



TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

TIN HỌC ĐẠI CƯƠNG

Bài 8. Mảng và cấu trúc dữ liệu

Nội dung

1. Mảng
2. Con trỏ
3. Cấu trúc dữ liệu

Nội dung

1. Mảng

- 1.1. Khái niệm mảng
- 1.2. Khai báo và sử dụng mảng
- 1.3. Các thao tác cơ bản trên mảng
- 1.4. Tìm kiếm trên mảng
- 1.5. Sắp xếp trên mảng

2. Con trỏ

3. Xâu kí tự

3

1.1. Khái niệm mảng

- Tập hợp hữu hạn các phần tử cùng kiểu, lưu trữ kế tiếp nhau trong bộ nhớ
- Các phần tử trong mảng có cùng tên (là tên mảng) nhưng phân biệt với nhau ở chỉ số cho biết vị trí của nó trong mảng
- Ví dụ:
 - Bảng điểm của sinh viên
 - Vector
 - Ma trận

4

1.2. Khai báo và sử dụng mảng

- Khai báo mảng (một chiều)
KieuDuLieu tenMang [kích_thước];
- Trong đó
 - KieuDuLieu: kiểu dữ liệu của các phần tử trong mảng
 - tenMang: tên của mảng
 - kích_thước: số phần tử trong mảng
- Ví dụ

```
int mangNguyen[10]; // khai báo mảng 10 phần tử có kiểu dữ liệu int
```

5

1.2. Khai báo và sử dụng mảng

- Cấp phát bộ nhớ
 - Các phần tử trong mảng được cấp phát các ô nhớ kế tiếp nhau trong bộ nhớ
 - Biến mảng lưu trữ địa chỉ ô nhớ đầu tiên trong vùng nhớ được cấp phát
- Ngôn ngữ C đánh chỉ số các phần tử trong mảng bắt đầu từ 0
 - Phần tử thứ i trong *mangNguyen* được xác định bởi *mangNguyen [i-1]*



6

1.2. Khai báo và sử dụng mảng

- Khai báo mảng nhiều chiều
KieuDuLieu tenMang[size₁][size₂]...[size_d];
Trong đó
 - size_i là kích thước chiều thứ i của mảng
- Mảng một chiều và mảng nhiều chiều
 - Mỗi phần tử của mảng cũng là một mảng
=> mảng nhiều chiều
- Ví dụ
 - int a[6][5] ; //mảng 2 chiều
 - int b[3][4][5]; // mảng 3 chiều

7

1.2. Khai báo và sử dụng mảng

- Sử dụng mảng
 - Truy cập vào phần tử thông qua tên mảng và chỉ số của phần tử trong mảng
tenMang[chỉ_số_phần_tử]
 - Chú ý: chỉ số bắt đầu từ 0
- Ví dụ
 - int a[4];
 - phần tử đầu tiên (thứ nhất) của mảng: a[0]
 - phần tử cuối cùng (thứ tư) của mảng: a[3]
 - a[i]: là phần tử thứ i+1 của a

8

1.2. Khai báo và sử dụng mảng

- Ví dụ (tiếp)
 - `int b[3][4];`
 - phần tử đầu tiên của mảng: `b[0]` là một mảng một chiều
 - phần tử đầu tiên của mảng `b[0]`: `b[0][0]`
 - `b[i][j]`: là phần tử thứ `j+1` của `b[i]`, `b[i]` là phần tử thứ `i+1` của `b`

9

Khai báo hằng số có kiểu mảng

- Sử dụng **#define**

```
#define TEN_MANG {Giá_trị_1, Giá_trị_2, ... Giá_trị_n}
```

- Lưu ý: không thể truy cập vào phần tử của mảng

- Sử dụng từ khóa **const**

```
const KiểuDuLieu TEN_MANG[Kích_thước] = {Giá_trị_1, Giá_trị_2, ..., Giá_trị_n};
```

Lưu ý:

- Nếu không khai báo `Kích_thước` thì kích thước của mảng là số lượng giá trị sử dụng khi khai báo
- Nếu số lượng giá trị nhỏ hơn `Kích_thước` mảng, các phần tử không được gán sẽ nhận giá trị 0

10

Khai báo hằng số có kiểu mảng – Ví dụ

```
const int CONST_ARR1[5] = {1,2,3,4,5}
```

CONST_ARR1:

1	2	3	4	5
---	---	---	---	---

```
const int CONST_ARR2[ ] = {1,2,3,4}
```

CONST_ARR2:

1	2	3	4
---	---	---	---

```
const int CONST_ARR3[5] = {1,2,3}
```

CONST_ARR3:

1	2	3	0	0
---	---	---	---	---

11

1.3. Các thao tác cơ bản trên mảng

a. Nhập dữ liệu cho mảng

- Khởi tạo giá trị cho mảng ngay khi khai báo
 - `int a[4] = {1,4,6,2};`
 - `int b[2][3] = { {1,2,3}, {4,5,6} };`
 - Số lượng giá trị khởi tạo không được lớn hơn số lượng phần tử trong mảng
 - Nếu số lượng này nhỏ hơn, các phần tử còn lại được khởi tạo giá trị 0

12

1.3. Các thao tác cơ bản trên mảng

a. Nhập dữ liệu cho mảng

- Có thể xác định kích thước mảng thông qua số giá trị khởi tạo nếu để trống kích thước mảng
- `int array1 [8] = {2, 4, 6, 8, 10, 12, 14, 16};`
- `int array2 [] = {2, 4, 6, 8, 10, 12, 14, 16};`

13

1.3. Các thao tác cơ bản trên mảng

a. Nhập dữ liệu cho mảng

- Nhập dữ liệu từ bàn phím bằng hàm `scanf`
 - `int a[10];`
 - Nhập dữ liệu cho `a[1]`: `scanf("%d", &a[1]);`
 - Nhập dữ liệu cho toàn bộ phần tử của mảng `a`
=> Sử dụng vòng lặp `for`
- Lưu ý
 - Tên mảng là một hằng (hằng con trỏ) do đó không thể thực hiện phép toán với tên mảng như phép gán sau khi đã khai báo

14

1.3. Các thao tác cơ bản trên mảng

```
#include <stdio.h>
#define MONTHS 12
int main(){
    int rainfall[MONTHS], i;
    for ( i=0; i < MONTHS; i++ ){
        printf("Nhap vao phan tu thu
                %d: ", i+1);
        scanf("%d", &rainfall[i] );
    }
    return 0;
}
```

15

1.3. Các thao tác cơ bản trên mảng

a. Nhập dữ liệu cho mảng

- Lưu ý

- Nếu số phần tử của mảng được nhập từ bàn phím và chỉ biết trước số phần tử tối đa tối đa => khai báo mảng với kích thước tối đa và sử dụng biến lưu số phần tử thực sự của mảng.
- Ví dụ: Khai báo mảng số nguyên a có tối đa 100 phần tử. Nhập từ bàn phím số phần tử trong mảng và giá trị các phần tử đó....

16

1.3. Các thao tác cơ bản trên mảng

```
#include<stdio.h>
#include<conio.h>
int main(){
    int a[100];
    int n, i;
    do{
        printf("\n Cho biet so phan tu cua mang: ");
        scanf("%d",&n);
    }while (n>100||n<=0);
    for(i = 0; i < n; i++){
        printf("a[%d] = ", i);
        scanf("%d",&a[i]);
    }
    getch();
    return 0;
}
```

17

1.3. Các thao tác cơ bản trên mảng

b. Hiển thị dữ liệu trong mảng

- Dùng hàm printf()
- Để hiển thị tất cả các phần tử: dùng vòng for
- Ví dụ
 - Hiển thị một phần tử bất kì
 - Hiển thị tất cả các phần tử, mỗi phần tử trên một dòng
 - Hiển thị tất cả các phần tử trên một dòng, cách nhau 2 vị trí
 - Hiển thị từng k phần tử trên một dòng

18

1.3. Các thao tác cơ bản trên mảng

```
#include <stdio.h>
#define MONTHS 12
int main(){
    int rainfall[MONTHS], i;
    for ( i=0; i < MONTHS; i++ ){
        printf("Nhap vao phan tu thu
                %d: ", i+1);
        scanf("%d", &rainfall[i] );
    }
    for ( i=0; i < MONTHS; i++ )
        printf( "%2d " , rainfall[i]);
    printf("\n");
    return 0;
}
```

