = 3;

**ชนิดของข้อมูล (data type)**

ชนิดของข้อมูลในภาษา C++ ที่เราจะเรียนในวิชานี้สามารถแบ่งออกได้เป็น 3 กลุ่มใหญ่ๆ คือ ข้อความ ค่าตัวเลข และค่าบูลีน

1) ข้อความ (text) มี 2 ชนิดคือ character ซึ่งใช้เก็บอักขระที่มีตัวเดียวหรือไม่มีก็ได้ และ string ซึ่งใช้เก็บข้อความที่มีอักขระอย่างน้อยหนึ่งตัวหรือไม่มีก็ได้
2) ค่าตัวเลข (number) มี 2 กลุ่มย่อยคือ จำนวนเต็ม ประกอบด้วย character, short, integer และ long และจำนวนจริง ซึ่งประกอบด้วย float และ double
3) ค่าบูลีน (Boolean) เป็นค่าความจริง มีชนิดเป็น bool สามารถเป็นได้ 2 ค่าคือ จริง (true) และเท็จ (false)

**อักขระ (Character)**

อักขระคืออักษร ตัวเลข (ไม่ใช่ค่าตัวเลข) หรือสัญลักษณ์ ที่เขียนด้วยตัวเดียว เวลาเราเขียนอักขระในภาษา C++ เราจะต้องเขียนอยู่ในอัญประกาศเดี่ยว (single quotation mark) เช่น ‘A’, ‘b’, ‘9’, ‘+’ นอกจากนี้ อักขระยังรวมถึงสัญลักษณ์พิเศษที่จะไม่แสดงผลออกมาโดยตรง คือ ‘\n’ ซึ่งหมายถึงแสดงการเริ่ม

บรรทัดใหม่ ‘\t’ ซึ่งหมายถึงแสดงย่อหน้า และ ‘\r’ ซึ่งหมายถึงการเริ่มบรรทัดใหม่หรือไปที่ต้นบรรทัด (แล้วแต่ระบบปฏิบัติการ ในวิชานี้เราแทบจะไม่ได้ใช้ ‘\r’ เลย)

ข้อมูลชนิดอักขระนี้จะมีค่าทางคณิตศาสตร์ได้ด้วย นั่นคือเราสามารถปฏิบัติการทางคณิตศาสตร์ เช่น บวก ลบ คูณ หารอักขระได้ โดยใช้ค่า ASCII code ของแต่ละอักขระ ดังแสดงในตารางต่อไปนี้

**ตารางที่ 1** ค่า ASCII code ของอักขระ

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |

Source: www.LookupTables.com

เราจะเห็นได้ว่าอักขระที่เป็นตัวเลขจะไม่ได้มีค่าทางคณิตศาสตร์ตามตัวเลขของมัน เช่น อักขระ ‘1’ มีค่าทางคณิตศาสตร์เท่ากับ 49 ไม่ใช่ 1 ดังนั้นเวลาเราเอาอักขระไปใช้จะต้องระวังด้วย

## สายอักขระ (String)

สายอักขระ หรือ สตริง (string) ประกอบด้วยอักขระมาต่อเข้าด้วยกัน พูดรวบง่ายๆ สายอักขระคือข้อความนั่นเอง เวลาเราเขียนสายอักขระในภาษา C++ เราจะต้องเขียนอยู่ในอัญประกาศ (double quotation mark) เช่น “hello”, “world”, “ ”, หรือ “12345” โดยปกติแล้วเราสามารถประกาศสตริงตามแบบตัวแปรทั่วไปได้ คือประกาศ

```
string s1;
```

การประกาศแบบนี้จะทำให้ s1 เป็นสตริงว่าง ส่วนการให้ค่าเริ่มต้นของตัวแปรก็ทำได้เหมือนตัวแปรชนิดอื่นๆ เช่นเดียวกัน เช่น

```
string s2 = “New York”;
```

หรือการคัดลอกค่าจากสตริงที่เคยประกาศไปแล้ว เช่น

```
string s3 = s2;
```

จะทำให้ค่า s3 มีค่าเท่ากับสตริง s2 คือ “New York” นอกจากการประกาศและให้ค่าตัวแปรที่เหมือนตัวแปรชนิดอื่นๆ แล้ว เรายังสามารถประกาศและให้ค่าเริ่มต้นของตัวแปรในแบบอื่นได้อีกด้วย เช่น

```
string s4(60, '*');
```

จะสร้างสตริง s4 โดยมีค่าเป็นดอกจัน * 60 ตัวต่อกันอยู่ในสตริง ส่วน

```
string s5(s2, 4, 2);
```

จะสร้างสตริง s5 โดยค่าเริ่มต้นของสตริงจะเอามาจากสตริง s2 โดยไปเอาค่าในตำแหน่งที่ 4 และคัดลอกมา 2 ตัว ซึ่งตำแหน่งในสตริงนั้นจะเริ่มจากตำแหน่งที่ 0 ดังนั้น ค่าในสตริง s5 จะเป็น “Yo”

นอกจากนี้ เราสามารถสร้างสตริงโดยเอาสตริงมาต่อกันด้วยเครื่องหมายบวก + อีกด้วย (string concatenation) เช่น

โค้ด:

```
#include <cstring>
int main(){
      string firstname = "John";
      string lastname = "Doe";
      string fullname = firstname + " " + lastname;
      cout << "Your full name is " << fullname << endl;
      return 0;
}
```

ในโค้ดนี้ เราเอาสตริง firstname มาต่อกับสตริงเว้นวรรคและสตริง last name ทำให้ผลลัพธ์ที่ได้เมื่อ cout ออกมาจะเป็น

