# Hướng dẫn Thực hành – HÀM và Module

## 1 Hàm không trả về giá trị và Hàm có trả về giá trị

### 1.1 Void functions (no return value)

*** Syntax**: **void** FunctionName ( [parameters] )

*** Note**:

- Kiểu void là kiểu không có hiệu lực, ứng với dạng không có giá trị gì cả. Như vậy hàm có kiểu trả về là void sẽ là hàm không trả về giá trị (và cuối hàm không cần phải có lệnh **return**).
- Hàm void vẫn có thể có tham số ([ parameters ]), và trong các tham số vẫn có thể có các tham biến chứa các kết quả trả ra của hàm.

*** Sample code:**

```
include <iostream>
using namespace std;

void Introduce () { // Function with no arguments and no return value
    cout << " Chuong trinh xuat ra cac so nguyen to nho hon N " << endl;
}

// Hàm in ra các số nguyên tố nhỏ hơn N
void printPrimeNumbers(int N) // Function with arguments and no return value
{
    cout << " Cac so nguyen to nho hon " << N << " la : ";
    for (int i = 2; i < N; i++) {
        bool check = true;          // Mac dinh la so nguyen to
        for (int j = 2; j < i; j++)
            if (i % j == 0) {
                check = false;      // Khong la so nguyen to
                break;
            }
        if (check == true)
            cout << i << " ";
    }
}

void main()
{
         int N;
         Introduce();
         cout << " Nhap gia tri N : ";
         cin >> N;
         printPrimeNumbers(N);      // Output: 2 3 5 7
}
```

## 1.2 Value-returning functions

*** Syntax**: **<Return_Type>** FunctionName ( [parameter] )

*** Note:**

- Return_Type phải là 1 kiểu dữ liệu bình thường - không phải kiểu **void**. Cuối hàm phải có lệnh **return <value>** (với value là 1 giá trị thuộc kiểu Return_Type (chỗ khác trong thân hàm cũng có thể có lệnh này) ).
- Phần nội dung trong cặp ngoặc khi định nghĩa hàm (viết hàm) gọi là phần tham số ( parameters ) còn khi sử dụng hàm (call /gọi hàm) gọi là phần đối số (*arguments*).
- Hàm loại này cũng có thể có tham số ([ parameters ]) hoặc không, và trong các tham số vẫn có thể có các tham biến chứa các kết quả trả ra khác của hàm.
- Hàm loại nào cũng có thể có nhiều tham số, khi có nhiều tham số thì phải dùng dấu phẩy (,) để phân cách các tham số (chứ không dùng chấm phẩy).
- Khi định nghĩa hàm nên có ghi chú rõ hàm làm gì, từng tham số mang ý nghĩa gì, giá trị trả về là gì.

*** Sample code:**

+ Xuất các số nguyên tố nhỏ hơn N:

```
// Hàm kiểm tra số N có phải là số nguyên tố
bool isPrimeNumber(int N)  // Function with arguments and a return value
{
     for (int i = 2; i < N; i++)
         if (N % i == 0) return false;
     return true;
}

// Hàm in ra các số nguyên tố có 2 chữ số và trả về số lượng số nguyên tố
int printPrimeNumbers()  //Function with no arguments and a return value
{
     int count = 0;
     for (int i = 10; i <= 99; i++)
         if (isPrimeNumber(i)) {
             std::cout << i << " ";
              count ++;
         }
     return count;
}

int main()  // Function with no arguments and a return value
{
     int num = printPrimeNumbers();
     std::cout << "\n Có tổng cộng "<< num << " số.";
     return 0;
}
```

+ Tìm số đảo ngược của một số nguyên không âm N.

```
// Hàm trả về giá trị bằng với số đảo ngược các chữ số của N (vd: 68 nếu N=86)
int reverse(int N) { // Function with arguments and a return value
    if (N < 10)  return N;

    int rev = 0;
    while (N != 0) {
        rev = rev * 10 + N % 10;
        N /= 10;
    }
    return rev;
}
void main() {  // Function with no arguments and no return value
    std::cout << reverse(2023);
}
```

