Condition

Which statement to execute

เงื่อนไขสำหรับคำสั่ง

- เราสามารถสร้างเงื่อนไขสำหรับคำสั่งได้ว่า จะให้โปรแกรมทำงานตามคำสั่งนี้ก็ต่อเมื่อเงื่อนไขบางอย่างเป็นจริง
- C++ จะใช้โครงสร้างที่เรียกว่า conditional statement ในการเขียนโปรแกรมแบบเงื่อนไข

expression นี้จะต้องมีค่าเป็นประเภท bool (คือสิ่งที่คำนวณค่าความจริงเท็จได้)

```
if ( expression ) {
  //statement ที่ทำงานเมื่อ
  //expression ข้างบนเป็นจริง
}
```

คำสั่งใด ๆ ที่อยู่ในส่วน { } ที่อยู่ข้างหลัง if (expression) จะทำงานก็ต่อเมื่อ expression เป็นจริง

ตัวอย่าง If

- โปรแกรมคำนวณราคาสินค้า เมื่อมีโปรโมชันคือถ้าราคารวมเกิน 200 บาท จะลดให้ 10%
  - ให้สังเกต expression `a >= 200` เป็น expression ที่มีค่าเป็น “จริง” หรือ “เท็จ” (หรือเรียกว่าประเภท Boolean)

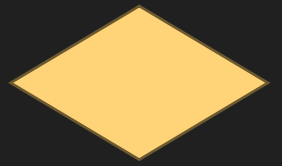
```
#include <iostream>
using namespace std;
int main() {
  int a;
  cout << "Enter a total value of your item: "; cin >> a;
  if ( a >= 200 ) {
    cout << "You receive a discount of 10%" << endl;
    a *= 0.9;
  }
  cout << "The price is " << a << endl;
}
```

คำสั่งที่ทำเมื่อเงื่อนไขเป็นจริง มักจะ “ย่อหน้า” เข้ามานิดหน่อยเพื่อให้อ่านง่าย

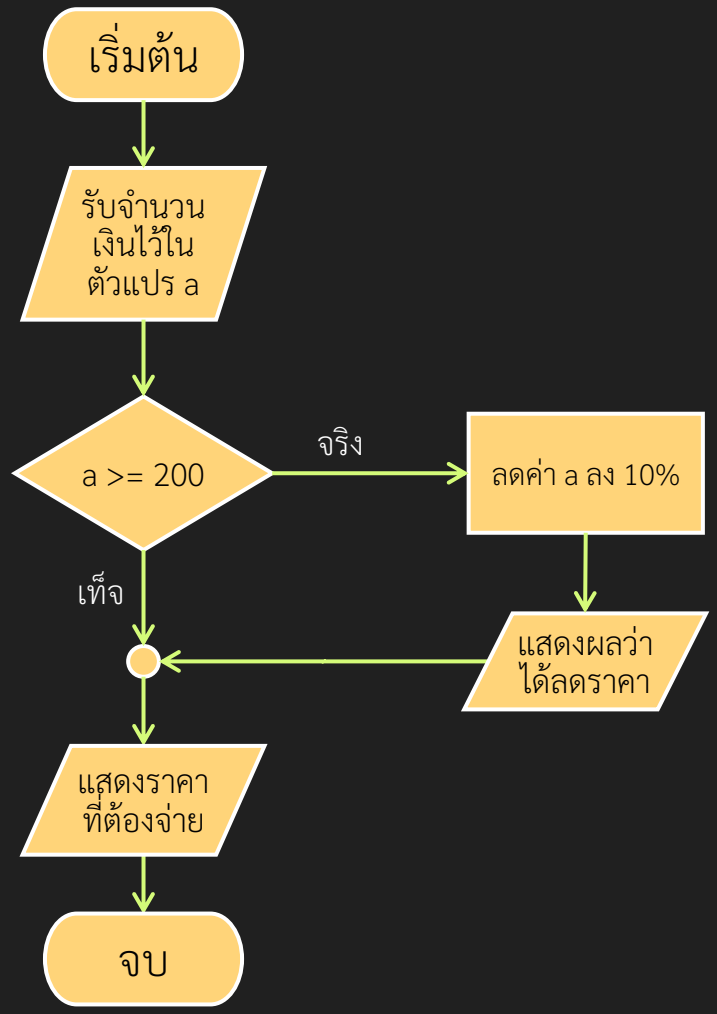
การที่จะย่อหน้าหรือไม่ ไม่มีผลต่อการทำงานของโปรแกรม

