Variable and Data Type

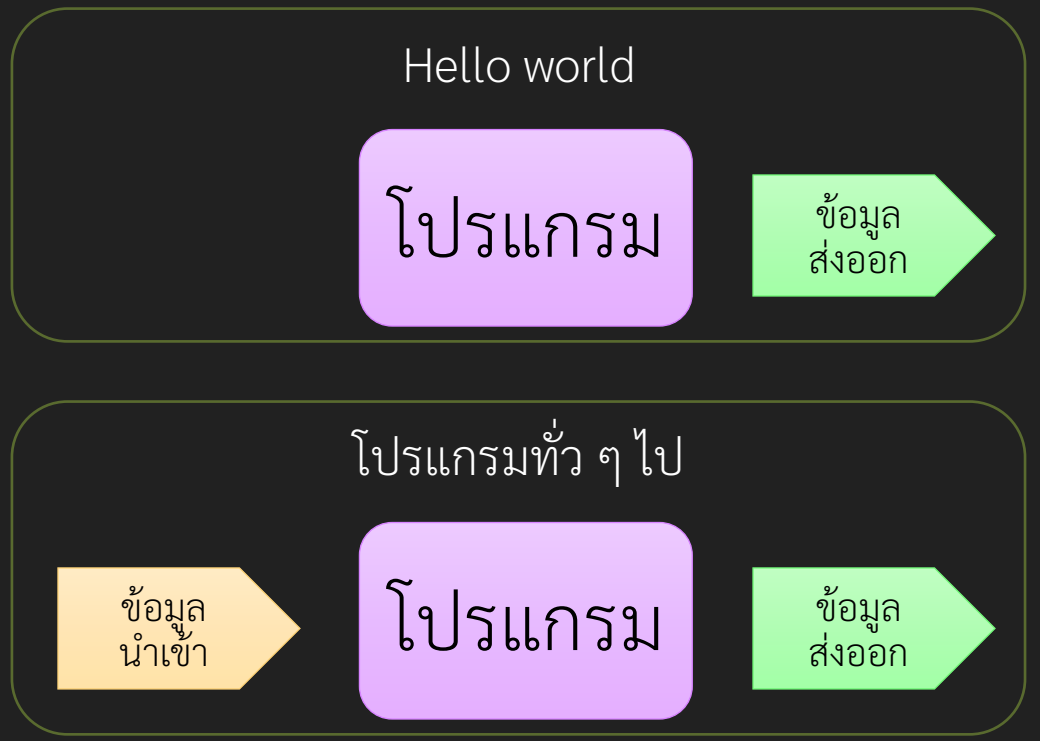
2110104 Computer Programming

Nattee Niparnan

Department of Computer Engineering
Chulalongkorn University

ตัวแปร (variable) คืออะไร?

- เป็นกล่องที่ใช้เก็บข้อมูลต่าง ๆ
  - เก็บค่าต่าง ๆ ที่ต้องเอามาใช้งาน
- ช่วยให้เขียนโปรแกรมที่มีประโยชน์ได้มากขึ้น
  - โปรแกรมคอมพิวเตอร์คือขั้นตอนในการทำงาน ประโยชน์ของมันมาจากการประมวลผลข้อมูลต่าง ๆ
  - มี ข้อมูลนำเข้า นำมาผ่านขั้นตอน แล้วสร้างข้อมูลส่งออก
  - ตัวแปรใช้เพื่อระบุค่าต่าง ๆ ที่อาจเปลี่ยนแปลงได้จากข้อมูลนำเข้า
    - ใช้งานอย่างอื่นก็ได้ เช่น ช่วยให้อ่านเข้าใจง่ายขึ้น



การใช้งานตัวแปร

- ก่อนใช้งานจะต้องประกาศตัวแปร โดยระบุชื่อและประเภทของตัวแปร
  - รูปแบบคำสั่งที่ใช้คือ `ประเภทตัวแปร ชื่อตัวแปร;`
  - เช่น `double a1;` เป็นการประกาศว่าจะมีตัวแปรชื่อ `a1` เป็นประเภท `double`
- ในภาษา C++ มีประเภทของตัวแปรอยู่มากมาย
  - แต่ละประเภทมีขอบเขตข้อมูลที่เก็บได้แตกต่างกัน และใช้เนื้อที่ (หน่วยความจำ) ในการเก็บข้อมูลแตกต่างกัน
  - วันนี้เอา 4 ประเภทก่อน

ชื่อของประเภท	ใช้สำหรับ	ขอบเขตข้อมูล	หน่วยความจำที่ใช้
int	ตัวเลขจำนวนเต็ม	$-(2^{32})$ ถึง $(2^{32}-1)$	4 bytes
double	ตัวเลขแบบมีทศนิยม	ประมาณ 16 หลัก (รวมทศนิยม)	8 bytes
bool	ค่า “จริง” หรือ “เท็จ”	{true, false}	1 bytes
char	ตัวอักษรพื้นฐาน 1 ตัว (หรือตัวเลขก็ได้)	0 ถึง 255	1 bytes

มาลองใช้ double กันก่อน

- double เอาไว้เก็บตัวเลขจำนวนจริง
- สามารถเก็บข้อมูลได้ประมาณ 16 หลัก (รวมทั้งส่วนก่อนทศนิยมและหลังทศนิยม)

- ทำไมเก็บได้แค่ 16 หลัก?
  - ตัวแปรต้องใช้พื้นที่หน่วยความจำในการเก็บ
  - double เป็นตัวแปรพื้นฐาน โดนกำหนดไว้ว่าใช้พื้นที่ไม่เกิน 8 bytes ซึ่งเพียงพอสำหรับเก็บข้อมูลแค่ประมาณ 16 หลัก
  - ถ้าอยากเก็บมากกว่านี้ ต้องใช้ตัวแปรประเภทอื่น ซึ่งอาจจะใช้พื้นที่เยอะกว่า 8 ก็ได้ ➔ เปลือง!

