

[Kỹ thuật lập trình(1): Cơ bản về ngôn ngữ lập trình]

Bộ môn Hệ thống thông tin
Khoa Công nghệ thông tin

[Nội dung]

- Giới thiệu chung
- Lệnh nhập/xuất
- Lệnh điều kiện
- Lệnh vòng lặp
- Hàm
- Kiểu mảng
- Xâu kí tự
- Kiểu cấu trúc (struct) và kiểu hợp (union)
- Làm việc với tệp

[Giới thiệu chung]

- Ngôn ngữ C ra đời năm 1972
- Phát triển thành C++ vào năm 1983
- Ngôn ngữ được sử dụng rất phổ biến
- Có nhiều trình biên dịch C khác nhau
 - Turbo C, Borland C
 - GCC
- Thực hành trên Turbo C
 - Cung cấp môi trường tích hợp cho phép soạn thảo và biên dịch

18-Jan-13

3

[Giới thiệu chung]

- Một số phím soạn thảo

Phím	Chức năng
←↑↓→	Di chuyển con trỏ sang trái, lên, xuống, sang phải
Home	Đưa con trỏ về đầu dòng
End	Đưa con trỏ về cuối dòng
PgUp	Đưa con trỏ về đầu một trang màn hình
PgDw	Đưa con trỏ về cuối một trang màn hình
Ctrl + →	Dịch con trỏ sang phải một chữ
Ctrl + ←	Dịch con trỏ sang trái một chữ

18-Jan-13

4

[Giới thiệu chung]

■ Một số phím soạn thảo

Phím	Chức năng
Enter	Xuống dòng
Insert	Chuyển đổi chế độ chèn/đề
Delete	Xóa kí tự ngay sau vị trí con trỏ
Back space	Xóa kí tự ngay trước vị trí con trỏ
Ctrl + Y	Xóa dòng kí tự chứa con trỏ
Ctrl + Q + Y	Xóa các kí tự từ vị trí con trỏ đến cuối dòng

[Giới thiệu chung]

■ Một số phím soạn thảo

Phím	Chức năng
Ctrl + K + C	Chép khối tới vị trí mới của con trỏ
Ctrl + K + V	Chuyển khối tới vị trí mới của con trỏ
Ctrl + K + Y	Xóa cả khối
Ctrl + K + W	Ghi một khối vào một tệp trên đĩa
Ctrl + K + R	Đọc một khối từ một tệp trên đĩa
Ctrl + Q + B	Dịch chuyển con trỏ về đầu khối
Ctrl + Q + K	Dịch chuyển con trỏ về cuối khối
Ctrl + Q + F	Tìm kiếm một cụm từ
Ctrl + Q + A	Tìm kiếm một cụm từ và sau đó thay thế bằng một cụm từ khác
Ctrl + Q + L	Lặp lại công việc Ctrl + Q + F hoặc Ctrl + Q + A cuối cùng

[Giới thiệu chung]

■ Từ khóa

- các từ dành riêng của ngôn ngữ C
- từ khóa phải được sử dụng đúng cú pháp
- một số từ khóa thông dụng

auto	break	case	char	continue	default
do	double	else	extern	float	for
goto	if	int	long	register	return
short	sizeof	static	struct	switch	typedef
union	unsigned	void	volatile	while	

[Giới thiệu chung]

■ Tên (identifier)

- Dùng để định danh các thành phần của chương trình
- Tên biến, tên hàm, tên hằng, ...
- Tên là một dãy các kí tự gồm các chữ cái [a-z, A-Z, 0-9] và gạch nối “_”
- Lưu ý:
 - tên không được chứa kí tự trống,
 - tên không được bắt đầu bằng một chữ số,
 - tên không được trùng với từ khóa
- Nên đặt các tên gọi nhớ, có ý nghĩa
- Tên chuẩn: một số tên có sẵn của trình biên dịch

[Giới thiệu chung]

- Hằng
 - là đại lượng có giá trị không thay đổi được trong chương trình
 - ví dụ
 - 111 hằng là một số
 - 'b' hằng là một kí tự
 - "lập trình" hằng là một chuỗi kí tự
- *Biến*
 - là đại lượng có thể thay đổi được giá trị trong chương trình
- *Biểu thức*
 - là một công thức tính toán để có một giá trị theo một qui tắc toán học
 - ví dụ: $x + y * z$

[Giới thiệu chung]

- Mỗi một câu lệnh C đều phải kết thúc bởi một dấu “;”
- Lời chú thích được đặt giữa hai dấu “/*” và “*/”
 - Ví dụ
/* Đây là một chú thích */
- Khi viết chương trình nên sử dụng các lời chú thích
- Trình biên dịch C phân biệt chữ in hoa và chữ in thường

[Giới thiệu chung]

- Các kiểu dữ liệu chuẩn
 - Kiểu kí tự
 - Kiểu số nguyên
 - Kiểu số thực

[Giới thiệu chung]

