Repetition

Runs same code again and again

Nattee Niparnan

Department of Computer Engineering
Chulalongkorn University

ทำงานแบบเดิมซ้ำ ๆ

- ในครั้งที่เราต้องการให้โปรแกรมทำงานแบบเดียว ๆ กันซ้ำกันหลาย ๆ ครั้ง เรามักจะใช้โครงสร้างแบบวงวน (loop)
- โครงสร้างแบบ loop ช่วยให้เขียนโปรแกรมได้สั้นขึ้น ชัดเจนขึ้น และปรับปรุงแก้ไขง่ายขึ้น
- มีโครงสร้าง 2 แบบที่จะศึกษากัน
  - `while` loop
  - `for` loop

`while` loop

ตัวอย่าง

- แสดงค่า 1 ถึง 10

```
int main() {
  cout << 1 << endl;
  cout << 2 << endl;
  cout << 3 << endl;
  cout << 4 << endl;
  cout << 5 << endl;
  cout << 6 << endl;
  cout << 7 << endl;
  cout << 8 << endl;
  cout << 9 << endl;
  cout << 10 << endl;
}
```

แต่ละบรรทัดทำงานแบบเดียวกัน คือพิมพ์ตัวเลขสักตัวออกทางหน้าจอ

```
int main() {
  int i = 1;
  while (i <= 10) {
    cout << i << endl;
    i++;
  }
}
```

สถานะเริ่มต้นของ loop

เงื่อนไขในการทำงาน

ยุบรวมส่วนที่ซ้ำกัน นำมาไว้ภายในโครงสร้าง while

ส่วนที่ทำงานซ้ำนี้ ในแต่ละรอบเรียกว่า iteration

คำสั่งเปลี่ยนแปลงสถานะของการทำงานแต่ละครั้ง

โครงสร้าง while

- รูปแบบคือ

```
while (เงื่อนไข) คำสั่ง
```

- คล้าย if
  - เงื่อนไขจะต้องเป็น expression ที่เป็น bool (หรือแปลงเป็น bool ได้)
  - คำสั่งจะใช้เพียง 1 คำสั่งเท่านั้น ถ้าต้องการมากกว่า 1 คำสั่งจะต้องใช้ compound statement (คือใช้ block `{ }` แทน)

DRY (Don’t Repeat Yourself)

- เป็นหลักการหนึ่งในการพัฒนาซอฟต์แวร์
- นำมาใช้กับการเขียนโปรแกรมคือ
  - อย่าเขียน code ที่ทำงานเดียวกันซ้ำ ๆ กัน
- ข้อดี
  - ลดจุดที่ต้องเปลี่ยนแปลง
  - แสดงความหมายได้มากขึ้น
- การพยายามใช้หลักการนี้บ่อย ๆ ในการเขียนโปรแกรมจะช่วยฝึกแนวคิดสำคัญคือ abstraction principle

ส่วนประกอบของ while

- เวลาจะเขียน `while` หรือเขียน loop อื่นใด ควรจะพิจารณาถึง 4 ส่วน
  - สิ่งที่จะทำซ้ำ
  - เงื่อนไขที่จะทำงาน
  - การเปลี่ยนสถานะในแต่ละรอบ
  - สถานะเริ่มต้นก่อนทำงาน

```
int main() {
  int i = 1;
  while (i <= 10) {
    cout << i << endl;
    i++;
  }
}
```

ตัวอย่างที่ใช้แต่ละส่วนต่าง ๆ กัน

```
i = 1;
while (i <= 3) {
  cout << i << endl;
  i++;
}
```

```
i = 1;
while (i <= 3) {
  cout << i++ << endl;
}
```

```
i = 0;
while (++i <= 3) {
  cout << i << endl;
}
```

```
i = 0;
while (i < 3) {
  i++;
  cout << i << endl;
}
```