Flow chart

- พอโปรแกรมมีเงื่อนไข การวาดแผนภาพบางอย่างมาช่วยอธิบายโปรแกรมจะช่วยให้ทำความเข้าใจโปรแกรมได้ง่ายขึ้น
- Flow chart คือแผนภาพที่แสดงลำดับการทำงานของโปรแกรม
  - ประกอบด้วยสัญลักษณ์กล่องต่าง ๆ
  - มีลูกศรเชื่อมกล่องต่าง ๆ เพื่อบอกลำดับการทำงาน

กล่อง	ความหมาย
	จุดเริ่มต้น/สิ้นสุดการทำงาน
	Input/output
	คำสั่งต่าง ๆ
	เงื่อนไข

ตัวอย่าง Flow chart

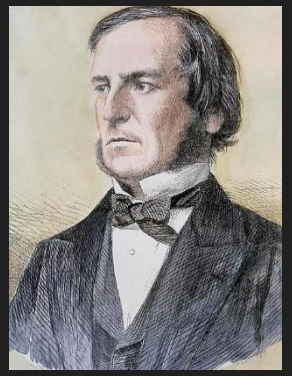



```cpp
#include <iostream>
using namespace std;
int main() {
  int a;
  cout << "Enter a total value of your item: "; cin >> a;
  if ( a >= 200 ) {
    cout << "You receive a discount of 10%" << endl;
    a *= 0.9;
  }
  cout << "The price is " << a << endl;
}
```

- Flow chart เน้นแสดง แนวคิด มากกว่า syntax ของภาษา

Boolean expression

George Boole



- boolean หมายถึงประเภทข้อมูลที่มีค่าเป็นจริงหรือเท็จ
- expression ใน `if (expression)` จะต้องเป็นประเภท boolean
- มักจะเกิดจากการใช้ operator ในแบบ comparison หรือ logical
  - เช่น a > b เป็น expression ที่ใช้ comparison operator > เพื่อเปรียบเทียบ a กับ b โดยจะคืนค่าเป็น “จริง” เมื่อ a มากกว่า b
- ใน C++ เราสามารถสร้างตัวแปรประเภท `bool` มาเพื่อใช้เก็บค่าจริงหรือเท็จ ได้
  - มีค่าคงที่ 2 ค่าที่เป็นประเภท bool คือ `true` และ `false`

Operator ที่ให้ผลลัพธ์เป็น bool

กลุ่ม	Expression	ความหมาย	ตัวอย่าง	
Comparison Operator (a และ b จะต้องเป็นค่าที่สามารถเปรียบเทียบกันได้)	`a == b`	เท่ากัน?	`5 == 3`	มีค่าเป็นเท็จ
	`a != b`	ไม่เท่ากัน?	`5 != 3`	มีค่าเป็นจริง
	`a < b`	น้อยกว่า?	`5 < 3`	มีค่าเป็นเท็จ
	`a > b`	มากกว่า?	`5 > 3`	มีค่าเป็นจริง
	`a <= b`	น้อยกว่าหรือเท่ากัน?	`4 <= 3`	มีค่าเป็นเท็จ
	`a >= b`	มากกว่าหรือเท่ากัน?	`4 >= 4`	มีค่าเป็นจริง
Logical Operator (a และ b จะต้องเป็นค่าแบบ boolean)	`!a`	not (สลับค่าความจริง)	`!(5 == 3)`	มีค่าเป็นจริง
	`a && b`	and (และ)	`(3 < 5) && (5 < 7)`	มีค่าเป็นจริง
	`a || b`	or (หรือ)	`(3 == 5) || (3 == 3)`	มีค่าเป็นจริง

