CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.1. Câu lệnh đơn

Một lệnh (gán, tăng giảm giá trị, lời gọi hàm ...) kết thúc bởi dấu chấm phẩy (;) là một câu lệnh đơn.

2.2. Khối lệnh (câu lệnh ghép)

Câu lệnh ghép (khối lệnh) là tập hợp các câu lệnh được bao bởi cặp dấu ngoặc nhọn "{" và "}".

- Ngôn ngữ C xem một khối lệnh cũng như là một câu lệnh riêng lẻ, vì vậy chỗ nào viết được một câu lệnh thì ở đó cũng viết được một khối lệnh.
- Có thể khai báo biến ở đầu khối lệnh
- Sự lồng nhau của các khối lệnh: Bên trong một khối lệnh có thể viết các khối lệnh khác.

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

- Phạm vi hoạt động của biến:

* Giá trị của biến khai báo bên trong một khối lệnh không thể đưa ra để sử dụng bên ngoài khối lệnh đó.
* ở bên ngoài khối lệnh ta không thể can thiệp đến các biến được khai báo bên trong khối lệnh.
* Nếu một biến đã khai báo ở ngoài một khối lệnh và không trùng tên với các biến bên trong khối lệnh này thì biến đó cũng có thể sử dụng cả bên trong cũng như bên ngoài khối lệnh.

```
{ // khối lệnh 1
   int a , b ;        // biến a ngoài khối lệnh 2
         . . . . .
         {   //  khối lệnh 2
            int a;  //  biến a trong khối lệnh 2
                  . . . . .
         }
         . . . . .
}
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.3. Lệnh if

***Dạng 1*:** **if (< biểu thức logic >) < lệnh > ;**

Hoạt động: Nếu < biểu thức logic > là đúng (có giá trị khác 0) thì máy sẽ thực hiện < lệnh >, sau đó nhảy tới thực hiện lệnh sau lệnh if. Nếu < biểu thức logic > là sai (có giá trị bằng 0) thì máy sẽ bỏ qua < lệnh > mà chuyển sang thực hiện lệnh sau lệnh if.

VD:

```
if ( max  <  x)  max = x;
if ( min  >  x)  min = x;
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

***Dạng 2*:**

if (< biểu thức logic>) < lệnh 1 > ; else < lệnh 2 > ;

Hoạt động: Nếu < biểu thức logic > là đúng (có giá trị khác 0) thì máy sẽ thực hiện < lệnh 1 >, sau đó nhảy tới thực hiện lệnh viết sau < lệnh 2 >. Nếu < biểu thức logic > là sai (có giá trị bằng 0) thì máy sẽ thực hiện < lệnh 2 > và sau đó thực hiện lệnh viết sau nó.

VD:

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

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

Sự lồng nhau của lệnh if: Các câu lệnh if có thể lồng nhau, nếu ta không sử dụng các dấu đóng mở khối lệnh { và } thì rất dễ gây ra sự lộn xộn. Máy tính sẽ gắn else với if không có else gần nhất trước đó.

else if: Khi muốn thực hiện một trong n quyết định, ta có thể sử dụng câu lệnh if dưới dạng sau:

```
if  < biểu thức 1 >
    < lệnh 1 > ;
else if  < biểu thức 2 >
    < lệnh 2 > ;
. . . . .
else if  < biểu thức n-1 >
    < lệnh n-1 > ;
else
     < lệnh n > ;
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

VD:

Mã	Dân tộc
1	Kinh
2	Tày
3	Nùng
4	Thái
5	Khơmer

Cỏc lệnh if lồng nhau:

```
if(ma==1) printf("Kinh");
else if(ma==2) printf("Tay");
else if(ma==3) printf("Nung");
else if(ma==4) printf("Thai");
else if(ma==5) printf("Khomer");
else printf("ma sai");
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.4. Lệnh switch

```
switch ( < biểu thức nguyên > )
{       case < giá trị 1 > : < lệnh 1 > ; [ break ; ]
        case < giá trị 2 > : < lệnh 2 > ; [ break ; ]
        . . . . .
        case < giá trị n > : < lệnh n > ; [ break ; ]
        [ default :  <lệnh n+1> ; [ break ; ] ]
}
```

Lệnh switch căn cứ vào < biểu thức nguyên > để chọn một nhánh. Giá trị của < biểu thức nguyên > được so sánh với các giá trị đứng sau case, nếu đúng thì sẽ thực hiện lệnh tương ứng với trường hợp đó.