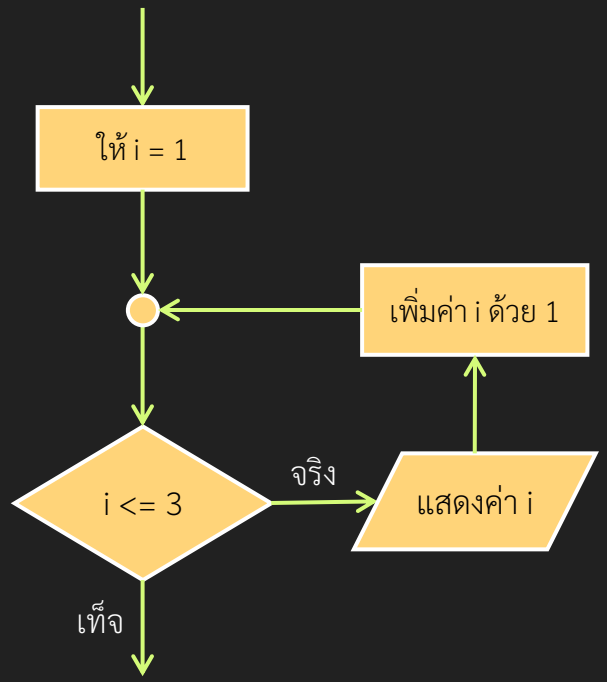
```
i = 0;
while (i < 3) {
  cout << ++i << endl;
}
```

```
i = 0;
while (i++ < 3) {
  cout << i << endl;
}
```

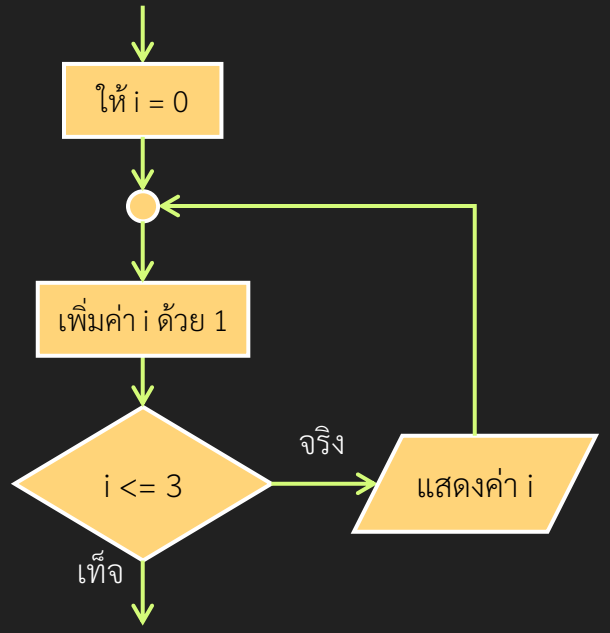
ทุกโปรแกรมพิมพ์ 1 ถึง 3 เหมือนกัน

เขียนเป็น Flow chart

```
i = 1;
while (i <= 3) {
  cout << i << endl;
  i++;
}
```




```
i = 0;
while (++i <= 3) {
  cout << i << endl;
}
```



ลองวาด Flow chart หน่อย

```
i = 0;
while (i < 3) {
  cout << ++i << endl;
}
```

ระวัง loop อนันต์

- บางครั้ง เงื่อนไขที่ใช้ใน loop อาจจะเป็นเงื่อนไขที่ไม่มีวันเป็นจริง (หรือใช้เวลานานมากกว่าจะเป็นจริง)
  - เรียกว่าเขียนโปรแกรมติด loop
- ต้องรู้วิธีที่จะสั่งให้โปรแกรมหยุดทำงาน

```
int main() {
  while (true) {
    cout << "yes" << endl;
  }
}
```

```
int main() {
  double d = 10;
  while (d > 1) {
    cout << d++ << endl;
  }
}
```

ตัวอย่างเพิ่มเติม

- นับเลขถอยหลัง

```
int main() {
  int i = 5;
  while (i >= 0) {
    cout << i << endl;
    i--;
  }
}
```

- ระวังประเภทข้อมูลด้วย!

คำสั่งนี้คือ `ui = ui - 1`

เมื่อ `ui` มีค่าเป็น 0 คำนั่งนี้ก็จะเท่ากับ `ui = -1`
แต่ ui มีค่าน้อยสุดที่เป็นได้คือ 0 ค่าก็จะวนกลายเป็นค่ามากสุดของ unsigned int

```
int main() {
  unsigned int ui = 5;
  while (ui >= 0) {
    cout << ui << endl;
    ui--;
  }
}
```

ตัวอย่างเพิ่มเติม: คำนวณ $\sum_{i=1}^{n} i$

```
int main() {
  int sum = 0;
  int i = 1, n;
  cout << "Enter n: "; cin >> n;
  while (i <= n) {
     sum += i;
     i++;
  }
  cout << "Summation is " << sum << endl;
}
```

- อยากจะคำนวณค่า$\sum_{i=1}^{n} i$
- $1 + 2 + \ldots + n$
  - (แน่นอนว่ามันมีสูตรแต่เราอยากจะลองใช้ loop)

```
int main() {
  int sum = 0, n;
  cout << "Enter n: "; cin >> n;
  while (n--) sum += n+1;
  cout << "Summation is " << sum << endl;
}
```

เขียนสั้นกว่า
อ่านเข้าใจยากกว่า

ทำงานผิดในบางค่า!!!
(ลองเดาดูว่าเมื่อ n
เป็นเท่าไรถึงจะผิด?)

