Mảng

Mảng một chiều:

1. Khai báo:

kiểu_giá_trị BIẾN[N];

- N ***hằng nguyên*** ≥ 1,

- Lệnh khai báo cho N biến: BIẾN[**0**], BIẾN[1], . . , BIẾN[N-1] có kiểu là **kiểu_giá_trị**.

Ví dụ 1 : Cho *mảng 3 phần tử kiểu nguyên*. Tính tổng mảng.

```
int a[3] ;
a[0]=10;                    ⇒ a[0] ← 10
a[1]=11;                    ⇒ a[1] ← 11
a[2]=12;                    ⇒ a[2] ← 12
T= a[0] + a[1] + a[2];      ⇒T ← 10 + 11 + 12
printf("%d", T);
```

Ví dụ 2 : Nhập và viết 10 số nguyên.

```
int a[10];
int i;

for (i=0 ; i < 10; i=i+1) {
    printf("Nhap a[%d] = ", i ) ;
    scanf("%d", &a[i]);
}

 for (i=0 ; i < 10; i=i+1) printf("%d   ", a[i]) ;
```

Ví dụ 3 :

```
int a[5], i;
i = 1;
a[i + 2] = 10;
```

Ví dụ 4 : Cho biết kết quả được viết ra màn hình. Giả sử giá trị của các phần tử mảng a khi khai báo có giá trị là 0.

1. int a[3] = {0, 0, 0};

2. i = 5;

3. a[i / 2] = 10;

4. printf("%d %d %d ", a[0], a[1],a[2]);

Các bài toán mảng một chiều :

Mảng n phần tử kiểu nguyên.

Bài 1 : Tổng mảng.

Bài 2 : Giá trị lớn nhất của mảng (Max của mảng).

Bài 3 : Đếm số phần tử có giá trị bằng x.

Bài 4 : Tìm phần tử đầu tiên trong mảng có giá trị bằng x (kết quả là chỉ số).

Bài 5 : Kiểm tra mảng đối xứng ?

Bài 1 : Tổng mảng M.

int M[4];

Nhập mảng M;

T = 0;

T = T + M[0];

T = T + M[1];

T = T + M[2];

T = T + M[3];

Bài 2 : Phần tử lớn nhất.

int M[4];

Nhập mảng M;

1. max = M[0];

2. if (max < M[1]) max = M[1];

3. if (max < M[2]) max = M[2];

4. if (max < M[3]) max = M[9];

Bài 3 : Đếm số phần tử có giá trị bằng x.

int M[4];

Nhập mảng M, x;

1. d = 0;

2. if (x = M[0]) d = d+1;

3. if (x = M[1]) d = d+1;

4. if (x = M[2]) d = d+1;

5. if (x = M[3]) d = d+1;

Bài 4 : Tìm phần tử đầu tiên có giá trị bằng x.

int M[5];

Nhập mảng M, x;

cs = -1;

if (x == M[0]) { cs = 0 ; **Kết thúc tìm**;}

if (x == M[1]) { cs = 1 ; **Kết thúc tìm**;}

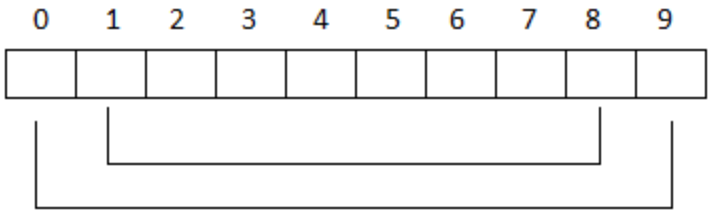
if (x == M[2]) { cs = 2 ; **Kết thúc tìm**;}

if (x == M[3]) { cs = 3 ; **Kết thúc tìm**;}

if (x == M[4]) { cs = 4 ; **Kết thúc tìm**;}

Kết thúc tìm : Viết kết quả;

Bài 5 : Mảng đã cho là đối xứng ?



d = 0;

if (M[0] = M[9]) d = d +1;

if (M[1] = M[8]) d = d +1;

if (M[2] = M[7]) d = d +1;

if (M[3] = M[6]) d = d +1;

if (M[4] = M[5]) d = d +1;

Viết kết quả;

Mảng hai chiều:

Khai báo :

kiểu_giá_trị a[M][N];

- M, N : hằng nguyên ≥ 1.

