



28TECH
Become A Better Developer



XÂU KÍ TỰ



Giới thiệu về xâu kí tự:



Trong ngôn ngữ lập trình C, để biểu diễn **một kí tự** ta dùng **kiểu dữ liệu char**, để có thể lưu nội dung là **một chuỗi** kí tự, một đoạn văn, tên người,... thì ta phải dùng **mảng kí tự**.



Để sử dụng các hàm xử lý **xâu và kí tự** trong ngôn ngữ lập trình C, các bạn khai báo 2 thư viện là **string.h** và **ctype.h**.



1. Khai báo mảng kí tự:

CÁCH KHAI BÁO MẢNG KÍ TỰ

Gán nội dung cho mảng kí tự bằng một chuỗi.

Liệt kê từng kí tự trong chuỗi. Kí tự null ('\0') đánh dấu kết thúc chuỗi.

```
#include <stdio.h>

int main(){
    char c[] = "28tech";
    char d[] = {'2', '8', 't', 'e', 'c', 'h', '\0'};
    return 0;
}
```

2. Nhập và xuất mảng kí tự:

●●●
Nhập xâu **không có dấu cách**, bạn sử dụng **scanf** với đặc tả là **"%s"**. Đối với mảng kí tự, bạn thường **khai báo sẵn kích thước** của nó, vì thế hãy chú ý tới giới hạn của đề bài.

Nhập xâu không có dấu cách:

```
#include <stdio.h>

int main(){
    char c[1000];
    printf("Nhap xau ki tu :");
    scanf("%s", c);
    printf("Xau ki tu vua nhap : %s", c);
    return 0;
}
```

INPUT

28tech

OUTPUT

Xau ki tu vua nhap : 28tech

2. Nhập và xuất mảng kí tự:

●●●
Nhập xâu có dấu cách, bạn sử dụng hàm **gets** hoặc **fgets**, nếu hàm **gets** ở chuẩn C các bạn đang sử dụng không sử dụng được thì bạn thay bằng **fgets**.

Nhập xâu có dấu cách:

```
#include <stdio.h>

int main(){
    char c[1000];
    printf("Nhap xau ki tu :");
    gets(c);
    printf("Xau ki tu vua nhap : %s", c);
    return 0;
}
```

INPUT

28tech dev

OUTPUT

Xau ki tu vua nhap : 28tech dev

2. Nhập và xuất mảng kí tự:

● ● ●
Nhập xâu bằng fgets, **hàm fgets có 3 tham số**, tham số thứ 1 là xâu bạn muốn nhập nội dung vào, tham số thứ 2 là số lượng kí tự tối đa bạn muốn nhập cho xâu, tham số thứ 3 là luồng vào. Nếu bạn nhập từ bàn phím thì luồng vào sẽ là stdin.

Nhập xâu có dấu cách:

```
#include <stdio.h>

int main(){
    char c[1000];
    printf("Nhap xau ki tu :");
    fgets(c, 1000, stdin);
    printf("Xau ki tu vua nhap : %s", c);
    return 0;
}
```

INPUT

28tech dev

OUTPUT

Xau ki tu vua nhap : 28tech dev



2. Nhập và xuất mảng kí tự:

Trước câu lệnh gets hoặc fgets nếu bạn sử dụng câu lệnh scanf thì **sẽ bị trôi** vì 2 hàm này **dừng nhập tới khi gặp dấu xuống dòng**. Scanf thì sẽ không xử lý kí tự enter mà bạn nhập mà để lại trong bộ đệm bàn phím, dẫn tới những lệnh gets, fgets dưới nó đọc phải kí tự enter này và sẽ kết thúc ngay.

Tình huống bị trôi lệnh:

```
#include <stdio.h>

int main(){
    int n;
    scanf("%d", &n); // sẽ để lại kí tự enter
    char c[1000];
    printf("Nhap xau ki tu :");
    gets(c); //Đọc phải kí tự enter và kết thúc ngay
    printf("Xau ki tu vua nhap : %s", c);
    return 0;
}
```

INPUT

100

OUTPUT

Nhap xau ki tu :Xau ki tu vua nhap :

2. Nhập và xuất mảng kí tự:

Cách xử lý: Trước khi sử dụng gets hoặc fgets hãy đảm bảo rằng **trước đó không còn kí tự enter nào trong bộ đệm bàn phím**. Các bạn có thể dùng **getchar** để **đọc một kí tự từ bàn phím** nhằm xóa đi kí tự enter này hoặc sử dụng `scanf("\n")` nếu trước gets có nhiều dấu cách rồi mới tới kí tự enter.

