


Function

in C++

ฟังก์ชันในภาษา C++

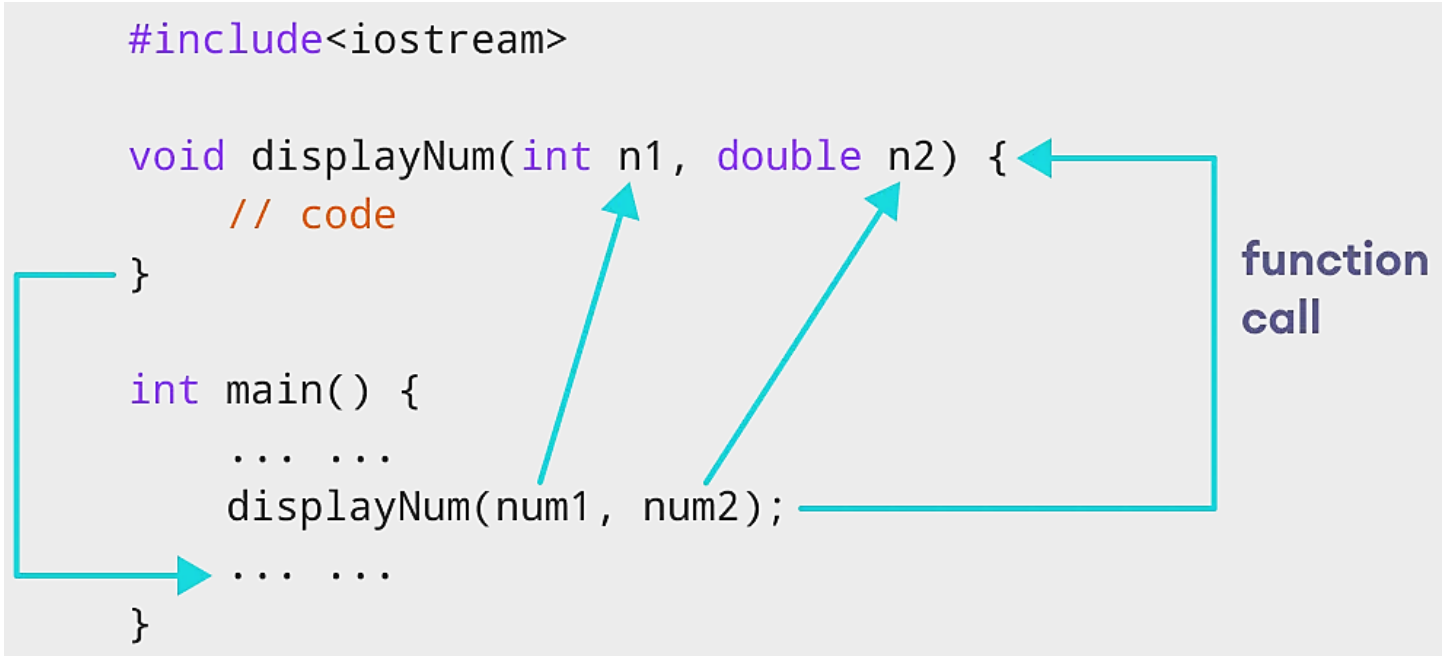
ฟังก์ชันในภาษา C++ ถูกรวมอยู่ใน Library ที่เก็บไว้ใน Header File

โดยฟังก์ชัน ในภาษา C++ คือกลุ่มของคำสั่งที่ถูกจัดกลุ่มไว้ด้วยกันเพื่อทำงานเฉพาะอย่างหนึ่ง ซึ่งฟังก์ชันจะช่วยในการทำงานที่ซ้ำ ๆ ลดความซับซ้อน และเพิ่มความสามารถในการใช้งานซ้ำของโค้ดได้ง่ายขึ้น ฟังก์ชันในภาษา C++ ประกอบด้วยสามส่วนหลัก ๆ คือ

การประกาศ (Declaration)

การนิยาม (Definition)