Logical Operator สำหรับคนที่ลืมไปแล้ว

A	B	A && B
false	false	false
false	true	false
true	false	false
true	true	true

A	B	A || B
false	false	false
false	true	true
true	false	true
true	true	true

A	!A
false	true
true	false

ลองดูตัวอย่างอีกหน่อย

```cpp
#include <iostream>
using namespace std;
int main() {
  cout << true << endl;
  cout << false << endl;
  cout << (1 < 2) << endl;
  cout << (1 > 2) << endl;
  cout << "-----" << endl;

  bool a,b,c;
  a = (1 == 2);    // false
  b = 'a' != 'b';  // true
  c = '1' < 'a';   // true  (char use ASCII number for comparison)
  cout << a << endl;
  cout << b << endl;
  cout << c << endl;
  cout << "-----" << endl;

  cout << (a || b) << endl;
  cout << (a && b) << endl;
  cout << (!a && b) << endl;
}
```

bool เวลาแสดงออกมาด้วย cout จะมีค่าเป็น 1 เมื่อเป็นจริง และเป็น 0 เมื่อเป็นเท็จ

output

```
1
0
1
0
-----
0
1
1
-----
1
0
1
```

ลำดับความสำคัญของ operator เพิ่มเติม

ความสำคัญ	Operator	ความหมาย
1	a++ a-- xxx()	Postfix increment Postfix decrement Function call
2	++a --a !	Prefix increment Prefix decrement Logical not
3	* / %	คูณ หาร หารเอาเศษ
4	+ -	บวก ลบ
5	<< >>	ใช้กับ cin, cout (และเป็น bitwise shift)
6	< <= > >=	มากกว่าน้อยกว่า
7	== !=	ความเท่ากัน
8	&&	Logical AND
9	||	Logical OR
10	= += -= *= /= %=	assignment ต่าง ๆ

- ! ทำก่อนบวกลบคูณหาร
- การเปรียบเทียบทำที่หลังบวกลบคูณหาร
- ด้วยลำดับดังกล่าว ทำให้

```
cout << 1 < 2 << endl
```

ทำงานไม่ได้ เพราะ 1 จะถูกพิมพ์ออกไปก่อนที่จะเปรียบเทียบกับ 2

- ต้องแก้เป็น

```
cout << (1 < 2) << endl
```

การเปรียบเทียบ string หรือ string literal

- จะใช้วิธีการเปรียบเทียบแบบพจนานุกรม (lexicographic order)
- พื้นฐานคือเปรียบเทียบตามตัวอักษร
  - ตัวอักษรแต่ละตัวเปรียบเทียบโดยใช้รหัส ASCII
  - เช่น `'1' < 'a'` เป็นจริง
- เปรียบเทียบไล่ไปทีละตัวอักษร ตั้งแต่ตัวแรก (ซ้ายสุด) ไล่ไปทางขวาทีละตัว
  - ถ้าตัวอักษรในตำแหน่งเดียวกันเป็นตัวเดียวกัน ให้ดูตัวถัดไป (ตัวทางขวา)
  - ถ้าไม่เท่ากัน ตัวอักษรของสตริงใดมาก่อน จะทำให้สตริงนั้นมีค่าน้อยกว่า
  - ถ้าในตำแหน่งเดียวกัน สตริงหนึ่งไม่มีตัวอักษรแล้ว (เพราะความยาวของสตริงน้อยกว่า) ให้ถือว่าตัวนั้นน้อยกว่า

```cpp
#include <iostream>
using namespace std;
int main() {
  cout << ("a" == "a") << endl;
  cout << ("a" < "a") << endl;
  cout << ("a" < "aa") << endl;
  cout << ("aa" < "aaa") << endl;
  cout << ("ab" < "aaa") << endl;
  cout << ("ab" < "a") << endl;
  cout << ("ab" < "ac") << endl;
}
```