- Kiểu kí tự
 - Kiểu **char**
 - Chiếm một byte
 - Biểu diễn các kí tự trong bảng mã ASCII
 - Ví dụ
 - 'a' có giá trị mã ASCII là 65
 - '0' có giá trị mã ASCII là 48
 - Kiểu kí tự đồng thời cũng là kiểu số nguyên
 - Có hai kiểu char: : **signed char** và **unsinged char**

Kiểu kí tự	Kích thước	Miền giá trị
signed char	1 byte	-128 -> 127
unsinged char	1 byte	0 -> 255

[Giới thiệu chung]

- Kiểu số nguyên
 - Có nhiều kiểu số nguyên

Kiểu số nguyên	Kích thước	Miền giá trị
int, short	2 byte	-32768 -> 32767
unsigned int, unsigned short	2 byte	0 -> 65535
long	4 byte	-2147483648 -> 2147483647
unsigned long	4 byte	0 -> 4294967295

[Giới thiệu chung]

- Kiểu số thực
 - Có nhiều kiểu số thực

Kiểu số thực	Kích thước	Miền giá trị
float	4 byte	3.4E-38 -> 3.4E+38
double	8 byte	1.7E-308 -> 1.7E+308
long double	10 byte	3.4E-4932 -> 1.1E+4932

[Giới thiệu chung]

■ Kiểu số thực

- Có hai cách biểu diễn số thực
 - Dạng thập phân: dùng dấu chấm để ngăn cách phần nguyên và phần thập phân
 - Ví dụ: -12.345672, 1203.8375
 - Dạng khoa học: gồm phần định trị và phần mũ của cơ số 10, hai phần cách nhau bởi chữ E hoặc e
 - Ví dụ: 6.123E+02

[Giới thiệu chung]

■ Chuyển kiểu (casting)

- Ngôn ngữ C cho phép chuyển kiểu: chuyển từ kiểu này sang kiểu khác
- Cú pháp: **(kiểu_mới)biểu_thức**
- Ví dụ

```
int i;
i = (int)10.45    /* i = 10 */
float x;
x = (float)1/3;   /* x = 1.0/3 = 0.3333 */
```


[Giới thiệu chung]

- Các phép toán
 - Các phép toán trên số nguyên
 - Cộng: +
 - Trừ: -
 - Nhân: *
 - Chia lấy phần nguyên: /
 - Chia lấy phần dư: %
 - Các phép toán trên số thực
 - Cộng: +
 - Trừ: -
 - Nhân: *
 - Chia: /

[Giới thiệu chung]

- Các phép toán
 - Các phép toán quan hệ (so sánh)
 - So sánh bằng nhau: ==
 - So sánh khác nhau: !=
 - So sánh lớn hơn: >
 - So sánh nhỏ hơn: <
 - So sánh lớn hơn hoặc bằng: >=
 - So sánh nhỏ hơn hoặc bằng: <=
 - Biểu thức chứa các phép toán quan hệ được gọi là biểu thức quan hệ
 - Biểu thức quan hệ có giá trị **đúng** hoặc **sai**

[Giới thiệu chung]

■ Các phép toán

○ Các phép toán logic

- Kiểu logic trong C không được định nghĩa một cách tường minh

- Một giá trị khác 0 là đúng, một giá trị bằng 0 là sai

Phép toán	Kí hiệu	Ví dụ
Và (AND)	&&	2 && 0 = sai
Hoặc (OR)		10 5 = đúng
Phủ định (NOT)	!	!0 = đúng

[Giới thiệu chung]

■ Các phép toán

○ Các phép toán trên bit

- Phép OR từng bit: |
- Phép AND từng bit: &
- Phép XOR từng bit: ^
- Phép đảo bit: ~
- Phép dịch trái (nhân 2): <<
- Phép dịch phải (chia 2): >>

○ Ví dụ

- $3 \& 5 = 1$
- $a \ll n$ $/* a \cdot (2^n) */$
- $a \gg n$ $/* a / (2^n) */$

Giới thiệu chung

- Khái niệm hàm
 - Là đoạn chương trình viết ra một lần, được sử dụng nhiều lần
 - Mỗi lần sử dụng chỉ cần gọi tên hàm và cung cấp các tham số
- Cấu trúc chương trình

```
#include <...> /* Gọi các tệp tiêu đề trong chương trình */
#define ... /* Khai báo hằng số */
typedef /* Định nghĩa kiểu dữ liệu */
/* Nguyên mẫu các hàm: khai báo tên hàm và các tham số */
/* Khai báo các biến toàn cục */

main()
{
    /* Khai báo biến */
    /* Các câu lệnh */
}

/* Định nghĩa các hàm */
```