ผลการทำงาน:

```
Your full name is John Doe
```

เราสามารถหาความยาวของสตริงได้โดยใช้ฟังก์ชัน strlen() จากไลบรารี cstring (เราจะเรียนเรื่องฟังก์ชันในบทต่อๆ ไป แต่ตอนนี้ขอให้จดจำวิธีการหาความยาวของสตริงไปก่อน) ความยาวของสตริงคือจำนวนอักขระที่อยู่ในสตริงนั้น สตริงสามารถมีความยาวได้ตั้งแต่ 0 ขึ้นไป สตริงที่มีความยาวเท่ากับ 0 เรียกว่า สตริงว่าง (empty string) อักขระแต่ละตัวจะมีความยาวเท่ากับ 1 ความยาวของอักขระพิเศษ เช่น “\n” ก็คือ 1 เช่นกันไม่ใช่ 2 ความยาวของเว้นวรรค (space) ก็มีความยาวเท่ากับ 1 เพราะถือว่าเป็นอักขระหนึ่งตัวเช่นเดียวกัน ตัวอย่างการหาความยาวของสตริงทำได้ดังนี้

โค้ด:

```
1    cout << strlen("Hello, World.\n") << '\n';
2    cout << strlen("Hello, World.") << '\n';
3    cout << strlen("Hello, ") << '\n';
4    cout << strlen("H") << '\n';
5    cout << strlen("") << '\n';
```

ผลการทำงาน:

```
1  14
2  13
3  7
4  1
5  0
```

ความยาวของสตริงในบรรทัดที่ 1, 2 และ 3 คือ 14, 13 และ 7 ตามลำดับ สังเกตว่าเรานับเว้นวรรคเป็น 1 อักขระ และนับ \n เป็นหนึ่งอักขระเช่นเดียวกัน สตริงในบรรทัดที่ 5 เป็นสตริงว่าง จึงมีความยาวเป็น 0

**จำนวนเต็ม (Integer)**

**ชนิดของข้อมูลที่ใช้สำหรับบรรจุจำนวนเต็มในภาษา C++ มีหลายชนิดตามขนาดของค่าที่ตัวแปรสามารถเก็บได้ ดังแสดงใน**

ตารางที่ 2 เราจะเห็นว่าเรามี char, short, int และ long โดยมีขนาดของข้อมูลเท่ากับ 8, 16, 32 และ 32 บิต ตามลำดับ ความสามารถในการเก็บค่าของตัวแปรจะขึ้นอยู่กับขนาดของข้อมูลนี้ นอกจากนี้ค่าสูงสุดและต่ำสุดของข้อมูลแต่ละชนิดจะมีตัวกำหนดอีกตัวคือ signed หรือ unsigned คำว่า signed หมายความว่าชนิดข้อมูลนั้นมี “เครื่องหมาย” คือสามารถมีค่าเป็นลบและบวกได้ ทำให้ตัวแปรสามารถเก็บค่าได้ตั้งแต่ $-2^{n-1}$ ถึง $2^{n-1}-1$ เมื่อ n เป็นขนาดของข้อมูล ส่วน unsigned คือข้อมูลนั้นไม่สามารถเป็นลบได้ เป็นบวกได้อย่างเดียว จึงทำให้ตัวแปรสามารถเก็บค่าได้ตั้งแต่ 0 ถึง $2^{n}-1$ เช่น ข้อมูลชนิด short มีขนาด 16 บิต สามารถเป็นได้ 2 ชนิดคือ

- short (จริงๆ คือ signed short แต่ไม่จำเป็นต้องเขียนว่า signed นำหน้า) เก็บข้อมูลได้ตั้งแต่ $-2^{16-5}$ = -32,768 ถึง $2^{16-1}-1$ = 32,767
- unsigned short (จำเป็นต้องเขียน unsigned นำหน้า) เก็บข้อมูลได้ตั้งแต่ 0 ถึง $2^{16}-1$ = 65,535

**ตารางที่ 2 ชนิดข้อมูลจำนวนเต็มและความสามารถในการเก็บข้อมูล**

| Types | Bits | Min | Max |
|---|---|---|---|
| char | 8 | -128 | 127 |
| unsigned char | 8 | 0 | 255 |
| short | 16 | -32,768 | 32,767 |
| unsigned short | 16 | 0 | 65,535 |
| int | 32 | -2,147,483,648 | 2,147,483,647 |
| unsigned int | 32 | 0 | 4,294,967,295 |
| long | 32 | -2,147,483,648 | 2,147,483,647 |
| unsigned long | 32 | 0 | 4,294,967,295 |

เราจะเห็นได้ว่าข้อมูลชนิดต่างๆ จะมีช่วงการเก็บค่าข้อมูลต่างๆ กัน เวลาเราเขียนโปรแกรมเราจึงต้องเลือกชนิดข้อมูลที่เหมาะสม อย่างเช่นเราต้องการเก็บค่า 1,000,000 เราต้องเลือกข้อมูลชนิด int หรือ long เพราะถ้าเลือก char ซึ่งเก็บข้อมูลได้สูงสุดเป็นหลักร้อย หรือ short ที่เก็บข้อมูลได้แค่หลักหมื่น เราจะไม่สามารถเก็บค่าที่ถูกต้องได้

**การหาขนาดของตัวแปรหรือข้อมูล**

เราสามารถหาขนาดของตัวแปรหรือข้อมูลในหน่วยไบต์ (byte) ได้โดยใช้ฟังก์ชัน sizeof() ดังตัวอย่างต่อไปนี้

โค้ด:

```
int main(){
      cout << "Size of char: " << sizeof(char) << endl;
      cout << "Size of short: " << sizeof(short) << endl;
      cout << "Size of int: " << sizeof(int) << endl;
      char a;
      int b;
      cout << "Size of a: " << sizeof(a) << endl;
      cout << "Size of b: " << sizeof(b) << endl;
      return 0;
}
```

ผลการทำงาน:

```
1
2
4
1
4
```

จะเห็นได้ว่าเราสามารถใส่ชนิดของข้อมูล (char, short, int) เข้าไปในฟังก์ชัน sizeof() ได้เลย หรือเราจะใส่ชื่อตัวแปร (a,b) เข้าไปก็ได้ โปรแกรมจะให้ค่าขนาดของข้อมูลชนิดต่างๆ ออกมาทางหน้าจอ เราสามารถใช้ฟังก์ชันนี้กับตัวแปรได้ทุกชนิด รวมถึง char, string, float, double, bool ด้วย