การเรียกใช้ (Call)



ฟังก์ชันในภาษา C++

การประกาศ (Declaration)

คือ การบอกคอมไพเลอร์ว่ามีฟังก์ชันอยู่ โดยระบุชื่อฟังก์ชัน ประเภทของค่าที่จะคืนค่า และพารามิเตอร์ที่ฟังก์ชันจะรับเข้า แต่ไม่ได้ระบุรายละเอียดหรือเนื้อหาของฟังก์ชัน

การนิยาม (Definition)

คือ การเขียนเนื้อหาของฟังก์ชัน ซึ่งรวมถึงการระบุคำสั่งที่จะให้ฟังก์ชันทำงานตามที่ต้องการ การนิยามฟังก์ชันจะบอกคอมไพเลอร์ว่าฟังก์ชันนั้นทำอะไรและจะคืนค่าอย่างไร

การเรียกใช้ (Call)

คือ กระบวนการที่ทำให้โปรแกรมสามารถใช้ฟังก์ชันที่ได้ถูกนิยามไว้ก่อนหน้านี้ โดยเมื่อฟังก์ชันถูกเรียกใช้โปรแกรมจะทำงานตามชุดคำสั่งที่กำหนดในฟังก์ชันนั้น ๆ

ฟังก์ชันในภาษา C++

การประกาศ (Declaration)

คือ การบอกคอมไพเลอร์ว่ามีฟังก์ชันอยู่ โดยระบุชื่อฟังก์ชัน ประเภทของค่าที่จะคืนค่า และพารามิเตอร์ที่ฟังก์ชันจะรับเข้า แต่ไม่ได้ระบุรายละเอียดหรือเนื้อหาของฟังก์ชัน

โครงสร้างของการประกาศฟังก์ชัน (Function Declaration)

return_type function_name(parameter_list);

return_type: ประเภทของค่าที่ฟังก์ชันจะส่งกลับ เช่น int, void, float

function_name: ชื่อของฟังก์ชัน

parameter_list: รายการของพารามิเตอร์ที่ฟังก์ชันจะรับเข้า (อาจมีหรือไม่มีพารามิเตอร์ก็ได้)

Ex.

```
int add(int a, int b);    // ประกาศฟังก์ชันชื่อ add ที่รับพารามิเตอร์สองตัวและคืนค่าเป็น int
void printMessage();      // ประกาศฟังก์ชันชื่อ printMessage ที่ไม่รับพารามิเตอร์และไม่คืนค่า
```

ฟังก์ชันในภาษา C++

การนิยาม (Definition)

คือ การเขียนเนื้อหาของฟังก์ชัน ซึ่งรวมถึงการระบุคำสั่งที่จะให้ฟังก์ชันทำงานตามที่ต้องการ

การนิยามฟังก์ชันจะบอกคอมไพเลอร์ว่าฟังก์ชันนั้นทำอะไรและจะคืนค่าอย่างไร

โครงสร้างของการนิยามฟังก์ชัน (Function Definition)

```
return_type function_name(parameter_list) {
    // ชุดคำสั่งที่ฟังก์ชันจะทำงาน
    return value;  // คืนค่าถ้าฟังก์ชันมีประเภทการคืนค่า (ไม่ใช่ void)
}
```

return_type: ชนิดของค่าที่ฟังก์ชันจะคืนค่า (int, float, void, ฯลฯ)

function_name: ชื่อของฟังก์ชัน

parameter_list: รายการของพารามิเตอร์ที่ฟังก์ชันจะรับเข้า

return value: ค่าที่จะคืนกลับจากฟังก์ชัน ถ้าเป็นฟังก์ชันประเภทที่ต้องคืนค่า

ฟังก์ชันในภาษา C++

การนิยาม (Definition)

return_type: ชนิดของค่าที่ฟังก์ชันจะคืนค่า (int, float, void, ฯลฯ)

function_name: ชื่อของฟังก์ชัน

parameter_list: รายการของพารามิเตอร์ที่ฟังก์ชันจะรับเข้า

return value: ค่าที่จะคืนกลับจากฟังก์ชัน ถ้าเป็นฟังก์ชันประเภทที่ต้องคืนค่า

Ex.

```
int add(int a, int b) {
    return a + b;  // คืนค่าผลรวมของ a และ b
}
```