ตัวอย่าง

- โปรแกรมหาค่าเฉลี่ยของตัวเลข 3 ตัว

```
#include <iostream>

using namespace std;

int main() {
  double a1;
  double a2;
  double a3;
  cout << "Enter the first value ";
  cin >> a1;
  cout << "Enter the second value ";
  cin >> a2;
  cout << "Enter the third value ";
  cin >> a3;
  cout << "Average is " << (a1 + a2 + a3) / 3 << endl;
}
```

การรับข้อมูลจาก Keyboard

- การแสดงผลจะใช้คำสั่ง `std::cin`
- ตามด้วยเครื่องหมาย `>>` แล้วตามด้วยตัวแปรที่ใช้เก็บข้อมูล เช่น
  - `std::cin >> a1;`
  - อย่าลืม! ตัวแปรที่จะรับข้อมูล ต้องประกาศก่อนใช้งาน

```
#include <iostream>

using namespace std;

int main() {
  double a1;
  cout << "Enter the first value ";
  cin >> a1;
  cout << "Enter the second value ";
  cin >> a2;
  cout << "Enter the third value ";
  cin >> a3;
  double a2;
  double a3;
}
```

ผิด!!! a2 ยังไม่ได้ประกาศก่อนใช้

a3 ก็ด้วย

Compilation Error

- โปรแกรมภาษา C++ จะถูก compile ก่อนใช้งาน
- การ compile จะตรวจสอบด้วยว่าโปรแกรมที่เขียนมานั้น ถูกต้องหรือไม่ หากไม่ถูก จะเกิด compilation error (การที่ C++ แจ้งว่าโปรแกรมของเราผิดกฎต่าง ๆ)
- คนที่แสดง Compilation error คือ Compiler
  - อาจจะอ่านยากหน่อย แต่ค่อย ๆ ลองดูไป

```
D:\Dae\Teaching\Com Prog CEDT\01-code\01-01-average.cpp:10:10: error: 'a2' was not declared in this scope
   cin >> a2;
          ^~
D:\Dae\Teaching\Com Prog CEDT\01-code\01-01-average.cpp:10:10: note: suggested alternative: 'a1'
   cin >> a2;
          ^~
          a1
D:\Dae\Teaching\Com Prog CEDT\01-code\01-01-average.cpp:12:10: error: 'a3' was not declared in this scope
   cin >> a3;
          ^~
```

การใช้ตัวแปรช่วยให้เขียนโปรแกรมง่ายขึ้น

```
#include <iostream>
using namespace std;
int main() {
  double a1, a2, a3;
  cout << "Enter the first value ";   cin >> a1;
  cout << "Enter the second value ";  cin >> a2;
  cout << "Enter the third value ";   cin >> a3;
  //calculation of the average of a1, a2, and a3
  double sum = a1+a2+a3;
  double average = sum / 3;
  cout << "Average is " << average << endl;
}
```

มีการสร้างตัวแปร sum และ average มาเพื่อเก็บค่าเพิ่มเติม

- สร้างตัวแปรเพื่อใช้เก็บค่าต่าง ๆ แยกตามหน้าที่ช่วยให้เวลาคนอื่น (หรือตัวเราเองในอนาคต) มาอ่านแล้วเข้าใจมากขึ้น

สังเกตเห็นอะไรบ้าง?

ประกาศตัวแปรประเภทเดียวกันพร้อมกันหลายตัวได้ โดยใช้รูปแบบ

ประเภทตัวแปร ชื่อ, ชื่อ, ..., ชื่อ

```
#include <iostream>
using namespace std;
int main() {
  double a1, a2, a3;
  cout << "Enter the first value ";   cin >> a1;
  cout << "Enter the second value ";  cin >> a2;
  cout << "Enter the third value ";   cin >> a3;
  //calculation of the average of a1, a2, and a3
  double sum = a1+a2+a3;
  double average = sum / 3;
  cout << "Average is " << average << endl;
}
```

คำสั่งมากกว่า 1 คำสั่งอยู่บรรทัดเดียวกันได้

การมี ; ทำให้ C++ แยกแยะได้ว่าคำสั่งจบที่ใด

เราสามารถกำหนดค่าเริ่มต้นให้ตัวแปรได้ตอนประกาศ โดยใช้รูปแบบ

ประเภทตัวแปร ชื่อตัวแปร = ค่าเริ่มต้น

อะไรก็ตามที่อยู่หลังเครื่องหมาย // คือ comment มีหน้าที่เอาไว้ ให้เราจดบันทึกต่าง ๆ ได้

C++ จะไม่สนในส่วน comment โปรแกรมเมอร์ที่ดีมักจะเขียน comment ไว้

การประกาศตัวแปรพร้อมระบุค่า

- เราสามารถประกาศตัวแปรพร้อมค่าเริ่มต้นได้โดยใช้
  - `ประเภทตัวแปร ชื่อตัวแปร = ค่าเริ่มต้น;`
  - มีรูปแบบอื่น ๆ อีกในการกำหนดค่า และประกาศตัวแปร ไว้คุยกันวันหลัง
- ทำไมต้องระบุค่าตอนประกาศ?
  - ไม่ระบุก็ได้ แต่ถ้าไม่ระบุ ค่าของตัวแปรก่อนระบุอาจจะเป็นอะไรก็ได้
  - มักจะเป็นจุดที่ทำให้โปรแกรมทำงานผิดพลาด

```cpp
#include <iostream>
using namespace std;
int main() {
  int x = 0;
  int y;
  cout << x << endl;
  cout << y << endl;
}
```

Y ไม่ได้ระบุค่ามาก่อน

output

```
0
16
```

16 เป็นค่ามั่ว ๆ ของ y ถ้าลองใหม่คราวหลังอาจจะไม่ใช่ 16 ก็ได้

การตั้งชื่อตัวแปร

- ชื่อตัวแปรสามารถใช้ได้เพียง ตัวอักษรภาษาอังกฤษ ตัวเลข และเครื่องหมาย _ (ขีดล่าง หรือ underscore)
  - และต้องไม่ขึ้นต้นด้วยตัวเลข
  - ตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ถือว่าแตกต่างกัน
  - ต้องไม่ซ้ำกับคำสงวน

ถูก	ผิด
`x`	`1a`
`x1`	`x!`
`TheCar`	`!x`
`sum_of_user`	`B(3)`
`____haha1`	`sum-of-something`
`_1_`	`double`

การกำหนดค่าให้กับตัวแปร (Assignment)

- คำสั่งในการกำหนดค่าคือ ชื่อตัวแปร = ค่าที่ต้องการ
- ให้มองว่า ชื่อตัวแปร คือกล่องที่จะเก็บข้อมูล และคำสั่งดังกล่าวคือการเอาค่าที่ต้องการมาเก็บในกล่องนั้น
- ตัวอย่าง

```
x = 3;
y = 3+5*7-2;
z = x + 1;
```

