I. Chuỗi trong C

1. Định nghĩa:

Trong ngôn ngữ lập trình C, chuỗi là một tập hợp các ký tự (**char**) được lưu trữ trên các ô nhớ liên tiếp và luôn luôn có 1 ký tự **null** là **\0** báo hiệu kết thúc chuỗi.

```
char s[] = "xin chao";
```

Như vậy, nếu bạn muốn khai báo chuỗi để lưu **n** ký tự, bạn cần mảng ký tự có kích thước tối đa ít nhất là **n+1**.

2. Khởi tạo

```
char a[] = "Full House chao moi nguoi";
char b[100] = "Full House chao moi nguoi";
char c[] = {'F','u','l','l',' ','H','o','u','s','e'};
char d[100] = {'F','u','l','l',' ','H','o','u','s','e'};
```

3. Cách nhập xuất

Chúng ta có thể sử dụng hàm **scanf** để nhập và hàm **printf** để xuất. Ở hàm **scanf** ta có thể bỏ đi dấu **&**. Chúng ta có thể nhập nhiều chuỗi một lúc trong hàm **scanf**.

```
3  int main(){
4      char s1[1000];
5      scanf("%s",&s1);
6  //  scanf("%s",a);
7      printf("%s\n",s1);
8
9      char s2[1000], s3[1000];
0      scanf("%s%s",s1,s2);
1      printf("%s\n%s",s1,s2);
2  }
```

4. Cách nhập chuỗi có khoảng trắng

a. Nhập bình thường

```
39 int main(){
40     char s1[1000], s2[1000];
41     fgets(s1,sizeof(s1),stdin);
42     gets(s2);
43
44     printf("%s%s\n",s1,s2);
45     puts(s2);
46 }
```

*# Hàm **gets** và hàm **fgets** đều là 2 hàm dùng để nhập chuỗi có chứa khoảng trắng nhưng hàm **fgets** sẽ đọc luôn kí tự **\n** còn hàm **gets** thì không.*

*# Hàm **gets** và hàm **fgets** đều là 2 hàm dùng để nhập chuỗi có chứa khoảng trắng nhưng hàm **fgets** sẽ đọc luôn kí tự **\n** còn hàm **gets** thì không.*

*# Hàm **printf** và hàm puts đều là 2 hàm dùng để in ra chuỗi nhưng hàm **puts** sẽ in ra thêm 1 dấu **\n** ở cuối và hàm này chỉ in duy nhất được 1 chuỗi còn hàm **printf** thì ngược lại.*

b. Lỗi không cho nhập chuỗi (getchar() và fflush(stdin);

- Chương trình lỗi

```
52  int main(){
53      float diem;
54      char Ho_ten[1000];
55
56      printf("Nhap diem: ");
57      scanf("%f",&diem);
58      printf("Nhap ho ten: ");
59      gets(Ho_ten);
60
61      printf("Ho ten: %s",Ho_ten);
62      printf("\nDiem: %.2f",diem);
63  }
```

Chương trình được fix

```
73  int main(){
74      float diem;
75      char Ho_ten[1000];
76
77      printf("Nhap diem: ");
78      scanf("%f",&diem);
79  //  fflush(stdin);
80      getchar();
81      printf("Nhap ho ten: ");
82      gets(Ho_ten);
83
84      printf("Ho ten: %s",Ho_ten);
85      printf("\nDiem: %.2f",diem);
86  }
```

Khi chúng ta nhập một biến trước khi sử dụng hàm **gets** hay **fgets** thì bộ nhớ sẽ lưu lại kí tự **\n**. Hàm **fflush(stdin)** và hàm **getchar()** đều dùng để xóa bộ nhớ đệm nhưng ta nên dùng hàm **getchar()**.

II. Một số hàm hỗ trợ

1. Hàm strlen().

- **Thư viện**: string.h
- **Mục đích**: Hàm **strlen()** trả về chiều dài của chuỗi, nó không đếm ký tự null '\0'.

Xây dựng hàm

```c
#include <stdio.h>

int strlen(char s[]){
    int i = 0;
    for(;s[i]!='\0';++i);
    return i;
}

int main(){
    char s[1000];
    gets(s);
    printf("%d",strlen(s));
}
```

Cú pháp: int strlen(const char* s)

- Ví dụ:

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

int main(){
    char s[1000];
    gets(s);
    printf("%d",strlen(s));
}
```

2. Hàm strcpy();

- **Thư viện**: string.h
- **Mục đích**: Sao chép nội dung của chuỗi source tới chuỗi destination. Đây là một hàm rất quan trọng khi xử lý chuỗi. Hàm strcpy chỉ sao chép dữ liệu từ mảng B[ ] sang mảng A[ ], nếu mảng A[ ] có kích thước nhỏ hơn mảng B[ ]thì sẽ xảy ra việc tràn dữ liệu của mảng A[ ].

Xây dựng hàm

