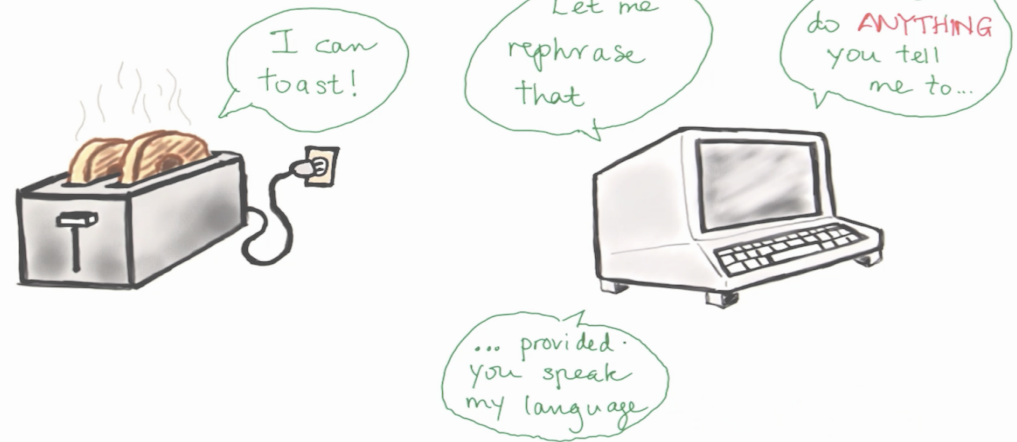


What is programming?



Nội dung 11

Chuỗi ký tự động (Dynamic String)

CT101 – Lập trình căn bản

Mục đích

Nhằm giới thiệu khái niệm “chuỗi động”,
mối quan hệ giữa con trỏ và chuỗi,
và sử dụng chuỗi động trong C

Yêu cầu

- ☐ Hiểu được **khái niệm** chuỗi động
- ☐ Hiểu **tại sao** phải dùng chuỗi động
- ☐ Biết cách **tạo và sử dụng** chuỗi động:
 - ☐ Biết khai báo chuỗi động
 - ☐ Hiểu khái niệm sao chép nông, sao chép sâu
 - ☐ Biết cách thay đổi chiều dài chuỗi
 - ☐ Biết tạo mảng các chuỗi động
- ☐ Biết cách tránh một số **lỗi thường gặp** trong sử dụng chuỗi động.

Nội dung

1. Tại sao cần chuỗi “động”?
2. Khai báo và sử dụng chuỗi động
 1. Khai báo và truy xuất chuỗi động
 2. Sao chép chuỗi
 3. Thay đổi chiều dài chuỗi
 4. Tạo mảng các chuỗi động (mảng các `char*`)
3. Các lỗi thường gặp khi sử dụng chuỗi
 1. Sử dụng chuỗi chưa cấp phát bộ nhớ
 2. Vấn đề với `scanf()` và `gets()`
 3. Gán giá trị vượt quá chiều dài chuỗi
4. Một số ví dụ về xử lý chuỗi (phụ lục)

1. Tại sao cần chuỗi động?

Chuỗi là gì?

- Chuỗi là một **mảng các ký tự**.
- Trong C, chuỗi ký tự luôn được **kết thúc bằng ‘\0’**

H	E	L	L	O		W	O	R	L	D	\0
---	---	---	---	---	--	---	---	---	---	---	----

- Khai báo: `char str[max_length];`
- Ví dụ: `char str[12]; strcpy(str, "HELLO");`

H	E	L	L	O	\0						
---	---	---	---	---	----	--	--	--	--	--	--

- Chuỗi \equiv mảng các ký tự \equiv hằng con trỏ

```
char str[10] = {'a', 'b', 'c', '\0'};
```

Vấn đề đối với chuỗi “thường”

- Viết một chương trình lấy ý kiến sinh viên.

- Cấu trúc dữ liệu:

- Câu hỏi:

- 17 câu
- 60 – 180 ký tự

- Ý kiến khác:

- 1 câu
- 33/80 sinh viên
- 0 – 140 ký tự

```
typedef struct {  
    char ques[xxx];  
    int ans;  
} Question;  
  
struct Feedback {  
    char mssv[12];  
    Question qList[17];  
    char comment[xxx];  
};
```

- Vậy, giá trị **xxx** là bao nhiêu là tối ưu?

- Nhỏ: tiết kiệm bộ nhớ nhưng có thể thiếu.
- Lớn: tránh thiếu bộ nhớ nhưng có thể hao phí.



Giải pháp

- Sử dụng kỹ thuật **cấp phát động** để tạo **chuỗi có kích thước có thể thay đổi**:
 - Chuỗi: như là một **mảng động** các ký tự
⇒ Cấp phát bộ nhớ cho mảng khi đã biết chiều dài chuỗi
 - Mảng động = con trỏ + cấp phát vùng nhớ động

```
typedef struct {  
    char ques[xxx];  
    int ans;  
} Question;
```

```
struct Feedback {  
    Question qList[17];  
    char comment[xxx]; };
```

```
typedef struct {  
    char *ques;  
    int ans;  
} Question;
```

```
struct Feedback {  
    char mssv[12];  
    Question qList[17];  
    char *comment;  
};
```


2. Khai báo & sử dụng chuỗi động

Khai báo chuỗi động – Cú pháp

- Chuỗi động: chuỗi với **chiều dài có thể thay đổi**.

- Khai báo chuỗi: như là 1 con trỏ

```
char *s;
```



- Khi đã có kích thước dữ liệu: cấp phát động bộ nhớ

```
s = (char*) malloc (n) ;
```



Lưu ý: phải cấp phát bộ nhớ cho **ký tự k/thức chuỗi**.