# 2 Hàm truyền tham trị và Hàm truyền tham biến

## 1.3 Pass by Value functions

*** Syntax of a parameter**: **Type Formal_Variable**

*** Note:**

- Khi gọi hàm dạng truyền tham trị, đối số tương ứng được truyền phải là một giá trị (có thể dưới dạng là một biểu thức).
- Với hàm có đặt giá trị mặc định cho tham số thì có thể gọi hàm mà không truyền đối số tương ứng (đối số sẽ mang giá trị mặc định). *(// các tham số có thiết lập giá trị mặc định nên để ở cuối)*
- C++ có hỗ trợ tính năng **function overloading** cho phép sử dụng cùng một tên gọi cho nhiều hàm (có cùng mục đích).

*** Sample code:**

+ Function Overloading

```
// Hàm tính tổng 2 số nguyên a và b
int Sum(int a, int b) {
    return a + b;
}
// Hàm tính tổng 3 số nguyên a, b, c
int Sum(int a, int b, int c) {
    return a + b + c;
}

// Hàm tính tổng 2 số thực a và b
float Sum(float a, float b) {
    return a + b;
}
```

```
void main()
{
     int a=1, b=2, c=3;
     float d=11.2022, e=2022.2023;

     cout << a << " + " << b << " = " << Sum(a, b) << endl; // Output: 1 + 2 = 3
     cout << a << "+2*" << b << "=" << Sum(a, 2*b) << endl; // Output: 1+2*2=5
     cout << Sum(d, e) << endl; // Output: 2033.4045
     cout << Sum(a, b, c) << endl; // Output: 6
     cout << Sum(a, b, c + Sum(d,e)); // Output: 2039
}
```

+ Default arguments

```
// Hàm tính tổng 4 số, cho phép gọi với 4/3/2 đối số
int Sum(int a, int b, int c=0, int d=0)
{
    return a + b + c + d ;
}

void main()
{
     cout << Sum(5, 10) << endl; // Output: 15
     cout << Sum(5, 10, 15) << endl; // Output: 30
     cout << Sum(5, 10, 15, 20) << endl; // Output: 50
}
```

## 1.4 Pass by Reference functions

*** Syntax of a parameter**: Type & Formal_Variable

*** Note:**

- Khi gọi hàm dạng truyền tham biến, đối số tương ứng được truyền bắt buộc phải là 01 biến.
- Với hàm sau khi xử lý cho ra đúng một kết quả, thì thường kết quả này được để ở giá trị trả về của hàm (và dùng lệnh **return <kết quả>** ), tức khi này sẽ thiết lập hàm ở dạng truyền tham trị (pass by value).
- Với hàm sau khi xử lý cho ra nhiều kết quả (ví dụ như hàm giải phương trình bậc nhất có 2 kết quả, là số lượng nghiệm và giá trị của nghiệm), thì **các kết quả sẽ được thiết lập ở dạng truyền tham biến** (pass by reference) *(trong đó có thể có 1 kết quả đặc biệt được thiết lập ở giá trị trả về của hàm, ví dụ như số lượng nghiệm ở hàm giải phương trình - prototype của hàm Giải phương trình bậc nhất có thể khai báo*

*như sau: int GiaiPTB1( float a, float b, **float &** x) // giá trị trả về của hàm là số lượng nghiệm và tham biến **x** chứa giá trị nghiệm nếu số nghiệm là 01 ).*

*- (Trường hợp đặc biệt: 1/ Khi muốn cập nhật giá trị của chính đối số đưa vào thì có thể không dùng giá trị trả về của hàm mà đặt tham biến ở đối số này - vd: **void** RutGonPhanSo( **PhanSo &** a ); 2/Khi đối số đưa vào có kích thước quá lớn thì dù không có cập nhật gì cũng có thể truyền tham biến để hệ thống không mất thời gian* và không gian tạo bản copy)

- Một hàm có thể có nhiều đối số ở cả 2 dạng (tham biến và tham trị), nhưng không nên quá nhiều.
- Để có thể lưu lại kết quả sau khi hàm kết thúc thì ngoài hình thức truyền tham biến, C/C++ còn hỗ trợ dạng truyền tham trỏ (pass by pointer) – sẽ giới thiệu ở môn Kỹ thuật Lập trình.