ตัวอย่างเพิ่มเติม หาว่าใน string มีอักขระหรือไม่

```
#include <iostream>
using namespace std;
int main() {
  string s,s2;
  char c;
  cout << "Enter a string: "; cin >> s;
  cout << "Enter a char: "; cin >> s2;
  c = s2[0];
  int i = 0, pos;
  bool found = false;
  while (i < s.length()) {
    if (s[i] == c) {
      found = true;
      pos = i;
    }
    i++;
  }
  if (found) {
    cout << "Found " << c << " at position "
      << pos << endl;
  } else {
    cout << "Not found" << endl;
  }
}
```

- เราจะลองเขียนบริการคล้าย ๆ `.find(a)` ของ `string` ดู
- รับค่า string, รับค่าอักขระ แล้วหาว่ามีอักขระดังกล่าวอยู่ที่ตำแหน่งใดใน string ดังกล่าว (ถ้าไม่มีก็แจ้งด้วย)
  - ให้สังเกตการใช้ตัวแปร `found` และ `pos`

คำสั่ง break

```
int main() {
  string s,s2;
  char c;
  cout << "Enter a string: "; cin >> s;
  cout << "Enter a char: "; cin >> s2;
  c = s2[0];
  int i = 0, pos;
  bool found = false;
  while (i < s.length()) {
    if (s[i] == c) {
      found = true;
      pos = i;
      break;
    }
    cout << "Checked at " << i << endl;
    i++;
  }
  if (found) {
    cout << "Found " << c << " at position "
      << pos << endl;
  } else {
    cout << "Not found" << endl;
  }
}
```

- ให้ลองนึกว่าในตัวอย่างที่แล้ว สมมติ s ยาวมาก ๆ แต่ว่าเจออักขระที่ต้องการตั้งแต่ช่องแรก ๆ
  - โปรแกรมจำเป็นต้องทำงานต่อหรือไม่?

- คำสั่ง `break` เมื่อใช้ภายใน block ของ loop ต่าง ๆ จะหยุดการทำงานของ loop ดังกล่าว

output

```
Enter a string: Manufacturing
Enter a char: c
Checked at 0
Checked at 1
Checked at 2
Checked at 3
Checked at 4
Checked at 5
Found c at position 6
```

loop ซ้อน loop

- คำสั่งใน loop จะเป็น loop ก็ได้เหมือนกัน
- ตัวอย่างโจทย์ พิมพ์ สี่เหลี่ยมขนาด R แถว C ช่อง ด้วยตัวอักษร "*"

R	C	ผลลัพธ์
3	5	***** ***** *****
1	7	*******
5	1	* * * * *
0	2	

```
#include <iostream>
using namespace std;
int main() {
  int r,c;
  cout << "Enter R and C: ";
  cin >> r >> c;
  int i = 0;
  while (i < r) {
    int j = 0;
    while (j < c) {
      cout << "*";
      j++;
    }
    cout << endl;
    i++;
  }
}
```

Scope ของตัวแปร

```cpp
#include <iostream>
using namespace std;
int main() {
  int a,b,c;
  cin >> a >> b;
  c = a + b;
  if (a < b) {
    int c;
    int diff = b - a;
    c = 20;
    cout << "in = " << c << endl;
  }
  cout << "out = " << c << endl;
  //cout << diff << endl;
}
```

c ตัวนี้เป็นคนละตัวกับข้างบน

diff ใช้ตรงนี้ไม่ได้

output

```
2 4
in = 20
out = 6
```

- เราสามารถประกาศตัวแปรไว้ใน block ต่าง ๆ ได้
  - ตัวแปรที่ประกาศใน block จะสามารถใช้งานได้ใน block นั้นเท่านั้น
  - ชื่อซ้ำกับตัวแปรข้างนอก block ก็ได้ (ถือว่าเป็นคนละตัวกัน)
  - block ใด ๆ ทั้งหมดเลย (main, if, while, ฯลฯ)
- ตัวแปรที่ตั้งใจใช้งานใน block ก็ควรจะประกาศใน block