Trường hợp có nhãn default, câu lệnh ứng với nó được thực hiện nếu không có trường hợp nào thỏa mãn.

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

VD:

Mã nghề	Nghề
1	Công nhân
2	Nông dân
3	Bộ đội
4	Doanh nhân

Lệnh switch như sau:

```
switch(ma)
    {     case 1: printf("Cong nhan"); break;
          case 2: printf("Nong dan"); break;
          case 3: printf("Bo doi"); break;
          case 4: printf("Doanh nhan"); break;
          default: printf("Cac nghe khac");
    }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.5. Lệnh lặp for

for ([<th/ phần 1>] ;[<th/ phần 2>] ;[< th/ phần 3 >])
< lệnh > ;

Hoạt động của lệnh for:

(1) Xác định <thành phần 1>

(2) Xác định <thành phần 2>

(3) Tùy thuộc vào tính đúng sai của <thành phần 2>:

+ Nếu <thành phần 2> là sai (có giá trị bằng 0) máy sẽ thoát ra khỏi for và chuyển tới lệnh sau for

+ Nếu <thành phần 2> là đúng (có giá trị khác 0), máy sẽ thực hiện <lệnh > (chính là thân vòng lặp), thực hiện bước (4)

(4) Tính <thành phần 3> , sau đó quay trở lại bước (2) để bắt đầu một vòng lặp mới

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

VD 1: tính tổng các số lẻ từ 1 đến 9999:

```
s = 0;
for( i = 1; i <= 9999; i = i+2) s = s + i  ;
```

VD 2: Tính n!! :

$$n!! = \begin{cases} 1.3.5...n & \text{nếu } n \text{ lẻ} \\ 2.4...n & \text{nếu } n \text{ chẵn} \end{cases}$$

```
gtc = 1;
for(i=n; i >= 1; i = i-2) gtc =  gtc *  i;
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

- Lệnh break và lệnh continue:

Lệnh break : Lệnh break dùng để thoát ra khỏi vòng lặp trực tiếp chứa nó khi thỏa điều kiện nào đó.

VD:

```
ngto = 1;
for(i=2; i <= floor(sqrt(n)); i++)
      if(n % i == 0) {ngto = 0; break;}
```

Lệnh continue: Lệnh continue dùng để bỏ qua một lần lặp khi thỏa mãn điều kiện nào đó.

```
for(i = 10; i <= 20; i++)
      {
            if(i==13 || i==17 ) continue;
            printf("%d\t", i);
      }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

Nhưng chú ý về lệnh lặp for:

(1) Các thành phần trong lệnh for có thể vắng mặt nhưng vẫn phải cách nhau bởi dấu chấm phẩy (;)

VD 1:

```
i = 1;
for (   ;i <= n; i++) s += x[i]; //Vắng thành phần thứ 1
```

VD 2:

```
for (i=1; i <= n;   )          // Vắng thành phần thứ 3
{
  s += x[i];  i++;
}
```

(2) Khi thành phần thứ hai vắng mặt thi nó được xem là đúng, muốn thoát khỏi vòng lặp for thi cần phải dùng lệnh break, hoặc lệnh return, hoặc lệnh goto.

VD 3:

```
for (i=1 ;   ; i++) // Vắng thành phần thứ 2
{
  s += x[i];
  if (i >= n) break;
}
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

(3) Khi gặp lệnh break máy tính sẽ thoát ra khỏi vòng for sâu nhất chứa lệnh break này.

(4) Trong thân vòng lặp for có thể dùng lệnh goto để đến một vị trí nào đó. Cũng có thể sử dụng lệnh return trong thân vòng lặp for để trở về một hàm nào đó.

(5) Trong thân vòng lặp for nếu gặp lệnh continue thi máy tính sẽ bỏ qua các lệnh còn lại trong thân for và chuyển đến thực hiện <thành phần thứ ba>.

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

(6) Các lệnh lặp for có thể lồng nhau.

VD:

```
for(i=1; i<=n-1; i++)
   for(j=i+1; j <=n; j++)
     if(x[i] > x[j])
     {
       tg = x[i]; x[i] = x[j]; x[j] = tg;
     }
```

(7) Trong lệnh lặp for, thân vòng lặp có thể rỗng.