- Truy xuất chuỗi giống như chuỗi bình thường
- Phải thu hồi vùng nhớ cấp phát cho chuỗi khi không còn sử dụng.

Khai báo chuỗi động – Ví dụ

- Tạo câu hỏi** trong chương trình lấy ý kiến SV.

```
#define N 2

typedef struct {
    char *ques;
    int ans;
} Question;
```

```
void taoCauhoi(Question qList[], int n) {
    char temp[200]; int i = 0;
    for (i = 0; i < N; i++) {
        printf("Cau hoi %d: ", i+1);
        gets(temp);
        int len = strlen(temp) + 1;
        qList[i].ques = (char*)malloc(len);
        strcpy(qList[i].ques, temp);
    }
}
```

```
int main() {
    Question q[N]; int i = 0;
    taoCauhoi(q, N);
    printf("\n*DS cau hoi:\n");
    for (i = 0; i < N; i++)
        printf("%s\n", q[i].ques);
}
```

```
D:\TCAN>ykien.exe
Cau hoi 1: Phuong phap danh gia hoc phan?
Cau hoi 2: Tac phong?

*DS cau hoi:
Phuong phap danh gia hoc phan?
Tac phong?
```



Chương trình còn vấn đề gì?

Khởi tạo chuỗi

- Ta có thể khai báo và k/tạo chuỗi động như sau:
`char *hoten = "Tran cong An";`
- Tuy nhiên, vùng nhớ chứa dữ liệu sẽ được khởi tạo trên **vùng nhớ chỉ đọc**.
 - Ví dụ, lệnh: `hoten[5] = 'C';` sẽ bị lỗi lúc thực thi

Cách khởi tạo	Nơi khởi tạo	Ghi chú
Mảng ký tự: <code>char s[] = "Hello";</code>	Vùng nhớ Stack	Ký tự '\0' sẽ tự động được thêm vào cuối chuỗi. Có thể t/đổi giá trị của <code>s</code> .
Con trỏ ký tự, chỉ đọc: <code>char *s = "Hello";</code>	Vùng nhớ chỉ đọc	Không t/đổi được giá trị của <code>s</code> (trừ khi cấp phát vùng nhớ khác cho <code>s</code>)
Con trỏ ký tự: <code>char *s;</code> <code>s = strdup("...");</code>	Vùng nhớ heap	Có thể thay đổi giá trị <code>s</code> trỏ đến (<code>strdup()</code> tương đương <code>malloc()</code> + <code>strcpy()</code>)

Phép gán chuỗi (sao chép giá trị)

- Sao chép giá trị cho chuỗi động:
 - Sao chép cạn (shallow copy)
 - Dùng phép **gán** = hai con trỏ (chuỗi)
 - Cái được copy là **địa chỉ** (tham chiếu) của hai con trỏ
 - hai con trỏ trỏ vào cùng một vùng nhớ
 - hai con trỏ sẽ luôn có cùng giá trị (nếu không t/đổi địa chỉ trỏ tới)
 - Sao chép sâu (deep copy)
 - Dùng **hàm** `strcpy()` hoặc `strdup()`
 - Cái được sao chép là **dữ liệu**
 - hai con trỏ sẽ không còn quan hệ sau khi copy

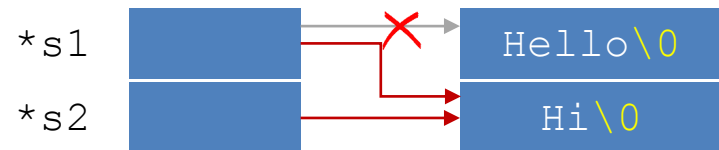
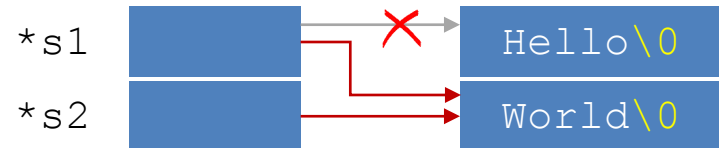
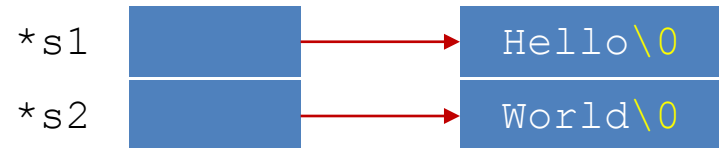
Sao chép cạn

```
int main() {
    char *s1 = strdup("Hello");
    char *s2 = strdup("World");

    s1 = s2; //gan: shallow copy
    printf("s1=%s\n", s1);
    printf("s2=%s\n", s2);

    strcpy(s2, "Hi");
    printf("s1=%s\n", s1);
    printf("s2=%s\n", s2);
}
```

```
D:\TCAN>shallow.exe
s1=World
s2=World
s1=Hi
s2=Hi
```