ตัวอย่าง scope เพิ่มเติม

```cpp
int main() {
  int r,c;
  cout << "Enter R and C: ";
  cin >> r >> c;
  int i = 0;
  while (i < r) {
    cout << "Row " << i << ": ";
    int j = 0;
    while (j < c) {
      cout << i;
      int i = 0;
      cout << "(" << i << ")";
      j++;
    }
    cout << endl;
    i++;
  }
  //cout << j << endl; // ทำไม่ได้
}
```

i ตัวนี้คือ ตัวเดียวกับข้างบน

i ตัวนี้คือจะคนละตัว

output

```
Enter R and C: 4 3
Row 0: 0(0)0(0)0(0)
Row 1: 1(0)1(0)1(0)
Row 2: 2(0)2(0)2(0)
Row 3: 3(0)3(0)3(0)
```

ค้นหาคำใน string

`getline` อ่านจาก cin จนกดปุ่ม Enter

```
int main() {
  string hay, needle;
  int b;
  cout << "Enter a string: "; getline(cin,hay);
  cout << "Enter another string: "; getline(cin,needle);
  cout << "Enter starting position: "; cin >> b;
  int pos;
  bool found = false;
  while (b < hay.length()) {
    int i = 0;
    bool match = true;
    while (i < needle.length()) {
      if (needle[i] != hay[b+i]) {
        match = false;
      }
      i++;
    }
    if (match) {
      found = true;
      pos = b;
    }
    b++;
  }
  if (found) {
    cout << "Found at " << pos << endl;
  } else {
    cout << "Can't find " << needle <<
      " in \"" << hay << "\"" << endl;
  }
}
```

- เขียน `.find(a, b)` แบบเป็น a เป็น string และเริ่มหาที่ตำแหน่ง b
- โปรแกรมนี้มีที่ผิด และ มีจุดที่ทำให้ดีขึ้นได้
- ลองคิดดูว่าด้วย input ต่อไปนี้ โปรแกรมจะตอบว่าอะไร
  - เมื่อ hay คือ `"This is a book"`

needle	b	ผลลัพธ์ที่ควรเป็น	ผลลัพธ์ที่ได้
This	0	0	
is	0	2	
is	3	5	
book	0	10	
booking	ไม่เจอ	ไม่เจอ	

ค้นหาคำใน string

- ยังมี “จุดที่ต้องพิจารณา” อื่น ๆ อีก เช่น
- Hay = aaaaaaaaaaaaaaaaaa
- Needle = aaaaaaaaaaaaaab
- การหาคำใน string เป็นปัญหาสำคัญมาก ๆ ในชีวิตจริง จงเรียกใช้ .find เถอะ

```
int main() {
  string hay, needle;
  int b;
  cout << "Enter a string: "; getline(cin,hay);
  cout << "Enter another string: "; getline(cin,needle);
  cout << "Enter starting position: "; cin >> b;
  int pos;
  bool found = false;
  while (b < hay.length()) {
    int i = 0;
    bool match = true;
    while (i < needle.length() && i < hay.length()) {
      if (needle[i] != hay[b+i]) {
        match = false;
        break;
      }
      i++;
    }
    if (match && i == needle.length()) {
      found = true;
      pos = b;
      break;
    }
    b++;
  }
  if (found) {
    cout << "Found at " << pos << endl;
  } else {
    cout << "Can't find " << needle <<
      " in \"" << hay << "\"" << endl;
  }
}
```

ถ้ารู้แล้วว่าไม่ใช่ ก็หยุดเลย

แก้กรณีที่คำที่หาตรงกับด้านท้ายของ hay

แก้กรณีที่เจอมากกว่า 1 ครั้ง

อีกสักตัวอย่าง ตรวจสอบจำนวนเฉพาะ

- จำนวนเฉพาะ คือ จำนวนเต็มบวกที่มีตัวประกอบเพียงสองตัวได้แก่ 1 และตัวมันเอง

```cpp
#include <iostream>
using namespace std;
int main() {
  int n;
  cout << "Enter a number: "; cin >> n;
  int divisor = 2;
  while (divisor < n) {
    if (n % divisor == 0) {
      cout << "Not prime" << endl;
      break;
    }
    divisor++;
  }
  if (divisor == n) {
    cout << "Is a prime" << endl;
  }
}
```

ถ้าเปลี่ยน n เป็น sqrt(n) จะยังถูกหรือไม่?