**การดำเนินการคำนวณ (Arithmetic operation)**

ในภาษา C++ เราสามารถเอาค่าที่เก็บในตัวแปรแบบจำนวนเต็ม มาดำเนินการคำนวณได้ดังกฎในรูปที่ 3 นั่นคือเราสามารถเพิ่มค่าทีละหนึ่ง (++) ลดค่าทีละหนึ่ง (--) ทำให้เป็นลบ (-) คูณ (*) หาร (/) หาเศษของการหาร (%) บวก (+) และลบ (-) ได้ โดยมีลำดับการคำนวณในสามกลุ่มจากบนลงล่าง คือถ้ามีการคำนวณมากกว่าหนึ่งตัวขึ้นไปแล้วไม่มีวงเล็บกำกับ ให้ทำกลุ่มข้างบนก่อนเสมอ จากนั้นทำการคำนวณกลุ่มตรงกลางและกลุ่มล่างสุดตามลำดับ แต่ถ้ามีตัวดำเนินการทางคณิตศาสตร์จากกลุ่มเดียวกัน ให้ทำจากซ้ายไปขวา เช่น

```
3 * 2 * 5 + 9 / 3 – 5
```

เครื่องหมายคูณและหารมีลำดับสูงสุดในตัวอย่างนี้ เราเลือกทำการคูณก่อน (จริงๆ เราทำการหารก่อนก็ได้ ไม่ผิดอะไร) เราจึงทำ 3 * 2 ก่อน จากนั้นเราจึงเอาผลลัพธ์ไป * 5 จะได้ 30 พักไว้ก่อน ถึงตอนนี้เราจะได้ 30 + 9 / 3 – 5 จากนั้นเราต้องทำ 9/3 เพราะการหารมีลำดับสูงกว่าการบวกและการลบ แล้วเราก็จะได้ 30 + 3 – 5 จากนั้นเราจะเห็นว่าการบวกกับลบอยู่ในระดับการดำเนินการเดียวกัน เราจึงทำจากซ้ายไปขวา ได้ 33 – 5 = 28 นั่นเอง

| Operator | Meaning | Example |
|---|---|---|
| ++/-- | Pre | ++n |
| ++/-- | Post | n++ |
| - | Negate | -n |
| * | Multiply | m * n |
| / | Divide | m / n |
| % | Remainder | m % n |
| + | Add | m + n |
| – | Subtract | m – n |

**รูปที่ 3** ตัวดำเนินการทางคณิตศาสตร์และกลุ่มลำดับการดำเนินการ

**การเพิ่มค่า (Increment)**

ในภาษา C++ เราสามารถเพิ่มค่าของตัวแปรทีละหนึ่งได้ โดยการเขียนเครื่องหมาย ++ ไว้ข้างหน้าหรือหลังตัวแปรนั้นได้เลย เช่น x++ หรือ ++x จะทำให้เราได้ค่าเสมือนเราให้ค่า x = x+1

```
varName++;
++Varname;
varName = varName + 1;
```

**รูปที่ 4** รูปแบบการเพิ่มค่าตัวแปรทีละหนึ่ง

ข้อแตกต่างระหว่าง x++ และ ++x นั้นคือในกรณีที่มีการคำนวณอย่างอื่นในคำสั่ง (statement) นั้นด้วย ถ้าเราเขียน x++ เราจะต้องใช้ค่า x เดิมก่อนการเพิ่มค่ามาคำนวณ จากนั้นเราจึงเพิ่มค่า x แต่ถ้าเราเขียน ++x เราจะต้องเพิ่มค่า x ก่อน แล้วจึงนำค่า x อันใหม่มาคำนวณ เช่น

```
a = x++ + 5;
```

ให้ทำ a = x + 5 (ใช้ x เก่า) แล้วจึงทำ x = x + 1 แต่ถ้าเป็น

```
a = ++x + 5;
```

ให้ทำ x = x + 1 ก่อน แล้วจึงทำ a = x + 5 (ใช้ x ใหม่)

**การลดค่า (Decrement)**

การลดค่าตัวแปรเหมือนการเพิ่มค่าทุกอย่าง ยกเว้นเราลดค่าทีละหนึ่งแทนที่จะเพิ่มค่าทีละหนึ่ง นั่นคือในภาษา C++ เราสามารถลดค่าของตัวแปรทีละหนึ่งได้ โดยการเขียนเครื่องหมาย -- ไว้ข้างหน้าหรือหลังตัวแปรนั้นได้เลย เช่น x-- หรือ --x จะทำให้เราได้ค่าเสมือนเราให้ค่า x = x - 1

```
varName--;
--Varname;
varName = varName - 1;
```

**รูปที่ 5** รูปแบบการลดค่าตัวแปรทีละหนึ่ง

ข้อแตกต่างระหว่าง x-- และ --x นั้นคือในกรณีที่มีการคำนวณอย่างอื่นในคำสั่ง (statement) นั้นด้วย ถ้าเราเขียน x-- เราจะต้องใช้ค่า x เดิมก่อนการลดค่ามาคำนวณ จากนั้นเราจึงลดค่า x แต่ถ้าเราเขียน --x เราจะต้องลดค่า x ก่อน แล้วจึงนำค่า x อันใหม่มาคำนวณ เช่น

```
a = x-- + 5;
```

ให้ทำ a = x + 5 (ใช้ x เก่า) แล้วจึงทำ x = x - 1 แต่ถ้าเป็น

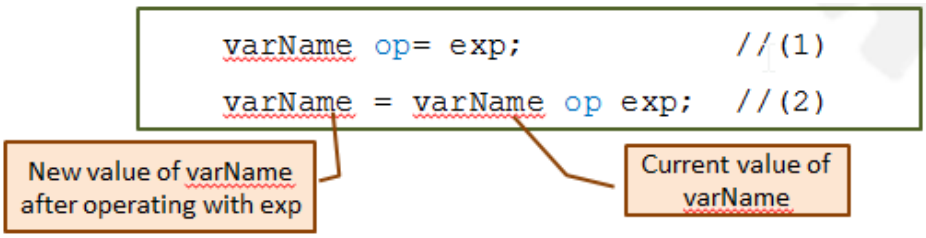
```
a = --x + 5;
```

ให้ทำ x = x - 1 ก่อน แล้วจึงทำ a = x + 5 (ใช้ x ใหม่)