Block structure

- จริง ๆ แล้วโครงสร้างของ if คือ `if ( expr ) คำสั่ง 1 คำสั่งที่จะทำเมื่อ expr เป็นจริง;`
  - เมื่อเงื่อนไขหลัง if เป็นจริง โปรแกรมจะทำงานคำสั่งที่อยู่หลัง if เป็นจำนวน 1 คำสั่งเท่านั้น!!!
- ถ้าอยากทำมากกว่า 1 คำสั่ง ต้องรวบคำสั่งที่จะทำให้อยู่ใน block
  - block จะระบุได้ด้วยการที่ถูกล้อมด้วยเครื่องหมาย { และ }
  - block ทั้ง block จะถูกนับรวมเป็น 1 คำสั่ง (เรียกว่า compound statement)
- ใน main นั้นเองก็เป็น block เช่นกัน

```
if ( expression ) {
  //statement ที่ทำงานเมื่อ
  //expression ข้างบนเป็นจริง
}
```

block

ตัวอย่างของ block

```cpp
#include <iostream>
using namespace std;
int main() {
  cout << "start" << endl;
  if (true)
    cout << "a" << endl;
  cout << "-----" << endl;

  if (1 > 2)
    cout << "b1" << endl;
    cout << "b2" << endl;
  cout << "-----" << endl;

  if (1 > 2) {
    cout << "c1" << endl;
    cout << "c2" << endl;
  }
  cout << "end" << endl;
}
```

output

```
start
a
-----
b2
-----
end
```

ทำไม b2 ถึงถูกพิมพ์ออกมา?

If ... else ...

- ในบางครั้ง สิ่งที่เราทำจะอยู่ในรูปแบบ
  - ถ้า xxx เป็นจริง ให้ทำ aaa
  - แต่ถ้าไม่จริง ให้ทำ bbb
- จะใช้โครงสร้างแบบ if .... else ....
  - ให้สังเกตว่า ตรงสิ่งที่ทำเมื่อเป็นจริง และ ส่งที่ทำเมื่อเป็นเท็จ เป็นคำสั่งเดียว
  - ถ้าอยากทำหลายคำสั่ง ก็ต้องทำเป็น block

```
if (xxx)
   คำสั่ง 1 คำสั่งที่จะทำเมื่อ xxx เป็นจริง;
else
   คำสั่ง 1 คำสั่งที่จะทำเมื่อ xxx เป็นเท็จ;
```

```
#include <iostream>
using namespace std;
int main() {
  int a;
  cout << "Enter a number: "; cin >> a;
  if (a % 2 == 0) {
    cout << "It is even." << endl;
  } else {
    cout << "It is odd." << endl;
  }
}
```

ตัวอย่าง if ... else .... กับ block

```cpp
#include <iostream>
using namespace std;
int main() {
  bool a = true;
  if (a) cout << "yes"; else cout << "no";

  if (a)
    cout << "yes" << endl;
  else
    cout << "no" << endl;

  if (a) {
    cout << "yes" << endl;
  } else {
    cout << "no" << endl;
  }

  if (a) { cout << "yes" << endl; } else {
    cout << "no" << endl;
  }
}
```

เชียร์แบบนี้

- ให้สังเกตว่าการขึ้นบรรทัดใหม่หรือไม่ ไม่มีผลต่อการทำงาน

output

```
yes
yes
yes
yes
```

ลองเดาหน่อยว่าโปรแกรมต่อไปนี้แสดงผลอะไรบ้าง

```cpp
#include <iostream>
using namespace std;
int main() {
  bool a = true;

  if (a)
    cout << "a1" << endl;
  else
    cout << "a2" << endl;
    cout << "a3" << endl;

  if (a)
    cout << "b1" << endl;
  else {
    cout << "b2" << endl;
    cout << "b3" << endl;
  }

  if (a) {
    cout << "c1" << endl;
    cout << "c2" << endl;
  } else
    cout << "c3" << endl;
    cout << "c4" << endl;
 }
```