ขออีกอัน: พลิกกลับ string

- รับ string มา แล้วสร้าง string ใหม่ที่เกิดจากอันเก่าสลับหัวท้าย

```
int main() {
  string s,s2;
  cout << "Enter a string: "; cin >> s;
  int i = 0;
  while (i < s.length()) {
    s2 = s2 + s[s.length() - i - 1];
    i++;
  }
  cout << s2 << endl;
}
```

i	n-i-1
0	6
1	5
2	4
3	3
4	2
5	1
6	0

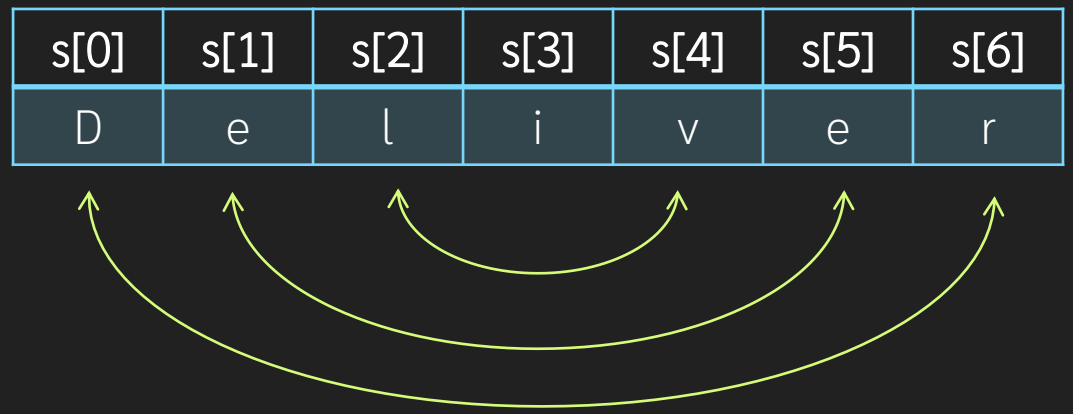
ขออีกอัน: พลิกกลับ string (in-place)

```
int main() {
  string s;
  cout << "Enter a string: "; cin >> s;
  int i = 0;
  while (i < s.length() / 2) {
    swap(s[i],s[s.length() - i - 1]);
    i++;
  }
  cout << s << endl;
}
```

swap(a,b) จะสลับค่าในตัวแปร a กับ b

จำนวนรอบน้อยกว่า

ไม่ต้องสร้าง string ใหม่



ลองทำโจทย์กันหน่อย

- โจทย์ รับจำนวนเต็มบวก n แล้วแสดงรูปสามเหลี่ยมดังตัวอย่าง

n	ผลลัพธ์
1	*
2	** *

3	***
	  **
	  *

4	****
	  ***
	  **
	  *

ออกแบบ

- สังเกต จำนวนบรรทัดเท่ากับ n
  - น่าจะมี loop ที่วิ่งเท่ากับจำนวนบรรทัด
- ในแต่ละบรรทัด จะต้องพิมพ์ตัวอักษร n ตัว
  - ก็น่าจะมี loop อีกอันอยู่ใน loop หลัก
  - จำนวน * ที่พิมพ์แตกต่างกัน
    - น่าจะมีตัวแปรมาเก็บจำนวนที่พิมพ์
  - จำนวนช่องว่างก็ต่างกัน
    - น่าจะมีตัวแปรเช่นกัน

n	ผลลัพธ์
4	**** *** ** *

แถว	จำนวน space	จำนวน *
1	0	4
2	1	3
3	2	2
4	3	1

เฉลย

```cpp
int main() {
  int n;
  cout << "Enter N: "; cin >> n;
  int row = 1;
  while (row <= n) {
    int col = 1;
    int num_space = row-1;
    int num_star = n - num_space;
    while (col <= n) {
      if (col <= num_space) {
        cout << " ";
      } else {
        cout << "*";
      }
      col++;
    }
    cout << endl;
    row++;
  }
}
```

```cpp
int main() {
  int n,row;
  cout << "Enter N: "; cin >> n;
  row = n;
  while (row--) {
    int col = 1;
    while (col <= n) {
      cout << (col++ < n - row ? " " : "*");
    }
    cout << endl;
  }
}
```

do...while loop

- โครงสร้าง `while` loop นั้นจะเช็คเงื่อนไขก่อนแล้วค่อยทำงานซ้ำ ๆ
- มีโครงสร้างอีกแบบที่ทำงานก่อน แล้วค่อยเช็คเงื่อนไข
  - เรียกว่า `do ... while` loop
- รูปแบบคือ