ในการช่วยจำ ให้เราท่องว่า ถ้าเครื่องหมายอยู่หน้า ให้ลดหรือเพิ่มค่าก่อน ถ้าเครื่องหมายอยู่หลัง ให้ลดหรือเพิ่มค่าทีหลัง

**การให้ค่าแบบประกอบ (Composite assignment)**

ในกรณีที่การให้ค่า (assignment) นั้นกระทำกับตัวแปรและเอาค่าไปเก็บไว้ที่ตัวแปรเดิม เช่น x = x + 5 เราสามารถเขียนคำสั่งการดำเนินการคำนวณ (arithmetic operation) แบบย่อได้ดังนี้




**รูปที่ 6** รูปแบบการให้ค่าแบบประกอบ

นั่นคือ โดยปกติแล้ว เช่นถ้าเราต้องการบวกค่าของ x ไป 5 แล้วเอาไปเก็บไว้ใน x เหมือนเดิม เราสามารถ เขียนได้ว่า x = x + 5 และเราก็สามารถเขียนแบบย่อได้คือ x += 5 ซึ่งมีความหมายแบบเดียวกัน เราสามารถใช้การเขียนแบบนี้ได้กับตัวดำเนินการคำนวณ + - * / และ %

**การมากเกินเก็บ (overflow) และ การน้อยเกินเก็บ (underflow)**

ดังที่กล่าวไปแล้วว่าข้อมูลแต่ละชนิดสามารถเก็บค่าได้ในช่วงที่จำกัด ถ้าเราพยายามที่จะเก็บค่าที่มากกว่าหรือน้อยกว่าค่าที่ข้อมูลชนิดนั้นรับได้ ก็จะเกิดการมากเกินเก็บ (overflow) หรือ การน้อยเกินเก็บ (underflow)

การมากเกินเก็บ (overflow) จะเกิดขึ้นเมื่อเราพยายามจะเพิ่มค่าของตัวแปรให้เกินค่าสูงสุดของมัน ดังนั้นเมื่อเราเพิ่มค่า ค่าที่ได้จะวนกลับไปในช่วงค่าต่ำสุดที่ข้อมูลชนิดนั้นสามารถเป็นไปได้ เช่น ถ้าเราพยายามเพิ่มข้อมูลเข้าตัวแปรชนิด char ที่มีค่า 127 ที่ละ 1 ค่าของตัวแปรนั้นจะเปลี่ยนเป็น -128, -127, -126 ตามลำดับ

การน้อยเกินเก็บ (underflow) จะเกิดขึ้นเมื่อเราพยายามจะลดค่าของตัวแปรให้น้อยกว่าค่าต่ำสุดของมัน ดังนั้นเมื่อเราลดค่า ค่าที่ได้จะวนกลับไปในช่วงค่าสูงสุดที่ข้อมูลชนิดนั้นสามารถเป็นไปได้ เช่น ถ้าเราพยายามลดข้อมูลเข้าตัวแปรชนิด char ที่มีค่า -128 ที่ละ 1 ค่าของตัวแปรนั้นจะเปลี่ยนเป็น 127, 126, 125 ตามลำดับ

**จำนวนจริง (Real)**

ในภาษา C++ เราแทนจำนวนจริงด้วยข้อมูลชนิด float (32 บิต), double (64 บิต), และ long double (64, 80, 96, หรือ 128 บิต) เราสามารถทำการบวกลบคูณหารจำนวนจริงเหมือนที่ทำกับจำนวนเต็ม แต่จะไม่สามารถเศษของการหาร (%) ได้ และเวลาหารนั้นโปรแกรมจะเก็บแค่ค่าประมาณเท่านั้น ทำให้อาจจะเกิดการประมาณค่าผิดพลาด (round-off errors) ดังตัวอย่างต่อไปนี้ เราจะเห็นได้ว่า แทนที่ y จะมีค่าเป็น 0 แต่กลับมีค่าเป็นค่าที่ใกล้เคียง 0 มากๆ เพราะค่าที่เก็บใน x เป็นแค่ค่าประมาณของ 1/3.0

โค้ด:

```
int main() {
     double x = 1/3.0;
     double y = (x * 3.0) - 1.0;
     cout << "y = " << y << endl;
     return 0;
}
```

ผลการทำงาน:

```
y = -5.55112e-017
```

**การกำหนดจำนวนตำแหน่งทศนิยมที่แสดงออกทางหน้าจอ (Precision Display)**

เราสามารถกำหนดตำแหน่งทศนิยมของเลขจำนวนจริงที่แสดงออกทางหน้าจอได้ โดยใช้คำสั่ง `std::fixed` และ `std::setprecision` ในไลบรารี iomanip เช่นสมมติว่าเราต้องการแสดงผลของตัวแปร y ซึ่งเป็นจำนวนจริง เราต้อง cout << std::fixed ก่อนหนึ่งบรรทัด จากนั้นในคำสั่ง cout ที่แสดง y ให้ใส่คำสั่ง `std::setprecision(n)` ไว้ก่อนการแสดง y โดยกำหนดให้ n คือตำแหน่งจำนวนทศนิยมที่เราต้องการแสดง ดังตัวอย่างต่อไปนี้

โค้ด:

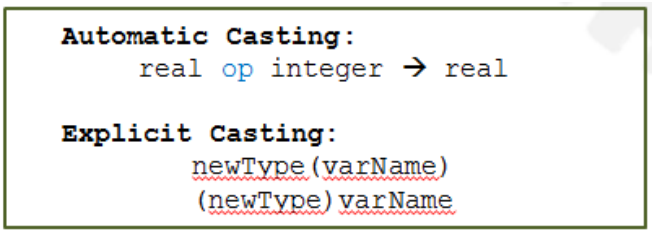
```
#include <iomanip>
int main() {
      double x = 1/3.0;
      double y = (x * 3.0) - 1.0;
      cout << "y = " << y << endl;
      cout << std::fixed;
      cout << "y = " << std::setprecision(5) << y << endl;
      cout << "y = " << std::setprecision(25) << y << endl;

      float a = 1/3.0;
      float b = (a * 3.0) - 1.0;
      cout << "b = " << std::setprecision(25) << b << endl;
      return 0;
}
```

ผลการทำงาน:

```
y = -5.55112e-017
y = -0.00000
y = -0.0000000000000000555111512
b = 0.0000000298023223876953125
```

## 4.4 การเปลี่ยนชนิดข้อมูล (Type casting)

ในภาษา C++ เราสามารถเปลี่ยนชนิดข้อมูลจากชนิดหนึ่งเป็นอีกชนิดหนึ่งได้ ในวิชานี้เราจะเรียนการเปลี่ยนระหว่างจำนวนจริงและจำนวนเต็ม และระหว่างอักขระ และจำนวนเต็ม ในการเปลี่ยนชนิดข้อมูลระหว่างจำนวนจริงและจำนวนเต็มนั้น เราสามารถทำได้ 2 แบบคือแบบอัตโนมัติ (automatic) และแบบชัดแจ้ง (explicit) ในแบบอัตโนมัติ เราสามารถเปลี่ยนข้อมูลโดยการดำเนินการคำนวณ (arithmetic operation) ระหว่างข้อมูลต่างชนิดกัน โดยถ้าเราคำนวณระหว่างจำนวนจริงและจำนวนเต็ม ผลลัพธ์ที่ได้จะเปลี่ยนเป็นจำนวนจริงโดยอัตโนมัติ แต่หากเราคำนวณระหว่างจำนวนจริงด้วยกัน ผลลัพธ์จะเป็นจำนวนจริงเหมือนเดิม เช่นเดียวกันกับหากเราคำนวณระหว่างจำนวนเต็มด้วยกัน ผลลัพธ์จะเป็นจำนวนเต็มเหมือนเดิม ส่วนการเปลี่ยนข้อมูลแบบชัดแจ้งนั้น เราเปลี่ยนชนิดของข้อมูลโดยการใส่ชนิดข้อมูลที่เราต้องการเปลี่ยนให้เป็นในวงเล็บหน้าตัวแปรนั้นๆ หรือเขียนชนิดข้อมูลใหม่ไว้หน้าตัวแปรแล้วใส่วงเล็บให้ตัวแปรนั้น เช่น ถ้าเราต้องการเปลี่ยน x จาก int ให้เป็น double เราก็เขียนว่า (int)x หรือ int(x) ก็เป็นอันเสร็จพิธี ตามตัวอย่างในรูปที่ 7




**รูปที่** 7 รูปแบบการเปลี่ยนชนิดข้อมูล

หมายเหตุ ถ้าเราเปลี่ยนชนิดข้อมูลจากจำนวนจริงเป็นจำนวนเต็มแล้ว เศษทศนิยมในจำนวนจริงนั้นจะหายไปในทุกกรณี (ปัดเศษลง) ดังนั้นเวลาที่เราเขียนโปรแกรม เราจะต้องระวังด้วยว่าตัวแปรที่เราเอามารับข้อมูลหนึ่งๆ นั้นเป็นตัวแปรชนิดใด เช่นถ้าเราเขียนว่า int x = 4.23 เราจะได้ค่า x เป็น 4 ไม่ใช่ 4.23 เพราะ x มีชนิดของข้อมูลเป็นจำนวนเต็ม เราจึงต้องเอาเศษออก

อีกกรณีที่เราต้องระวังคือการใช้วงเล็บ โดยปกติแล้วเราจะคำนวณสิ่งที่อยู่ในวงเล็บก่อน ดังนั้นเราจึงต้องดูด้วยว่าค่าที่อยู่ในวงเล็บเป็นค่าชนิดใด หรือเป็นการคำนวณระหว่างข้อมูลชนิดใด โดยไม่ต้องสนใจสิ่งที่อยู่นอกวงเล็บในเวลานั้น อย่างเช่น ถ้าเราเขียนว่า float y = float(1/5) เราต้องทำในวงเล็บก่อนคือ 1/5 แต่เราจะไม่ได้ค่า 0.2 เพราะ 1 และ 5 ต่างก็เป็นจำนวนเต็ม เพราะฉะนั้นผลลัพธ์ที่ได้ต้องเป็นจำนวนเต็ม คือ 0 (1/5 ได้ 0.2 แต่ปัดเศษลงเพราะเป็นจำนวนเต็ม) เพราะฉะนั้นค่า y จะเป็น 0 ไม่ใช่ 0.2 หากเราต้องการทำให้ y เป็น 0.2 เราสามารถทำได้โดยให้ 1/5 เป็นการหารระหว่างจำนวนจริงกับจำนวนใดๆ นั่นคือ 1.0/5 หรือ 1/5.0 หรือ 1.0/5.0 หรือเราสามารถทำ float(1)/5 โดย float(1) จะทำให้ 1 กลายเป็นจำนวนจริง แล้วเอา 5 ซึ่งเป็นจำนวนเต็มมาหาร ผลลัพธ์ออกมาก็จะได้ 0.2 ซึ่งเป็นจำนวนจริง

ส่วนการเปลี่ยนชนิดข้อมูลระหว่างอักขระ (char) และจำนวนเต็มนั้น ในกรณีที่พบบ่อยคือเราจะใช้เมื่อต้องการแสดงข้อมูลออกทางหน้าจอ เรารู้มาก่อนว่า char ถือว่าเป็นชนิดหนึ่งของจำนวนเต็ม โดยค่าที่อยู่ในอักขระจะเป็นค่า ASCII code ของอักขระนั้นๆ ทำให้เราสามารถดำเนินการคำนวณ char โดยใช้ค่า ASCII ของมันได้เลยโดยไม่ต้องเปลี่ยนแปลงอะไร เช่น ถ้าเราเขียนว่า

```
char thisChar = 'a';
thisChar++;
```