19

1.3. Các thao tác cơ bản trên mảng

c. Tìm giá trị lớn nhất, nhỏ nhất

- Tìm giá trị lớn nhất
 - Giả sử phần tử đó là phần tử đầu tiên
 - Lần lượt so sánh với các phần tử còn lại
 - Nếu lớn hơn hoặc bằng => so sánh tiếp
 - Nếu nhỏ hơn => coi phần tử này là phần tử lớn nhất và tiếp tục so sánh
 - Cách làm?
- Tìm giá trị nhỏ nhất: tương tự

20

1.4. Tìm kiếm trên mảng

- Bài toán
 - Cho mảng dữ liệu a và một giá trị k
 - Tìm các phần tử trong mảng a có giá trị bằng (giống) với k . Nếu có in ra vị trí (chỉ số) các phần tử này. Ngược lại thông báo không tìm thấy
- Cách làm
 - Duyệt toàn bộ các phần tử trong mảng
 - Nếu $a[i]$ bằng (giống) k thì lưu lại chỉ số i
 - Sử dụng một biến để xác định tìm thấy hay không tìm thấy

21

1.4. Tìm kiếm trên mảng

- Phân tích
 - Duyệt toàn bộ các phần tử
 - Vòng lặp for (while, do while)
 - Lưu lại i nếu $a[i]$ bằng (giống) k
 - Sử dụng mảng lưu chỉ số
 - Biến xác định tìm thấy hay không tìm thấy
 - Biến nhận giá trị 0 hoặc 1
 - Biến nhận giá trị 0 hoặc ≥ 1 (tìm thấy thì tăng giá trị)

22

1.4. Tìm kiếm trên mảng

```
#include <stdio.h>
#include <conio.h>
int main(){
    int a[100], chi_so[100];
    int n;//n la số phần tử trong mảng
    int i, k, kiem_tra;
    printf("Nhap vao so phan tu cua mang:");
    scanf("%d",&n);
    printf("Nhap vao giá trị tìm kiếm:");
    scanf("%d",&k);
```

23

1.4. Tìm kiếm trên mảng

```
    kiem_tra = 0;
    // Duyệt qua tất cả các phần tử
    for(i = 0;i<n;i++)
        if(a[i] == k)
        {
            chi_so[kiem_tra] = i;
            kiem_tra ++;
        }
```

24

1.4. Tìm kiếm trên mảng

```
if(kiem_tra > 0){  
    printf("Trong mang co %d phan tu co  
    gia tri bang %d",kiem_tra,k);  
    printf("\nChi so cua cac phan tula:");  
    for(i = 0;i < kiem_tra;i++)  
        printf("%3d",chi_so[i]);  
} else  
    printf("\n Trong mang khong co phan  
    tu nao co gia tri bang %d",k);  
getch();  
return 0;  
}
```

25

1.5. Sắp xếp mảng

- Bài toán

- Cho mảng a gồm n phần tử. Sắp xếp các phần tử của mảng a theo thứ tự tăng dần/giảm dần

23	78	45	8	32	56
----	----	----	---	----	----



8	23	32	45	78	56
---	----	----	----	----	----

26

1.5. Sắp xếp mảng

- Giải thuật sắp xếp
 - Sắp xếp thêm dần (insertion sort)
 - Sắp xếp lựa chọn (selection sort)
 - Sắp xếp nổi bọt (bubble sort)
 - Sắp xếp vun đống (heap sort)
 - Sắp xếp nhanh (quick sort)
 - Sắp xếp trộn (merge sort)
 -

27

Giải thuật sắp xếp lựa chọn

Ý tưởng

- Lần sắp xếp thứ 1:
 - Đoạn đã được sắp xếp: chưa có phần tử nào
 - Đoạn chưa được sắp xếp: có n phần tử từ a_0, a_1, \dots, a_{n-1}
 - So sánh a_0 với a_j ($1 \leq j \leq n-1$). Nếu $a_0 > a_j$ thì đổi chỗ
 - Sau lượt sắp xếp này a_0 đã đúng thứ tự
- Lần sắp xếp thứ 2:
 - Đoạn đã được sắp xếp: a_0
 - Đoạn chưa được sắp xếp: a_1, a_2, \dots, a_{n-1}
 - So sánh a_1 với a_j ($2 \leq j \leq n-1$). Nếu $a_1 > a_j$ thì đổi chỗ
 - Sau lượt sắp xếp này a_1 đã đúng thứ tự