- Khai báo cho các biến **a[i][j]**, i =0, .., M-1 , j = 0, . . , N-1 có kiểu giá trị là **kiểu_giá_trị**.

Ví dụ :

int a[2][3] ;

Các biến :

a[0][0], a[0][1], a[0][2],

a[1][0], a[1][1], a[1][2],

có kiểu là **int**.

Ví dụ : Nhập mảng 2 chiều kiểu nguyên , kích thước 2x3. In ra màn hình các phần tử hàng 0.

```
int a[2][3];
for (i=0 ; i < 2 ; i=i+1)
    for (j=0 ; j < 3 ; j=j+1) {
          scanf("%d", &a[i][j]);
    }

for (j=0 ; j < 3 ; j=j+1)
         printf("%d   ", a[0][j]);
```

Ví dụ : Nhập mảng 2 chiều kiểu nguyên , kích thước 2x3. In ra màn hình tất cả các hàng.

```
int a[2][3];
for (i=0 ; i < 2 ; i=i+1)
    for (j=0 ; j < 3 ; j=j+1) {
        scanf("%d", &a[i][j]);
    }
for (i=0 ; i < 2 ; i=i+1) {
     for (j=0 ; j < 3 ; j=j+1)
            printf("%d   ", a[i][j]);
     printf("\n");
}
```

Ví dụ : Tính tổng 2 mảng a[2][3] và b[2][3] chứa kết quả vào mảng c[2][3]. Phần tử c[i][j] = a[i][j] + b[i][j] .

```
int a[2][3], b[2][3], c[2][3];
Nhập a, b;
for (i=0 ; i < 2 ; i=i+1)
    for (j=0 ; j < 3 ; j=j+1)
        c[i][j] = a[i][j] + b[i][j];
Viết c;
```

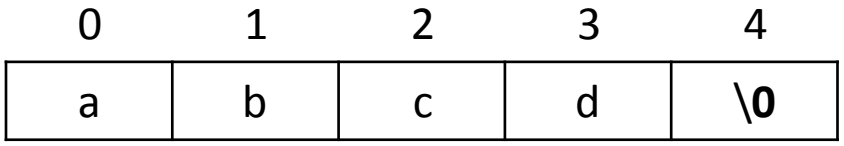
CHUỖI

Khai báo: char S[N];

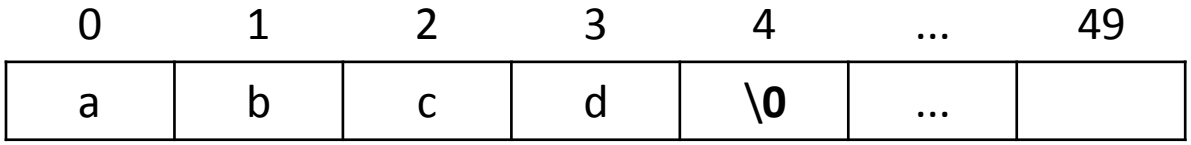
- N : hằng nguyên ≥ 2,
- Chuỗi S có tối đa N-1 ký tự.

Ví dụ:

char c1[] = "abcd";



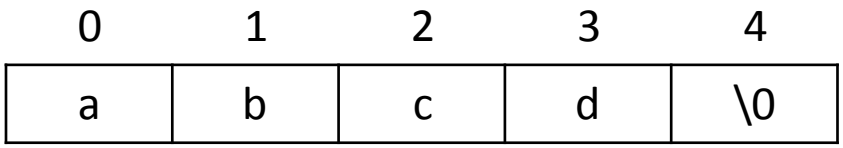
char c2[50] = "abcd";



Ví dụ:

char c3[] = {'a', 'b', 'c', 'd', '\0'};

char c4[5] = {'a', 'b', 'c', 'd', '\0'};