เราจะได้ thisChar เป็นอักขระ b เพราะเมื่อเราทำ thisChar++ โปรแกรมจะบวกค่า ASCII ของอักขระ ‘a’ ไปหนึ่ง ทำให้ค่า ASCII ของมันกลายเป็นค่า ASCII ของ ‘b’ นั่นเอง

อย่างไรก็ตาม เมื่อเราต้องการแสดงผลลัพธ์ออกทางหน้าจอ ถ้าเรา cout << thisChar ออกไป เราจะได้ตัวอักขระออกมา แต่ถ้าเราต้องการแสดงค่า ASCII code ของอักขระนั้นๆ เราต้องทำการเปลี่ยนชนิดข้อมูลก่อน โดยการเปลี่ยนข้อมูลแบบชัดแจ้ง (explicit type casting) คือเขียน cout << (int)thisChar หรือ cout << int(thisChar) ก็ได้

**ค่าบูลลีน (Boolean)**

ข้อมูลชนิดบูลีนเป็นค่าทางตรรกะ คือมีค่าเป็นจริง (true) หรือเท็จ (false) โดยปกติแล้วเราจะใช้ค่าชนิดนี้ในการทำงานกับเงื่อนไข (condition) ซึ่งจะกล่าวในบทต่อไป ส่วนบทนี้ให้นักศึกษาทราบถึงลักษณะทั่วไปของข้อมูลก็พอ การเก็บค่าบูลีนในภาษา C++ จะเก็บเป็นค่าตัวเลข 1 สำหรับค่าจริง (true) และ 0 สำหรับค่าเท็จ (false) โดยเมื่อเราใช้คำสั่ง cout เพื่อแสดงค่าของตัวแปรบูลีน โปรแกรมจะแสดงออกมาเป็น 1 หรือ 0 ไม่ใช่ true หรือ false การให้ค่ากับตัวแปรบูลีน เราสามารถให้ค่าได้ 3 แบบคือ

1) ให้ค่า true หรือ false ได้เลย
2) ให้ค่าเป็นตัวเลข โดยการให้ค่าเป็น 0 จะทำให้ตัวแปรมีค่าเป็น false นอกเหนือจากนั้น (non-zero) จะให้ค่าเป็น true
3) ให้ค่าโดยการเปรียบเทียบ เช่น 2 > 3 ถ้าผลของการเปรียบเทียบเป็นจริง ตัวแปรก็จะมีค่าเป็น true แต่ถ้าผลเป็นเท็จ ตัวแปรก็จะมีค่าเป็น false

**ตัวอย่างที่** 1

โจทย์: จงหาผลรันของโปรแกรมต่อไปนี้

โค้ด:

```
int main() {
    bool a = true, b = false;
    bool c = 3 < 4;
    bool d = 3 > 4;
    bool e = 0.4, f = 0;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "c = " << c << endl;
    cout << "d = " << d << endl;
    cout << "e = " << e << endl;
    cout << "f = " << f << endl;
    return 0;
}
```

ตัวอย่างนี้แสดงการให้ค่าแบบต่างๆ ของตัวแปรชนิดบูลีน ตัวแปร a และ b ถูกให้ค่าแบบที่ 1 ผลการรันจึงเป็น 1 และ 0 ตามลำดับ (ตามที่กล่าวไปแล้วว่า ค่าที่ถูกเก็บในตัวแปรจะเป็นค่า 1 หรือ 0) ส่วนตัวแปร c และ d ถูกให้ค่าโดยการเปรียบเทียบ เนื่องจาก 3 มีค่าน้อยกว่า 4 จึงทำให้ตัวแปร c และ d มีค่า true และ false และผลการรับเป็น 1 และ 0 ตามลำดับ และสุดท้ายตัวแปร e และ f ถูกให้ค่าเป็นตัวเลข ตัวแปร e ถูกให้ค่าเป็น 0.4 แม้จะมีค่าไม่ถึง 1 แต่ไม่ใช่ 0 จึงมีค่าเป็น 1 (true) และ ตัวแปร f มีค่าเป็น 0 (false) ตามลำดับ

ผลการทำงาน:

```
a = 1
b = 0
c = 1
d = 0
e = 1
f = 0
```

ดังที่ได้กล่าวไปแล้วว่าแม้ค่าบูลีนจะเป็น true หรือ false แต่การเก็บค่าและการแสดงผลของตัวแปรจะเป็น 1 หรือ 0 หากเราต้องการให้แสดงผลเป็น true หรือ false เราต้องใช้คำสั่ง std::boolalpha ก่อนการแสดงผลของตัวแปรบูลีน ดังโค้ดต่อไปนี้

โค้ด:

```
int main() {
      bool b = true;
      cout << std::boolalpha << b << '\n';
      cout << std::noboolalpha << b << '\n';
      return 0;
}
```

ผลการทำงาน:

```
true
1
```

**ค่าคงที่ (Constant)**

ค่าคงที่คือตัวแปรชนิดใดๆ ที่เราไม่สามารถเปลี่ยนค่าของมันได้ ณ เวลาที่โปรแกรมรันอยู่ เราจะกำหนดค่าคงที่ได้โดยวางคำสั่ง const ไว้หน้าการประกาศตัวแปรตามปกติ ดังรูปที่ 8

```
const type varName = value;
```

**รูปที่ 8** รูปแบบการประกาศตัวแปรให้เป็นค่าคงที่

ถ้าเราเขียนโปรแกรมที่มีคำสั่งเปลี่ยนค่าของค่าคงที่แล้ว โปรแกรมนั้นจะคอมไพล์ไม่ผ่าน ไม่สามารถรันได้

