Function

***ผู้เขียน**: ทศพร แสงจ้า*

- ตัวอย่างฟังก์ชันที่มีใน `bits/stdc++.h`
- วิธีการเขียน user-defined functions
- เทียบมุมมองของฟังก์ชันทางคณิตศาสตร์และทางคอมพิวเตอร์

แนวคิด

อ่าน official tutorial เรื่อง functions ที่ `https://cplusplus.com/doc/tutorial/functions/`

การเขียนฟังก์ชันในโปรแกรมเป็นการรวมกลุ่มคำสั่งเพื่อทำงานตามจุดประสงค์ โดยสามารถเรียกใช้งานภายหลังในโปรแกรมนั้นได้

การเขียนโปรแกรม

ตัวอย่างฟังก์ชันพร้อมเรียกใช้งาน

ใน C++ Standard Template Library (STL) จะมีฟังก์ชันพร้อมเรียกใช้งานให้ใช้ ซึ่งเมื่อ

```cpp
#include <bits/stdc++.h>
```

แล้ว ในทางการเขียนโปรแกรมเชิงแข่งขันจะทำการ `include` STL ที่จำเป็นทั้งหมดมาให้ ซึ่งตัวอย่างฟังก์ชันที่ใช้งานบ่อยครั้งมีดังนี้

- `abs`, `pow`, `sqrt`, `log2`, `exp` จาก `cmath`
- `swap`, `min`, `max`, `__gcd`, `next_permutation` จาก `algorithm`

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
  cout << "cmath examples\n";
  cout << "abs(-5): " << abs(-5) << "\n";
  cout << "pow(2, 3): " << pow(2, 3) << "\n";
  cout << "sqrt(64): " << sqrt(64) << "\n";
  cout << "log2(64): " << log2(64) << "\n";
  cout << "exp(3): " << exp(3) << "\n";
  cout << "\n";

  cout << "algorithm examples\n";
  int a = 4, b = 7;
  cout << "a b: " << a << " " << b << "\n";
  swap(a, b);
  cout << "a b: " << a << " " << b << "\n";
  cout << "min(a, b): " << min(a, b) << "\n";
  cout << "max(a, b): " << max(a, b) << "\n";
  cout << "__gcd(a, b): " << __gcd(a, b) << "\n";
}
```

```
cmath examples
abs(-5): 5
pow(2, 3): 8
sqrt(64): 8
log2(64): 6
exp(3): 20.0855

algorithm examples
a b: 4 7
a b: 7 4
min(a, b): 4
max(a, b): 7
__gcd(a, b): 1
```

User-defined functions

เราสามารถเขียนฟังก์ชันด้วยตนเองได้เช่นกัน โดยการระบุ

- ชนิด (type) ของค่าที่จะคืนค่า
- ชื่อ (name) ที่ใช้เรียกฟังก์ชันนั้น
- พารามิเตอร์ (parameters) ที่ไว้รับค่าเข้าสู่ฟังก์ชัน
- ชุดคำสั่ง (statements) ที่ระบุวิธีการทำงานของฟังก์ชันนั้น

ในรูปแบบ

```
type name ( param1, param2, ...) { statements }
```

ตัวอย่างต่อไปนี้เป็นการเขียนฟังก์ชันที่

- คืนค่าเป็นชนิด `int`
- ชื่อ `addition`
- รับพารามิเตอร์สองตัว `a, b`
- มีชุดคำสั่งดำเนินการบวกเลขสองตัว

```
#include <bits/stdc++.h>
using namespace std;

int addition(int a, int b) {
  int r;
  r = a + b;
  return r;
}

int main() {
  int z;
  z = addition(5, 3);
  cout << "The result is " << z;
}
```

```
The result is 8
```

มุมมองของฟังก์ชัน

ทางคณิตศาสตร์จะเขียนฟังก์ชันอยู่ในรูป

$$y = f(x)$$

ซึ่งเมื่อเทียบกับทางคอมพิวเตอร์แล้วจะเทียบเท่าเป็นฟังก์ชันที่

- ไม่ได้ระบุชนิดค่าที่จะคืนออกมา แต่ทางคณิตศาสตร์จะมี range ของฟังก์ชันกำกับ
- ชื่อ `f`

- รับพารามิเตอร์ x
- ไม่ได้ระบุชุดดำเนินการคำสั่ง ขึ้นอยู่กับรูปแบบฟังก์ชันทางคณิตศาสตร์

ตัวอย่าง ฟังก์ชันเอกลักษณ์

$$f(x) = x$$

สามารถเขียนในภาษา C++ ได้ดังนี้

```cpp
int f(int x) {
  return x;
}
```

โจทย์ตัวอย่าง - ค่าถัดไปของฟังก์ชันเวียนเกิด

เนื้อหาโจทย์

จงเขียนฟังก์ชัน `next_a(int p, int pn)` หาค่า

$$a_n = a_{n-1} + n \times 2$$

เมื่อทราบค่า a_{n-1}

แนวคิด

หาค่าตามสมการที่โจทย์กำหนด

รหัสเทียม

```cpp
int next_a(int p, int pn) {
    return p + (pn + 1) * 2
}
```

โจทย์ตัวอย่าง - ระยะห่างระหว่างสองจุด

เนื้อหาโจทย์

กำหนดให้ $A(x_1, y_1)$ และ $B(x_2, y_2)$ เป็นจุดสองจุดบนระนาบ จงเขียนฟังก์ชัน `distance(x1, y1, x2, y2)` เพื่อหาระยะระหว่าง A และ B

แนวคิด

$$AB = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

รหัสเทียม