28

Giải thuật sắp xếp lựa chọn

- Lần sắp xếp thứ i :
 - Đoạn đã được sắp xếp: a_0, a_1, \dots, a_{i-2}
 - Đoạn chưa được sắp xếp: $a_{i-1}, a_i, \dots, a_{n-1}$
 - So sánh a_{i-1} với a_j ($i \leq j \leq n-1$). Nếu $a_i > a_j$ thì đổi chỗ
 - Sau lượt sắp xếp này a_{i-1} đã đúng thứ tự
- Thuật toán dừng khi dãy chưa được sắp xếp chỉ có 1 phần tử

29

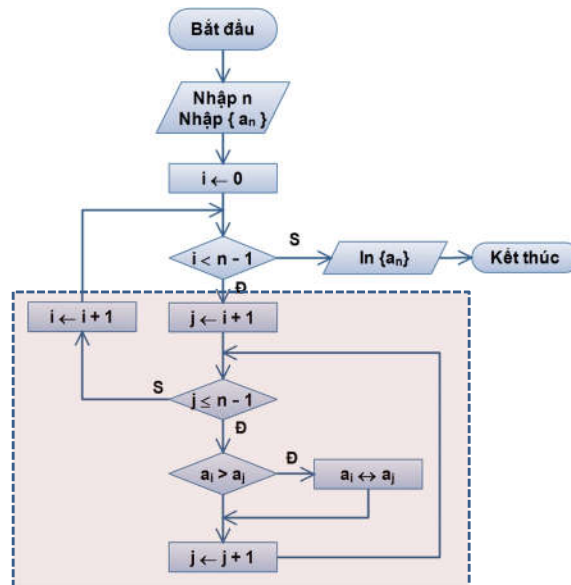
1.5. Sắp xếp mảng

- $A = \{ 12, 5, 3, 4 \}$;

	Lượt 1	Lượt 2	Lượt 3
12	3	3	3
5	12	4	4
3	5	12	5
4	4	5	12

30

Lưu đồ thuật toán sắp xếp lựa chọn



31

1.5. Sắp xếp mảng

```
//Khai bao cac bien
int a[100];
int i, j, tmp;

//Sap xep
for (i = 0; i < n-1; i++)
    for (j = i+1; j <n ; j++)
        if ( a[i] > a[j]){
            tmp = a[i];
            a[i]= a[j];
            a[j]= tmp;
        }
}
```

32

Nội dung

1. Mảng
2. Con trỏ
 - 2.1. Khái niệm, khai báo con trỏ
 - 2.2. Toán tử địa chỉ, toán tử nội dung
 - 2.3. Phép toán trên con trỏ
 - 2.4. Con trỏ và mảng
3. Xâu kí tự

33

2.1. Khái niệm, khai báo con trỏ

- Địa chỉ và giá trị của một biến
 - Bộ nhớ như một dãy các byte nhớ.
 - Các byte nhớ được xác định một cách duy nhất qua một *địa chỉ*.
 - Biến được lưu trong bộ nhớ.
 - Khi khai báo một biến
 - Chương trình dịch sẽ cấp phát cho biến đó một số ô nhớ liên tiếp đủ để chứa nội dung của biến. Ví dụ một biến số nguyên (int) được cấp phát 2 byte.
 - Địa chỉ của một biến chính là địa chỉ của byte đầu tiên trong số đó.

34

Địa chỉ và giá trị của một biến (tiếp)

- Một biến luôn có 2 đặc tính:
 - Địa chỉ của biến
 - Giá trị của biến

- Ví dụ

```
int i, j;  
i = 3;  
j = i + 1;
```

Biến	Địa chỉ	Giá trị
i	FFEC	3
j	FFEE	4

35

Khái niệm và khai báo con trỏ

- Con trỏ là một biến mà giá trị của nó là địa chỉ của một vùng nhớ
- Khai báo con trỏ:
`KieuDuLieu *tenBienConTro;`
- Một con trỏ chỉ có thể trỏ tới một đối tượng cùng kiểu.
- Ví dụ:

```
int i = 3;  
int *p;  
p = &i;
```

Biến	Địa chỉ	Giá trị
i	FFEC	3
p	FFEE	FFEC

36

2.2. Toán tử địa chỉ và toán tử nội dung

- Toán tử **&**: Trả về địa chỉ của biến.
- Toán tử *****: Trả về giá trị chứa trong vùng nhớ được trỏ bởi con trỏ.
- Cả hai toán tử ***** và **&** có độ ưu tiên cao hơn tất cả các toán tử số học ngoại trừ toán tử đảo dấu.

