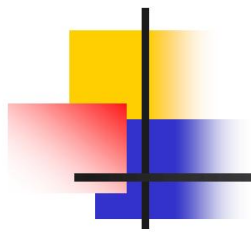


# Chức năng

Chỉ dành cho Trung tâm Aptech sử dụng

## Phiên 9



# Mục tiêu

---

Giải thích cách sử dụng hàm

Giải thích cấu trúc của hàm      Giải thích

khai báo hàm và nguyên mẫu hàm      Giải thích các loại biến

khác nhau      Giải thích cách gọi hàm      Gọi

theo giá trị      Gọi theo tham chiếu

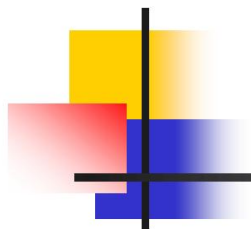
Giải thích các

quy tắc phạm vi cho

hàm      Giải thích các hàm trong chương trình

đa tệp      Giải thích các lớp lưu trữ      Giải

thích các con trỏ hàm



# Chức năng

---

Một hàm là một phân đoạn chương trình độc lập thực hiện một nhiệm vụ cụ thể, được xác định rõ ràng

Các hàm thường được sử dụng như các chữ viết tắt cho một loạt các lệnh được thực hiện nhiều lần

Chỉ dành cho Trung tâm Aptech sử dụng

## Các hàm dễ viết và dễ hiểu

Việc gỡ lỗi chương trình trở nên dễ dàng hơn khi cấu trúc của chương trình rõ ràng hơn, do hình thức mô-đun của nó

Các chương trình có chứa các chức năng cũng dễ bảo trì hơn, vì các sửa đổi, nếu cần, sẽ được giới hạn ở một số chức năng nhất định trong chương trình



# Cấu trúc chức năng

---

Cú pháp chung của hàm trong C là:

```
type_specifier function_name (arguments)
{
    body of the function
}
```

Chỉ sử dụng tại Trung tâm Aptech

`type_specifier` chỉ định kiểu dữ liệu của giá trị, chức năng sẽ trả về.

Một tên hàm hợp lệ sẽ được chỉ định để xác định chức năng

Các đối số xuất hiện trong dấu ngoặc đơn cũng được gọi là tham số chính thức .

# Đổi số của một hàm

```
#include <stdio.h>
main()
{
    int i;
    for (i =1; i <=10; i++)
        printf ("\nsquare of %d is %d ", i, squarer(i));
}

squarer(int x)
/* int x; */
{
    int j;
    j = x * x;
    return (j);
}
```

Chỉ dành cho Trung tâm Aptech sử dụng

Lập luận thực tế

Lập luận chính thức

Chương trình tính bình phương các số từ 1 đến 10

Dữ liệu được truyền từ hàm main() đến hàm squarer()

Hàm này hoạt động trên dữ liệu bằng cách sử dụng các đối số



# Trở về từ hàm

```
squarer (int x)
/* int x; */
{
    int j;
    j = x * x;
    return (j);
}
```

Chỉ sử dụng tại Trung tâm Aptech

Nó chuyển quyền điều khiển từ hàm trở lại chương trình gọi ngay lập tức.

Bất cứ thứ gì nằm trong dấu ngoặc đơn theo sau lệnh return được trả về dưới dạng giá trị cho chương trình gọi.



# Kiểu dữ liệu của một hàm

---

```
type_specifier function_name (arguments)
```

```
{
```

```
    body of the function
```

```
}
```

Chỉ dành cho Trung tâm Aptech sử dụng

Type\_specifier không được viết trước hàm squarer(), vì squarer() trả về giá trị kiểu số nguyên

Type\_specifier không bắt buộc nếu kiểu số nguyên của giá trị được trả về hoặc nếu không có giá trị nào được trả về

Tuy nhiên, để tránh sự không nhất quán, nên có một kiểu dữ liệu đã chỉ định



# Gọi một hàm

---

Dấu chấm phẩy được sử dụng ở cuối câu lệnh khi một hàm được gọi, nhưng không phải sau định nghĩa hàm

Dấu ngoặc đơn là bắt buộc sau tên hàm, bất kể hàm có đối số hay không

Chỉ có một giá trị có thể được trả về bởi một hàm

Chương trình có thể có nhiều hơn một chức năng

Hàm gọi một hàm khác được gọi là hàm/chương trình gọi

Hàm được gọi được gọi là hàm/chương trình được gọi





# Khai báo hàm

---

Việc khai báo một hàm trở nên bắt buộc khi hàm đó được sử dụng trước khi định nghĩa nó

Hàm `address()` được gọi trước khi nó được định nghĩa

Một số trình biên dịch C trả về lỗi nếu hàm không được khai báo trước khi gọi

```
#include <stdio.h>
main()
{
    :
    :
    address()
    :
}

address()
{
    :
    :
    :
}
```

Điều này đôi khi được gọi là tuyên bố ngầm định



# Nguyên mẫu chức năng

---

Chỉ định kiểu dữ liệu của các đối số

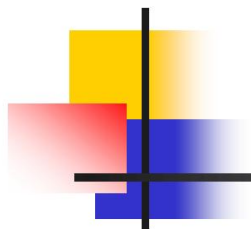
```
ký tự abc(int x, nt y);
```

Chỉ dành cho Trung tâm Aptech sử dụng

Lợi thế :