18-Jan-13

21

Giới thiệu chung

- Các khai báo
 - **#include**: dùng để gọi tệp tiêu đề
 - Khai báo biến: muốn sử dụng biến thì phải khai báo trước
 - Cú pháp: **kiểu_dữ_liệu danh_sách_các_biến;**
 - Ví dụ
 - `int x, y;`
 - `float a = 10.5, b; /* khai báo và khởi gán */`
 - `int a, b, c = 1;`

18-Jan-13

22

[Giới thiệu chung]

■ Các khai báo

- Khai báo hằng
 - Có hai cách để khai báo hằng, hoặc sử dụng **#define** hoặc sử dụng từ khóa **const**
- ```
#define tên_hằng giá_trị_hằng
```
- ```
const kiểu_dữ_liệu tên_hằng = giá_trị_hằng;
```
- Ví dụ
- ```
#define PI 3.14
const float PI = 3.14;
```

# [ Giới thiệu chung ]

## ■ Phép gán

- Gán giá trị cho một biến
- Cú pháp: **tên\_biến = biểu\_thức;**
- Ví dụ
  - `x = 0;`
  - `y = z + 1;`
- Phép gán kép
  - `x = y = z = 1;`
  - `x = y + (z = 2);`

## [ Giới thiệu chung ]

- Phép tăng 1 (++), giảm 1 (--)
  - Ngôn ngữ C cung cấp hai phép toán tăng 1 và giảm 1
  - Ví dụ
    - $x = x + 1;$  sẽ được viết thành:  $++x;$  hoặc  $x++;$
    - $y = y - 1;$  sẽ được viết thành:  $--y;$  hoặc  $y--;$
  - Sự khác nhau giữa khi toán tử ++ hoặc -- đứng trước hoặc sau biến là thể hiện trong phép gán: *biến = biểu\_thức*
    - Nếu toán tử ++x (--x) xuất hiện trong *biểu\_thức* thì x sẽ được tăng (giảm) 1 trước khi thực hiện phép gán
    - Nếu toán tử x++ (x--) xuất hiện trong *biểu\_thức* thì thực hiện phép gán trước khi x được tăng (giảm) 1
  - Ví dụ
    - $a = 5; b = ++a;$                       kết quả ?
    - $a = 5; b = a++;$                       kết quả ?

## [ Giới thiệu chung ]

- Tóm lại
  - Các từ khóa, tên
  - Các kiểu dữ liệu chuẩn
  - Các phép toán
  - Cấu trúc chung một chương trình C
  - Các khai báo
  - Phép gán
  - Phép tăng 1, giảm 1

# [ Nội dung ]

- Giới thiệu chung
- **Lệnh nhập/xuất**
- Lệnh điều kiện
- Lệnh vòng lặp
- Hàm
- Kiểu mảng
- Xâu kí tự
- Kiểu cấu trúc (struct) và kiểu hợp (union)
- Làm việc với tệp

18-Jan-13

27

# [ Lệnh nhập/xuất ]

- Lệnh xuất / hiển thị **printf**

- Ví dụ


```
#include <stdio.h>
void main()
{
 printf("Chào các bạn.\n");
}
```

- Cú pháp

**printf**(chuỗi\_điều\_khiển [, danh\_sách\_các\_tham\_số]);  
Chuỗi điều khiển dùng để định dạng dữ liệu cần hiển thị

- Ví dụ

```
printf("a = %f\n", a);
```



18-Jan-13

28

# [ Lệnh nhập/xuất ]

- Chuỗi điều khiển bao gồm 3 loại kí tự
  - Các kí tự điều khiển
    - \n sang dòng mới
    - \f sang trang mới
    - \b xóa kí tự bên trái
    - \t dấu tab
  - Các kí tự để đưa ra màn hình
  - Các kí tự định dạng và khuôn in
    - Các kí tự định dạng theo sau kí tự %
    - Ví dụ
      - %f
      - %d

18-Jan-13

29

# [ Lệnh nhập/xuất ]

- Các kí tự định dạng thường dùng

| Kí tự định dạng | Ý nghĩa                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------|
| <b>c</b>        | In ra một kí tự kiểu <b>char</b>                                                                     |
| <b>d</b>        | In ra số nguyên kiểu <b>int</b>                                                                      |
| <b>u</b>        | In ra số nguyên không dấu kiểu <b>unsigned int</b>                                                   |
| <b>ld</b>       | In ra số nguyên kiểu <b>long</b>                                                                     |
| <b>lu</b>       | In ra số nguyên kiểu <b>unsigned long</b>                                                            |
| <b>f</b>        | In ra số thực dạng m...m.n...n với phần thập phân có 6 chữ số, áp dụng cho kiểu <b>float, double</b> |
| <b>s</b>        | In ra xâu kí tự                                                                                      |
| <b>x</b>        | In ra số nguyên dưới dạng cơ số 16 (hexa)                                                            |
| <b>o</b>        | In ra số nguyên dưới dạng cơ số 8                                                                    |
| <b>e, E</b>     | In ra số thực dạng khoa học m...m.E[+ hoặc -]xx, áp dụng cho kiểu <b>float, double</b>               |
| <b>g, G</b>     | Dùng %e hoặc %f tùy thuộc loại nào ngắn hơn                                                          |