VD:

```
for (i=1; i<=n ; s += x[i],i++) ;
```

(8) Trong lệnh lặp for mỗi thành phần không những có thể là một biểu thức, mà còn có thể là một dãy biểu thức phân cách bởi dấu phẩy (,), thứ tự thực hiện từ trái sang phải; tính đúng / sai của <thành phần thứ hai> được hiểu là tính đúng / sai của biểu thức cuối cùng trong dãy này.

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.6. Lệnh lặp while

Dạng lệnh:

while (< biểu thức >)
< lệnh > ;

Sự hoạt động của while:

(1) Xác định giá trị của < biểu thức >

(2) Tùy thuộc vào giá trị của < biểu thức > này, máy sẽ lựa chọn :

- Nếu < biểu thức > có giá trị 0 (sai), máy sẽ ra khỏi chu trình.
- Nếu < biểu thức > có giá trị khác 0 (đúng), máy sẽ thực hiện < lệnh >, sau đó quay về bước (1)

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

VD: Tính $$P = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + ... + \frac{(-1)^n}{2n+1} + ...$$

Với sai số $\frac{1}{2n+1} < \varepsilon$ $\quad$ *với* $\varepsilon > 0$ *đủ bé*

Đoạn chương trinh như sau:

```
scanf ("%f", &ep);
p = 1. ; n = 1;
while (1.0 / (2*n+1) >= ep)
      {
         p +=  pow(-1, n)/(2*n+1);
         n++;
      }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.7. Lệnh lặp do ... while …

Dạng lệnh:

```
do
{
        < lệnh >
}
while   (< biểu thức >) ;
```

Sự hoạt động của do ... while:

(1) Thực hiện <lệnh> trong thân do ... while ...

(2) Khi gặp dấu ngoặc nhọn "}" cuối cùng của thân do ... while máy sẽ xác định giá trị của <biểu thức>

(3) Máy sẽ phân nhánh theo giá trị của <biểu thức>:

- Nếu biểu thức có giá trị đúng (khác 0), máy sẽ trở lại bước (1) để tiếp tục thực hiện vòng lặp mới của chu trình.
- Nếu biểu thức có giá trị sai (bằng 0), máy sẽ thoát ra khỏi chu trình.

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

VD 1: Nhập một số nguyên dương:

```
do
      {     printf("Nhap so nguyen duong:  ");
            scanf("%d",  &n);
      } while (n <= 0);
```

VD 2: Tính n!!

```
gtc = 1.;
do
      {
            gtc *= n;
            n = n-2;
      }  while (n >= 1);
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.8. Mảng 1 chiều

2.8.1. Khai báo mảng 1 chiều

a/ Cách 1: Khai báo trực tiếp biến mảng

<kiểu dữ liệu> <tên mảng> [<số phần tử tối đa của mảng>] ;

Ví dụ: int a [100] ; float b [50] ;

b/ Cách 2: Khai báo một kiểu dữ liệu mới là mảng, sau đó sử dụng kiểu đó để khai báo biến.

Ví dụ :

typedef int mangi1[100] ;

typedef float mangt1[50] ;

mangi1 a ;

mangt1 b ;

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.8.2. Truy xuất phần tử mảng 1 chiều

Chỉ việc viết như sau:

<tên mảng> [<chỉ số của phần tử>]

2.8.3. Nhập xuất mảng 1 chiều

```
for(i=0; i < n; i++)
  {
    printf("Nhap phan tu thu %d : ", i+1);
    scanf("%d", &a[i]);
  }
printf("\nMang 1 chieu:\n");
for (i = 0; i < n; i++ )
printf("a[%d] = %d \t ", i,a[i]) ;
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.8.4. Tìm kiếm trên mảng 1 chiều

a/ Tìm phần tử lớn nhất, phần tử nhỏ nhất trong mảng 1 chiều

Cách 1: Tìm theo giá trị

```
max = -32768; min = 32767; //Gan gia tri dau cho max
                           // va min
for(i=0; i < n; i++)
      {
            printf("Nhap phan tu thu %d : ", i+1);
            scanf("%d", &a[i]);
            if (max < a[i]) max = a[i];
            if (min > a[i]) min = a[i];
      }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

Cách 2: Tìm max, min theo chỉ số.

```
imax = 1; imin = 1;     //Gan gia tri dau cho imax va
imin
for(i=0; i < n; i++)
      {
            printf("Nhap phan tu thu %d : ", i+1);
            scanf("%d", &a[i]);
            if (a[imax] < a[i]) imax = i;
            if (a[imin] > a[i]) imin = i;
      }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

b/ Tìm phần tử đầu tiên trong mảng 1 chiều có giá trị bằng số x nào đó.