```
double distance(double x1, double y1, double x2, double y2) {
    return sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
}
```

โจทย์ตัวอย่าง - ระยะห่าง Manhattan

เนื้อหาโจทย์

กำหนดให้ $A(x_1, y_1)$ และ $B(x_2, y_2)$ เป็นจุดสองจุดบนระนาบ จงเขียนฟังก์ชัน `manhattan(x1, y1, x2, y2)` เพื่อหาระยะ Manhattan ระหว่าง A และ B

แนวคิด

$$\texttt{manhattan}(A, B) = |x_1 - x_2| + |y_1 - y_2|$$

รหัสเทียม

```cpp
double manhattan(double x1, double y1, double x2, double y2) {
    return abs(x1 - x2) + abs(y1 - y2);
}
```

ปัญหาที่พบบ่อย

การเขียนฟังก์ชันขึ้นเองมีข้อควรระวังดังนี้

Scope of variables

แต่ละฟังก์ชันจะมีสิ่งที่เรียกว่า scope ของตัวเอง โดยตัวแปรในภาษา C++ จะแบ่งเป็น

- Global variables
- Local variables

พิจารณา

```cpp
#include <bits/stdc++.h>
using namespace std;

int g;

int addition(int a, int b) {
  int r;
  g = b;
  r = a + b;
  return r;
}

int main() {
  int z;
  z = addition(5, 3);
  cout << "z is " << z << "\n";
  cout << "g is " << g;
}
```

```
z is 8
g is 3
```

มีตัวแปรทั้งหมดคือ g, a, b, r, z โดย

- g เป็น global variable และสามารถเรียกใช้ได้จากทุกฟังก์ชัน
- a, b, r เป็น local variable ของฟังก์ชัน `addition` จะสามารถเรียกใช้ได้จากฟังก์ชัน `addition` ได้เท่านั้น
- z เป็น local variable ของฟังก์ชัน `main` จะสามารถเรียกใช้จากฟังก์ชัน `main` ได้เท่านั้น

ซึ่งหากมีการเรียกใช้นอก scope of variable เช่น

```cpp
#include <bits/stdc++.h>
using namespace std;

int g;

int addition(int a, int b) {
  int r;
  g = b;
  r = a + b;
  cout << "z is " << z << "\n";
  return r;
}

int main() {
  int z;
  z = addition(5, 3);
  cout << "z is " << z << "\n";
  cout << "g is " << g;
}
```

จะเกิด error

```
sol.cpp: In function ‘int addition(int, int)’:
sol.cpp:10:22: error: ‘’z was not declared in this scope
   10 |   cout << "z is " << z << "\n";
      |                      ^

shell returned 1
```

หรือ

```cpp
#include <bits/stdc++.h>
using namespace std;

int g;

int addition(int a, int b) {
  int r;
  g = b;
```

```
  r = a + b;
  return r;
}

int main() {
  int z;
  z = addition(5, 3);
  cout << "a is " << a << "\n";
  cout << "z is " << z << "\n";
  cout << "g is " << g;
}
```

```
sol.cpp: In function ‘int main()’:
sol.cpp:16:22: error: ‘’a was not declared in this scope
   16 |   cout << "a is " << a << "\n";
      |                      ^

shell returned 1
```

return/type

หากประกาศชนิดของค่าที่จะคืนแล้วลืม `return` จะเกิด warning

```
#include <bits/stdc++.h>
using namespace std;

int addition(int a, int b) {
  int r;
  r = a + b;
}

int main() {
  int z;
  z = addition(5, 3);
  cout << "z is " << z << "\n";
}
```

```
sol.cpp: In function ‘int addition(int, int)’:
sol.cpp:7:1: warning: no return statement in function returning
    non-void [-Wreturn-type]
    7 | }
      | ^
```

ซึ่งหากไม่ต้องการคืนค่าใด ๆ เช่น ในกรณีที่ฟังก์ชันไว้สำหรับแสดงผลด้วยคำสั่ง `cout` เท่านั้นให้ใช้ return type เป็น `void`

Arguments passed by value and by references

หากต้องการเปลี่ยนแปลงค่าของ parameters ให้ส่งผลถึงภายนอกฟังก์ชัน ให้ใช้วิธี pass by references ด้วยการใส่ & ข้างหน้า parameters

ตัวอย่างที่แก้ไขค่าไม่สำเร็จ

```cpp
#include <bits/stdc++.h>
using namespace std;

void swap(int a, int b) {
  int c = a;
  a = b;
  b = c;
  cout << "in swap: " << a << " " << b << "\n";
}

int main() {
  int a = 0, b = 1;
  cout << "in main: " << a << " " << b << "\n";
  swap(a, b);
  cout << "in main: " << a << " " << b << "\n";
}
```

```
in main: 0 1
in swap: 1 0
in main: 0 1
```

ตัวอย่างที่แก้ไขค่าได้สำเร็จ

```cpp
#include <bits/stdc++.h>
using namespace std;

void swap(int &a, int &b) {
  int c = a;
  a = b;
  b = c;
  cout << "in swap: " << a << " " << b << "\n";
}

int main() {
  int a = 0, b = 1;
  cout << "in main: " << a << " " << b << "\n";
  swap(a, b);
  cout << "in main: " << a << " " << b << "\n";
}
```

```
in main: 0 1
in swap: 1 0
in main: 1 0
```