18-Jan-13

30

# [ Lệnh nhập/xuất ]

## ■ Ví dụ

**printf**("%%c và %%c có mã ASCII tương ứng là %d và %d\n", 'a', 'A', 'a', 'A');

Kết quả: a và A có mã ASCII tương ứng là 97 và 65

**printf**("%%f", x); /\* phần thập phân được hiển thị ngầm định là 6 chữ số \*/

x = 4.2

kết quả: 4.200000

X = 4.2345678

kết quả: 4.234568 /\*làm tròn\*/

**printf**("Ví dụ \nxa\b kí\b tự\b trái\b\n");

Kết quả:

Ví dụ

xo k t tra

# [ Lệnh nhập/xuất ]

## ■ Khuôn in

- Quy định cách thức in ra dữ liệu và chỉ rõ số chỗ dữ liệu sẽ chiếm, canh lề trái hay phải

- Khuôn in có dạng: **%m** hay **%m.n**

- Đối với số nguyên, mẫu ghi là **%md**

- m là số nguyên chỉ ra số vị trí mà số nguyên chiếm

- Ví dụ: **printf**("x = %4d", x);

- Kết quả: nếu x = 12 in ra ^^12  
nếu x = 12345 in ra 12345

- Đối với số thực, mẫu ghi là **%m.nf**

- m là tổng số chữ viết ra, n là số chữ số phần thập phân

- Ví dụ: **printf**("x = %4.2f", x);

- Kết quả: nếu x = 1.234 in ra ^1.23



# [ Lệnh nhập/xuất ]

## ■ In kí tự đặc biệt

- `\'` In ra dấu '
- `\"` In ra dấu "
- `\\` In ra dấu \

## ■ Các lệnh xuất dữ liệu khác

- `puts(chuỗi_kí_tự)`: hiển thị chuỗi kí tự
  - Ví dụ: `puts("Chào bạn");`
- `putchar(kí_tự)`: hiển thị một kí tự
  - Ví dụ: `putchar('a');`

# [ Lệnh nhập/xuất ]

## ■ Lệnh nhập dữ liệu **scanf**

### ■ Ví dụ

```
#include <stdio.h>
void main()
{
 float r, dien_tich;
 printf("Nhập vào bán kính: ");
 scanf("%f", &r);
 dien_tich = 3.14 * r * r;
 printf("Diện tích là: %f\n", dien_tich);
 getch();
}
```

### ■ Cách sử dụng lệnh **scanf** gần giống với lệnh **printf**

# [ Lệnh nhập/xuất ]

- Lệnh scanf
  - Cú pháp  
**scanf**(chuỗi\_điều\_khiển [, danh\_sách\_tham\_số]);
    - chuỗi\_điều\_khiển cho phép định dạng dữ liệu nhập vào
    - danh\_sách\_tham\_số là địa chỉ các biến cần nhập dữ liệu
  - Để lấy địa chỉ một biến, sử dụng toán tử &

# [ Lệnh nhập/xuất ]

- Lệnh scanf

| Kí tự định dạng | Ý nghĩa                                                                                                        |
|-----------------|----------------------------------------------------------------------------------------------------------------|
| <b>c</b>        | Nhập vào một kí tự kiểu <b>char</b>                                                                            |
| <b>d</b>        | Nhập vào số nguyên kiểu <b>int</b>                                                                             |
| <b>u</b>        | Nhập vào số nguyên không dấu kiểu <b>unsigned int</b>                                                          |
| <b>ld</b>       | Nhập vào số nguyên kiểu <b>long</b>                                                                            |
| <b>lu</b>       | Nhập vào số nguyên kiểu <b>unsigned long</b>                                                                   |
| <b>f</b>        | Nhập vào số thực dạng m...m.n..n với phần thập phân có 6 chữ số, áp dụng cho kiểu <b>float</b> , <b>double</b> |
| <b>s</b>        | Nhập vào xâu kí tự, không chứa dấu cách (space)                                                                |
| <b>x</b>        | Nhập vào số nguyên dưới dạng cơ số 16 (hexa)                                                                   |
| <b>o</b>        | Nhập vào nguyên dưới dạng cơ số 8                                                                              |

## [ Lệnh nhập/xuất ]

- Một số lệnh nhập dữ liệu khác
  - `gets(char *str)`: nhận chuỗi kí tự vào từ bàn phím cho đến khi gặp “\n”
  - `getchar()`: nhận kí tự nhập vào
    - Ví dụ: `ch = getchar();`
  - `getch()`: nhận kí tự nhập vào và không cho hiển thị kí tự đó trên màn hình
  - `getche()`: nhận kí tự nhập vào và cho hiển thị kí tự đó trên màn hình