เอาค่า 3 ไปเก็บไว้ใน x

เอาค่า 36 ไปเก็บไว้ใน y

เอาค่า x+1 ไปเก็บไว้ใน z

มาลองใช้ int กันบ้าง

- ตัวแปรประเภท int เอาไว้เก็บจำนวนเต็มต่าง ๆ
- เก็บจำนวนแบบมีทศนิยมไม่ได้
  - แล้วถ้าพยายามเอาไปเก็บจะเป็นอย่างไร? เช่น `int b = 30.99;`

```
#include <iostream>
using namespace std;
int main() {
  double a = 30.99;
  int b = 30.99;
  cout << a << " , " << b << endl;

  a = 12345678900;
  b = 12345678900;
  cout << a << " , " << b << endl;
}
```

output

```
30.99 , 30
1.23457e+10 , -539222988
```

การแปลงค่า

- คำสั่งกำหนดค่าจะเอาค่าที่ต้องการมาเก็บในตัวแปร
- ถ้าประเภทของตัวแปรกับค่ามันเป็นคนละประเภทจะมีการพยายามแปลง

ตัวเลขในภาษา C++

- `1.23457e+10` คืออะไร?
  - วิธีการเขียนตัวเลขแบบ scientific ความหมายคือ $1.23457 * 10^{10}$
- ตัวอย่าง
  - `1.32e+4` คือ $1.32 * 10^{4}$ ซึ่งเท่ากับ `13200`
  - `9.42e-2` คือ $9.42 * 10^{-2}$ ซึ่งเท่ากับ `0.0942`

12345678900 ??

```
double a = 12345678900;
int b = 12345678900;
cout << a << " , " << b << endl;
```

```
1.23457e+10 , -539222988
```

- ทำไม a ได้ `1.23457e+10`?
  - ทำไมไม่เป็น `1.23456789e+10` หรือทำยังไงให้พิมพ์ `12345678900`
  - เพราะโดยปรกติแล้ว `cout` จะพิมพ์แบบย่อ ๆ มาให้สำหรับ `double`
  - อยากให้พิมพ์ยาว ๆ ให้ใช้ `setprecision(xxx)` โดย `xxx` คือจำนวนหลักที่ต้องการ
  - `cout << setprecision(15) << a << endl;`
  - ลองดู
- ทำไม b ได้ -5392229888?
  - เพราะ 12,345,678,900 มันมากกว่าขอบเขตค่าที่ int รับได้
  - ขอบเขตของ int คือ -2,147,483,648 ถึง 2,147,483,647
  - แล้วทำไมมันติดลบ???
  - เพราะเกิดสิ่งที่เรียกว่า Overflow ขึ้น (ไว้ไปคุยรายละเอียดกันในบท Data Representation)

งั้นทำไมไม่ใช้ double ไปเลย?

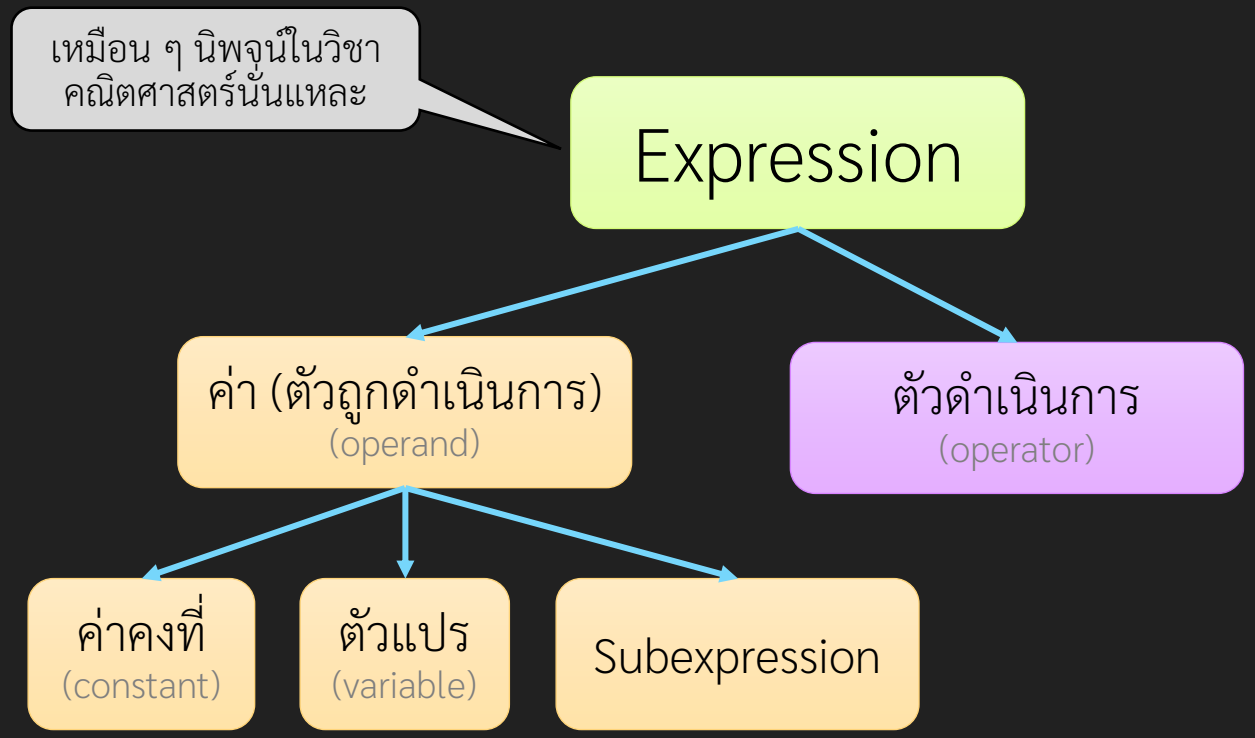
- เหมือนว่า double จะเก็บข้อมูลได้เยอะกว่า int ตลอด ทำไมถึงต้องใช้ int ด้วย
- คำตอบ:
  - เนื่องจาก double เองเก็บตัวเลขได้แค่ประมาณ 16 หลัก เพราะฉะนั้นค่าของ double จึงเป็นการประมาณ
  - แถมมีปัญหาอื่น ๆ อีก
  - เดี๋ยวจะได้เห็นละเอียด ๆ ในเรื่อง Data Representation
- ตัวอย่าง 49.0 / 23.0
  - ค่าที่แท้จริงคือ 2.130434782608695652173913043478 3...
  - ค่าที่เก็บใน double เป็นประมาณ ๆ 2.13043478260 87