- Ví dụ:

```
int main()
{
    int i = 3, *p;
    p = &i;
    printf("p = %X và &i = %X \n", p, &i);
    printf("*p = %d và i = %d \n", *p, i);
    getch();
    return 0;
}
```

37

2.2. Toán tử địa chỉ và toán tử nội dung

- Một biến con trỏ có thể được gán bởi:
- Địa chỉ của một biến khác:

```
bienConTro = &bienKhac;
```

- Giá trị của một con trỏ khác (tốt nhất là cùng kiểu):

```
bienConTro1 = bienConTro2;
```

- Giá trị NULL (số 0):

```
bienConTro = 0; // hoặc bienConTro = NULL
```

- Gán giá trị cho biến con trỏ:

```
*bienConTro = GIA_TRI;
```

38

Ví dụ 1

```
int main()
{
    int i = 3, j = 6;
    int *p1, *p2;
    p1 = &i;
    p2 = &j;
    *p1 = *p2;
    getch();
    return 0;
}
```

Biến	Địa chỉ	Giá trị
i	FFEC	3
j	FFEE	6
p1	FFDA	FFEC
p2	FFDC	FFEE

Biến	Địa chỉ	Giá trị
i	FFEC	6
j	FFEE	6
p1	FFDA	FFEC
p2	FFDC	FFEE

39

Ví dụ 1

```
int main()
{
    int i = 3, j = 6;
    int *p1, *p2;
    p1 = &i;
    p2 = &j;
    *p1 = *p2;
    getch();
    return 0;
}
```

Biến	Địa chỉ	Giá trị
i	FFEC	3
j	FFEE	6
p1	FFDA	FFEC
p2	FFDC	FFEE

Biến	Địa chỉ	Giá trị
i	FFEC	3
j	FFEE	6
p1	FFDA	FFEE
p2	FFDC	FFEE

40

Con trỏ `void`

- Khai báo: `void *ten_bien_con_tro;`
- Con trỏ đặc biệt, không có kiểu
- Có thể nhận giá trị là địa chỉ của một biến thuộc bất kỳ kiểu dữ liệu nào.

- Ví dụ:

```
void *p, *q;  
int x = 21;  
float y = 34.34;  
p = &x; q = &y;
```

41

2.3. Các phép toán làm việc với con trỏ

- Cộng/trừ con trỏ với một số nguyên (`int`, `long`) → Kết quả là một con trỏ cùng kiểu
`ptr-- ; //ptr trỏ đến vị trí của phần tử đứng trước`
- Trừ hai con trỏ cho nhau
 - Kết quả là một số nguyên
 - Kết quả này nói lên khoảng cách (số phần tử thuộc kiểu dữ liệu của con trỏ) ở giữa hai con trỏ.

- Các phép toán: Cộng, nhân, chia, lấy số dư với toán hạng là con trỏ là không hợp lệ.
- Ví dụ: (p2 trỏ đến số nguyên nằm ngay sau x trong bộ nhớ)

```
int x, *p1, *p2;  
p1 = &x;  
p2 = p1+1;
```

42

9.2.4. Con trỏ và mảng

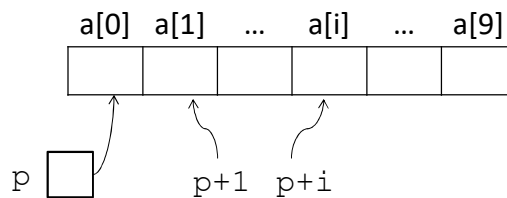
- Với mảng một chiều có tên là a thì
 - Giá trị của a là địa chỉ ô nhớ đầu tiên của vùng nhớ chứa mảng a .
 - Giá trị của a có giá trị bằng $\&a[0]$
 - Giá trị của a không thể thay đổi.
- Giả sử có khai báo: `int a[10], *p;`

Nếu $p = a$ thì

p trỏ tới $a[0]$

$p + 1$ trỏ tới $a[1]$

$p + i$ trỏ tới $a[i]$



43

Nội dung

1. Mảng
2. Con trỏ
3. Xâu kí tự
 - 3.1. Khái niệm xâu kí tự
 - 3.2. Khai báo và sử dụng xâu
 - 3.3. Các hàm xử lý kí tự
 - 3.4. Các hàm xử lý xâu