## [ Lệnh nhập/xuất ]

- Một số lệnh khác liên quan đến xuất/nhập
  - `fflush()`: xóa vùng đệm bàn phím
  - `kbhit()`: kiểm tra bộ đệm bàn phím, bộ đệm rỗng trả về giá trị 0, ngược lại trả về giá trị khác 0
  - `clrscr()`: xóa màn hình
  - `gotoxy(int x, int y)`: di chuyển con trỏ màn hình đến vị trí cột x (1→80), và dòng y (1→25)

# [ Lệnh nhập/xuất ]

## ■ Bài tập

- Nhập vào 3 số thực, tính tổng của chúng và in ra màn hình
- Tính diện tích tam giác khi biết chiều cao và cạnh đáy

# [ Tóm lại ]

## ■ Lệnh nhập dữ liệu

- printf
- putchar
- puts

## ■ Lệnh xuất dữ liệu

- scanf
- getchar
- gets

## ■ Một số lệnh liên quan khác

# [ Nội dung ]

- Giới thiệu chung
- Lệnh nhập/xuất
- **Lệnh điều kiện**
- Lệnh vòng lặp
- Hàm
- Kiểu mảng
- Xâu kí tự
- Kiểu cấu trúc (struct) và kiểu hợp (union)
- Làm việc với tệp

18-Jan-13

41

# [ Lệnh điều kiện ]

- Lệnh
  - Một câu lệnh nhằm thực hiện một công việc nào đó
  - Câu lệnh kết thúc bởi dấu “;”
  - Ví dụ
    - `printf("một câu lệnh\n");`
    - `i++;`
- Khối lệnh
  - Là dãy các lệnh được đặt giữa cặp ngoặc nhọn “{” và “}”
  - Khối lệnh thường được sử dụng khi muốn chúng thực hiện dưới một điều kiện nào đó

```
{
/* các lệnh */
}
```

18-Jan-13

42

# [ Lệnh điều kiện ]

## ■ Lệnh if

- Thực hiện một trong hai khối lệnh tùy thuộc vào giá trị của biểu thức điều kiện
- Lệnh **if** có hai dạng: dạng đầy đủ **if ... else** và dạng chỉ có **if**
- Cú pháp  
    **if** (biểu thức điều kiện) (*dạng 1*)  
        khối lệnh 1;  
    **else**  
        khối lệnh 2;  
Hoặc  
    **if** (biểu thức điều kiện) (*dạng 2*)  
        khối lệnh 1;

# [ Lệnh điều kiện ]

## ■ Lệnh if

- Ý nghĩa
  - Dạng 1: nếu biểu thức điều kiện có giá trị đúng (có giá trị khác không), khối lệnh 1 sẽ được thực hiện; nếu điều kiện là sai (có giá trị bằng không) thì khối lệnh 2 sẽ được thực hiện
  - Dạng 2: nếu biểu thức điều kiện là đúng (có giá trị khác không), khối lệnh 1 sẽ được thực hiện; nếu điều kiện là sai (có giá trị bằng không) thì thực hiện câu lệnh đứng sau khối lệnh 1
- Mô tả hai dạng của lệnh if bằng sơ đồ khối
  - ???

# [ Lệnh điều kiện ]

## ■ Lệnh if

- Ví dụ 1: tính giá trị nhỏ nhất của hai số

```
#include <stdio.h>
main()
{
 int a, b, min;
 printf("Nhập vào hai số nguyên a và b.\n");
 printf("a = ");
 scanf("%d", &a);
 printf("b = ");
 scanf("%d", &b);
 if (a < b)
 min = a;
 else
 min = b;
 printf("min = %d\n", min);
}
```

18-Jan-13

45

# [ Lệnh điều kiện ]

## ■ Lệnh if

- Ví dụ 2: viết lại chương trình tìm giá trị nhỏ nhất của 2 số sử dụng dạng **if** không có **else**
- Ví dụ 3: trường hợp sử dụng khối lệnh

```
if (a > b)
{
 max = a;
 min = b;
}
else
{
 max = b;
 min = a;
}
```

18-Jan-13

46

# [ Lệnh điều kiện ]

## ■ Lệnh if

- Có thể sử dụng các toán tử “&&” và “||” để xây dựng các biểu thức điều kiện phức tạp hơn
- Chẳng hạn
  - if ((đk1 && đk2) || đk3)
- Ví dụ: viết biểu thức điều kiện kiểm tra 3 số thực là 3 cạnh tam giác