ฟังก์ชัน add(int a, int b) รับค่าพารามิเตอร์สองตัว a และ b
แล้วทำการคืนค่าผลรวมของทั้งสอง
โดยประเภทของค่าที่คืนคือ int

ฟังก์ชันในภาษา C++

การนิยาม (Definition)

ตัวอย่างการนิยามฟังก์ชันหลัง main

```
#include <iostream>
using namespace std;
int multiply(int a, int b); // การประกาศฟังก์ชัน (Declaration หรือ Prototype)
int main() {
    int result = multiply(5, 10);  // เรียกใช้ฟังก์ชัน multiply
    cout << "Result: " << result << endl;  // แสดงผลลัพธ์
    return 0;
}
int multiply(int a, int b) { // การนิยามฟังก์ชัน (Definition)
    return a * b;
}
```

การประกาศฟังก์ชัน อยู่ก่อน main() เพื่อบอกคอมไพเลอร์ว่ามีฟังก์ชันนี้อยู่
การนิยามฟังก์ชัน อยู่หลัง main() ซึ่งเป็นเนื้อหาจริงของฟังก์ชัน

ฟังก์ชันในภาษา C++

การเรียกใช้ (Call)

คือ กระบวนการที่ทำให้โปรแกรมสามารถใช้ฟังก์ชันที่ได้ถูกนิยามไว้ก่อนหน้านี้ โดยเมื่อฟังก์ชันถูกเรียกใช้โปรแกรมจะทำงานตามชุดคำสั่งที่กำหนดในฟังก์ชัน

รูปแบบการเรียกใช้ฟังก์ชัน (Function Call)

การเรียกใช้ฟังก์ชันสามารถทำได้โดยใช้ชื่อฟังก์ชัน พร้อมกับระบุค่าที่จะส่งเข้าไปในฟังก์ชัน (ถ้ามี)

function_name(argument_list);

function_name : ชื่อของฟังก์ชันที่ต้องการเรียกใช้

argument_list : รายการของค่าหรือพารามิเตอร์ที่ต้องการส่งเข้าไปในฟังก์ชัน (ถ้ามี)

```
int add(int a, int b) {
    return a + b;  // คืนค่าผลรวมของ a และ b
}
int main() {
int sum = add(5, 10);  // เรียกใช้ฟังก์ชัน add
cout << "Sum: " << sum << endl;  // แสดงผล: Sum: 15
return 0;}
```

ฟังก์ชันในภาษา C++

การเรียกใช้ (Call)

การเรียกใช้ฟังก์ชันหลายครั้ง ฟังก์ชันสามารถถูกเรียกใช้หลายครั้งในโปรแกรม โดยสามารถส่งค่าต่าง ๆ ไปยังฟังก์ชันได้ตามต้องการ

```
#include <iostream>
using namespace std;
double calculateArea(double radius) { // การนิยามฟังก์ชัน calculateArea
    return 3.14 * radius * radius;  // คืนค่าพื้นที่วงกลม
}
int main() {
    double area1 = calculateArea(5.0);  // เรียกใช้ฟังก์ชัน calculateArea กับรัศมี 5.0
    double area2 = calculateArea(10.0); // เรียกใช้ฟังก์ชัน calculateArea กับรัศมี 10.0
    cout << "Area of circle with radius 5.0: " << area1 << endl;  // แสดงผลพื้นที่
    cout << "Area of circle with radius 10.0: " << area2 << endl; // แสดงผลพื้นที่
    return 0;
}
```