## 4.5 การนำเข้าข้อมูลพื้นฐาน

เราใช้คำสั่ง cin เพื่อนำเข้าข้อมูลจากแป้นพิมพ์มายังตัวแปรต่างๆ ในโปรแกรม ซึ่งลักษณะการใช้จะคล้ายกับคำสั่ง cout มาก ยกเว้นว่าเครื่องหมายที่ตามหลัง cin จะเป็นเครื่องหมาย >> เพื่อป้องกันการสับสน เราสามารถจดจำการใช้เครื่องหมายสำหรับ cin และ cout ได้โดยจำว่า สำหรับ cout เราแสดงผลออกไปยังหน้าจอ ดังนั้นเครื่องหมายจึงต้องชี้ไปที่ cout เป็น cout << x แต่สำหรับ cin เรานำค่าเข้ามาเก็บไว้ในตัวแปร ดังนั้นเครื่องหมายจึงต้องชี้ไปที่ตัวแปร เป็น cin >> x

ในการอ่านข้อมูลเข้ามานั้น ข้อมูลจะถูกเก็บไว้ในตัวแปรอย่างน้อยหนึ่งตัวเสมอ นั่นคือเราเอาตัวแปรไปรองรับเก็บค่าที่อ่านเข้ามาจากแป้นพิมพ์ เราไม่สามารถจะ cin ค่าตรงๆ ได้ ซึ่งต่างกับ cout ในเมื่อมีการใช้ตัวแปรในคำสั่ง cin เราจึงต้องประกาศตัวแปรที่จะเอามาเก็บค่าจากแป้นพิมพ์ก่อนที่จะสั่ง cin เสมอ เช่น ถ้าเราต้องการอ่านค่าจำนวนเต็มเข้ามา 4 ตัว เราต้องประกาศ

```
int a, b, c, d;
```

ก่อนที่จะนำเข้าข้อมูลโดย

```
cin >> a >> b >> c >> d;
```

เสมอ อีกประการหนึ่งที่เราต้องระวังคือ การนำเข้าค่าหลายค่าในคราวเดียวกัน เราต้องใช้เครื่องหมาย >> คั่นระหว่างตัวแปรต่างๆ เราไม่สามารถใช้เครื่องหมาย , คั่นได้ นั่นคือ เราไม่สามารถเขียนว่า cin >> a,b,c,d ได้ โปรแกรมจะสามารถคอมไพล์ผ่าน แต่จะรับค่ามาแค่ค่าแรกเท่านั้น ทำให้โปรแกรมทำงานผิดพลาด นอกจากนี้ค่าที่เราอ่านเข้ามานั้นจะถูกเก็บในตัวแปรต่างๆ ตามลำดับที่อ่านเข้ามา เช่นถ้าเราเขียนว่า

```
cin >> a >> b >> c >> d;
```

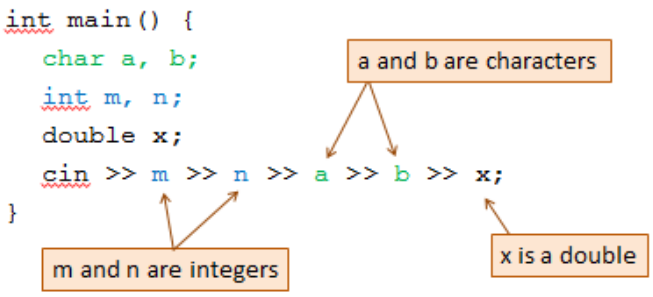
และเราใส่อินพุตที่แป้นพิมพ์เป็น 1 2 3 4 ค่าของตัวแปร a, b, c, d จะเป็น 1, 2, 3, 4 ตามลำดับ

**การทำงานของ cin**

เมื่อเราใส่อินพุตผ่านทางแป้นพิมพ์ อินพุตจะถูกผ่านเข้ามาในโปรแกรมด้วย istream ซึ่งจะมีวัตถุไปรองรับแต่ละอักขระใน istream นั้น เมื่อเราใช้ cin โปรแกรมจะทำการแปลผลข้อมูลที่อ่านเข้ามาตามชนิดของตัวแปรที่เอาไปรองรับโดยอัตโนมัติ เช่นถ้า istream ที่ใส่เข้ามาเป็น

```
46\n
```

นั่นคือมีช่องว่าง (space) 4 ตัว จากนั้นเป็นอักขระ '4' '6' และอักขระพิเศษ '\n' ถ้าเราต้องการรับอินพุตเป็น int มันก็จะพยายามหาตัวเลขจำนวนเต็มเพื่อเอามาใส่ในตัวแปรชนิดจำนวนเต็มที่มารองรับนั้น ในกรณีนี้ มันจะไม่สนใจช่องว่าง 4 ตัวแรก แต่พอเจอตัวเลข มันจะอ่านอักขระที่เป็นตัวเลขเข้ามาเรื่อยๆ จนกว่าจะเจออักขระที่ไม่ได้เป็นตัวเลขแล้วมันจึงหยุดอ่าน จากนั้นมันจึงประมวลผลและแปลงอักขระต่างๆ ที่อ่านเข้ามาให้เป็นเลขจำนวนเต็ม ได้เป็น 46 เช่นเดียวกับตัวแปรชนิดอื่นๆ เช่น char, string, double รวมทั้งการรับตัวแปรหลายๆ ค่าพร้อมกัน cin ก็จะทำงานค้นหาค่าอินพุตที่เหมาะสมกับตัวแปรที่มารองรับแต่ละตัวไปตามลำดับด้วย ดังตัวอย่างในรูปที่ 9




**รูปที่ 9** ตัวอย่างการนำเข้าตัวแปรหลายชนิดพร้อมกัน

**จากรูปที่ 9 ถ้าเราพิมพ์อินพุต** 1.5 2 3 h 4.5 **เข้าทางแป้นพิมพ์เราจะได้ค่า**

ตารางที่ 3

**ตารางที่ 3** ค่าของตัวแปรหลังจากการนำเข้าข้อมูล

| ตัวแปร | ค่า |
|---|---|
| a | '3' |
| b | 'h' |
| m | 1 |
| n | 2 |
| x | 4.5 |