# [ Lệnh điều kiện ]

## ■ Một số lưu ý khi sử dụng lệnh if

- Biểu thức điều kiện phải luôn đặt trong hai dấu “(“ và “)”
- Biểu thức điều kiện là đúng, nếu nó có giá trị khác 0 và là sai nếu nó có giá trị bằng 0
- Biểu thức điều kiện có thể là số nguyên hoặc thực
- Nếu sau **if** hoặc **else** là một dãy các câu lệnh, thì các câu lệnh này phải được đặt trong cặp dấu ngoặc “{“ và “}”



# [ Lệnh điều kiện ]

## ■ Sử dụng lệnh **if** lồng nhau

- Ví dụ: chương trình tính nghiệm phương trình  $ax+b=0$

```
#include <stdio.h>
main()
{
 float a, b, x;
 printf("Nhap vao a va b:"); scanf("%f%f", &a, &b);
 if (a == 0){
 if (b == 0)
 printf("Phuong trinh co vo so nghiem\n");
 else
 printf("Phuong trinh vo nghiem\n");
 }
 else{
 x = -b/a;
 printf("Phuong trinh co nghiem x = %f\n", x);
 }
}
```

18-Jan-13

49

# [ Lệnh điều kiện ]

## ■ Sử dụng lệnh **if** lồng nhau

- Khi sử dụng các lệnh **if** lồng nhau, nên sử dụng các dấu đóng mở ngoặc “{ }” để tránh gây ra sự hiểu nhầm **if** nào tương ứng với **else** nào
- Ví dụ

```
if (a != 0)
 if (a > b)
 y = b/a;
 else
 y = -b/a;
```



```
if (a != 0)
{
 if (a > b)
 y = b/a;
 else
 y = -b/a;
}
```

18-Jan-13

50

# [ Lệnh điều kiện ]

## ■ Sử dụng **else if**

- Khi muốn sử dụng một trong n quyết định, sử dụng dạng lệnh **if** như sau

```
if (điều kiện 1)
 khối lệnh 1;
else if (điều kiện 2)
 khối lệnh 2;
...
else if (biểu thức n-1)
 khối lệnh n-1;
else
 khối lệnh n;
```

# [ Lệnh điều kiện ]

## ■ Sử dụng **else if**

- Ví dụ: Chương trình xếp loại kết quả học tập của một sinh viên

```
#include <stdio.h>
main()
{
 float diem;
 printf("Nhap diem vao"); scanf("%f", &diem);
 if (diem < 5)
 printf("Xep loai: kem");
 else if (diem < 7)
 printf("Xep loai: trung binh");
 else if (diem < 8)
 printf("Xep loai: kha");
 else if (diem < 9)
 printf("Xep loai: gioi");
 else
 printf("Xep loai: xuat sac");
}
```

## [ Lệnh điều kiện ]

- Bài tập
  - Viết chương trình giải một phương trình bậc 2

## [ Toán tử “?:” ]

- Có thể sử dụng toán tử “?:” thay cho lệnh **if**
- Cú pháp  
**(điều kiện) ? lệnh 1 : lệnh 2;**  
nếu điều kiện là đúng lệnh 1 sẽ được thực hiện, nếu không lệnh 2 sẽ được thực hiện
- Ví dụ  
 $(a > b) ? \max = a : \max = b;$ 
  - Hoặc  
 $\max = (a > b) ? a : b;$

## [ Lệnh switch ... case ]

- Lệnh **if** chỉ cho phép chọn một trong hai phương án
- Lệnh **switch ... case** cho phép chọn một trong nhiều phương án khác nhau
- Cú pháp  
**switch** (biểu thức nguyên)  
{  
  **case**  $n_1$ :  
    Các câu lệnh;  
  **case**  $n_2$ :  
    Các câu lệnh;  
  ...  
  **case**  $n_k$ :  
    Các câu lệnh;  
  [default: Các câu lệnh;]  
}

18-Jan-13

55

## [ Lệnh switch ... case ]

- Ý nghĩa câu lệnh
  - Nếu biểu thức nguyên có giá trị bằng nhãn  $n_i$  thì máy sẽ nhảy đến thực hiện các lệnh của nhãn đó, nếu không thì máy sẽ nhảy đến thực hiện các lệnh trong thành phần tùy chọn **default**
  - Máy sẽ ra khỏi toán tử **switch** khi nó gặp câu lệnh **break**, **return** hoặc nó gặp dấu “}” của câu lệnh **switch**
  - Chú ý, khi máy nhảy tới nhãn  $n_i$ , nếu kết thúc dãy lệnh trong nhãn này không có câu lệnh **break** hoặc **return** thì máy sẽ tiếp tục thực hiện các lệnh trong nhãn  $n_{i+1}$
  - Thường cuối mỗi dãy lệnh của một nhãn có một lệnh **break**