- โปรแกรมด้านซ้ายนี้พิมพ์อะไรมาบ้าง

เมื่อ a = true	เมื่อ a = false

ตัวอย่างผิด

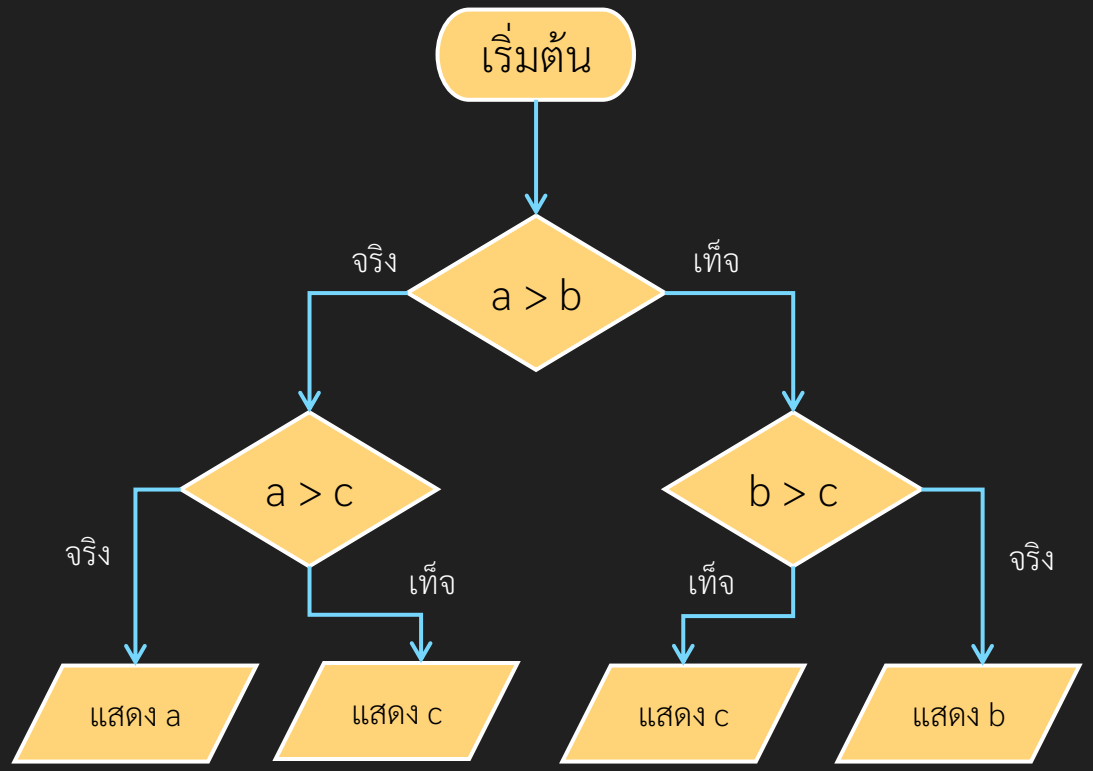
- Compile ไม่ผ่าน เพราะ อยู่ ๆ เจอ else

```
#include <iostream>
using namespace std;
int main() {
  bool a = true;

  if (a)
    cout << "c1" << endl;
    cout << "c2" << endl;
  else
    cout << "c3" << endl;
    cout << "c4" << endl;
}
```

if ซ้อนกัน

```cpp
#include <iostream>
using namespace std;
int main() {
  int a,b,c;
  cout << "Enter 3 numbers: ";
  cin >> a >> b >> c;

  if (a > b) {
    if (a > c) {
      cout << "Largest is " << a << endl;
    } else {
      cout << "Largest is " << c << endl;
    }
  } else {
    if (b > c) {
      cout << "Largest is " << b << endl;
    } else {
      cout << "Largest is " << c << endl;
    }
  }
}
```