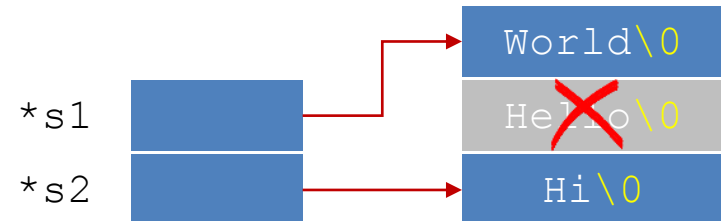
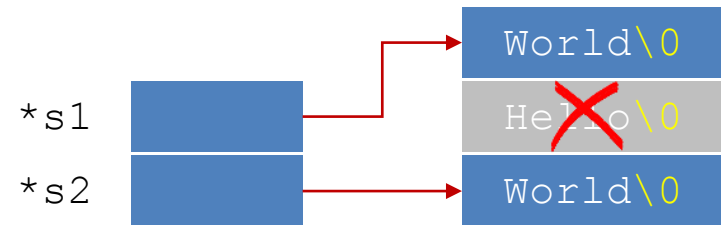
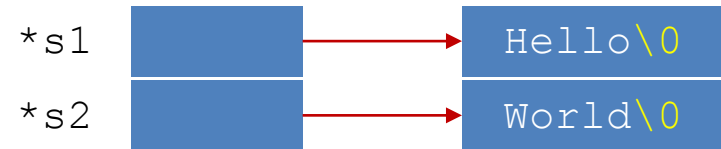
Sao chép sâu

```
int main() {
    char *s1 = strdup("Hello");
    char *s2 = strdup("World");

    free(s1);
    s1 = strdup(s2); //copy g/trị
    printf("s1=%s\n", s1);
    printf("s2=%s\n", s2);

    strcpy(s2, "Hi");
    printf("s1=%s\n", s1);
    printf("s2=%s\n", s2);
}
```

```
D:\TCAN>deep.exe
s1=World
s2=World
s1=World
s2=Hi
```



Thay đổi chiều dài chuỗi


- Sau khi đã cấp phát bộ nhớ và lưu trữ dữ liệu, giá trị của chuỗi có thể thay đổi
⇒ chiều dài có thể thay đổi.
- Ví dụ: trong chương trình quản lý SV có thể có chức năng **sửa tên SV** (nếu có nhập sai).
 - Cấp phát vùng nhớ mới
 - Nhập giá trị mới
 - Giải phóng vùng nhớ cũ



Thay đổi chiều dài chuỗi

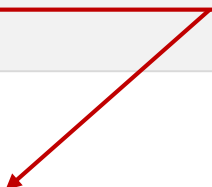
```
typedef struct {
    char *hoten;
    //các trường khác...
} Sinhvien;
```

```
void suaHoten1(Sinhvien *sv) {
    printf("Nhap ho ten: ");
    gets((*sv).hoten);
}
```



```
void nhapThongtin(Sinhvien *sv) {
    char temp[200];
    printf("Nhap ten SV: ");
    gets(temp);
    (*sv).hoten = strdup(temp);
    //nhập các trường khác...
}
```

```
void suaHoten2(Sinhvien *sv) {
    char temp[200];
    printf("Nhap ho ten: ");
    gets(temp);
    free((*sv).hoten);
    (*sv).hoten = strdup(temp);
}
```



```
(*sv).hoten = (char*)malloc(strlen(temp) + 1);
strcpy((*sv).hoten, temp);
```

Tạo mảng các chuỗi động

- Mảng có kích thước cố định:

```
char *dsTen[N];          //N: số phần tử của mảng
char *(dsTen2[N]);
dsTen[i] = strdup(temp); //cấp phát+gán giá trị
```

- Mảng động:

```
//khai báo
char **dsTen;

//cấp phát vùng nhớ N cho con trỏ char*
dsTen = (char**)malloc(sizeof(char*) * N);

//cấp phát + gán giá trị
dsTen[i] = strdup(temp);
for(i=0;i<N;i++) free(dsTen[i]);
free(dsTen);
```

Tạo mảng các chuỗi động

```
int main() {
    int soSV, i;
    char **dsTen; //char *(*dsTen);

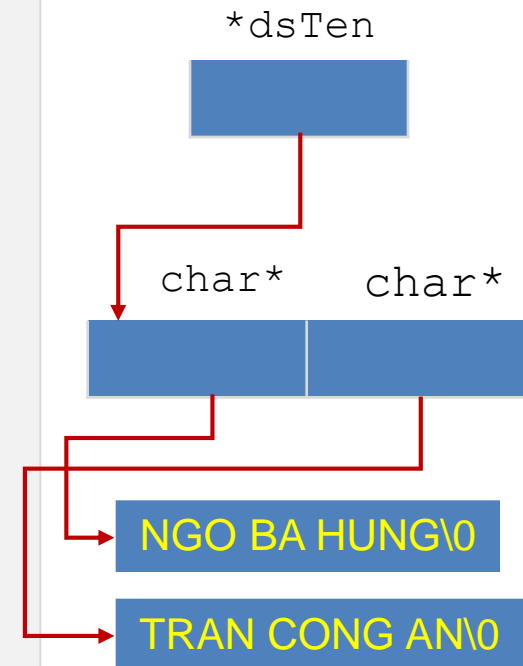
    printf("Nhap so SV: ");
    scanf("%d", &soSV);
    fflush(stdin);

    dsTen = (char**)malloc(sizeof(char*) * soSV);
    for (i = 0; i < soSV; i++) {
        char temp[200];
        printf("Nhap ten SV %d: ", i+1); gets(temp);
        dsTen[i] = strdup(temp);
    }

    for (i = 0; i < n; i++)
        printf("Sinh vien %d: %s\n", i+1, dsTen[i]);

    // ...
    for (i = 0; i < n; i++) free(dsTen[i]);
}
```

```
D:\TCAN>sinhvien.exe
Nhap so SV: 2
Nhap ten SV 1: Ngo Ba Hung
Nhap ten SV 2: Tran Cong An
Sinh vien 1: Ngo Ba Hung
Sinh vien 2: Tran Cong An
```