44

2.1. Khái niệm chuỗi ký tự

- Chuỗi ký tự (string) là một dãy các ký tự viết liên tiếp nhau
 - Ký tự kết thúc chuỗi: '0' (mã ASCII là 0)
 - Độ dài chuỗi là số ký tự có trong chuỗi không gồm ký tự kết thúc chuỗi
- Ví dụ: “Tin học”, “String”
- Lưu trữ: kết thúc chuỗi bằng ký tự '\0' hay NUL (mã ASCII là 0)

'T'	'i'	'n'	' '	'h'	'o'	'c'	'\0'
-----	-----	-----	-----	-----	-----	-----	------

45

2.1. Khái niệm chuỗi ký tự

- So sánh
 - Chuỗi ký tự và mảng ký tự?
 - Tập hợp các ký tự viết liên tiếp nhau
 - Sự khác biệt: chuỗi ký tự có ký tự kết thúc chuỗi, mảng ký tự không có ký tự kết thúc chuỗi
 - Chuỗi ký tự “A” và ký tự ‘A’?
 - ‘A’ là 1 ký tự
 - “A” là 1 chuỗi ký tự, ngoài ký tự ‘A’ còn có ký tự '\0' => gồm 2 ký tự

46

2.2. Khai báo và sử dụng chuỗi

a. Khai báo chuỗi

- Cú pháp 1

char tenXau [kích_thước];

- Lưu ý:

– Để lưu trữ một chuỗi có n ký tự chúng ta cần một mảng có kích thước n+1

- Ví dụ

– Để lưu trữ chuỗi “Tin học” chúng ta phải khai báo chuỗi có số phần tử tối đa ít nhất là 8
`char str [8];`

47

2.2. Khai báo và sử dụng chuỗi

a. Khai báo chuỗi

- Cú pháp 2

char tenXau [kích_thước] = Giá_trị;

- Lưu ý:

– *kích_thước* ≥ độ dài (***Giá_trị***) + 1

– Có thể không cần khai báo *kích_thước*: khi đó kích thước biến *tenXau* là độ dài (***Giá_trị***) + 1

- Ví dụ

`char str1[10] = “BKHN”;`

`char str2[5] = “SoICT”; //Lỗi`

`char str3[] = “SoICT”; //Đúng`

48

2.2. Khai báo và sử dụng chuỗi

b. Truy cập vào một phần tử của chuỗi

- Cú pháp:

tenXau [chỉ_số_của_kí_tự]

- Ví dụ

```
char quequan[10];
```

Giả sử chuỗi này có nội dung là "Ha noi"

```
⇒ quequan[0]    lưu trữ    'H'  
   quequan[1]                'a'  
   quequan[6]                'i'  
   quequan[7]                '\0'
```

49

Khai báo hằng số có kiểu chuỗi ký tự

- Sử dụng **#define**

```
#define TEN_XAU Giá_trị
```

Ví dụ: `#define DAI_HOC "BKHN"`

- Sử dụng từ khóa **const**

```
const char TEN_XAU[Kích_thước] = Giá_trị;
```

Ví dụ: `const char DAI_HOC[5] = "BKHN";`

- Khi khai báo với từ khóa **const**, kích thước phải đủ để chứa ký tự '\0'

- Tốt hơn: không khai báo kích thước hằng chuỗi ký tự

50

2.3. Các hàm xử lý kí tự

- Tập tiêu đề sử dụng: `ctype.h`
- `char toupper(char ch)`: chuyển kí tự thường thành kí tự hoa

```
char ch = 'a';  
ch = toupper(ch); // ch = 'A';  
ch = toupper('B'); // ch = 'B';
```

- `char tolower(char ch)`: chuyển kí tự hoa thành kí tự thường

```
char ch = 'A';  
ch = tolower(ch); // ch = 'a';  
ch = tolower('b'); // ch = 'b';
```