18-Jan-13

56

# [ Lệnh switch ... case ]

## ■ Ví dụ

```
#include <stdio.h>
main()
{
 int n;
 printf(" Nhập vào một số nguyên từ 0 đến 2: ");
 scanf("%d", &n);
 switch(n)
 {
 case 0: printf("Số không\n");
 break;
 case 1: printf("Số một\n");
 break;
 case 2: printf("Số hai\n");
 break;
 default: printf("Không đúng\n");
 }
 printf("Kết thúc\n");
}
```

18-Jan-13

57

# [ Lệnh switch ... case ]

## ■ Ví dụ: thiếu lệnh **break**

```
#include <stdio.h>
main()
{
 int n;
 printf(" Nhập vào một số nguyên từ 0 đến 2: ");
 scanf("%d", &n);

 switch(n)
 {
 case 0: printf("Số không\n");
 case 1: printf("Số một\n");
 case 2: printf("Số hai\n");
 }
 printf("Kết thúc\n");
}
```

18-Jan-13

58

## [ Lệnh switch ... case ]

- Bài tập
  - Viết chương trình nhập vào hai số thực  $a$ ,  $b$  và một ký hiệu  $op$ ,  $op$  là một trong các ký hiệu  $+$ ,  $-$ ,  $*$ ,  $/$ . Hãy xuất kết quả của biểu thức  $a \text{ } op \text{ } b$  ra màn hình.

## [ Nội dung ]

- Giới thiệu chung
- Lệnh nhập/xuất
- Lệnh điều kiện
- Lệnh vòng lặp
- Hàm
- Kiểu mảng
- Xâu kí tự
- Kiểu cấu trúc (struct) và kiểu hợp (union)
- Làm việc với tệp

# [ Lệnh vòng lặp ]

- Thực hiện một công việc nào đó được lặp đi lặp lại nhiều lần
- Ví dụ
  - In ra màn hình các số từ 1 đến 10, mỗi số trên một dòng
    - Giải pháp đơn giản
      - `printf("1\n");`
      - `printf("2\n");`
      - ...
      - `printf("10\n");`
    - Giải pháp tổng quát
      - Dùng vòng lặp

# [ Lệnh vòng lặp ]

- Các lệnh vòng lặp
  - Lệnh **for**
  - Lệnh **while**
  - Lệnh **do ... while**

# [ Lệnh vòng lặp ]

- Lệnh lặp **for**
  - Cú pháp  
**for** ([biểu thức 1]; [biểu thức 2]; [biểu thức 3])  
khối lệnh;
    - Các thành phần trong ngoặc "[" và "]" là tùy chọn, không bắt buộc
    - Các dấu ";" và cặp ngoặc "(" và ")" là bắt buộc phải có
  - Ý nghĩa câu lệnh: lệnh **for** hoạt động theo các bước
    1. Tính biểu thức 1.
    2. Tính biểu thức 2. Nếu biểu thức 2 có giá trị 0 (sai), máy sẽ ra khỏi **for** và chuyển tới câu lệnh sau thân **for**. Nếu biểu thức 2 có giá trị khác 0 (đúng), máy thực hiện các câu lệnh trong thân **for**, sau đó chuyển tới bước 3.
    3. Tính biểu thức 3, sau đó quay trở lại bước 2 để bắt đầu các bước lặp mới.

# [ Lệnh vòng lặp ]

- Ví dụ

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

- Có thể viết cách khác đoạn chương trình trên không ?
- Cách biểu diễn bằng sơ đồ khối lệnh **for** như thế nào ?



# [ Lệnh vòng lặp ]

- Ví dụ: tính tổng n số tự nhiên đầu tiên

```
#include <stdio.h>
main()
{
 int n, s, i;
 /*nhập n*/
 printf("n = ");
 scanf("%d", &n);
 s = 0;
 for (i = 1; i <= n; i++)
 s = s + i;
 printf("Tổng của %d số tự nhiên đầu tiên là: %d\n", n, s);
}
```

# [ Lệnh vòng lặp ]

- Nhận xét
  - Biểu thức 1 chỉ được tính một lần
  - Biểu thức 2, biểu thức 3 và khối lệnh trong thân lệnh **for** được lặp đi lặp lại nhiều lần
  - Khi biểu thức 2 vắng mặt thì nó được xem là đúng
    - Để thoát khỏi lệnh for trong trường hợp này phải dùng lệnh **break** hoặc **return**
  - Có thể sử dụng các lệnh **for** lồng nhau
- Câu lệnh sau làm gì ?
  - `for(;;){}`

# [ Lệnh vòng lặp ]

## ■ Lệnh lặp **while**

### ○ Cú pháp

**while** (biểu thức)  
khối lệnh;

### ○ Ý nghĩa

- Trong khi *biểu thức* có giá trị đúng, tức khác 0, thì còn phải thực hiện khối lệnh. Việc lặp dừng lại khi biểu thức có giá trị sai (bằng 0).