```
itim = -1;  // Gán giá trị đầu cho itim
for(i=0; i < n; i++)
      if(x == a[i])
            {
                  itim = i;
                  break;
            }
if(itim == -1)printf("\nDay khong co phan tu nao thoa man");
else printf("x = %d, phan tu thu %d co gia tri bang x", x, itim);
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.8.5. Phương pháp đếm

Ví dụ: Đếm các phần tử chia hết cho 5 trong mảng các số nguyên.

```
dem = 0;
for(i=0; i < n; i++)
      {
            printf("Nhap phan tu thu %d : ", i);
            scanf("%d", &a[i]);
            if (a[i] % 5 == 0) dem++;
      }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.8.6. Tính tổng, trung bình cộng

Tính tổng các phần tử dương và tính trung bình cộng các phần tử âm trong mảng các số nguyên.

```
sd = 0; sa = 0; dem = 0;
for(i=0; i < n; i++)
      {
            printf("Nhap phan tu thu %d : ", i+1);
            scanf("%d", &a[i]);
            if (a[i] > 0) sd = sd + a[i];
            if (a[i] < 0)
                  { dem++; sa = sa + a[i]; }
      }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.8.7. Sắp xếp tăng dần mảng 1 chiều

Thuật toán sắp xếp lựa chọn trực tiếp:

```
for (i=1; i <= n-1; i++)
     for (j=i+1; j <= n; j++)
          if (a[i] > a[j])
               {
                    tg = a[i];
                    a[i] = a[j];
                    a[j] = tg;
               }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.9. Mảng 2 chiều

2.9.1. Khái niệm

Mảng 2 chiều thực chất là mảng 1 chiều trong đó mỗi phần tử của mảng là một mảng 1 chiều, và được truy xuất bởi hai chỉ số là dòng và cột.

2.9.2. Khai báo mảng 2 chiều

a/ Cách 1: Khai báo trực tiếp mảng 2 chiều:

<kiểu dữ liệu> <tên mảng> [<số dòng tối đa>]
[<số cột tối đa>] ;

Ví dụ:

```
int  a[10][5] ;
float x[15] [20] ;
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

b/ Cách 2: Khai báo kiểu dữ liệu mảng 2 chiều mới, sau đó sử dụng kiểu dữ liệu mới này để khai báo mảng 2 chiều.

Ví dụ :

```
typedef int mangi2 [10] [ 5 ] ;
typedef float mangt2 [15] [20] ;
mangi2  a;
mangt2   x ;
```

2.9.3. Truy xuất phần tử mảng 2 chiều

Dạng truy xuất phần tử mảng như sau:

<tên mảng> [<chỉ số dòng>] [<chỉ số cột>]

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.9.4. Ma trận vuông và một vài khái niệm liên quan

a/ Khái niệm*:* Ma trận vuông là ma trận có số dòng bằng số cột.

b/ Một vài tính chất của ma trận vuông*:*

- *Đường chéo loại 1*: Đường chéo loại 1 bao gồm đường chéo chính và những đường chéo song song với đường chéo chính. Trong đó đường chéo chính là đường chéo chứa các phần tử có chỉ số dòng = chỉ số cột.

Truy xuất các phần tử trên đường chéo loại 1: Dựa vào chỉ số dòng và chỉ số cột như sau : cột – dòng = hằng số .

Ví dụ: Cho ma trận vuông $a_{n \times n}$, gọi (i_0, j_0) là tọa độ điểm xuất phát, chúng ta có thể duyệt đường chéo loại 1 xuất phát từ (i_0, j_0) như sau :

```
for (i = i0,j = j0; i < n; i++,j++)
printf(''%6d'', a[i][j]);
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

- *Đường chéo loại 2* : Đường chéo loại 2 bao gồm đường chéo phụ và những đường chéo song song với nó. Trong đó đường chéo phụ là đường chéo chứa các phần tử có chỉ số dòng + chỉ số cột = số dòng (hoặc số cột)

Truy xuất các phần tử trên đường chéo loại 2 : Dựa vào chỉ số dòng và chỉ số cột như sau : cột + dòng = hằng số.

Ví dụ: Cho ma trận vuông $a_{n \times n}$, gọi (i_0, j_0) là tọa độ điểm xuất phát, chúng ta có thể duyệt đường chéo loại 2 xuất phát từ (i_0, j_0) như sau :