51

2.3. Các hàm xử lý kí tự

- `int isalpha(char ch)`: kiểm tra xem kí tự có phải chữ cái hay không ('a'...'z', 'A',...'Z')
 - `int isdigit(char ch)`: kiểm tra chữ số ('0', '1',...'9')
 - `int islower(char ch)`: kiểm tra chữ thường
 - `int isupper(char ch)`: kiểm tra chữ hoa
 - `int iscntrl(char ch)`: kiểm tra kí tự điều khiển (0-31)
 - `int isspace(char ch)`: kiểm tra kí tự dấu cách (mã 32), xuống dòng ('\n' 10), đầu dòng ('\r' 13), tab ngang ('\t' 9), tab dọc ('\v' 11)
- trả về khác 0 nếu đúng, ngược lại trả về 0

52

2.3. Các hàm xử lý kí tự

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
int main(){
    char ch;
    printf("Nhap vao mot ki tu: ");
    fflush(stdin);
    scanf("%c", &ch);
```

53

2.3. Các hàm xử lý kí tự

```
if(isupper(ch)){
    printf("Ki tu nay la chu hoa\n");
    printf("Ki tu chu thuong tuong
    ung %c\n", tolower(ch));
}else if(islower(ch)){
    printf("Ki tu nay la chu thuong\n");
    printf("Ki tu chu hoa tuong ung
    %c\n", toupper(ch));
}
getch();
return 0;
}
```

54

2.3. Các hàm xử lý kí tự

Vào ra chuỗi kí tự

- Tập tiêu đề: `stdio.h`
- Nhập chuỗi kí tự
 - `gets(tenXâu);`
 - `scanf("%s",&tenXâu);`
- Hiển thị chuỗi kí tự
 - `puts(tenXâu);`
 - `printf("%s", tenXâu);`
- Sự khác nhau giữa `gets` và `scanf`?

55

2.4. Các hàm xử lý chuỗi kí tự

Tập tiêu đề: `string.h`

- `int strlen(char[] ten_xau):` trả về độ dài chuỗi tính đến trước ký tự `'\0'` đầu tiên trong chuỗi

```
char s[] = "Tin hoc dai cuong";
int n;
n = strlen(s); // n = ?
s[7] = 0;
n = strlen(s); // n = ?
printf("Chuoi: %s",s)//Hiển thị?
```

56

2.4. Các hàm xử lý chuỗi ký tự

- **strcpy(char[] xauDich, char[] xauNguon)**: sao chép nội dung xauNguon và xauDich
 - Lưu ý: không dùng phép gán giá trị cho các biến chuỗi

```
char s1[] = "Tin hoc", s2[10];  
  
strcpy(s2, s1); // s2 = "Tin hoc"  
strcpy(s1, "Tin hoc dai cuong");
```

57

2.4. Các hàm xử lý chuỗi ký tự

- **int strcmp(char[] xau_thu_nhat, char[] xau_thu_hai)**: so sánh hai chuỗi
 - giá trị 0 : hai chuỗi giống nhau
 - giá trị <0: chuỗi thứ nhất nhỏ hơn chuỗi thứ hai
 - giá trị >0: chuỗi thứ nhất lớn hơn chuỗi thứ hai
- Quy tắc: so sánh lần lượt các ký tự từ đầu mỗi chuỗi.
 - Chuỗi nào xuất hiện ký tự có mã ASCII lớn hơn trước thì lớn hơn
 - Tất cả các ký tự giống nhau thì hai chuỗi bằng nhau
- Ví dụ: "Tin hoc" > "TIN hoc dai cuong"
"Tin hoc" = "Tin hoc"

58

Hàm int `strcmp()` – Ví dụ

```
char s1[30], s2[20];
printf("Nhap xau thu nhat:");
fflush(stdin); gets(s1);
printf("Nhap xau thu hai:");
fflush(stdin); gets(s2);
if (strcmp(s1,s2)>0)
    printf ("Xau thu nhat lon hon");
else if(strcmp(s1,s2)<0)
    printf ("Xau thu hai lon hon");
else printf ("Hai xau giong nhau");
```

Không so sánh trực tiếp 2 xâu

59

2.4. Các hàm xử lý xâu kí tự

- `char[] strcat(char[] xauDich, char[] xauNguon):`
ghép nội dung xâu nguồn vào sau nội dung xâu đích

```
char s1[] = "Hello,", s2[20];
printf("Nhap ten:");
fflush(stdin); gets(s2);
strcat(s1, s2); //s1 = ?
strcat(s1, ".Start game!"); //s1 = ?
```

60

Thảo luận



61