ฟังก์ชันในภาษา C++

ตัวอย่างการสร้างและใช้งานฟังก์ชันใน C++

1. ฟังก์ชันที่ไม่มีการคืนค่า (void)

```
#include <iostream>
using namespace std;
void printMessage() { // การนิยามฟังก์ชัน
    cout << "Hello, C++!" << endl;  // แสดงข้อความ
}
int main() {
    // เรียกใช้ฟังก์ชัน printMessage
    printMessage();  // แสดงผล: Hello, C++
    return 0;
}
```

```
Hello, C++!

--------------------------------
Process exited after 0.03775 seconds with return value 0
Press any key to continue . . .
```

ฟังก์ชันในภาษา C++

ตัวอย่างการสร้างและใช้งานฟังก์ชันใน C++

2. ฟังก์ชันที่มีการคืนค่า

```
#include <iostream>
using namespace std;
int add(int a, int b) { // การนิยามฟังก์ชัน
    return a + b;  // คืนค่าผลรวมของ a และ b
}
int main() {
    // เรียกใช้ฟังก์ชัน add
    int result = add(10, 20);  // ส่งค่า 10 และ 20 ไปยังฟังก์ชัน
    cout << "Sum: " << result << endl;  // แสดงผล: Sum: 30
    return 0;
}
```

```
Sum: 30

--------------------------------
Process exited after 0.03653 seconds with return value 0
Press any key to continue . . .
```

ฟังก์ชันในภาษา C++

โจทย์การสร้างฟังก์ชันในภาษา C++

ฟังก์ชันเช็คจำนวนเฉพาะ (Prime Number)

- สร้างฟังก์ชันชื่อ isPrime ที่รับตัวเลขจำนวนเต็มเป็นพารามิเตอร์ โดยคืนค่าเป็น true หากตัวเลขนั้นเป็นจำนวนเฉพาะ และคืนค่า false หากไม่ใช่
- เรียกใช้ฟังก์ชันนี้ใน main() โดยรับตัวเลขจากผู้ใช้ แล้วแสดงผลว่าตัวเลขนั้นเป็นจำนวนเฉพาะหรือไม่

ตัวอย่างผลลัพธ์

Enter a number: 11

11 is a prime number

ฟังก์ชันเช็คจำนวนเฉพาะ

```
#include <iostream>
using namespace std;
bool isPrime(int n) {
        if (n ≤ 1) return false;   // ตัวเลขที่น้อยกว่าหรือเท่ากับ 1 ไม่ใช่จำนวนเฉพาะ
        for (int i = 2; i ≤ n / 2; i++) {
                if (n % i == 0) return false;   // ถ้า n หารด้วย i ลงตัว คือค่า false แสดงว่าไม่ใช่จำนวนเฉพาะ
        }
        return true;   // ถ้าไม่มีตัวหาร คืนค่า true แสดงว่าเป็นจำนวนเฉพาะ
}
int main() {
        int number;
        cout << "Enter a number: ";
        cin >> number;
        if (isPrime(number)) {
                cout << number << " is a prime number." << endl;
        } else {
                cout << number << " is not a prime number." << endl;
        }
return 0;
}
```

ฟังก์ชันในภาษา C++

หลักการและเหตุผลในการใช้ Function

1. **ลดความซ้ำซ้อน** หากเขียนชุดคำสั่งเดิมซ้ำ ๆ กันเกินกว่าหนึ่งครั้งในโปรแกรม ควรสร้างฟังก์ชัน การเขียนโค้ดซ้ำทำให้แก้ไขยากและมีโอกาสเกิดข้อผิดพลาดสูง เมื่อมีชุดคำสั่งที่ต้องทำงานเดิมซ้ำ ๆ แต่ใช้ข้อมูลที่แตกต่างกัน (Input) ให้สร้างฟังก์ชันเพื่อจัดการชุดคำสั่งนั้น
2. **เพิ่มความสามารถในการอ่านและทำความเข้าใจ** การแตกโค้ดชุดใหญ่ให้เป็นฟังก์ชันย่อย ๆ ที่มีชื่อเฉพาะเจาะจง (เช่น isPrime(), calculateArea(), sortData()) จะทำให้โค้ดใน main() หรือในฟังก์ชันอื่น ๆ สั้นลงและเข้าใจง่ายขึ้น
3. **แยกความรับผิดชอบและแก้ไขง่าย** การแยกโค้ดเป็นฟังก์ชันย่อย ๆ แต่ละฟังก์ชันจะทำหน้าที่รับผิดชอบงานเดียว (Single Responsibility Principle) หากเกิดข้อผิดพลาดขึ้นจะสามารถจำกัดขอบเขตการค้นหา (Debugging) ได้ง่ายและรวดเร็ว