Bất kỳ chuyển đổi kiểu bất hợp pháp nào giữa các đối số được sử dụng để gọi một hàm và định nghĩa kiểu của các tham số của nó đều được báo cáo

```
char noparam (không có giá trị);
```



# Biến số

---

## Biến cục bộ

Được khai báo bên trong một hàm

Được tạo khi vào một khối và bị hủy khi thoát  
từ khối

Chỉ dành cho Trung tâm Aptech sử dụng

## Tham số chính thức

Được khai báo trong định nghĩa của hàm như các tham số Hoạt  
động giống như bất kỳ biến cục bộ nào bên trong một hàm

## Biến toàn cục

Được khai báo bên ngoài tất cả các  
hàm Giữ giá trị trong suốt quá trình thực thi chương trình



# Lớp lưu trữ-1

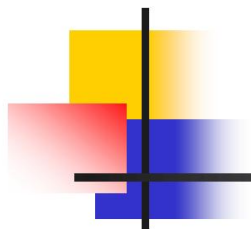
---

Mỗi biến C có một đặc điểm được gọi là lớp lưu trữ

Lớp lưu trữ xác định hai đặc điểm của biến:

Chỉ dành cho Trung tâm Aptech sử dụng

- Tuổi thọ - Tuổi thọ của một biến là khoảng thời gian nó giữ một giá trị cụ thể
- Khả năng hiển thị - Khả năng hiển thị của một biến xác định các phần của chương trình có thể nhận dạng biến đó



# Lớp lưu trữ-2

---

tự động

Chỉ dành cho Trung tâm Aptech sử dụng  
bên ngoài

tĩnh

đăng ký



# Quy tắc phạm vi chức năng

---

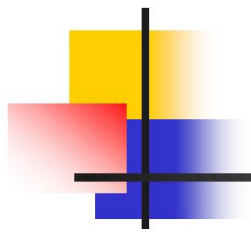
Quy tắc phạm vi - Quy tắc quản lý xem một đoạn mã có biết hoặc có quyền truy cập vào một đoạn mã hoặc dữ liệu khác hay không

Chỉ dành cho Trung tâm Aptech sử dụng  
Mã trong một hàm là riêng tư hoặc cục bộ đối với hàm đó  
chức năng

## Hai hàm có phạm vi khác nhau

Hai hàm ở cùng cấp độ phạm vi

Một hàm không thể được định nghĩa trong một hàm khác  
chức năng



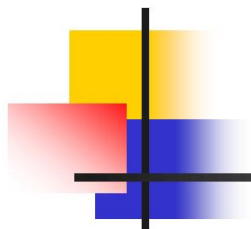
# Gọi các hàm

---

Gọi theo giá trị

Chỉ dành cho Trung tâm Aptech sử dụng

Gọi theo tham chiếu



# Gọi theo giá trị

---

Trong C, theo mặc định, tất cả các đối số hàm được truyền theo giá trị

Khi các đối số được truyền cho hàm được gọi, các giá trị là  
được truyền qua các biến tạm thời

Tất cả các thao tác chỉ được thực hiện trên các biến tạm thời này

Các đối số được cho là được truyền theo giá trị khi giá trị của biến được truyền cho hàm được gọi và bất kỳ thay đổi nào trên giá trị này đều không ảnh hưởng đến giá trị ban đầu của biến được truyền





# Gọi theo tham chiếu

---

Trong lệnh gọi theo tham chiếu, hàm được phép truy cập vào vị trí bộ nhớ thực tế của đối số và do đó có thể thay đổi giá trị của các đối số của chương trình gọi

Chỉ dành cho Trung tâm Aptech sử dụng

Định nghĩa

```
getstr(char *ptr_str, int *ptr_int);
```

Gọi

```
gettr(pstr, &var);
```



# Gọi hàm lồng nhau

---

```
chủ yếu()
```

```
{
```

```
·
```

```
· Chỉ dành cho Trung tâm Aptech sử dụng
```

```
·
```

```
xuôi ngược đều giống nhau();
```

```
·
```

```
·
```

```
}
```

```
xuôi ngược đều giống nhau()
```

```
{
```

```
getstr();
```

```
đảo ngược();
```

```
cmp();
```

```
·
```

```
·
```

```
}
```



# Các chức năng trong chương trình Multifile

---

Các hàm cũng có thể được định nghĩa là tĩnh hoặc bên ngoài

Các hàm tĩnh chỉ được nhận dạng trong tệp chương trình và phạm vi của chúng không mở rộng ra ngoài tệp chương trình

```
static fn_type fn_name (danh sách đối số);
```

Chức năng bên ngoài được nhận dạng thông qua tất cả các tập tin của chương trình

```
extern fn_type fn_name (danh sách đối số);
```



# Con trỏ hàm

Địa chỉ là điểm vào của hàm      Hàm có vị trí vật lý trong bộ nhớ có thể được gán cho một con trỏ

```
#include <stdio.h>
#include <chuỗi.h>
kiểm tra void(char *a, char *b, int (*cmp)());
```

chủ yếu()

```
{
    ký tự s1[80];
    số nguyên (*p)();
    p = strcmp;
    được(s1);
    được(s2);
    kiểm tra(s1, s2, p);
}
```

```
void kiểm tra(char *a, char *b, int (*cmp)())
{
    printf("kiểm tra sự bằng nhau \n");
    nếu (!(*cmp)(a,b))
        printf("Bằng");
    khác
        printf("Không bằng");
}
```