Expression

นิพจน์ (Expression)

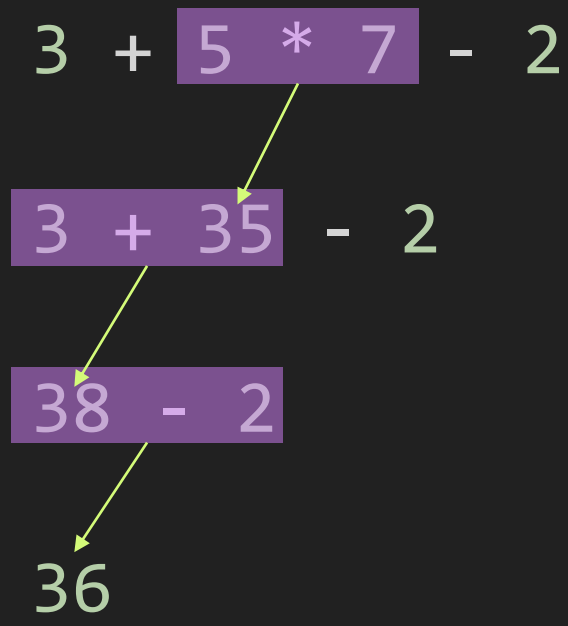
- คำสั่งต่าง ๆ ของภาษา C++ จะมีการประมวลผลออกมาเป็นค่าต่าง ๆ ได้
- Expression คือกลุ่มคำสั่งที่ประกอบด้วย ค่า, ตัวแปร, ตัวดำเนินการ (operator) ต่าง ๆ ที่เราสามารถคำนวณค่าของทั้ง Expression ออกมาได้
- เช่น 3+5*7-2 เป็น expression ที่เกิดจากค่า 4 ค่า (3, 5, 7, 2) มาผ่านตัวดำเนินการ 3 ตัว คือ +, *, - เพื่อคำนวณค่าออกมา (ได้เป็น 36)
- ตัวอย่างการใช้งานที่เห็นมาแล้วเช่น `y = 3+5*7-2;` เป็นการใช้ expression ดังกล่าว (คือ `3+5*7-2`) มาเป็นค่าในคำสั่งกำหนดค่าให้กับ `y`

โครงสร้างของ Expression และการคำนวณค่า



- ตัวเลขต่าง ๆ
- ค่าเฉพาะ

- Expression คำนวณตามลำดับความสำคัญของ operator
  - เหมือน ๆ กับที่นิยามไว้ในวิชาเลข



Operator ต่าง ๆ และลำดับความสำคัญ

ความสำคัญ	Operator	ความหมาย
1	a++ a-- xxx()	Postfix increment Postfix decrement Function call
2	++a --a	Prefix increment Postfix decrement
3	* / %	คูณ หาร หารเอาเศษ
4	+ -	บวก ลบ
5	= += -= *= /= %=	assignment ต่าง ๆ

- ลำดับความสำคัญเป็นตัวระบุว่าทำอะไรก่อนหลัง
- จริง ๆ มีอีกเยอะ แต่เอาที่ใช้บ่อย ๆ ในช่วงนี้ก่อน ตัวอื่น ๆ เดี๋ยวจะค่อย ๆ เจอ

Operator แปลก ๆ ในภาษา C++

- การหารเอาเศษ `a % b` คือการเอา a หารด้วย b โดยมีค่าเป็นเศษของการหาร
  - เช่น `5 % 3` มีค่าเป็น 2
- ทั้ง a และ b ต้องเป็น expression แบบจำนวนเต็ม
  - `3.77 % 2.15` ไม่ได้ (compile ไม่ผ่าน)
- ตัวอย่างการใช้งานเช่น
  - ต้องการทราบหลักหน่วยของตัวเลข ใช้ % 10
    - เช่น `321478 % 10` มีค่าเป็น 8
  - ต้องการทราบว่าหารด้วย x ลงตัวหรือไม่ ใช้ `% x` แล้วดูว่าค่าเป็น 0 หรือไม่

Operator แปลก ๆ ในภาษา C++

- Prefix/postfix increment/decrement operator
  - ต้องใช้กับตัวแปร
  - ทำการ เพิ่ม/ลด ค่าของตัวแปรไป 1 หน่วย
  - ค่าของ expression จะเป็นค่า หลัง/ก่อน เพิ่ม/ลด

```
#include <iostream>
using namespace std;
int main() {
  int x = 5;
  int y = 0;
  cout << "(1) x before " << x << endl;
  y = ++x;       // The operator
  cout << "(2) x after "  << x << endl;
  cout << "(3) y "  << y << endl;
}
```

output

```
Value before 5
Value after 6
Y is 6
```

Prefix/postfix increment/decrement operator

expression	(1)	(2)	(3)
y = ++a	5	6	6
y = a++	5	6	5
y = --a	5	4	4
y = a--	5	4	5

```
#include <iostream>
using namespace std;
int main() {
  int a = 5;
  int y = 0;
  cout << "(1) a before " << a << endl;
  // expression
  cout << "(2) a after "  << a << endl;
  cout << "(3) y "  << y << endl;
}
```

- ใช้ทำอะไร?
  - การเพิ่ม/ลด 1 หน่วยเป็นสิ่งที่เกิดขึ้นบ่อยมาก
  - จะเขียนเป็น a = a+1 ก็ได้ แต่ ++a เขียนสั้นกว่า เข้าใจง่ายกว่า

Function Call คืออะไร?