- ของที่อยู่ใน block ก็มี if อยู่ข้างในได้
- ตัวอย่างโปรแกรมหาค่ามากสุดจาก ตัวเลข 3 ตัว



ทำงานแบบเดียวกันเขียนได้หลายแบบ

- โปรแกรมนี้มีที่ผิดอยู่ ดูออกหรือไม่ว่าจะผิดเมื่อ a b c มีค่าเป็นเท่าไร?
- ต้องแก้ยังไงให้ถูก

```
#include <iostream>
using namespace std;
int main() {
  int a,b,c;
  cout << "Enter 3 numbers: ";
  cin >> a >> b >> c;

  if (a > b && a > c) cout << "Largest is " << a << endl;
  if (b > a && b > c) cout << "Largest is " << b << endl;
  if (c > a && c > b) cout << "Largest is " << c << endl;
}
```

ลำดับของ operator แบบ logical

- && ทำก่อน ||
- ! ทำก่อนทั้ง && และ ||

Expression	ค่าที่ได้
`false || true && false`	`false`
`!true || true && true || false`	`true`
`false && true || true && true`	`true`

Short Circuit

- Boolean expression ในรูปแบบ `a1 && a2 && a3 && … && an` จะถูกคำนวณจาก a1 ไป a2 ไปเรื่อย ๆ จนถึงตัวแรกที่เป็น false แล้วจะหยุดเลย (ไม่คำนวณ a ตัวถัดไป และ expression มีค่าเป็น false)
- Boolean expression ในรูปแบบ `a1 || a2 || a3 || … || an` จะถูกคำนวณจาก a1 ไป a2 ไปเรื่อย ๆ จนถึงตัวแรกที่เป็น true แล้วจะหยุดเลย (ไม่คำนวณ a ตัวถัดไป และ expression มีค่าเป็น true)
- ช่วยให้เขียนโปรแกรมได้ง่ายขึ้น

Short Circuit

```
#include <iostream>
using namespace std;
int main() {
  string s; int a;
  cout << "Enter a string: ";   cin >> s;
  cout << "Enter a position that contain a letter 'z’:";   cin >> a;
  if (a >= 0 && a < s.length())
    if (s[a] == 'z') {
      cout << "correct!!" << endl;
}
```

```
#include <iostream>
using namespace std;
int main() {
  string s; int a;
  cout << "Enter a string: ";   cin >> s;
  cout << "Enter a position that contain a letter 'z': ";   cin >> a;
  if (a >= 0 && a < s.length() &&  s[a] == 'z')
      cout << "correct!!" << endl;
}
```

Ternary operator

- ในหลายครั้ง เรามักจะใช้ if เพื่อกำหนดค่าของตัวแปรในรูปแบบ
- ถ้า xxx เป็นจริง ให้ y = c1 แต่ถ้าไม่ใช่ ให้ y = c2
- รูปแบบนี้มีการใช้บ่อย C++ จึงมี ternary operator ให้ใช้
- เป็น operator แบบหนึ่ง สามารถเขียนเป็น expression ได้ในรูปแบบ

```
a ? b : c
```

- a เป็น bool
- expression มีค่าเป็น b เมื่อ a เป็นจริง แต่มีค่าเป็น c เมื่อ a เป็นเท็จ
- ตัวอย่าง

```
a >= 0 ? a : -a;
```

```
int a;
cout << "Enter an integer: "; cin >> a;

int abs;
if (a >= 0) abs = a; else abs  = -a;
cout << "The absolute value is " << abs << endl;
```

```
int a;
cout << "Enter an integer: "; cin >> a;

int abs = a >= 0 ? a : -a;
cout << "The absolute value is " << abs << endl;
```

ลองเอาหลาย ๆ อย่างมารวมกัน

```cpp
#include <iostream>
using namespace std;
int main() {
  int a = 5;
  if (a++ == 6) {
    cout << "a" << endl;
  }

  if (a++ == 6) cout << "b" << endl;
  else { cout << "d" << endl; }

  int b = 10;

  if (a < 50 || b++ == 10) {
    cout << "The value of b: " << b << endl;
  }
}
```