3. Các lỗi thường gặp

Sử dụng khi chưa cấp phát bộ nhớ

- Sử dụng con trỏ chưa được cấp phát bộ nhớ:

```
int main() {  
    char *s;  
    printf("Nhap s: ");  
    gets(s);  
}
```

- Lỗi này thường dẫn đến treo máy.
- Khắc phục:
 - Luôn khởi tạo các con trỏ: `char *s = NULL;`
(chưa g/quyết được vấn đề nhưng có thể tránh treo máy)
 - Luôn cấp phát bộ nhớ cho con trỏ trước khi truy xuất

Nhập chuỗi với `scanf()` và `gets()`

- Hai hàm này không k/tra số ký tự được phép nhập
 - Người dùng có thể nhập vượt quá chiều dài chuỗi
 - Dẫn đến các lỗi rất khó phát hiện.

```
int main() {  
    char s1[15], s2[15];  
    printf("Nhap s1: ");  
    scanf("%s", s1);  
    printf("Nhap s2: ");  
    scanf("%s", s2);  
    printf("s1=%s\n", s1);  
    printf("s2=%s\n", s2);  
}
```

```
D:\TCAN>comm_err.exe  
Nhap s1: 1234567890  
Nhap s2: 1234567890  
s1=1234567890  
s2=1234567890
```

```
D:\TCAN>comm_err.exe  
Nhap s1: 12345678901234567890  
Nhap s2: 12345678901234567890  
s1=7890  
s2=12345678901234567890
```

- Khắc phục: dùng hàm `fgets()`


```
char* fgets (char *str, int num, FILE *stream);
```

Gán giá trị vượt quá chiều dài chuỗi

- Thay đổi nội dung của chuỗi dài hơn so với giá trị trước đó đã được cấp phát bộ nhớ

```
int main() {  
    char *hoten, temp[200];  
    printf("Nhap ten: ");  
    gets(temp);  
    hoten = strdup(temp);  
  
    //...  
    printf("Sua ho ten: ");  
    gets(hoten);  
    //...  
}
```

```
int main() {  
    char *hoten, temp[200];  
    printf("Nhap ten: ");  
    gets(temp);  
    hoten = strdup(temp);  
  
    //...  
    printf("Sua ho ten: ");  
    gets(temp);  
    ???  
    hoten = strdup(temp);  
    //...  
}
```