- ใช้เมื่อมีงานที่ต้องทำแน่ ๆ อย่างน้อย 1 รอบ

ตัวอย่าง

- โจทย์พิมพ์พิระมิด
  - รับค่า n
  - ถ้า n ไม่ใช่ -1 ให้พิมพ์พิระมิดขนาด n แล้ววนกลับไปรับค่า n
  - หยุดทำงานเมื่อ n = -1

```
#include <iostream>
using namespace std;
int main() {
  int n;
  do {
    cout << "Enter N: "; cin >> n;
    int row = 0;
    while (row++ < n) {
      int col = n-row;
      while(col--) cout << " ";
      col = row*2-1;
      while(col--) cout << "*";
      cout << endl;
    }
  } while (n != -1);
}
```

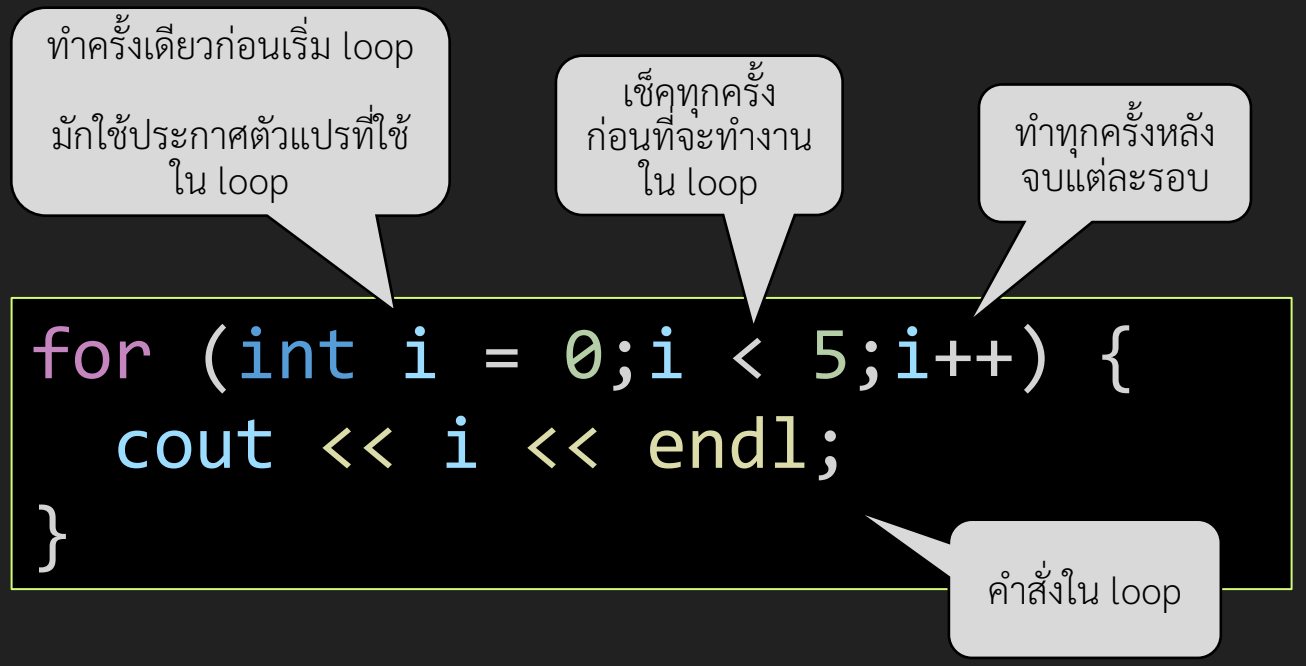
for loop

loop อีกรูปแบบ

- จะเห็นว่า loop ที่ผ่านมามีองค์ประกอบ 4 ส่วนเสมอ ๆ
  - สิ่งที่จะทำซ้ำ
  - เงื่อนไขที่จะทำงาน
  - การเปลี่ยนสถานะในแต่ละรอบ
  - สถานะเริ่มต้นก่อนทำงาน
- for loop เป็นการเขียน loop อีกแบบที่ทำให้เห็นองค์ประกอบชัดเจน
  - มี syntax แบบอื่นด้วย

