Hàm

- Bài toán: Viết chương trình nhập 10 mảng và xuất ra màn hình 10 mảng đó.
- Giải quyết vấn đề: Sử dụng hàm có thể viết gọn hơn.

Hàm

- Hàm(function) là một đoạn các câu lệnh có thể tái sử dụng. Hàm cho phép lập trình viên cấu trúc chương trình thành những phân đoạn khác nhau để thực hiện những công việc khác nhau.
- Để sử dụng được hàm phải xác định được 2 giá trị là đầu ra và đầu vào.

Khai báo hàm

```
[KIỂU DỮ LIỆU TRẢ VỀ] [TÊN HÀM]([CÁC THAM SỐ]) {
    //định nghĩa các lệnh
    return [BIẾN HOẶC GIÁ TRỊ CỦA KIỂU TRẢ VỀ]
}
```

Định nghĩa hàm

```
int max(int num1, int num2) {
   int result;
   if (num1 > num2)
      result = num1;
   else
      result = num2;
   return result;
}
int main() {
    int a, b = 3, c = 4;
    a = max(b, c);
    cout<<a<<endl;
    return 1;
}
```

Định nghĩa hàm

```
int max(int num1, int num2) {
   //num1 = b
   //num2 = c
   int result;
   if (num1 > num2)
      result = num1;
   else
      result = num2;
   return result;
}
max(b, c) -> a = result
```

Định nghĩa hàm

```
void print(int num1, int num2) {
   cout<<num1<<endl;
   cout<<num2<<endl;
}
print(4, 5);
```

Định nghĩa hàm

```
void print() {
   cout<<"Nguyen Van A"<<endl;
   cout<<"Nguyen Van B"<<endl;
}
print();
```

Định nghĩa hàm

```
void print(int num) {
   cout<<num;
}
void print(int num1, int num2) {
   cout<<num1<<endl;
   cout<<num2<<endl;
}
print(4);
print(6, 7);
```

Định nghĩa hàm

```
void print(int num1, int num2) {
    if (num1 > num2) {
        return;
    }
    cout<<num1 - num2<<endl;
}
```

Định nghĩa hàm

```
int print(int num1, int num2) {
    if (num1 > num2) {
        return num1;
    }
}
//Lỗi
```

Định nghĩa hàm

```
int print(int num1, int num2) {
    if (num1 > num2) {
        return num1;
    } else {
        return num2;
    }
}
```

Định nghĩa hàm

```
int print(int num1, int num2 = 0) {
    if (num1 > num2) {
        return num1;
    } else {
        return num2;
    }
}
print(5) -> 5
print(-1) -> 0
print(4, 8) -> 8
```

Định nghĩa hàm

```
int total(int a[], int n) {
    int s = 0;
    for(int i = 0; i < n; i++) {
        s += a[i];
    }
    return s;
}
int a[5];
...
cout<<total(a, 5);
```

Định nghĩa hàm

```
int total(int a[10][], int n) {
    int s = 0;
    for(int i = 0; i < n; i++) {
        s += a[i][i];
    }
    return s;
}
int a[10][10];
...
cout<<total(a, 10);
```

Phạm vi của biến

- Biến cục bộ: Bên trong một hàm hoặc một khối, được gọi là biến cục bộ
- Biến toàn cục: Bên ngoài của tất cả hàm kể cả hàm main, được gọi là biến toàn cục.

Biến toàn cục

- Có tác dụng trong cả chương trình, bất kể ở đâu.
- Giải phóng sau khi kết thúc chương trình.

Biến toàn cục

```
int c;
void add(int x, int y) {
    c = x + y;
}
int main() {
    int x = 6, y = 9;
    add(x, y);
    cout <<c<< endl;
    return 0;
}
```

Biến cục bộ

- Chỉ có tác dụng trong hàm hoặc khối block.
- Giải phóng sau khi kết thúc hàm hoặc khối block.
- Nếu trong hàm hoặc khối block khai báo 1 biến cục bộ giống tên 1 biến nằm ở phía ngoài. Thì khi tính toán sẽ ưu tiên cho biến vừa khai báo.

Biến cục bộ

```
int main() {
     int n = 6;
     double d = 9;
     cout << "Enter a value for n: ";
     cin >> n;
     cout << "You entered: " << n << endl;
     return 0;
}
```

Biến cục bộ

```
int c;
int add(int x, int y) {
    int c = x + y;
    return c;
}
int main() {
    int x = 6, y = 9;
    cout << add(x, y) << endl;
    cout << c << endl;
    return 0;
}
```