*** Sample code:**

+ Hàm hoán vị 2 giá trị *a* và *b*.

```cpp
#include <iostream>

using namespace std;

void Swap(int &a, int &b)
{
    int temp = a;
    a = b;
    b = temp;
}
```

```cpp
int main()
{
    int a, b;
    a = 2; b = 3;

    Swap(a, b);
    cout << a << " " << b;
    // Output: 3 2

    return 0;
}
```

Lưu ý: Ta không thể truyền hằng số vào các hàm sử dụng tham biến. Ví dụ ở mã nguồn trong mục 2, ta ***không thể gọi*** `mySwap(2, 3)` như `reverse(2023)` ở mục 1.2.

# 3 Cách tổ chức các hàm trong Project

## 3.1. Cách phân chia module cho các thành viên trong project

Một dự án (project) thường có nhiều xử lý trên nhiều loại đối tượng khác nhau, và cần một nhóm nhiều thành viên cùng thực hiện. Ta cần chia các công việc phải xử lý thành các module tương ứng (với từng loại đối tượng) và phân mỗi module cho một thành viên (hoặc 1 nhóm con) giải quyết, rồi sau đó kết hợp lại với nhau.

Ví dụ, với bài toán yêu cầu giải phương trình không chỉ trên miền Số thực mà còn phải đáp ứng trên trường Phân số và Số phức (các hệ số và nghiệm đều là số thực, hoặc đều là phân số, hoặc đều là số phức) thì ta có thể chia thành các module như sau:

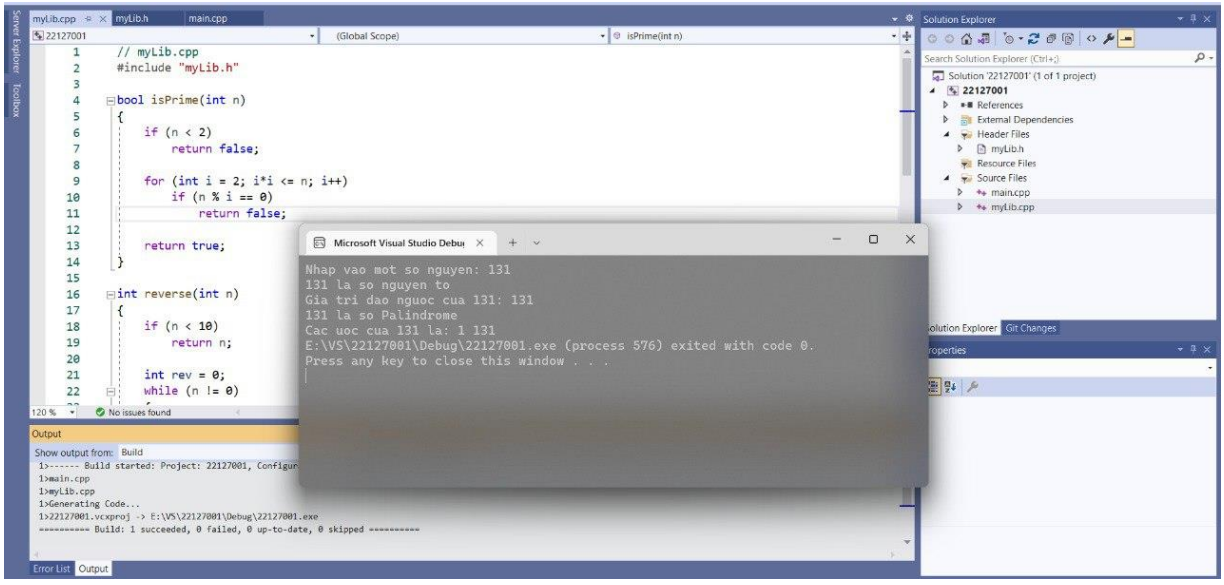
- Module Phân_số : chứa các hàm nhập /xuất phân số, cộng /trừ /nhân /chia phân số, rút gọn phân số (về dạng tối giản), ...
- Module Số_phức : chứa các hàm nhập /xuất số phức, cộng /trừ /nhân /chia số phức, ...
- Module Giải_Phương_trình: chứa các hàm giải các dạng phương trình theo yêu cầu, nhập các hệ số, xuất kết quả giải, ...