Xử lí tình huống bị trôi lệnh:

```
#include <stdio.h>

int main(){
    int n; scanf("%d", &n);
    getchar(); // đọc kí tự enter do scanf để lại
    char c[1000];
    printf("Nhap xau ki tu :");
    gets(c); // Sẽ đợi bạn nhập và ấn enter
    printf("Xau ki tu vua nhap : %s", c);
    return 0;
}
```

INPUT

100
28tech dev

OUTPUT

Xau ki tu vua nhap : 28tech dev

2. Nhập và xuất mảng kí tự:

Trong trường hợp bạn scanf sau đó cố tính ấn nhiều dấu cách rồi mới enter ta xử lý bằng cách sau.

Cách xử lí:

```
#include <stdio.h>

int main(){
    int n;
    scanf("%d", &n); //bạn có thể nhập n và nhiều dấu cách
    while(getchar() != '\n');
    char c[1000];
    printf("Nhap xau ki tu :");
    gets(c);
    printf("Xau ki tu vua nhap : %s", c);
    return 0;
}
```

2. Nhập và xuất mảng kí tự:

Chú ý: Nếu trước gets là gets thì bạn không cần xóa kí tự enter thừa vì gets hay fgets đều không để lại kí tự enter như scanf.

Ví dụ:

```
#include <stdio.h>

int main(){
    int n; scanf("%d", &n);
    while(getchar() != '\n');
    char c[1000];
    gets(c);
    char d[1000];
    gets(d); //Không cần xóa kí tự enter
    printf("%s\n%s", c, d);
    return 0;
}
```

3. Hàm length:



Đối với các bài toán về mảng 1 chiều các số nguyên thì số lượng phần tử trong mảng thường được cho biết trước còn đối với chuỗi ký tự thì không. Vì thế ta cần sử dụng hàm để biết số lượng ký tự trong mảng thực sự được lưu là bao nhiêu.

Ví dụ:

```
#include <stdio.h>
#include <string.h>

int main(){
    char c[1000];
    gets(c);
    printf("Chieu dai cua xau : %d", strlen(c));
    return 0;
}
```

INPUT

28tech dev

OUTPUT

Chieu dai cua xau : 10



3. Hàm length:



Sự khác nhau giữa gets và fgets: fgets sẽ đọc cả kí tự enter và cho vào xâu, ngược lại gets thì không.

Ví dụ:

```
#include <stdio.h>
#include <string.h>

int main(){
    char c[100], d[100];
    gets(c);
    fgets(d, 100, stdin);
    printf("%d %d", strlen(c), strlen(d));
    return 0;
}
```

INPUT

28tech
28tech

OUTPUT

6 7



3. Hàm length:



Cách loại bỏ kí tự enter nếu bạn dùng fgets đó là cho kí tự đó thành kí tự null.

Ví dụ:

```
#include <stdio.h>
#include <string.h>

int main(){
    char c[100], d[100];
    gets(c);
    fgets(d, 100, stdin);
    d[strlen(d) - 1] = '\0';
    printf("%d %d", strlen(c), strlen(d));
    return 0;
}
```

INPUT

28tech
28tech

OUTPUT

6 6



4. Duyệt chuỗi:



Bạn có thể duyệt qua từng ký tự của chuỗi thông qua chỉ số

Ví dụ:

```
#include <stdio.h>
#include <string.h>

int main(){
    char c[] = "28techdev";
    for(int i = 0; i < strlen(c); i++){
        printf("%c ", c[i]);
    }
    return 0;
}
```

OUTPUT

2 8 t e c h d e v



5. Nhắc lại các hàm kiểm tra kí tự:



Ở bài if else các bạn đã học cách tự viết các câu lệnh if để kiểm tra loại kí tự, bây giờ các bạn có thể sử dụng các hàm có sẵn trong thư viện `<ctype.h>`.