โครงสร้าง for loop

- `for (ทำตอนแรก; เงื่อนไข;ทำหลังจบแต่ละรอบ) คำสั่ง`



เปรียบเทียบกับ while loop

```
for ( a ; b ; c ) d
```

```
{
  a;
  while (b) {
    d
    c;
  }
}
```

- มีความแตกต่างเล็กน้อยในเรื่อง scope ของตัวแปร
  - ยังไม่ต้องสนใจตอนนี้

รูปแบบที่เห็นบ่อย ๆ ของ for loop

- วนรอบ n ครั้ง

```
int main() {
  int n;
  cout << "Enter round: "; cin >> n;
  for (int i = 0;i < n;i++) {
    cout << i << endl;
  }
}
```

ทำตรงนี้ n รอบ

- ไล่ดูข้อมูลแต่ละตัวในชุดข้อมูล

```
int main() {
  string s;
  cout << "Enter a string: "; cin >> s;
  for (int i = 0;i < s.length();i++) {
      cout << s[i] << endl;
  }
}
```

ทำตรงนี้ s.length() ครั้ง
ค่า i เป็นตั้งแต่ 0 ถึง s.length() - 1

ตัวอย่าง for

- หาผลรวม 1 ถึง n

```
int main() {
  int sum = 0, n;
  cout << "Enter n: "; cin >> n;
  for (int i = 1;i <= n;i++) sum += i;
  cout << "Summation is " << sum << endl;
}
```

```
int main() {
  int sum = 0, n;
  cout << "Enter n: "; cin >> n;
  for (int i = n;i > 0;i--) sum += i;
  cout << "Summation is " << sum << endl;
}
```

ลองทำโจทย์ที่เคยทำแล้วด้วย for

- พิมพ์สามเหลี่ยม

```
int main() {
  int n;
  cout << "Enter n: "; cin >> n;
  for (int i = 0;i < n;i++) {
    for (int j = 0;j < i;j++) cout << " ";
    for (int j = 0;j < n-i;j++) cout << "*";
    cout << endl;
  }
}
```

ลองทำโจทย์ที่เคยทำแล้วด้วย for

- ตรวจสอบจำนวนเฉพาะ

```cpp
#include <iostream>
using namespace std;
int main() {
  int n;
  cout << "Enter a number: "; cin >> n;
  for (int div = 2; div*div <= n;div++) {
    if (n % div == 0) {
      cout << "Not prime" << endl;
      break;
    }
  }
  //if (div == n) {
  //  cout << "Is a prime" << endl;
  //}
}
```

ทำไม่ได้ เพราะ div ประกาศอยู่ใน for ใช้ได้ใน for เท่านั้น

```cpp
#include <iostream>
using namespace std;
int main() {
  int n;
  cout << "Enter a number: "; cin >> n;
  bool prime = true;
  for (int div = 2; div*div <= n;div++) {
    if (n % div == 0) {
      prime = false;
      break;
    }
  }
  if (prime) {
    cout << "Is a prime" << endl;
  } else {
    cout << "Not prime" << endl;
  }
}
```

Range-based for loop

- การไล่ดูข้อมูลแต่ละตัวในชุดข้อมูลมีการใช้บ่อยมาก จึงมี syntax แบบสั้นขึ้นมา

```
int main() {
  string s;
  cout << "Enter a string: "; cin >> s;
  for (int i = 0;i < s.length();i++) {
      cout << s[i] << endl;
  }
}
```

```
int main() {
  string s;
  cout << "Enter a string: "; cin >> s;
  for (char c : s) {
      cout << c << endl;
  }
}
```

- อีกสักพักจะได้เห็นชุดข้อมูลแบบอื่นอีก
  - เช่น vector, set, map

```
int main() {
  string s;
  cout << "Enter a string: "; cin >> s;
  for (auto c : s) {
      cout << c << endl;
  }
}
```

auto หมายถึงให้ C++ ช่วยคิดให้ว่าเป็นตัวแปรประเภทใดถึงจะเหมาะ (ซึ่งจะเป็นประเภทเดียวกับข้อมูลแต่ละตัวในชุดข้อมูล)