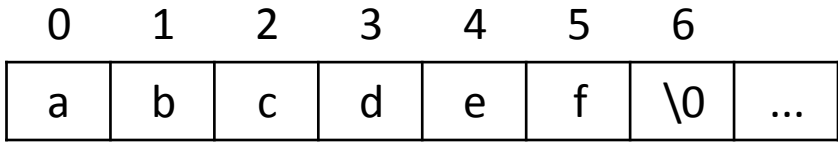
Ví dụ: Xét đoạn chương trình :

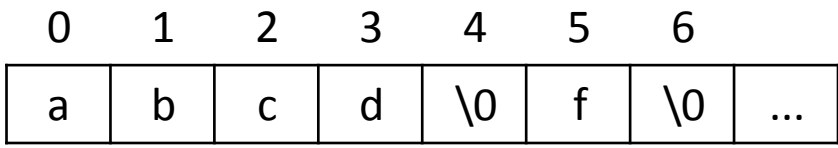
char c[20]= "abcdef";

printf("%s", c); → abcdef

c[4]=0; // c[4]='\0';

printf("%s", c); → abcd





Gán giá trị cho chuỗi: Lệnh gán sau sẽ bị báo lỗi “

```
char c[20];
c = "abcdef";  // error
```

error : *'=' : cannot convert from 'char [7]' to 'char [20]'*

Nhập và viết giá trị: scanf, printf

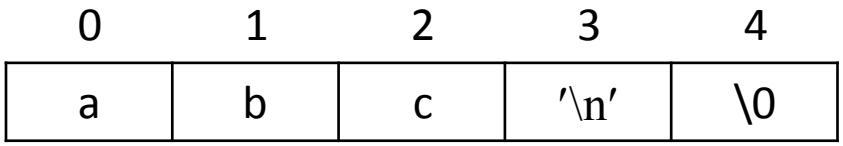
```
#include <stdio.h>
int main()
{
    char c[20];
    scanf("%s", c);
    printf("%s.", c);
}
```

- scanf : đọc các ký tự vào c cho đến khi gặp khoảng trắng, enter, tab.

Nhập và viết giá trị: fgets

```
#include <stdio.h>
int main()
{
    char c[20];
    fgets(c, sizeof(c), stdin);  // read string
    printf("%s", c);
}
```

Chú ý : Lệnh **fgets** lưu vào chuỗi cả *ký tự* **enter** ('\n') dùng để kết thúc nhập chuỗi. VD : nhập chuỗi **abc** :



Chí ý: Nếu ngay trước lệnh *fgets* là lệnh *scanf* thì ta thực hiện như sau :

```
#include <stdio.h>
int main()
{
    char c[20];
    char kt;
    int n;
    scanf("%d", &n);
    scanf("%c", &kt);
    fgets(c, sizeof(c), stdin);
    printf("%s", c);
}
```

Hàm chuỗi : strlen, strcmp, strcpy, strcat

#include <**string.h**>

char c[20]="abcd ";

printf("%d", **strlen**(c)); → 4

char c1[20]="abcd", c2[30]="abcd";

printf("%d", **strcmp**(c1, c2)); → 0

char c1[20]="abc**d** ", c2[30]="abc**e**";

printf("%d", **strcmp**(c1, c2)); → khác 0

Khi so sánh:

- có tính khoảng trắng,
- có phân biệt chữ in, chữ thường.

Hàm chuỗi : strlen, strcmp, strcpy, strcat

#include <**string.h**>

char c1[20]=″abcd″;

strcpy(c2, c1);

printf(“%s”, c2); → abcd

Nhập và viết giá trị: cin.getline

```
#include <iostream>
int main()
{
    char c[20];
    std::cin.getline(c, 10);  // read string
    std::cout << c;
}
```

Chú ý : Lệnh **cin.getline** không lưu *ký tự* **enter** ('\n') vào chuỗi (dùng để kết thúc nhập chuỗi). VD : nhập chuỗi **abc**