- เช่นเดิม ลองคิดหน่อยว่าโปรแกรมนี้พิมพ์อะไรออกมาบ้าง

สรุป

- if (expr) เอาไว้เลือกว่าจะทำงานส่วนใดตามเงื่อนไขที่ดูจาก expr
- มีรูปแบบ if ... else ...
- รู้จัก block
- มี operator ใหม่ ๆ ให้ใช้
  - Comparison
  - Logical
  - ternary

รายละเอียดยิบย่อยของ C++

expression งง ๆ

การแปลงข้อมูลเป็นประเภท bool

- ข้อมูลประเภทตัวเลข สามารถแปลงเป็น bool ได้
  - เช่น เมื่อ ให้ `a = b` แล้ว `b` เป็น `bool` แต่ `a` เป็นพวกตัวเลข
  - เช่น `if (a)` แล้ว `a` เป็นตัวเลข (เพราะ if ต้องการ bool แต่เราใส่ตัวเลขเข้าไปแทน)
- กฎในการแปลงคือ ค่าใด ๆ ที่ไม่ใช่ 0 จะมีค่าเป็น true
  - ส่วน 0 คือค่า false
- ปรกติไม่ค่อยแนะนำ แต่ก็ทำได้ในบางกรณี
  - แทนที่จะ `if (a)` ให้ใช้ `if (a != 0)` แทน
  - เหมือนกัน แต่แบบหลังอ่านเข้าใจง่ายกว่า

```cpp
#include <iostream>
using namespace std;
int main() {
  int a = 20;
  bool b = -1;
  double d = 0.0001;
  char c = 'A';

  // ลองเปลี่ยน a เป็น b, c, d ดู
  // ลอง a && b && c && d ดู
  if (a) {
    cout << "This is true" << endl;
  }

  // แนะนำแบบนี้
  if (a != 0) {
    cout << "I like this" << endl;
  }
}
```

การเปรียบเทียบ double

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main() {
  double a = 987654.23456789;
  double b = 0.1234567;
  double c = 345789123;

  double a1 = a / b / c;
  double a2 = a / c / b;

  if (a1 == a2) {
    cout << "YES" << endl;
  } else {
    cout << "NO" << endl;
  }
  cout << setprecision(20);
  cout << a1 << endl << a2 << endl;

  if (fabs(a1 - a2) < 1e-9) {
    cout << "YES" << endl;
  }
}
```

เปรียบเทียบ double ว่าเท่ากันโดยตรวจสอบว่าต่างกันไม่เกินค่า น้อย ๆ

- ปัญหาของประเภทข้อมูลแบบทศนิยมนั้นคือความไม่ละเอียด
  - มีต้นเหตุมาจากการเก็บข้อมูลทศนิยมในรูปแบบฐานสองในคอมพิวเตอร์ (จะพูดละเอียดอีกทีในหัวข้อ Data Representation)
- เวลาคำนวณด้วยตัวเลขเยอะ ๆ แล้วบางครั้งผลอาจจะไม่เป็นตามที่คิดไว้

เงื่อนไขเศษการหาร

- ตัวอย่างการตรวจว่าเป็นเลขคู่
  - `if (a % 2 == 0)`
  - ถ้าใช้ `if (a % 2 != 1)` ได้มั้ย???
  - จะเกิดอะไรขึ้นถ้าตัวตั้งหรือตัวหารมีค่าติดลบ?

```
#include <iostream>
using namespace std;
int main() {
  cout << 5 % 3 << endl;
  cout << -5 % 3 << endl;
  cout << 5 % -3 << endl;
  cout << -5 % -3 << endl;
}
```

สำหรับเลขติดลบนั้น Operator modulo มีพฤติกรรมไม่เหมือนกันในภาษาต่าง ๆ

output

```
2
-2
2
-2
```

ลำดับการคำนวณและประเภทข้อมูล

- ให้ลองรัน code ด้านขวานี้ แล้วดูค่าที่ได้
  - ทำไมถึงเป็นเช่นนั้น?