- Khắc phục: cấp phát lại bộ nhớ khi d/liệu thay đổi

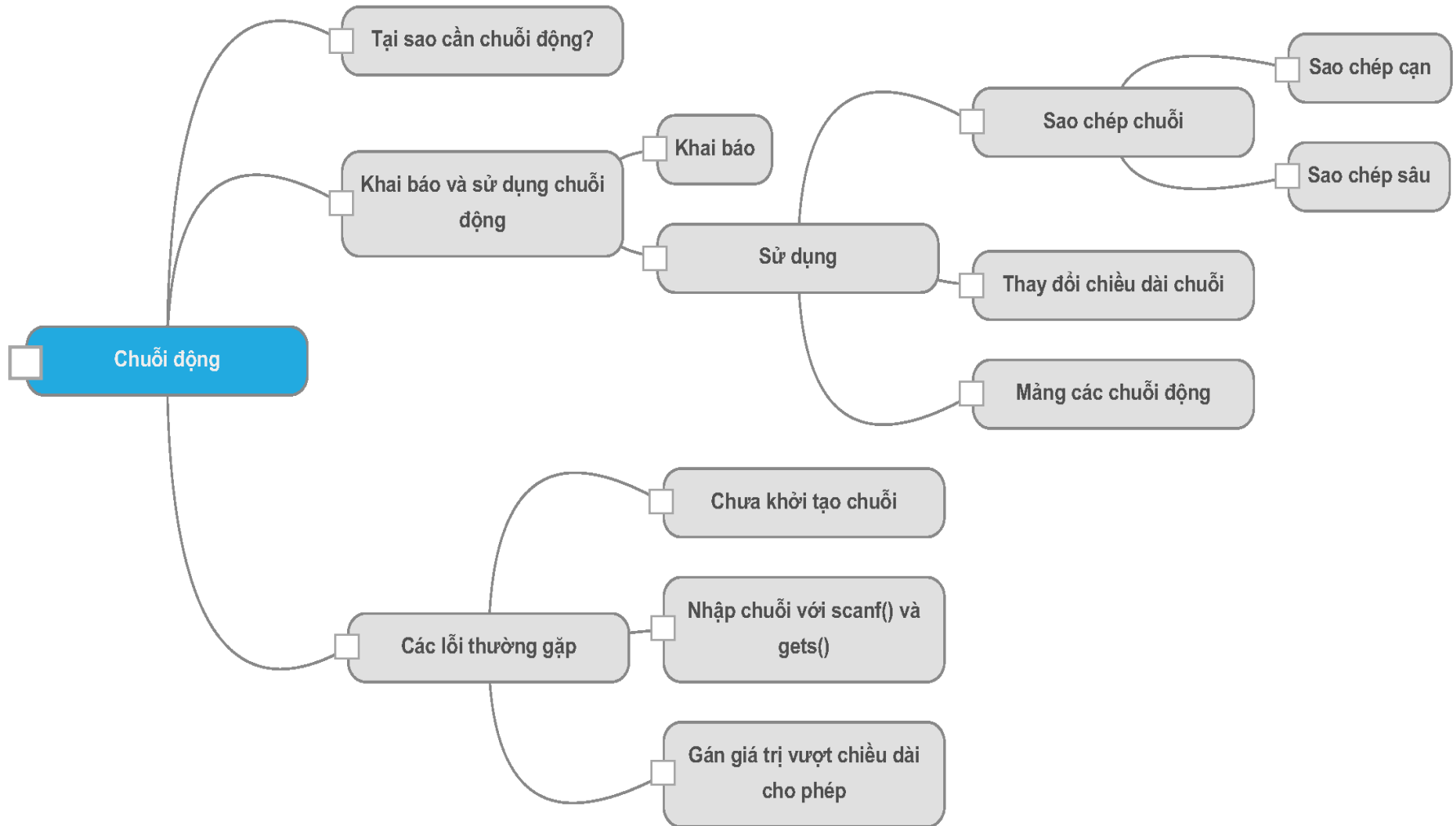
Tổng kết

- Chuỗi động: **kích thước có thể thay đổi** được.
- Chuỗi động về bản chất là một **mảng động**.
- Khai báo chuỗi động: `char *s;`
- Khởi tạo: `char *s = "...";` sẽ cấp phát vùng nhớ **chỉ đọc**.
- Thay đổi chiều dài chuỗi động: cấp phát lại vùng nhớ và sao chép giá trị
 - `strdup() = malloc() + strcpy()`
- Cấp phát vùng nhớ:
 1. Dùng hàm `malloc()` (chưa có dữ liệu)
 2. Dùng hàm `strdup()` (cấp phát + gán giá trị)

Tổng kết

- Gán/Sao chép chuỗi:
 - Sao chép cạn: **sao chép địa chỉ** con trỏ
 - Sao chép sâu: **sao chép dữ liệu** của chuỗi
- Mảng các chuỗi động: mảng các con trỏ
 - Mảng cố định: `char *dsTen[N];`
 - Mảng động: `char **dsTen;`
- Một số vấn đề cần lưu ý:
 - Phải cấp phát vùng nhớ cho chuỗi trước khi sử dụng
 - Phải lưu ý vấn đề kích thước dữ liệu khi sử dụng hàm `scanf()` và `gets()`
 - Phải lưu ý k/thước của chuỗi động trước khi gán giá trị
- Phải **giải phóng bộ nhớ** khi không còn s/dụng.

Check what you achieved.
Re-learn what you haven't mastered.



Kiểm tra kiến thức

1. Ký tự nào sau đây là ký tự kết thúc chuỗi?

- a) `'.'`
- b) `' '`
- c) `'\0'`
- d) `'\n'`

2. Hàm nào dùng để so sánh hai chuỗi?

- a) `compare()` ;
- b) `stringcompare()` ;
- c) `cmp()` ;
- d) `strcmp()` ;

Kiểm tra kiến thức

3. Khai báo chuỗi nào sau đây là chưa chính xác?

- a) `char *s1;`
- b) `char *s1 = "Hi";`
- c) `char s2[] = "Hi";`
- d) `char s2[2] = "Hi";`
- e) `char s3[] = "Hi\0";`
- f) `char s4[] = {'H', 'i', '\0'};`
- g) `char s5[] = {'H', 'i'};`
- h) `char s5[3] = {'H', 'i'};`
- i) `char *s6 = strdup("Hi");`
- j) `char *s7 = strcpy("Hi");`

Kiểm tra kiến thức

5. Cho hai câu khai báo sau:

```
char *p = "Sanjay"; char a[] = "Sanjay";
```

hãy chọn câu phát biểu **sai**:

- a) p là một non-const pointer trỏ tới một non-const string còn a là một const pointer trỏ tới một non-const string
- b) Có thể cho con trỏ p trỏ tới một chuỗi khác, còn a thì không mà chỉ có thể thay đổi giá trị của a.
- c) Trong cả hai trường hợp, ký tự '\0' sẽ được thêm vào cuối chuỗi.

Kiểm tra kiến thức

6. Chương trình hiển thị kết quả gì?

```
int main() {  
    char *str=NULL;  
    strcpy(str,"Hello World");  
    printf("%s",str);  
}
```

- a) Hello World
- b) Hello World\0
- c) NULLHello World
- d) Không hiển thị gì hết (hoặc bị lỗi)

Kiểm tra kiến thức

7. Hãy cho biết kết quả của chương trình sau:

```
int main() {  
    char *x="HELLO";  
    x+=3;  
    printf("%s",x);  
    ;  
}
```

- a) HELLO
- b) ELLO
- c) LLO
- d) LO

Kiểm tra kiến thức

8. Hãy cho biết kết quả của chương trình sau:

```
void myStrcat(char *a, char *b) {  
    int m = strlen(a);  
    int n = strlen(b);  
    int i;  
    for (i = 0; i <= n; i++)  
        a[m+i] = b[i];  
}
```

```
int main() {  
    char *str1 = "HELLO ";  
    char *str2 = "World";  
    myStrcat(str1, str2);  
    printf("%s ", str1);  
}
```

- a) HELLO world
- b) HELLOworld
- c) HwEoLrLIOD
- d) Chương trình bị lỗi.

Nếu chọn câu d) thì giải thích nguyên nhân.