- Lệnh **while** kiểm tra điều kiện trước khi thực hiện khối lệnh

### ○ Hãy vẽ sơ đồ khối biểu diễn lệnh **while**

# [ Lệnh vòng lặp ]

## ■ Ví dụ

- Viết lại chương trình tính tổng n số tự nhiên đầu tiên sử dụng lệnh **while**

```
#include <stdio.h>
main()
{
 int n, s, i;
 printf("n = "); scanf("%d", &n);
 s = 0;
 i = 1;
 while (i <= n) {
 s = s + i;
 i++;
 }
 printf("Tổng của %d số tự nhiên đầu tiên là: %d\n", n, s);
}
```

# [ Lệnh vòng lặp ]

- Nhận xét
  - Biểu thức điều kiện luôn được đặt trong cặp dấu "(" và ")"
  - Biểu thức điều kiện sẽ được tính toán đầu tiên nên phải có giá trị xác định
- Câu lệnh sau làm gì ?
  - `while(0) printf("nothing\n");`
- Hãy chuyển lệnh **for** dạng tổng quát thành lệnh **while**

# [ Lệnh vòng lặp ]

- Lệnh lặp **do ... while**
  - Cú pháp
    - do**  
khối lệnh;
    - while** (biểu thức);
  - Ý nghĩa
    - Thực hiện khối lệnh trong khi biểu thức có giá trị đúng, tức là khác 0
    - Thực hiện khối lệnh trước khi kiểm tra biểu thức điều kiện
    - Khối lệnh được thực hiện ít nhất 1 lần
  - Hãy vẽ sơ đồ khối biểu diễn lệnh **do ... while**

# [ Lệnh vòng lặp ]

## ■ Ví dụ

- Viết chương trình nhập vào một số lớn hơn 10

```
#include <stdio.h>
main()
{
 int n;
 do
 {
 printf(" Hãy cho một số > 10 :");
 scanf("%d", &n);
 printf(" Bạn đã đọc một số %d\n", n);
 }while (n <= 10);

 printf(" Đúng số lớn hơn 10 rồi.");
}
```

18-Jan-13

71

# [ Lệnh vòng lặp ]

## ■ Ví dụ (tiếp)

- Nếu dùng lệnh while

```
#include <stdio.h>
main()
{
 int n;

 printf(" Hãy cho một số > 10 :");
 scanf("%d", &n);
 printf(" Bạn đã đọc một số %d\n", n);
 while (n <= 10)
 {
 printf(" Hãy cho một số > 10 :");
 scanf("%d", &n);
 printf(" Bạn đã đọc một số %d\n", n);
 }
 printf(" Đúng số lớn hơn 10 rồi.");
}
```

18-Jan-13

72

## [ Lệnh vòng lặp ]

- Hãy chuyển lệnh **do ... while** thành lệnh **while** tương ứng
- Hãy viết lại chương trình tính tổng n số tự nhiên đầu tiên dùng **do ... while**
- Viết câu lệnh nhập vào các kí tự và dừng lại khi kí tự nhập vào là '@'

## [ Lệnh break ]

- Lệnh **break** thường được sử dụng kết hợp lệnh lặp
- Lệnh **break** dùng để thoát khỏi vòng lặp
- Nếu có nhiều lệnh lặp lồng nhau thì lệnh **break** chỉ thoát vòng lặp trực tiếp chứa nó
- Lệnh **break** cũng dùng để thoát khỏi lệnh **switch ... case**

## Lệnh break

- Ví dụ

```
#include <stdio.h>
main()
{
 int i;
 for (i=1; i <=5; i++)
 {
 printf("Bắt đầu vòng %d\n", i);
 printf("Chào bạn\n");
 if (i==3) break;
 printf("Kết thúc vòng %d\n", i);
 }
 printf("Hết vòng lặp.");
}
```

- Kết quả ?
- Hãy vẽ sơ đồ khối mô tả chương trình trên

## Lệnh continue

- Lệnh **continue** dùng để quay trở lại từ đầu để thực hiện lần lặp mới mà không cần thực hiện phần còn lại
- Ví dụ

```
#include <stdio.h>
main()
{
 int i;
 for (i=1; i <= 5; i++)
 {
 printf(" Bắt đầu %d\n", i);
 if (i<4) continue;
 printf(" Chào bạn\n");
 }
}
```

## [ Lệnh continue ]

- Đoạn chương trình sau làm gì ?

```
for (i = 1; i <= 4; i++)
 for (j = 1; j <= 10; j++)
 {
 printf("%d", j);
 if (j != 10) continue;
 printf("\n");
 }
```

## [ Tóm lại ]

- Lệnh điều kiện
  - Lệnh **if**
  - Toán tử "?:"
  - Lệnh **switch ... case**
- Lệnh lặp
  - Lệnh **for**
  - Lệnh **while**
  - Lệnh **do ... while**
- Các lệnh liên quan
  - Lệnh **break**
  - Lệnh **continue**