```
for(i=i0,j=j0;i<n && j>=0; i++,j--)
printf(''%6d``,a[i][j]) ;
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.9.5. Một số thao tác trên mảng 2 chiều

a/ Nhập - xuất ma trận

```
for(i = 0; i < m; i++)
      for(j = 0; j < n; j++)
      { printf("a[%d][%d] = ", i+1, j+1);
       scanf("%d", &tg);
       a[i][j] = tg;
      }
   printf("Ma tran nhap vao:\n");
   for (i = 0; i < m; i++)
      { for (j = 0; j < n; j++) printf("%6d",
a[i][j]);
       printf("\n");
      }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

b-1/ *Tìm max và min của ma trận*

- Cách 1 : Tìm max, min của ma trận theo giá trị.

```
max = -32768; min = 32767; //Gan gia tri dau
cho max, min
for(i = 0; i < m; i++)
  for(j = 0; j < n; j++)
  {
     printf("a[%d][%d] = ", i+1, j+1);
     scanf("%d", &tg);
     a[i][j] = tg;
     if(max < a[i][j]) max = a[i][j];
     if(min > a[i][j]) min = a[i][j];
  }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

Cách 2 : Tìm max, min của ma trận theo chỉ số.

```
imax = jmax = imin = jmin = 0;
  for(i = 0; i < m; i++)
   for(j = 0; j < n; j++)
   {
      printf("a[%d][%d] = ", i+1, j+1);
      scanf("%d", &tg);
      a[i][j] = tg;
      if(a[imax][jmax] < a[i][j])
      { imax = i; jmax = j; }
      if(a[imin][jmin] > a[i][j])
      { imin = i; jmin = j; }
   }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

b-2/ *Tìm phần tử thỏa mãn điều kiện nào đó.*

Ví dụ tìm phần tử ma trận các số nguyên thỏa mãn điều kiện là số lẻ và lớn hơn 100.

```
co = 0;
for(i = 0; i < m; i++)
   for(j = 0; j <n; j++)
   if(a[i][j] % 2 != 0 && a[i][j] > 100)
      {
         co = 1;
         it = i+1;
         jt = j+1;
         break;
      }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

c/ Phương pháp đếm

Ví dụ: Đếm các phần tử lẻ trong một ma trận các số nguyên.

```
demle = 0;
for(i = 0; i < m; i++)
   for(j = 0; j < n; j++)
   {
   printf("a[%d][%d] = ", i+1, j+1);
   scanf("%d", &tg);
   a[i][j] = tg;
   if(a[i][j] % 2 != 0) demle++;
   }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

d/ Tính tổng các phần tử dương và trung bình cộng các phần tử âm trong một ma trân.

```
      sd = sa = demam = 0;
   for(i = 0; i < m; i++)
      for(j = 0; j < n; j++)
      {
      printf("a[%d][%d] = ", i+1, j+1);
      scanf("%d", &tg);
      a[i][j] = tg;
      if(a[i][j] > 0) sd += a[i][j];
      if(a[i][j]  <  0)  {  sa  +=  a[i][j];
demam++; }
      }
   if(demam != 0) tbcam = sa / demam;
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

e/ Sắp xếp ma trận tăng dần theo thứ tự từ trái sang phải và từ trên xuống dưới (không dùng mảng phụ).

```
for(i = 0; i <= m*n - 1; i++)
      for(j = i+1; j <= m*n ; j++)
      if(a[i/n][i%n] < a[j/n][j%n])
         {
            tg = a[i/n][i%n];
            a[i/n][i%n] = a[j/n][j%n];
            a[j/n][j%n] = tg;
         }
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.10.1. Xâu ký tự

2.10.1. Khai báo biến xâu ký tự

char <biến xâu> [<độ dài tối đa>] ;

Ví dụ:

```
char     s [ 50 ] ;
```

Có thể khởi đầu cho một xâu ngay lúc khai báo, ví dụ:

```
char  hoten [ ] = "Nguyen Van X" ;
char  tentinh [ 30 ] = "Nam Dinh" ;
char   thudo  [25]  =  {'H',  'A',  'N',  'O',
'I','\0'} ;
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.10.2. Cách tổ chức lưu trữ xâu ký tự

Các ký tự sẽ được lưu trong mảng theo thứ tự từ trái sang phải và bắt đầu từ chỉ số 0 trở đi, ký tự cuối cùng dùng làm ký tự kết thúc xâu được C quy định là ký tự '\0' (ký tự này có mã ASCII là 0, tức là ký tự NULL).