Kiểm tra kiến thức

9. Hãy cho biết kết quả của chương trình sau:

```
void swap(char *str1, char *str2) {  
    char *temp = str1;  
    str1 = str2;  
    str2 = temp;  
}
```

```
int main() {  
    char *str1 = "Hello";  
    char *str2 = "World";  
    swap(str1, str2);  
    printf("str1 is %s, str2"  
        " is %s", str1, str2);  
}
```

- a) str1 is Hello, str2 is World
- b) str1 is World, str2 is Hello
- c) str1 is World, str2 is World
- d) str1 is Hello, str2 is Hello

Kiểm tra kiến thức

10. Hãy cho biết kết quả của chương trình sau:

```
int main() {  
    char str1[] = "Hello World";  
    char str2[] = {'H', 'e', 'l', 'l', 'o'};  
    int n1 = sizeof(str1)/sizeof(str1[0]);  
    int n2 = sizeof(str2)/sizeof(str2[0]);  
    printf("n1 = %d, n2 = %d", n1, n2);  
}
```

- a) $n1 = 10, n2 = 9$
- b) $n1 = 10, n2 = 10$
- c) $n1 = 9, n2 = 9$
- d) $n1 = 9, n2 = 10$

Kiểm tra kiến thức

11. Hãy cho biết kết quả của chương trình sau:

```
int main() {  
    char *s= "hello";  
    char *p = s;  
    printf("%c\t%c", p[0], s[1]);  
}
```

- a) Bị lỗi khi thực thi
- b) h h
- c) h e
- d) h l

Kiểm tra kiến thức

12. Câu lệnh nào sau đây là sai?

- a) `char str1[5] = "Texas";`
- b) `char str2[] = "A character string";`
- c) `char str3[2] = "A";`
- d) `char str4[2] = "TX";`

13. Cho một biến con trỏ `char *ptr_ch`, những câu lệnh nào sau đây là đúng?

- a) `*ptr_ch = 'a';`
- b) `ptr_ch = "A character string";`
- c) `ptr_ch = 'x';`
- d) `*ptr_ch = "This is Quiz 2.";`

Bài tập tổng kết

1. Viết hàm đảo chuỗi, dùng con trỏ để duyệt chuỗi.
2. Viết hàm tách từ
 - Input: chuỗi bất kỳ
 - Output: mảng các từ
3. Viết hàm alltrim() để cắt các khoảng trắng dư thừa giữa chuỗi (tham khảo 2 hàm trong phụ lục).
 - Input: Chuỗi bất kỳ
 - Output: Chuỗi đã cắt khoảng trắng dư thừa bên trong

Bài tập tổng kết

4. Viết chương trình lấy ý kiến SV như đã mô tả trong phần Đặt vấn đề.

- Một câu hỏi bao gồm số câu hỏi (qNo) và câu hỏi (qText).
- Một câu trả lời của SV bao gồm số câu hỏi (qNo) và trả lời (ans).
- Một feedback bao gồm 5 câu trả lời cho 5 câu hỏi và comment.
- Chức năng:
 - Nhập danh sách câu hỏi (tất cả các SV đều trả lời cùng các câu hỏi).
 - Nhập số SV, tạo danh sách các feedback cho các SV.
 - Nhập feedback: Hiển thị từng câu hỏi và nhận câu trả lời từ SV
 - Hiển thị feedback của SV: Bao gồm MSSV, các hỏi và câu trả lời tương ứng, comment nếu có.
 - Thêm một số chức năng như (làm thêm):
 - Chấm điểm feedback dựa vào ý kiến của SV.

Bài tập tổng kết

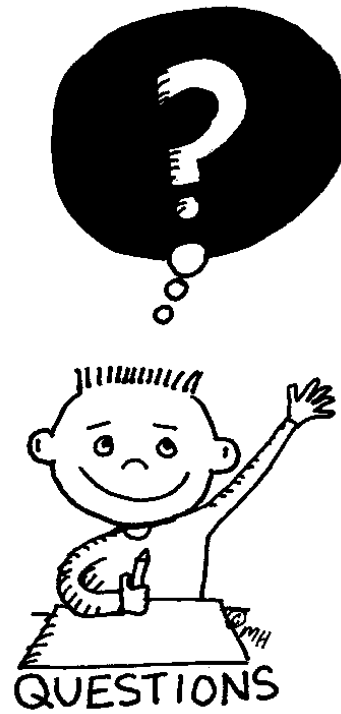
- Gợi ý:
 - Cấu trúc dữ liệu đề nghị như sau:

```
typedef struct {  
    int qNo;  
    char *qText;  
} Question;
```

```
typedef struct {  
    int qNo;  
    int ans;  
} Answer;
```

```
type struct {  
    char mssv[12];  
    Answer aList[5];  
    char *comment;  
} Feedback;
```

- Một số biến chính:
 - Danh sách câu hỏi (chung cho tất cả các SV):
`Question qList[5]; //5 câu hỏi`
 - Danh sách ý kiến của SV:
`Feedback *ykien; int soSV;`
`//tạo mảng ykien sau khi có số lượng SV`
`ykien = (Feedback*)malloc(sizeof(Feedback)*soSV);`



CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT

4. Phụ lục

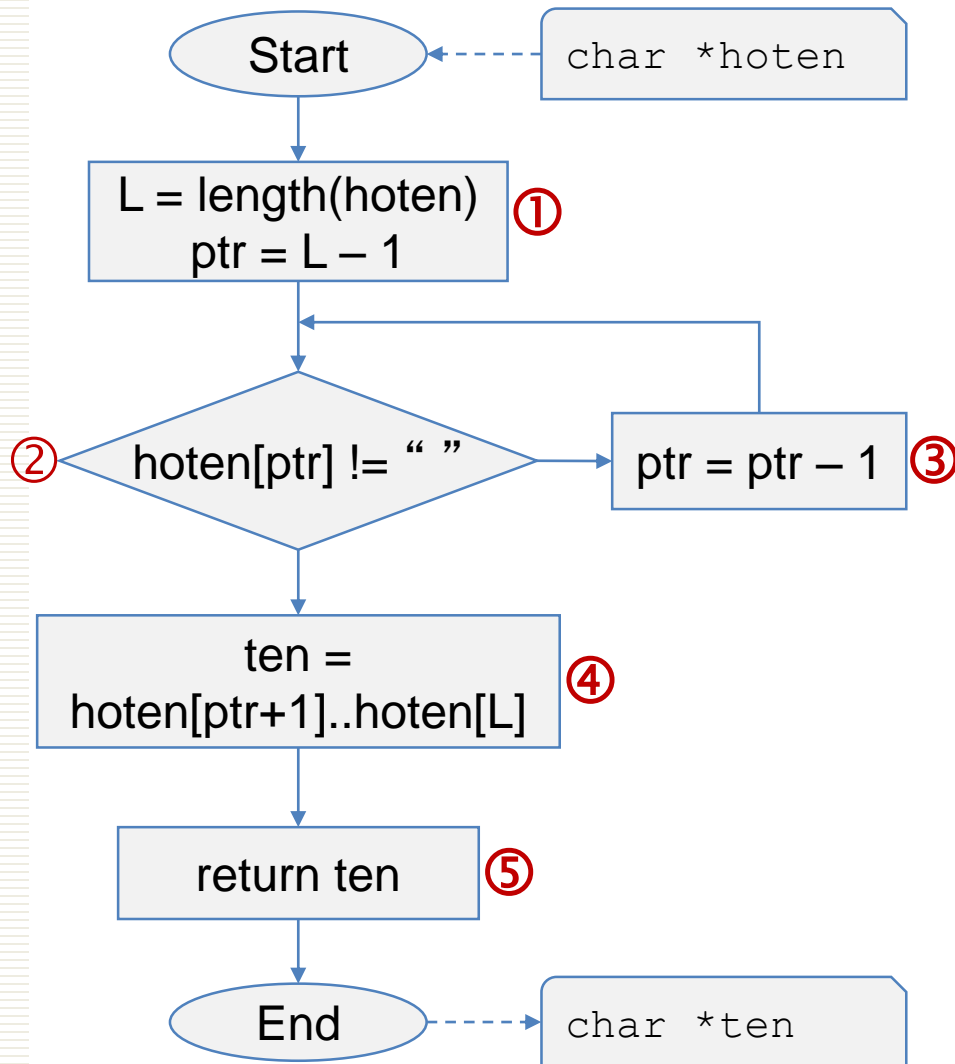
Một số ví dụ về xử lý chuỗi

Tách tên

- Bài toán:
 - Cho: một chuỗi lưu trữ họ tên
 - Yêu cầu: **trả về** tên trong chuỗi họ tên
- Giải thuật:
 - Giả thuyết:
 - chuỗi đầu vào có họ tên đầy đủ
 - tên không có khoảng trắng dư thừa
 - Mô tả phương pháp:
 - Bắt đầu từ cuối chuỗi
 - Dò ngược lại cho đến khi gặp ký tự khoảng trắng
 - Từ ký tự sau vị trí hiện tại đến cuối chuỗi là tên cần tìm



Tách tên



```

char* tachten(char* hoten) {
    char *ptr;
    int len = strlen(hoten);
    ptr = hoten + len - 1;
    while (*ptr != ' ' && ptr != hoten)
        ptr--;
    return (ptr + 1);
}
  
```

```

int main() {
    char hoten[]="Tran Cong An";
    char *ten = tachten(hoten);
    printf("Ten: %s", ten);
}
  