- เหมือนฟังก์ชันในวิชาเลข เช่น sin, cos, tan
  - C++ มีฟังก์ชันให้เรียกใช้มากมาย
  - เราสร้างเองได้ (และเราจะสร้างบ่อยมาก ๆ ๆ ๆ ด้วยในเร็ว ๆ นี้)
- ฟังก์ชัน จะ รับข้อมูลนำเข้า แล้วคำนวณค่าตามที่ฟังก์ชันนั้นกำหนดแล้วคืนค่าออกมา
  - ค่าที่รับเข้า และ ค่าที่ส่งออก มีประเภทต่าง ๆ กัน
  - ดูรายการฟังก์ชันทางคณิตศาสตร์ที่ C++ มีมาให้แล้วได้ที

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main() {
  double a1;
  cout << M_PI << endl;
  a1 = cos(M_PI);
  cout << a1 << endl;
  cout << sin(3.14) << endl;
  cout << sin(3.141) << endl;
  cout << sin(3.1415) << endl;
  cout << sin(M_PI) << endl;
}
```

output

```
3.14159
-1
0.00159265
0.000592654
9.26536e-05
1.22465e-16
```

ตัวอย่าง Function Call

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main() {
  double a1;
  cout << M_PI << endl;
  a1 = cos(M_PI);
  cout << a1 << endl;
  cout << sin(3.14) << endl;
  cout << sin(3.141) << endl;
  cout << sin(3.1415) << endl;
  cout << sin(M_PI) << endl;
}
```

ฟังก์ชันในตัวอย่าง (เช่น sin, cos) อยู่ใน library ชื่อ cmath

M_PI เป็นค่าคงที่ในภาษา C++ ของค่า π

output

```
3.14159
-1
0.00159265
0.000592654
9.26536e-05
1.22465e-16
```

sin(M_PI) ไม่ใช่ 0 เพราะ M_PI “ใกล้เคียง” กับ π แต่ไม่ใช่ π

ตัวอย่างฟังก์ชันต่าง ๆ ที่ C++ มีมาให้แล้ว

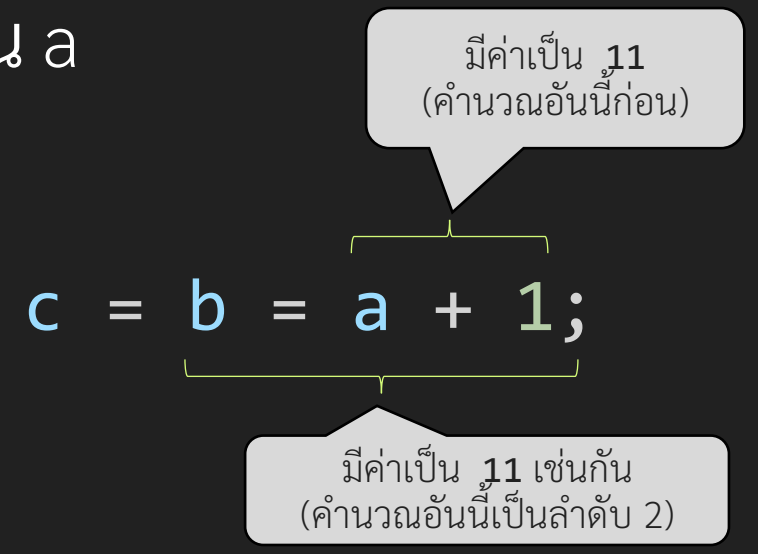
- ดูรายการฟังก์ชันทางคณิตศาสตร์บางส่วนที่ C++ มีอยู่แล้วได้ที่ https://en.cppreference.com/w/cpp/header/cmath
  - ตอนนี้อาจจะอ่านแล้วงง ๆ แต่พอเรียนเรื่องฟังก์ชันแล้วจะอ่านเข้าใจมากขึ้น

ฟังก์ชัน	สิ่งที่ทำ
sin(x) cos(x) tan(x)	คำนวณค่า sin cos tan ของ x (หน่วยเป็น radian)
abs(x)	หาค่า absolute ของ x
floor(x) ceil(x)	ค่า $\lfloor x \rfloor$ และ $\lceil x \rceil$
sqrt(x)	ค่ารากที่ 2 ของ x
log10(x)	ค่า log ฐาน 10 ของ x
log(x)	ค่า log ฐาน e ของ x
pow(a, b)	ค่า a ยกกำลังด้วย b
max(a, b)	คืนค่ามากสุดระหว่าง a กับ b
min(a, b)	คืนค่าน้อยสุดระหว่าง a กับ b

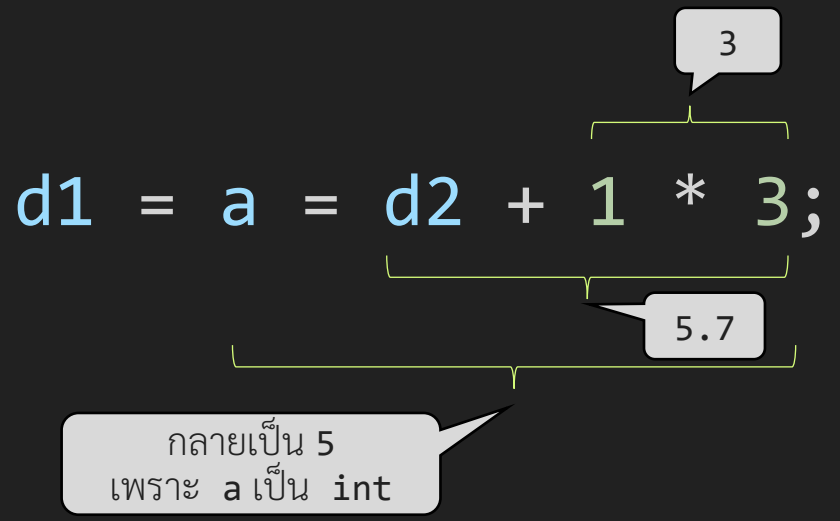
การ assignment ก็นับเป็น operator

- a = b จะมีค่าเป็นค่าที่เก็บเข้าไปใน a

```
int main() {
  int a,b,c;
  a = 10;
  c = b = a + 1;
  cout << c << endl;
}
```




```
int main() {
  int a;
  double d1, d2;
  d2 = 2.7;
  d1 = a = d2 + 1 * 3;
  cout << d1 << endl;
}
```



Compound Assignment

- ในการทำงานจริง มักจะมีความจำเป็นต้องเปลี่ยนแปลงค่าตัวแปรไปมา เช่น “เพิ่ม x ไป 10” “เอาค่า x ไปยกกำลังสอง” “ทำให้ x เหลือ 1 ใน 3”
- คำสั่งเหล่านี้เขียนได้ง่าย ๆ เช่น
  - x = x + 10 หรือ x = x * x หรือ x = x / 3
- เนื่องจากใช้บ่อยมาก จึงมีรูปแบบการเขียนแบบ compound assignment เช่น
  - `x += 10;`
  - `x *= x;`
  - `x /= 3;`