Ví dụ:

char s[ ] = 'ABC';

Khi đó xâu s được tổ chức như sau:

A	B	C	'\0'

Trong đó s[0] = 'A', s[1] = 'B', s[2] = 'C', s[3] = '\0'.

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.10.3. Mảng các xâu ký tự

Khai báo mảng các xâu ký tự:

char <tên biến> [m] [n] ;

Trong đó m là số dòng (số xâu có thể lưu trong mảng), n là số cột (số ký tự tối đa của mỗi xâu).

Ví dụ: char s [100] [50] ;

Cột / Dòng	0	1	2	3	4	5	6	7	8	9	10	…	49
0	H	A		N	O	I	\0						
1	V	I	E	T		N	A	M	\0				
2	W	O	R	L	D	\0							
…													
99													

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.10.2. Một số hàm xử lý xâu ký tự

Hàm	Ý nghĩa
puts(s)	Hiện ra màn hinh xâu s
gets(s)	Nhập từ bàn phím một xâu
scanf("%s", &s)	Nhập từ bàn phím một xâu không kể ký tự trống
strlen(s)	Hàm tra lại độ dài của xâu s
strcpy(s,p)	Hàm copy xâu p vào xâu s
strncpy(s,p,n)	Hàm copy n ký tự đầu tiên từ xâu p vào xâu s
strnset(s, c, n)	Hàm sao n lần ký tự c vào chuỗi s
strcat(s,p)	Hàm nối xâu p vào sau xâu s
strncat(s,p, n)	Hàm nối n ký tự đầu tiên của xâu p vào sau xâu s

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

Hàm	Ý nghĩa
strcmp(s,p)	Hàm tra lại giá trị dơng nếu xâu s lớn hơn xâu p, tra lại giá trị âm nếu xâu s nhỏ hơn xâu p, tra lại giá trị 0 nếu xâu s đúng bằng xâu p.
strncmp(s,p,n)	Hàm so sánh n ký tự đầu tiên của xâu s và p
strcmpi(s,p)	Tơng tự nh strcmp nhng không phân biệt ch hoa và ch thờng
strstr(s,p)	Hàm tra lại vị trí của xâu p trong xâu s, nếu p không có mặt trong s thi hàm tra lại con trỏ NULL
strchr(s, c)	Hàm tim lần xuất hiện đầu tiên của ký tự c trong chuỗi s, nếu không tim thấy hàm tra về NULL
strrev(s)	Hàm đao xâu s theo thứ tự ngợc lại
strupr(s)	Hàm đổi chuỗi s sang ch hoa
strlwr(s)	Hàm đổi chuỗi s sang ch thờng

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

2.10.3. Một số thao tác trên xâu ký tự

Đếm ký tự:

```
dem = 0;
for(i=0; i < strlen(s); i++)
      if(s[i] thoả mãn điều kiện) dem++;
printf("So ky tu thoa man dieu kien la  %d ", dem);
```

Đếm từ:

```
dem = s[0] = = ' ' ? 0 : 1;
for (i=1; i < strlen(s); i++)
      if((s[i] = = ' ') && (s[i-1] != ' ')) dem++;
printf("So tu la  %d ", dem);
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

Xoá dấu trống ở đầu xâu:

```
while (s[0] == ' ')
   for(i=1;s[i]!=0;i++)s[i-1]=s[i];
```

Hoặc:

```
k=strlen(s);
i=0;
while (s[i] == ' ' && i < k) i++;
strcpy(&s[0],&s[i]);
```

Xoá dấu trống ở cuối xâu:

```
k = strlen(s);
while(s[k-1]==' ' && k>0) s[--k]=0;
```

CHƯƠNG 2: CÁC LỆNH ĐIỀU KHIỂN

Xoá dấu cách thừa giữa hai từ:

```
k = strlen(s);
   for(i=1;i<k; )
      if((s[i]==' ') && (s[i-1]==' '))
            for(j=i;s[j]!=0;j++)s[j]=s[j+1];
      else i++;
```

Hoặc:

```
   k = strlen(s);
   i=0;
   while(i < k-1)
   if(s[i]==' ' && s[i+1]==' ') strcpy(&s[i],&s[i+1]);
   else i++;
```