Hàm	Chức năng
<code>isdigit(char c)</code>	Kiểm tra chữ số
<code>islower(char c)</code>	Kiểm tra chữ in thường
<code>isupper(char c)</code>	Kiểm tra in hoa
<code>isalpha(char c)</code>	Kiểm tra chữ cái
<code>int tolower(char c)</code>	Chuyển thành chữ in thường
<code>int toupper(char c)</code>	Chuyển thành chữ in hoa



6. Hàm strlwr và strupr:

a. Hàm strlwr:

Cú pháp:

```
char* strlwr(char[]);
```



Chức năng: Chuyển chuỗi kí tự về dạng in thường, ngoài ra còn trả về con trỏ tới mảng này sau khi chuyển đổi.



Chú ý: Hàm này có thể không còn được sử dụng ở một số chuẩn C nên bạn hãy tự xây dựng nó trong trường hợp không thể dùng.

Ví dụ:

```
#include <stdio.h>
#include <string.h>

int main(){
    char c[] = "28TECH";
    strlwr(c);
    printf("%s", c);
    return 0;
}
```

OUTPUT

28tech



6. Hàm strlwr vàstrupr:

a. Hàm strlwr:

Xây dựng hàm strlwr:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void lower(char c[]){
    for(int i = 0; i < strlen(c); i++){
        c[i] = tolower(c[i]);
    }
}

int main(){
    char c[] = "28TECHDEV";
    lower(c);
    printf("%s", c);
    return 0;
}
```

OUTPUT

28techdev

6. Hàm strlwr và strupr:

b. Hàm strupr:

Cú pháp:

```
char* strupr(char[]);
```



Chức năng: Chuyển chuỗi kí tự về dạng in hoa, ngoài ra còn trả về con trỏ tới mảng này sau khi chuyển đổi.



Chú ý: Hàm này có thể không còn được sử dụng ở một số chuẩn C nên bạn hãy tự xây dựng nó trong trường hợp không thể dùng.

Ví dụ:

```
#include <stdio.h>
#include <string.h>

int main(){
    char c[] = "28tech";
    strupr(c);
    printf("%s", c);
    return 0;
}
```

OUTPUT

28TECH



6. Hàm strlwr vàstrupr:

b. Hàmstrupr:

Xây dựng hàmstrupr:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void upper(char c[]){
    for(int i = 0; i < strlen(c); i++){
        c[i] = toupper(c[i]);
    }
}

int main(){
    char c[] = "28techdev";
    upper(c);
    printf("%s", c);
    return 0;
}
```

OUTPUT

28TECHDEV

7. Hàm strcmp:

Cú pháp:

```
int strcmp(char a[], char b[]);
```



Chức năng: Hàm này dùng để so sánh 2 xâu theo thứ tự từ điển.



Giá trị trả về:

- Nếu $a < b$ trả về -1.
- Nếu $a > b$ trả về 1.
- Nếu $a = b$ trả về 0.

Ví dụ:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(){
    char c[] = "28techdev";
    char d[] = "28ztechdev";
    char e[] = "28techdev";
    printf("%d\n", strcmp(c, d));
    printf("%d\n", strcmp(c, e));
    printf("%d\n", strcmp(d, e));
    return 0;
}
```

OUTPUT

-1 0 1



8. Hàm strcat:

Cú pháp:

```
char* strcat(char đích[], char nguồn[]);
```



Chức năng: Nối xâu nguồn vào xâu đích.

Ví dụ:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(){
    char c[] = "28tech ";
    char d[] = "developer";
    strcat(c, d);
    printf("%s", c);
    return 0;
}
```

OUTPUT

28tech developer



9. Hàm strcpy:

Cú pháp:

```
char* strcpy(char dich[], char nguon[]);
```



Chức năng: Copy nội dung từ chuỗi nguồn vào chuỗi đích, chú ý rằng trước khi copy, toàn bộ nội dung của chuỗi đích sẽ bị xóa hết.

Ví dụ:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(){
    char c[] = "28tech ";
    char d[] = "developer";
    strcpy(c, d);
    printf("%s", c);
    return 0;
}
```

OUTPUT

developer



10. Hàm strrev:

Cú pháp:

```
char* strrev(char c[]);
```



Chức năng: Lật ngược xâu c.

Ví dụ:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(){
    char c[] = "28tech";
    strrev(c);
    printf("%s", c);
    return 0;
}
```

OUTPUT

hcet82



11. Hàm strstr:

Cú pháp:

```
char* strstr(char c[], char d[]);
```



Chức năng: Trả về con trỏ tới vị trí xuất hiện đầu tiên của chuỗi d trong chuỗi c, nếu chuỗi d không xuất hiện trong chuỗi c nó trả về con trỏ NULL. Hàm này được dùng để kiểm tra sự tồn tại của chuỗi con.