การหารตามประเภทข้อมูลของค่า

```cpp
#include <iostream>
using namespace std;
int main() {
  cout << 3 / 5 << endl;
  cout << 3.0 / 5 << endl;
  cout << 3 / 5.0 << endl;
  cout << "------" << endl;
  cout << 3 / 5 * 2.0 << endl;
  cout << 3.0 / 5 * 2 << endl;
}
```

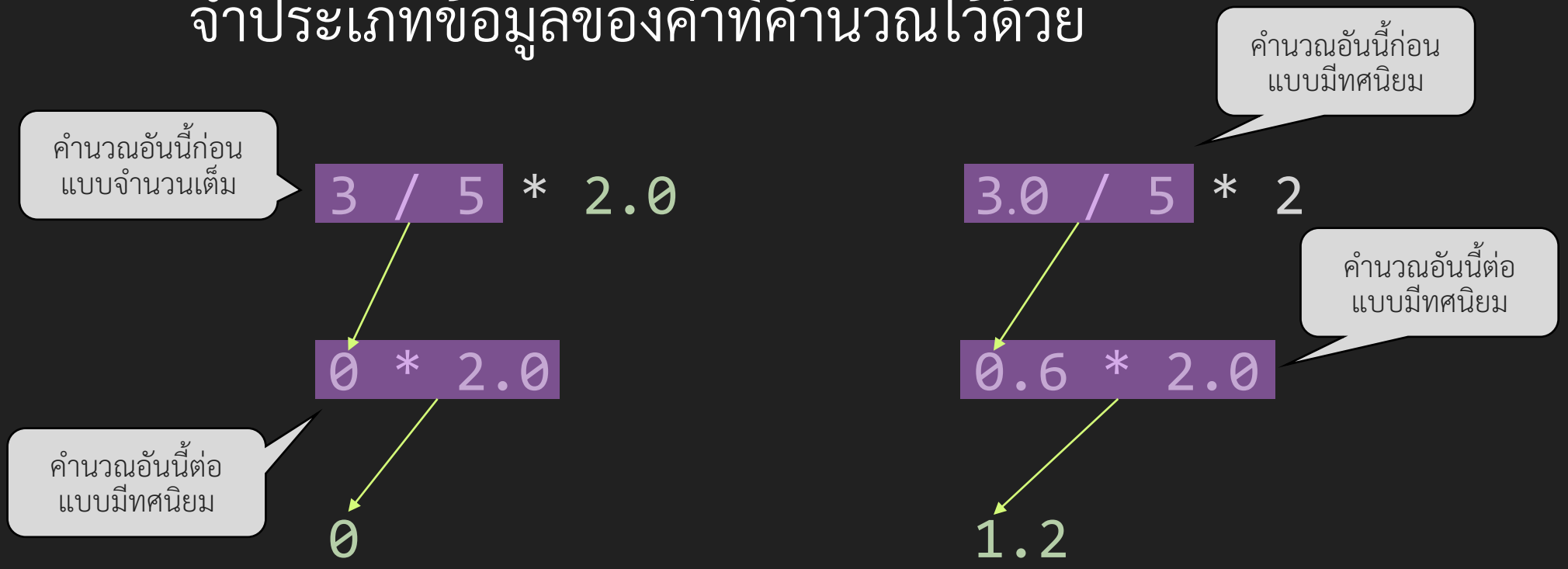
output

```
0
0.6
0.6
------
0
1.2
```

- การหาร (operator /) นั้นทำงานตามประเภทของค่าที่เอามาคำนวณ
  - หากตัวตั้งและตัวหารเป็นจำนวนเต็ม จะใช้วิธีหารปัดเศษให้เป็นจำนวนเต็ม
  - หากมีตัวตั้งหรือตัวหารเป็นแบบทศนิยม จะใช้การหารแบบทศนิยม

ประเภทข้อมูลของ expression

- ประเภทข้อมูลมีการเปลี่ยนแปลงได้จาก operator ต่าง ๆ
- C++ จะคำนวณค่าของ expression ตามลำดับของ operator พร้อม ๆ กับจำประเภทข้อมูลของค่าที่คำนวณไว้ด้วย



ลองเดาอันนี้หน่อย

```
int a;
cout << 1 - (a = 3.0 / 5 * 2) * 5.0 << endl;
```

- ค่าที่จะพิมพ์ออกมาเป็นเท่าไร?

- ในชีวิตจริงไม่ค่อยทำอะไรแบบนี้กัน

ตัวอย่างแบบยาว ๆ

expression	ค่าของ z
`int z = 5+7*3-1*4;`	22
`int z = (5+7)*(3-1)*4;`	96
`double z = 20 + sin(30 * M_PI / 180) * sqrt(100);`	25
`int z = max(log10(1000000),pow(2,3))+1*0`	8
`int z = 3;` `z += z++ * 5 + abs(-3*4);`	31
`int z = 3;` `z += ++z * 5 + abs(-3*4);`	36

สรุป

- ตัวแปร
  - เอาไว้เก็บข้อมูล
  - ต้องระบุประเภทข้อมูล
  - ประเภทแตกต่างกันเก็บข้อมูลได้แตกต่างกัน
  - รู้จักการกำหนดค่าตัวแปร
- Expression
  - คำสั่งคำนวณค่าต่าง ๆ
  - เรียกใช้ function ได้
  - มี operator หลายตัวให้ใช้

char

ประเภทข้อมูลแบบตัวอักษร

อักขระ

- ประเภทข้อมูล char มีไว้เก็บ ตัวอักขระ 1 ตัว
  - เช่น `a b c` หรือเครื่องหมายวรรคตอนเช่น `! , - “`
- เราสามารถเขียนอักขระ 1 ตัวในภาษา C++ ได้ด้วย expression `'a'`

```cpp
#include <iostream>
using namespace std;
int main() {
  char c1 = 't', c2 = 'h', c3 = 'e';
  cout << c1 << c2 << c3 << '-' << endl;
}
```

สังเกตว่ามันเป็น
single quote (')
ไม่ใช่ double quote (")

char จริง ๆ แล้วเก็บตัวเลข!!!