```
4 / 3 * M_PI * r * r * r;
```

4 / 3 ถูกคำนวณก่อนแบบจำนวนเต็ม ได้ผลเป็น 1 (ซึ่งมันควรจะเป็น 1.3333...)

```
r * r * r * 4 / 3 * M_PI;
```

ตอนคิด * 4 นั้น ด้านซ้ายของ * เป็น double แล้ว ก็เลยคิดแบบ double

ตอนคิด / 3 ก็เช่นกัน มันเป็น double มาก่อนแล้ว

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main() {
  double r = 30;
  double v1 = 4 / 3 * M_PI * r * r * r;
  double v2 = r * r * r * 4 / 3 * M_PI;

  cout << v1 << endl;
  cout << v2 << endl;
}
```

output

```
84823
113097
```

การบังคับเปลี่ยนประเภทข้อมูล

- เราเคยเห็นตัวอย่างการแปลงประเภทข้อมูลแล้วจากคำสั่ง `a = b` เมื่อ a และ b เป็นคนละประเภทกัน (หรือใน `if (a)` เมื่อ a ไม่ใช่ bool)
  - การแปลงเกิดขึ้นเพราะ ต้องใช้ข้อมูลประเภทหนึ่ง แต่ expression ให้ข้อมูลอีกประเภทหนึ่งมา
- เราสามารถบังคับการเปลี่ยนประเภทได้ โดยใช้รูปแบบ
  - `(ประเภท) ค่า`
  - เช่น `(double)4` จะทำให้ expression นี้ (ถึงแม้จะเป็นจำนวนเต็ม 4) จะถูกเปลี่ยนเป็นประเภท double
- ข้อมูลบางประเภทเปลี่ยนไม่ได้
  - ต้องเขียนโปรแกรมเพิ่มเติมเพื่อให้เปลี่ยน
  - (ว่ากันวันหลัง)

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main() {
  double r = 30;
  double v = (double)4 / 3 * M_PI * r * r * r;
  cout << v << endl;
  cout << 'a' << endl;
  cout << (int)'a' << endl;
  cout << (char)105 << endl;
  cout << (int)3.15 << endl;
  //cout << (char)"asd" << endl;
  //cout << (string)23 << endl;
}
```

output

```
113097
a
97
i
3
```

02-Quiz

บางเรื่องที่ควรรู้ก่อน quiz

Nattee Niparnan

Department of Computer Engineering
Chulalongkorn University

แนะนำประเภทข้อมูลเพิ่มเติม

ประเภทข้อมูล	ค่าน้อยสุด	ค่ามากสุด	Size
int	-2^{31}	$2^{31}-1$	4 bytes
unsigned int	0	2^{32}	4 bytes
long long int	-2^{63}	$-2^{63}-1$	8 bytes
unsigned long long int	0	2^{64}	8 bytes
short int	-2^{15}	$2^{15}-1$	2 bytes
unsigned short int	0	$2^{16}-1$	2 bytes

```cpp
#include <iostream>
using namespace std;
int main() {
  long long x = -9223372036854775808;
  int a       = -9000000000000000000;
  cout << x << endl;
  cout << a << endl;
}
```

long long คือชื่อสั้น ๆ ของ long long int

output

```
-9223372036854775808
494665728
```

แนะนำฟังก์ชันเพิ่มเติม

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main() {
  cout << round(2.4) << endl;
  cout << round(2.5) << endl;
  cout << round(2.6) << endl;
  cout << round(-2.4) << endl;
  cout << round(-2.5) << endl;
  cout << round(-2.6) << endl;
}
```

output

```
2
3
3
-2
-3
-3
```

- `round(x)` เป็นการปัดเศษ x
  - ปัดไปหาจำนวนเต็มที่ใกล้ x มากที่สุด
  - ถ้า ทศนิยมเป็น 0.5 พอดี จะปัดให้ไกล 0 มากที่สุด