Ví dụ:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>


int main(){
    char c[] = "28tech_dev";
    char d[] = "tec";
    char *ptr = strstr(c, d);
    if(ptr == NULL){
        printf("NOT FOUND\n");
    }
    else{
        printf("%s", ptr);
    }
    return 0;
}
```

OUTPUT

tech_dev



12. Hàm memset:

 **Chức năng:** Hàm memset được dùng để **gán giá trị cho các ô nhớ**, các bạn có thể dùng hàm này để gán giá trị 0 hoặc -1 cho mảng int.

Ví dụ:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(){
    int a[] = {1, 2, 3, 4, 5};
    memset(a, 0, sizeof(a));
    for(int i = 0; i < 5; i++){
        printf("%d ", a[i]);
    }
    return 0;
}
```

OUTPUT

0 0 0 0 0

Ví dụ:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(){
    int a[] = {1, 2, 3, 4, 5};
    memset(a, -1, sizeof(a));
    for(int i = 0; i < 5; i++){
        printf("%d ", a[i]);
    }
    return 0;
}
```

OUTPUT

-1 -1 -1 -1 -1



13. Hàm atoi, atoll:



Chức năng: chuyển đổi các chuỗi ký tự sang số tương ứng.



Trong đó atoi dùng để chuyển 1 chuỗi ký tự sang số int, atoll dùng để chuyển 1 chuỗi ký tự sang số long long.

Ví dụ:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(){
    char c[] = "0123";
    int n = atoi(c);
    printf("%d\n", n);
    char d[] = "001231823812831823";
    long long m = atoll(d);
    printf("%lld\n", m);
}
```

OUTPUT

123

1231823812831823



14. Hàm strtok:



Hàm strtok có chức năng tách các từ trong chuỗi theo một ký tự chỉ định cho trước, thông thường ta sẽ dùng strtok để tách các từ trong chuỗi theo dấu cách

Ví dụ:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(){
    char c[] = "28tech    developer GUI    become    dev";
    char *token;
    token = strtok(c, " ");
    while(token != NULL){
        printf("%s\n", token);
        token = strtok(NULL, " ");
    }
    return 0;
}
```

OUTPUT

```
28tech
developer
GUI
become
dev
```



14. Hàm strtok:



Hàm strtok còn có thể được dùng để tách theo nhiều kí tự khác nhau.

Ví dụ:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(){
    char c[] = "28tech/developer GUI-become-dev";
    char *token;
    token = strtok(c, " -/");
    while(token != NULL){
        printf("%s\n", token);
        token = strtok(NULL, " -/");
    }
    return 0;
}
```

OUTPUT

```
28tech
developer
GUI
become
dev
```



15. Các bài toán liên quan tới tần suất xuất hiện của kí tự trong xâu:



Bài toán: Đếm số lần xuất hiện của các kí tự trong xâu sau đó liệt kê theo thứ tự từ điển tăng dần



Để đếm tần suất xuất hiện của các kí tự xuất hiện trong xâu các bạn có thể **sử dụng mảng để đếm**, vì các kí tự thường gặp đều có mã ASCII từ 0 tới 255 nên sử dụng **mảng đếm có 256 phần tử** là có thể **đếm được** kí tự xuất hiện trong xâu.



15. Các bài toán liên quan tới tần suất xuất hiện của kí tự trong xâu:

Chương trình mẫu

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(){
    char c[] = "28tech_dev";
    int cnt[256] = {0};
    for(int i = 0; i < strlen(c); i++){
        cnt[c[i]]++;
    }
    for(int i = 0; i < 256; i++){
        if(cnt[i]){
            printf("%c %d\n", i, cnt[i]);
        }
    }
    return 0;
}
```

OUTPUT

```
2 1
8 1
_ 1
c 1
d 1
e 2
h 1
t 1
v 1
```

16. Xử lý số lớn:



Đối với các bài toán mà số lượng chữ số của số đầu bài cho lên tới hàng nghìn, triệu chữ số thì bạn cần dùng mảng kí tự để lưu.

Tính tổng chữ số của một số nguyên có tối đa 1000 chữ số

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(){
    char c[1000]; gets(c);
    int sum = 0;
    for(int i = 0; i < strlen(c); i++){
        sum += c[i] - '0'; // ???
    }
    printf("%d", sum);
    return 0;
}
```