- คอมพิวเตอร์เองรู้จักแต่ตัวเลข
- เพื่อให้มีมาตรฐาน มีการตกลงกันไว้ว่าจะแทนที่ตัวอักษรต่าง ๆ ด้วยตัวเลข
- มาตรฐานมีชื่อว่า ASCII
  - A แทนด้วยตัวเลข 65
  - B แทนด้วยตัวเลข 66
  - ...
  - Z แทนด้วยตัวเลข 90
  - a แทนด้วยตัวเลข 97

ลองใช้ char แบบตัวเลข

- เอา char + ตัวเลข ได้ตัวเลข
- cout จะพิมพ์ตัวอักษรหรือตัวเลข ขึ้นอยู่กับของที่พิมพ์ว่าเป็น char หรือเป็นตัวเลข

```
#include <iostream>
using namespace std;
int main() {
  cout << 'a'  << endl;
  cout << 'a' + 1 << endl;

  char c = 'a';
  c++;
  cout << c << endl;
  cout << c - 1 << endl;
  c = 100;
  cout << c << endl;
}
```

output

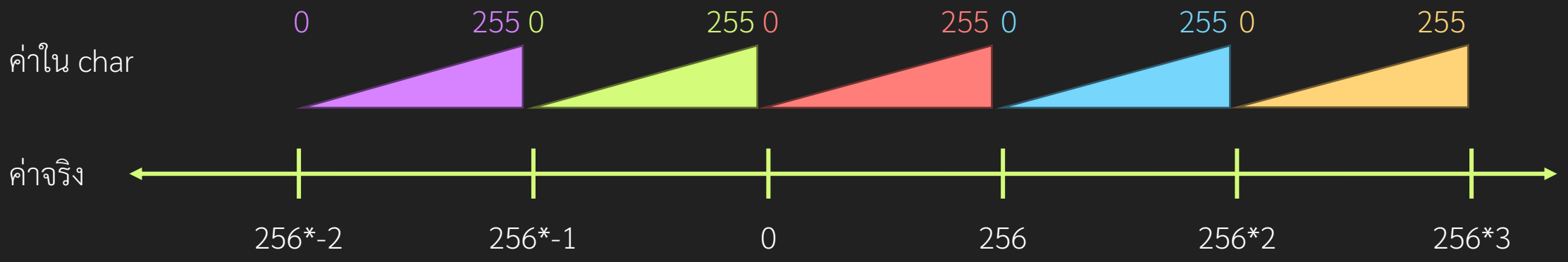
```
a
98
b
97
d
```

ขอบเขตของ char และการแปลงค่า

- char ใช้เนื้อที่ 1 byte ในหน่วยความจำ
- เก็บค่าได้แค่ตัวเลข 0 ถึง 255 เท่านั้น
  - ให้สังเกตว่า 255 คือ 2^8-1
- หากกำหนดค่าให้ char ด้วยค่านอกเหนือขอบเขต C++ จะแปลงค่าให้
- วิธีแปลงจะใช้วิธีวนค่า

ค่าที่กำหนดให้ (เช่น char c = 999)	ค่าที่ char เก็บจริง
-257	255
-256	0
-255	1
-254	2
…	…
-2	254
-1	255
0	0
1	1
2	2
…	…
254	254
255	255
256	0
257	1
…	…
510	255
511	256
512	0

การแปลงค่าแบบวน ๆ



- ตอน int ก็แปลงแบบเดียวกัน แต่ขอบเขตจะอยู่ที่ 0 ถึง 2^{32}-1 แทน
- มีอธิบายละเอียดอีกทีในบท data representation

แค่ 0 – 255 เก็บอักขระภาษาต่าง ๆ พอได้อย่างไร?

- ไม่พอ!!!
- char ใช้มาตั้งแต่สมัยที่ใช้แต่ตัวอักษรภาษาอังกฤษ
- ปัจจุบันเราใช้ Unicode ที่ 1 ตัวอักษรเป็นได้ตั้งแต่ 1 ถึง 4 bytes
  - วิชานี้ไม่ครอบคลุมถึง unicode

ตัวอย่างการใช้งานรหัส ASCII เพิ่มเติม

- ให้สังเกตว่าค่าตัวเลขในรหัส ASCII นั้นเรียงจาก A → Z (และ a → z)
- อยากรู้ว่า อักขระในตัวแปร c ที่มีประเภทข้อมูลเป็น char เป็นตัวอักษรที่เท่าไรในภาษาอังกฤษ
  - ใช้ c – ‘a’

```
c = 'd';
cout << c - 'a' << endl;
```

เมื่อไรจะใช้ char

- ใช้เพื่อเก็บตัวเลขน้อย ๆ
- ใช้เพื่อเก็บตัวอักษรภาษาอังกฤษตัวเดียว

- ใช้เป็นพื้นฐานของประเภทข้อมูลถัดไป คือ string

std::string

char ที่ต่อ ๆ กันหลาย ๆ ตัว

string คืออะไร?

- เป็นตัวแปรสำหรับเก็บตัวอักษรที่ต่อ ๆ กัน (เรียกว่าสายอักขระก็ได้)
- ใช้เก็บคำ ประโยคต่าง ๆ
- มีประโยชน์สำหรับการประมวลผลเอกสาร รวมถึงตัวเลขต่าง ๆ

ตัวอย่าง

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
  string s = "asdf";
  cout << "The string is " << s << endl;

  cout << "Please enter your name: ";
  cin >> s;
  cout << "Your name is " << s << endl;

  string s1, s2;
  cout << "Please enter two words: ";
  cin >> s1 >> s2;
  cout << "The first word is " << s1 << endl;
  cout << "The second word is " << s2 << endl;
}
```

output