```

```

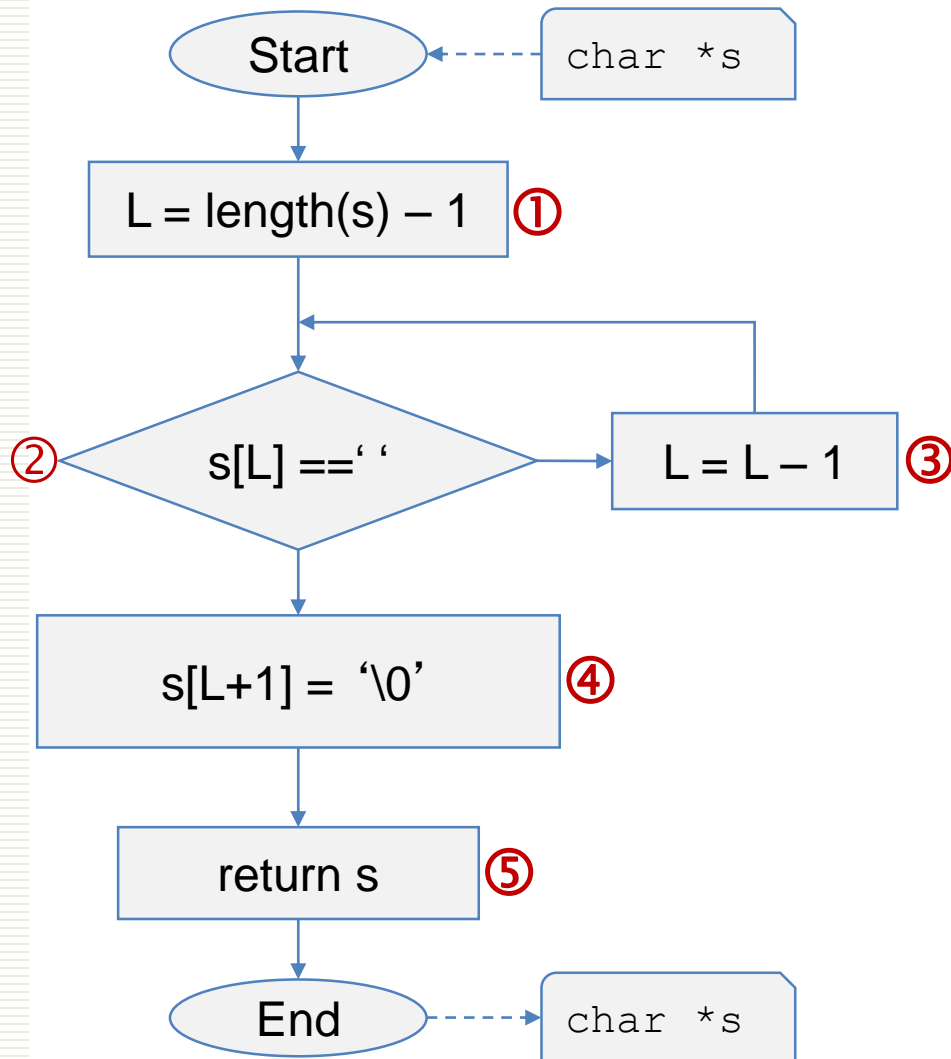
D:\TCAN>tachten.exe
Ten: An
  
```

Cắt khoảng trắng hai đầu chuỗi

- Bài toán: Cho một chuỗi bất kỳ
 - Yêu cầu: trả về chuỗi đã cắt bỏ khoảng trắng ở hai đầu
- Giải thuật:
 - Cắt khoảng trắng ở cuối chuỗi:
 - Bắt đầu từ cuối chuỗi
 - Dò ngược lên đầu chuỗi cho đến khi gặp ký tự != khoảng trắng
 - Đặt ký tự phía sau vị trí hiện tại là ký tự kết thúc chuỗi
 - Cắt khoảng trắng ở đầu chuỗi:
 - Bắt đầu từ đầu chuỗi
 - Dò ngược về cuối chuỗi cho đến khi gặp ký tự != khoảng trắng
 - Trả về chuỗi từ vị trí hiện tại đến cuối chuỗi.



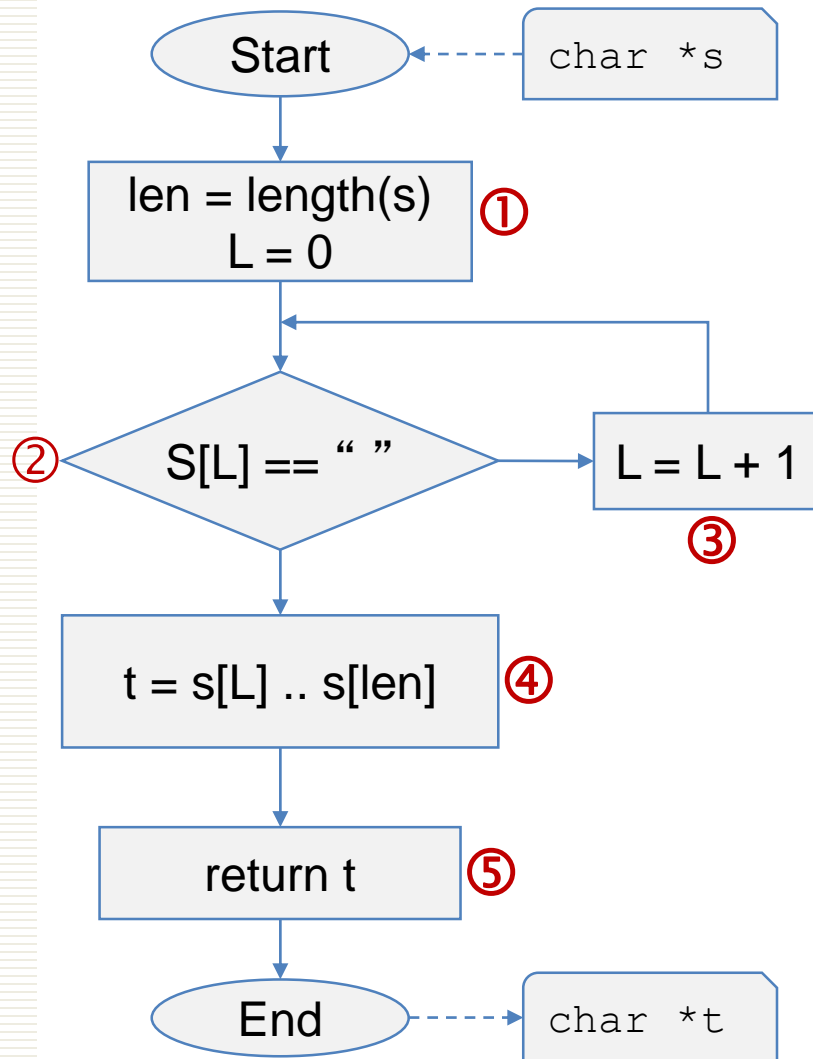
Cắt khoảng trắng ở cuối chuỗi



```

char* rTrim(char* s) {
    int L = strlen(s) - 1;
    while (s[L] == ' ' && L > 0)
        L--;
    s[L + 1] = '\\0';
    return (s);
}
  
```

Cắt khoảng trắng ở đầu chuỗi



```

char* lTrim(char* s) {
    while (s[0] == ' ') ①
        s++;           ②③④⑤
    return s;
}
  
```

```

int main() {
    char hoten[]=" Tran Cong An ";
    printf("Ho ten: [%s]\n", hoten);

    char *rt = lTrim2(hoten);
    printf("Left trim: [%s]\n", rt);

    rTrim(hoten);
    printf("Right trim: [%s]\n", rt);
}
  
```

```

D:\TCAN>rlTrim.exe
Ho ten: [   Tran Cong An   ]
Left trim: [Tran Cong An   ]
Right trim: [Tran Cong An]
  
```