## 3.2. Khai báo prototype (khai báo hàm)

Mỗi module có một cặp tập tin là header (.h hoặc .hpp) và source (.c hoặc .cpp).

Trong đó, tập tin header chứa: các khai báo hàm, các khai báo kiểu dữ liệu…; tập tin source chứa định nghĩa hàm (thân hàm).

Trong một **project Visual Studio**, tập tin header được đặt trong thư mục **Header Files** như hình:



Ví dụ: Viết chương trình cho phép nhập vào số nguyên và thực hiện các yêu cầu:

- Kiểm tra số vừa nhập có là số nguyên tố, nếu không thì in ra tất cả các ước của nó.
- Kiểm tra số đó có phải là số Palindrome (Số Palindrome là số mà bạn đọc các chữ số theo chiều xuôi và chiều ngược thì đều như nhau. Ví dụ như 191, 6886, … là các số Palindrome).

⇨ Với đề bài trên, ta có thể thiết kế một module đặt tên là myLib, với tập tin header có tên myLib.h và tập tin source có tên myLib.cpp.

```
// myLib.h
// Function declarations

bool isPrime(int); // Hàm kiểm tra 1 số nguyên có phải là nguyên tố
int reverse(int); // Hàm trả về giá trị bằng với số đảo ngược các chữ số
bool isPalindrome(int); // Hàm kiểm tra 1 số nguyên có phải là số Palindrome
```

```
// myLib.cpp
#include "myLib.h"

// Hàm kiểm tra n có phải là số nguyên tố
bool isPrime(int n)
{
    if (n < 2)
        return false;

    for (int i = 2; i*i <= n; i++)
        if (n % i == 0)
            return false;

    return true;
}

// Hàm trả về giá trị bằng với số đảo ngược các chữ số của m (vd: 68 nếu n=86)
int reverse(int n)
{
    if (n < 10)
        return n;

    int rev = 0;
    while (n != 0)
    {
        rev = rev * 10 + n % 10;
        n /= 10;
    }

    return rev;
}
// Hàm kiểm tra n có phải là số Palindrome (vd nhu 191, 6886)
```

```
bool isPalindrome(int n)
{
    return n == reverse(n);
}
```

Sau đó, chương trình chính (`main`) sẽ gọi các hàm trong module myLib để thực thi.

```
// main.cpp
#include <iostream>
#include "myLib.h"

using namespace std;

// Hàm in ra các ước số của n
void printDivisors(int n)
{
    for (int i = 1; i <= n; i++)
        if (n % i == 0)
            cout << i << " ";
}

int main()
{
    int n;

    cout << "Nhap vao mot so nguyen: ";
    cin >> n;

    cout << n << " la ";
    if (isPrime(n))
        cout << "So nguyen to.\n";
    else {
        cout << "Hop so, co cac uoc la: ";
        printDivisors(n);
     }

    cout << n;
    if (!(isPalindrome(n)))
        cout << " khong";
    cout << " la so Palindrome.\n";

    return 0;
}
```

HẾT