ลำดับแรก โปรแกรมรับค่า m และ n มาก่อน ดังนั้นสองค่าแรก 1.5 และ 2 จึงถูกเก็บใน m และ n แต่เพราะว่า m และ n เป็นชนิด int ซึ่งเป็นจำนวนเต็ม โปรแกรมเลยปัดเศษ 1.5 ให้เหลือ 1 แล้วจึงเก็บในตัว

แปร m ต่อมาโปรแกรมรับค่า a และ b มา ดังนั้นค่าถัดมาคือ 3 กับ h จะถูกเก็บไว้ใน a และ b ตามลำดับ แต่เพราะ a และ b จะต้องเก็บอักขระ โปรแกรมจึงเก็บ 3 ที่เป็นอักขระเลขสาม ไม่ใช่ค่าทางคณิตศาสตร์ 3 เข้าในตัวแปร a ส่วนตัวสุดท้าย x ได้รับค่า 4.5 มันถูกประกาศเป็น double ซึ่งสามารถเก็บจำนวนจริงได้ x เลยมีค่า 4.5 ตามที่รับมาจากแป้นพิมพ์เลย

ในการใส่ค่าผ่านทางแป้นพิมพ์สำหรับการรับค่าหลายค่าในคราวเดียวกัน เราสามารถใส่ได้สองแบบคือ การเว้นวรรคระหว่างค่าต่างๆ หรือการกด enter ระหว่างค่าต่างๆ โปรแกรมจะสามารถรู้เองว่าจะต้องรับค่าเข้ามากี่ค่า เพราะฉะนั้นถึงเราจะกด enter โปรแกรมก็จะไม่ทำงานต่อไปจนกว่าจะรับค่าตามจำนวนที่มันต้องการเสร็จสิ้นก่อน

**ตัวอย่างที่ 2** การนำเข้าและส่งออกข้อมูล

โค้ด:

```
int main() {
      int m,n;
      cin >> m >> n;
      cout << "m + n = " << m+n << endl;
}
```

ถ้าเราใส่ค่า 3 4 เข้าทางแป้นพิมพ์ เราจะได้ค่า m เท่ากับ 3 และ n เท่ากับ 4 และได้ผลรันดังนี้

ผลการทำงาน:

```
7
```

จากที่เราเรียนไปแล้ว การนำเข้าข้อมูลจะต้องมีตัวแปรมารองรับ ดังนั้นเราจึงต้องประกาศตัวแปรก่อนนำเข้าข้อมูล นอกจากนี้ ในหลายกรณี เราควรจะมีการแสดงข้อความเพื่อขออินพุตทางหน้าจอด้วย ข้อความนี้จะต้องอยู่ก่อนคำสั่งนำเข้าข้อมูลเนื่องจากผู้ใช้จะต้องเห็นข้อมูลก่อนโปรแกรจะหยุดเพื่อรอรับข้อมูล เราสรุปได้ว่าการนำเข้าข้อมูลโดยทั่วไปนั้นจะมี 3 ขั้นตอนคือ

1) ประกาศตัวแปร
2) cout แสดงข้อความขอข้อมูลจากผู้ใช้
3) cin ค่าที่ต้องการนำเข้า

**ตัวอย่างที่ 3** ลักษณะทั่วไปของการนำเข้าข้อมูล

โค้ด:

```
1  int main() {
2        int h, w;
3        cout << "Please enter your height and weight: ";
4        cin >> h >> w;
5        cout << "height = " << h << " weight = " << w<< endl;
6        return 0;
7  }
```

ตัวอย่างนี้โปรแกรมขอค่าน้ำหนักและส่วนสูงของผู้ใช้ ถ้าเราไม่มีคำสั่งในบรรทัดที่ 3 ซึ่งแสดงข้อความขอข้อมูลจากผู้ใช้ ผู้ใช้ก็จะไม่ทราบว่าเราต้องการนำเข้าข้อมูลอะไร เมื่อโปรแกรมรัน จะได้ผลรันก่อนใส่อินพุตดังนี้

```
Please enter your height and weight:
```

จากนั้นโปรแกรมจะหยุดรอรับอินพุต และถ้าเราใส่ค่า 160 50 เข้าทางแป้นพิมพ์ เราจะได้ค่า h เท่ากับ 160 และ w เท่ากับ 50 โดยผลรันแบบครบถ้วนของโปรแกรมนี้เป็นดังนี้

ผลการทำงาน:

```
Please enter your height and weight: 160 50
height = 160 weight = 50
```

หมายเหตุ: เลข 160 50 ตัวหนาที่อยู่ในผลการทำงานข้างบนคือค่าที่ผู้ใช้พิมพ์เข้าไปในคอมพิวเตอร์ ไม่ใช่ค่าที่โปรแกรมแสดงออกมาโดยอัตโนมัติ

**สรุปสิ่งที่ได้เรียนมา**

1) โครงสร้างการเขียนโปรแกรมของภาษา C++

2) การส่งออกข้อมูลไปยังหน้าจอใช้คำสั่ง cout กับเครื่องหมาย << คั่นระหว่างค่าต่างๆ ที่ต้องการส่งออก
3) การนำเข้าข้อมูลจากแป้นพิมพ์ใช้คำสั่ง cin กับเครื่องหมาย >> คั่นระหว่างตัวแปรต่างๆ ที่ต้องการรับค่าเข้ามา
4) ตัวแปร ชนิดของตัวแปร และคุณสมบัติของตัวแปร ควรเลือกใช้ให้เหมาะกับชนิดและขนาดข้อมูลที่เราต้องการใช้ และต้องประกาศตัวแปรก่อนนำไปใช้เสมอ
5) การดำเนินการคำนวณ (arithmetic operation) สำหรับข้อมูลชนิดตัวเลข จะมีลำดับการคำนวณ ให้ระวังการมากเกินเก็บ (overflow) และ การน้อยเกินเก็บ (underflow) ด้วย

**แบบฝึกหัดท้ายบท**

จะเพิ่มเติมทีหลัง