ลองทำโจทย์ที่เคยทำแล้วด้วย for

- ค้นหาตัวอักษรใน string
- ใช้ range-based for loop

```
int main() {
  string s,s2;
  cout << "Enter a string: "; cin >> s;
  cout << "Enter a char: "; cin >> s2;
  int pos = 0;
  bool found = false;
  for (auto c : s) {
    if (s2[0] == c) {
      found = true;
      break;
    }
    pos++;
  }
  if (found) {
    cout << "Found " << s2[0] << " at position "
      << pos << endl;
  } else {
    cout << "Not found" << endl;
  }
}
```

หาว่าใน string มีอักขระซ้ำกันหรือไม่

```
int main() {
  string s;
  cout << "Enter a string: "; cin >> s;
  bool unique = true;
  for (auto c1 : s) {
    for (auto c2 : s) {
      if (c1 == c2)
      unique = false;
      break;
    }
  }
  if (unique) {
    cout << "S is unique" << endl;
  } else {
    cout << "S has repetitive char" << endl;
  }
}
```

- มีที่ผิด ผิดตรงไหนดูออกหรือไม่?

หาว่าใน string มีอักขระซ้ำกันหรือไม่

```cpp
int main() {
  string s;
  cout << "Enter a string: "; cin >> s;
  bool unique = true;
  int p1 = 0;
  for (auto c1 : s) {
    int p2 = 0;
    for (auto c2 : s) {
      if (c1 == c2 && p1 != p2) {
        unique = false;
        break;
      }
      p2++;
    }
    p1++;
  }
  if (unique) {
    cout << "S is unique" << endl;
  } else {
    cout << "S has repetitive char" << endl;
  }
}
```

แบบนี้ทำงาน
น้อยกว่า 1 เท่า

```cpp
int main() {
  string s;
  cout << "Enter a string: "; cin >> s;
  bool unique = true;
  for (int p1 = 0;p1 < s.length();p1++) {
    for (int p2 = p1+1;p2 < s.length();p2++) {
      if (s[p1] == s[p2]) {
        unique = false;
        break;
      }
    }
  }
  if (unique) {
    cout << "S is unique" << endl;
  } else {
    cout << "S has repetitive char" << endl;
  }
}
```

ให้สังเกตรูปแบบ for loop 2 ชั้นสำหรับ
ตรวจสอบทุกคู่ตัวอักษร

คำสั่ง continue

- ขณะอยู่ใน loop หากต้องการข้ามการทำงานของ loop รอบนั้นไปยังรอบถัดไป สามารถใช้คำสั่ง continue ได้
- continue ไม่ใช่ break
  - break จะหยุดรอบปัจจุบัน และออกจาก loop ไปเลย
  - continue จะไม่ทำรอบปัจจุบัน แต่ไปทำรอบต่อไป
- ใช้ได้ทั้ง for และ while
  - การใช้ continue ใน for(a ; b ; c) จะทำคำสั่ง c ด้วย

ตัวอย่าง continue

```cpp
#include <iostream>
using namespace std;
int main() {
  int sum = 0;
  for (int i = 0;i < 10;i++) {
    cout << "round " << i << endl;
    if (i % 2 == 0)
      continue;
    sum += i;
  }
  cout << "Sum of odd number is " << sum << endl;
}
```

output

```
round 0
round 1
round 2
round 3
round 4
round 5
round 6
round 7
round 8
round 9
Sum of odd number is 25
```

- ให้สังเกตว่าพิมพ์ round ทุกรอบ แต่ `sum += i` นั้นทำเฉพาะรอบที่เป็นเลขคี่
- ถ้าจะหา sum เลขคี่จริง ๆ ทำแบบนี้ดีกว่า

```cpp
for (int i = 1;i < 10;i+=2)
  sum += i;
```

continue กับ while

- ระวังการใช้ continue กับ while เมื่อคำสั่งในการเปลี่ยนสถานะทำงานที่หลังการ continue

```
#include <iostream>
using namespace std;
int main() {
  int sum = 0;
  int i = 0;
  while (i < 10) {
    cout << "round " << i << endl;
    if (i % 2 == 0)
      continue;
    sum += i;
    i++;
  }
  cout << "Sum of odd number is " << sum << endl;
}
```

ติด loop อยู่ที่รอบ i = 0

สรุป

- รู้จักการทำงานวงวน
- while loop
- for loop
  - range-based for loop
- รู้จักวิธีปรับเปลี่ยนการทำงานในแต่ละรอบด้วย continue, break
- เห็น pattern การวน loop แบบต่าง ๆ
- รู้จักเรื่อง scope ของตัวแปร