```
The string is asdf
Please enter your name: Dae
Your name is Dae
Please enter two words: AA BB
The first word is AA
The second word is BB
```

สีชมพูคือส่วนที่ผู้ใช้พิมพ์เข้าไป

- สังเกตอะไรบ้าง
  - การใช้ string ต้อง include <string>
  - กำหนดค่าเริ่มต้นได้ด้วย = "xxx";
  - สามารถใช้ cin เพื่อรับข้อมูลจาก keyboard เข้า string ได้
    - cin จะรับข้อมูลจนถึงอักขระช่องว่างต่าง ๆ

string เป็น object

- Object เป็นรูปแบบหนึ่งของตัวแปร
- ตัวแปรแบบ object นั้นจะมีบริการต่าง ๆ ให้เรียกใช้งานได้
  - บริการดังกล่าวจะทำงานกับตัวแปรที่เรียกนั้น
  - วิธีเรียกใช้บริการจะเป็น `object.บริการ`
- ตัวอย่างเช่น `string` มีบริการ `.length()` เพื่อใช้คำนวณจำนวนตัวอักษรใน `string`
- อีกสักพัก จะได้รู้จัก object อื่น ๆ อีกมากมาย `string` เป็นตัวอย่างเริ่มต้น

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
  string s = "Somchai";
  cout << "The string is " << s << endl;
  cout << "The length is " << s.length() << endl;
  cout << "First char: " << s[0] << endl;
  cout << "last char: " << s[s.length() - 1] << endl;
}
```

output

```
The string is Somchai
The length of the string is 7
First char: S
last char: i
```

อักขระแต่ละตัวใน string

- บริการ `[x]` ของ `string` เป็นการเข้าถึงอักขระตัวที่ x ใน string นั้น โดยตัวแรกสุดคือตัวที่ 0
- `s[x]` มีประเภทข้อมูลเป็น `char`
  - ใช้งานได้เหมือน char
- หาก x มีค่าเกินขอบเขต (จำนวนอักขระใน string) โปรแกรมอาจจะทำงานผิดพลาดได้
  - C++ ไม่ตรวจสอบสิ่งเหล่านี้ให้เรา

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
  string s = "Somchai";
  cout << "The string is " << s << endl;
  s[2] = '3';
  cout << "Now s is " << s << endl;
  s[3]++;
  cout << "Now s is " << s << endl;
  s[4] = 90;
  cout << "Now s is " << s << endl;
  int x = s[0];
  cout << x << endl;
  s[1000] = 999;  // this will crash
}
```

output

```
The string is Somchai
Now s is So3chai
Now s is So3dhai
Now s is So3dZai
83
```

บริการตัดเอาบางส่วนของ string

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
  string s = "Somchai";
  cout << s.substr(0,4) << endl;
  cout << s.substr(3,2) << endl;
  cout << s.substr(6,0) << endl;
  cout << s.substr(1) << endl;
  cout << s.substr(4,999) << endl;
  string s2 = s.substr(1,4);
  cout << s2.substr(0,1).length() << endl;
}
```

output

```
Somc
ch

omchai
hai
1
```

- `.substr(a, b)` เป็นการสร้าง string ที่ประกอบด้วยอักขระตัวที่ a จนถึงก่อนตัวที่ a + b ของ string ที่เรียกใช้
  - b คือจำนวนตัวที่ต้องการ
  - ถ้ามากกว่าจำนวนที่มี ก็เอาเท่าที่มี
- `.substr(a)` เอา string ตั้งแต่ a จนถึงตัวสุดท้าย
- คืนค่าเป็น string

แปลง string เป็นตัวเลข

แปลงอักขระตัวเดียว ใช้วิธีแบบ char คือ - '0'

แปลงทั้ง string ใช้ฟังก์ชัน stoi

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
  string s = "789";
  int a;
  a = s[0] - '0';
  cout << a << endl;
  a = s[1] - '0';
  cout << a << endl;

  //a = s;  //ทำไม่ได้
  a = stoi(s);
  cout << a + 1000 << endl;
}
```

output

```
7
8
1789
```

แปลงเลขฐานอื่น ๆ

- `stoi` ใช้แปลงเลขฐานอื่น ๆ ได้ด้วย
- `stoi(a, b, c);` เป็นการแปลง string a โดยพิจารณาจากตำแหน่ง b และคิดแบบเลขฐาน c

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string s;   int a;

  s = "1FF";
  a = stoi(s,0,16);
  cout << a << endl;

  s = "100101";
  a = stoi(s,0,2);
  cout << a << endl;
}
```

output

```
511
37
```

“xxxx” ไม่ใช่ string

- เวลาเราเขียน `"Somchai"` ใน expression นั้น มันไม่ใช่ string แต่เรียกว่า string literal
- String literal ไม่ใช่ string object ดังนั้นไม่สามารถเรียกใช้บริการต่าง ๆ ของ string ได้
  - `"123".length()` ทำไม่ได้
- แต่มันแปลงเป็น string ได้
  - `stoi("123")` ทำงานได้ตามปรกติ ถึงแม้ stoi ต้องการ string แต่พอเราให้ string literal ไป มันก็จะแปลงเป็น string ให้อัตโนมัติ

ค้นหาใน string

- อยากหาว่ามีคำ `"xxx"` ใน string หรือไม่ ให้ใช้บริการ `.find(a, b)` ของ string
  - โดย `a` คือคำที่ต้องการหา
  - `b` คือตำแหน่งเริ่มต้นที่จะหา (ถ้าไม่ระบุคือเริ่มตั้งแต่แรก)
- ถ้าหาไม่เจอ `find` จะคืนค่าเป็น `string::npos`
  - `string::npos` มีค่าเป็นตัวเลขแปลก ๆ ขึ้นอยู่กับระบบคอมพิวเตอร์ที่ใช้

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
  string s = "This box has a book";
  cout << s.find("has") << endl;
  cout << s.find("bo") << endl;
  cout << s.find("bo", s.find("bo")+1) << endl;

  cout << "---" << endl;
  cout << s.find("car") << endl;
  if (s.find("car") == string::npos) {
    cout << "Not found" << endl;
  }
}
```

output

```
9
5
15
---
18446744073709551615
Not found
```

`if` คืออะไร?

- เรียกว่า control flow structure
- เป็นโครงสร้างของภาษา C++ ที่เลือกคำสั่งตามเงื่อนไข
  - ไว้คุยกันในบทถัดไป

สรุป

- Char
  - เอาไว้เก็บตัวอักษร 1 ตัว
  - สถานะเป็นเหมือนตัวเลข
- String
  - เก็บสายอักขระ
  - มีชนิดเป็น object
  - มีบริการต่าง ๆ ให้ใช้
    - มีฟังก์ชันใช้งานเฉพาะด้วย
  - รู้จัก operator [ ] ของ string และรู้ว่ามันสามารถพังได้

อยากรู้ว่า string มีบริการอะไรอีก? ดูได้ที่
https://en.cppreference.com/w/cpp/string/basic_string