โจทย์การสร้างฟังก์ชันในภาษา C++

การหาจำนวนการลบที่น้อยที่สุดเพื่อให้เป็น Palindrome (Minimum Deletions for Palindrome)

ให้เขียนโปรแกรมรับ String S จากนั้นคำนวณและแสดงผลจำนวนตัวอักษร น้อยที่สุด ที่ต้อง ลบ ออกจาก S เพื่อให้ String ที่เหลือกลายเป็น Palindrome

#	Input String (S)	Length (N)	Output Min Deletions	คำอธิบาย (String Palindrome ที่ยาวที่สุด)
1	LEVEL	5	0	LEVEL เป็น Palindrome อยู่แล้ว (ลบ 0 ตัว)
2	AGBGCBA	7	2	LPS คือ A G B G A หรือ A B G B A (ความยาว 5) Min Deletions : 7-5=2
3	AXBCXA	2	2	LPS คือ A B C A (ความยาว 4) Min Deletions : 6-4=2
4	ABCDCBAZ	8	1	LPS คือ ABCDCBA (ความยาว 7) Min Deletions : 8-7=1 (ลบแค่ตัว Z)


```cpp
#include <bits/stdc++.h>
using namespace std;
// ตารางสำหรับเก็บผลลัพธ์ที่คำนวณแล้ว
int memo[10][10];
string S_global; // เก็บ String S ไว้ใน Global
int minDeletions(int i, int j) {
     if (i ≥ j) {// 1.ถ้าเหลือตัวอักษร 1 ตัว หรือ String ว่าง
          return 0;
     }
     // 2. CHECK MEMO: ถ้าเคยคำนวณค่านี้แล้ว ให้คืนค่าที่บันทึกไว้
     if (memo[i][j] ≠ -1) {// -1 หมายถึงยังไม่เคยคำนวณ
          return memo[i][j];
     }
     // 3. RECURSIVE STEP:
     if (S_global[i] == S_global[j]) {
          // กรณีที่ 1: ตัวอักษรคู่กัน (ไม่ต้องลบ)
          // ผลลัพธ์คือการลบที่เหลือตรงกลาง
          memo[i][j] = minDeletions(i + 1, j - 1);
     } else {
          // กรณีที่ 2: ตัวอักษรไม่เท่ากัน (ต้องเลือกลบตัวใดตัวหนึ่ง)
          // 1 + min(ลองลบตัวแรก, ลองลบตัวสุดท้าย)
          memo[i][j] = 1 + min(minDeletions(i + 1, j),
          minDeletions(i, j - 1));
     }
     return memo[i][j];
}
```

```cpp
int main() {
     cout << "Input String (S): ";
     cin >> S_global; // รับ String ตัวแปร Global
     int N = S_global.length();
     // ตั้งค่าตาราง Memoization ทั้งหมดเป็น -1
     memset(memo, -1, sizeof(memo));
     // เริ่มต้นคำนวณจาก String ทั้งหมด (ดัชนี 0 ถึง N-1)
     int K_best = minDeletions(0, N - 1);
     cout << "--- Result ---" << endl;
     cout << "Length of S: " << N << endl;
     cout << "Minimum Deletions: " << K_best << endl;
     cout << "Length of Palindromic: " << N - K_best << endl;

     return 0;
}
```