What is Function 😍

ফাংশন হলো কোডের একটি ব্লক যা নির্দিষ্ট কাজ সম্পাদনের জন্য ডিজাইন করা হয়েছে। যেমনঃ আমাদের হাত, পা, চোখ, নাক; এগুলো একেকটা একেক রকমের কাজ করে, ঠিক তেমনি ফাংশনও নির্দিষ্ট একটি কাজ সম্পন্ন করে।

আমাদের শরীরে মাথা হলো প্রধান নিয়ন্ত্রক, যেখান থেকে নির্দেশ আসে। মাথা যখন বলে হাতকে কিছু ধরতে, হাত তখন সেই কাজটি সম্পন্ন করে। আবার, মাথা যদি বলে চোখকে কিছু দেখতে, চোখ তখন সেই কাজটি করে। ঠিক তেমনিভাবে, যখন আপনি কোনো ফাংশনকে **কল / ইনভোক** করেন, তখন এটি কাজ শুরু করে এবং নির্দিষ্ট কাজটি সম্পন্ন করে।

উদাহরণস্বরূপ:

- মাথা (main) নির্দেশ দেয় হাতকে (a function) কিছু ধরতে।
- হাত (function) তখন সেই নির্দেশ অনুসারে কাজটি সম্পন্ন করে।
- আবার মাথা নির্দেশ দেয় চোখকে (another function) কিছু দেখতে।
- চোখ (function) তখন সেই নির্দেশ অনুসারে কাজটি সম্পন্ন করে।

এইভাবে, ফাংশনগুলোর মাধ্যমে কোডকে ছোট ছোট ব্লকে বিভক্ত করে রাখা হয় এবং প্রয়োজন অনুযায়ী সেই ফাংশনগুলোকে কল করে কাজ সম্পন্ন করা হয়। এতে করে কোড সহজে বুঝতে ও পরিচালনা করতে সুবিধা হয়।

উদাহরণ:

```cpp
#include <iostream>
using namespace std;

// হাতের কাজ করার জন্য ফাংশন
void handFunction() {
    cout << "হাত দিয়ে কিছু ধরা হচ্ছে।" << endl;
}

// চোখের কাজ করার জন্য ফাংশন
void eyeFunction() {
    cout << "চোখ দিয়ে কিছু দেখা হচ্ছে।" << endl;
}

int main() {
    // মাথার নির্দেশ
    handFunction();  // হাতকে কল করা
    eyeFunction();   // চোখকে কল করা

    return 0;
}
```

এই উদাহরণের মাধ্যমে আমরা দেখতে পারছি যে, মাথার (main function) নির্দেশ অনুযায়ী হাত ও চোখ (others function) কাজ সম্পন্ন করছে ✌️

Parts of a Function

1. **Function Type**: ফাংশনটি কোন ধরনের মান রিটার্ন করবে তার উপর ভিত্তি করে type নির্বাচন করতে হবে। যদি কোনো কিছু রিটার্ন না করে, তাহলে `void` ব্যবহার করতে হবে।
2. **Function Name**: ফাংশনটিকে সনাক্ত করার জন্য একটি ইউনিক এবং অর্থবহ নাম দিতে হবে।
3. **Function Body**: ফাংশনের মধ্যে **{ }** বা সেকেন্ড ব্রাকেটের মধ্যকার অংশটিকে বলা হয়। একে কোড-ব্লকও বলা হয়।
4. **Return**: ফাংশনের রিটার্ন হলো ফাংশন তার কাজ সম্পন্ন করার পর যা ফেরত দেয়, যেমন চোখ দেখে মাথাকে যা জানায়। এটা মূলত ফাংশনের আউটপুট, যা অন্য কোথাও ব্যবহার করা যেতে পারে।
5. **Parameters (Optional)**: প্যারামিটার হলো ফাংশনের **()**-এর মধ্যে যে ভেরিয়েবল বা ডেটা পাস করা হয় যাতে ফাংশন কাজ সম্পাদন করতে পারে। এই প্যারামিটারগুলি ফাংশন কলের সময় **Arguments** হিসেবে পাঠানো ভ্যালুকে রিসিভ করে। প্যারামিটারকে আমরা ফাংশনের কান হিসেবে কল্পনা করতে পারি।
6. **Arguments**: ফাংশনের আর্গুমেন্ট হলো ফাংশন কলের সময় পাস করা ভেরিয়েবল বা ডেটা। আর্গুমেন্টগুলি ফাংশনের প্যারামিটারগুলির মান হিসাবে ব্যবহৃত হয়। যা পরে ফাংশনের বডির মধ্যে ব্যবহৃত হয়ে ফাংশন কাজ সম্পাদন করতে সাহায্য করে।

Syntax of a Function

```
functionType functionName(parameters) {
    // Function body: code to be executed
```

```
}
```

Example of Function

```cpp
#include <iostream>
using namespace std;

// Function declaration
int sum(int a, int b) {
    int result = a + b;  // Function body
    return result;       // Returning the result
}

int main() {
    int num1 = 5;
    int num2 = 3;
    int total = sum(num1, num2);  // Function call
    cout << "Sum: " << total << endl;  // Output: Sum: 8
    return 0;
}
```

Breaking Down the Example:

1. **Function Declaration**: `int sum(int a, int b)`
    - `int` : রিটার্ন টাইপ, যা নির্দেশ করে ফাংশন একটি পূর্ণসংখ্যা রিটার্ন করবে।
    - `sum` : ফাংশনের নাম।
    - `int a, int b` : এগুলো হলো প্যারামিটার যা `int` টাইপের।
2. **Function Body**:
    - `int result = a + b;` : প্যারামিটার দুটিকে যোগ করে।
    - `return result;` : যোগফলটি রিটার্ন করে।
3. **Function Call**: `sum(num1, num2)`
    - এটি `sum` ফাংশনকে `num1` এবং `num2` দিয়ে কল করে।
    - ফলাফলটি `total` ভ্যারিয়েবলে সংরক্ষণ করে।
4. **Output**: `cout << "Sum: " << total << endl;`
    - কনসোলে ফলাফলটি প্রিন্ট করে।

Why Use Functions?

1. **Reusability**: একবার কোড লিখে এটি অনেকবার ব্যবহার করতে পারবেন।
2. **Organization**: বড় কাজকে ভেঙ্গে ছোট ছোট আকারে ভাগ করে কাজ করা যায়।
3. **Maintainability**: কোড আপডেট এবং পরিচালনা করা সহজ।

Function বিষয়ে আরও নোট ধাপে ধাপে শেয়ার করা হবে ইনশাআল্লাহ 🤗→