```
135 char* strcpy(char s1[],char s2[]){
136     int i=0, len_s2 = strlen(s2);
137     for(;i<strlen(s2);++i){
138         s1[i] = s2[i];
139     }
140     s1[i]='\0';
141     return s1;
142 }
143
144 int main(){
145     char s1[1000], s2[1000];
146     gets(s1);
147     gets(s2);
148     strcpy(s1,s2);
149     printf("%s",s1);
150     printf("%s",strcpy(s1,s2));
151 }
```

- **Cú pháp**: char *strcpy(char *destination, const char *source)

- **Ví dụ:**

```
153  #include <stdio.h>
154  #include <string.h>
155
156  int main(){
157      char s1[1000],s2[1000];
158      gets(s1);
159      gets(s2);
160      strcpy(s1,s2);
161      printf("%s",s1);
162  //  printf("%s",strcpy(s1,s2));
163  }
```

3. Hàm strcat();

- **Thư viện**: string.h
- **Mục đích**: Dùng để nối 2 chuỗi. Kết quả được lưu vào chuỗi đầu tiên.
- **Xây dựng hàm:**

```
182 char* strcat(char s1[],char s2[]){
183     int i= 0, len_s1 = strlen(s1);
184     for(;i<strlen(s2);++i){
185         s1[len_s1+i] = s2[i];
186     }
187     s1[i+len_s1]='\0';
188     return s1;
189 }
190
191 int main(){
192     char s1[1000], s2[1000];
193     gets(s1);
194     gets(s2);
195     strcat(s1,s2);
196     printf("%s",s1);
197 //  printf("%s",strcat(s1,s2));
198 }
```

Cú pháp: char *strcat(char *des, const char *source)

- **Ví dụ:**

```
200  #include <stdio.h>
201  #include <string.h>
202
203  int main(){
204      char s1[1000],s2[1000];
205      gets(s1);
206      gets(s2);
207      strcat(s1,s2);
208      printf("%s",s1);
209  //  printf("%s",strcat(s1,s2));
210  }
211
```

4. Hàm strcmp()

- **Thư viện**: string.h
- **Mục đích**: Dùng để so sánh hai chuỗi với nhau. Hàm sẽ so sánh hai chuỗi với nhau và trả về các giá trị tương ứng.

  Ta có 2 mảng s1[] và s2[]

  Nếu s1[] > s2[], kết quả trả về là 1

  Nếu s1[] == s2[], kết quả trả về là 0

  Nếu s1[] < s2[], kết quả trả về là -1

- **Xây dựng hàm:**

```
226 int strcmp(char s1[],char s2[]){
227     int len_s1 = strlen(s1);
228     int len_s2 = strlen(s2);
229     int out;
230     for(int i=0;i<len_s1 && i<len_s2;++i){
231         out = s1[i]-s2[i];
232         if(out) {
233             if(out<0) return -1;
234             else if(out>0) return 1;
235         }
236     }
237     if(len_s1<len_s2) return -1;
238     else if(len_s1>len_s2) return 1;
239     return 0;
240 }
241
242 int main(){
243     char s1[1000], s2[1000];
244     gets(s1);
245     gets(s2);
246     printf("%d",strcmp(s1,s2));
```

- **Cú pháp:** int strcmp(const char*s1, const char *s2)

- **Ví dụ:**

```
249  #include <stdio.h>
250  #include <string.h>
251
252  int main(){
253      char s1[1000],s2[1000];
254      gets(s1);
255      gets(s2);
256      printf("%d",strcmp(s1,s2));
257  }
258
```

5. Hàm strchr()

- **Thư viện**: string.h
- **Mục đích**: Dùng để tìm kiếm sự xuất hiện đầu tiên của ki tự c trong chuỗi s1. Kết quả trả về của hàm là một con trỏ chỉ đến phần tử đầu tiên của chuỗi s1 có chứa kí tự c hoặc giá trị NULL nếu kí tự c không có trong chuỗi s1.

Xây dựng hàm

```
271 char* strchr(char s1[],char c){
272     int len_s1 = strlen(s1);
273     for(int i=0;i<len_s1;++i){
274         if(s1[i]==c){
275             return s1+i;
276         }
277     }
278     return NULL;
279 }
280
281 int main(){
282     char s1[1000], c;
283     gets(s1);
284     scanf("%c",&c);
285     if(strchr(s1,c)==NULL)
286         printf("Khong tim thay ki tu trong s1");
287     else printf("%s",strchr(s1,c));
288 }
```

- **Cú pháp**: char *strchr(const char *s1, char s2)

- **Ví dụ:**

```
290 #include <stdio.h>
291 #include <string.h>
292
293 int main(){
294     char s1[1000],c;
295     gets(s1);
296     scanf("%c",&c);
297     if(strchr(s1,c)==NULL)
298         printf("Khong tim thay chuoi s2 trong s1");
299     else printf("%s",strchr(s1,c));
300 }
```

6. Hàm strstr()

- **Thư viện**: string.h
- **Mục đích**: Dùng để tìm kiếm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1. Kết quả trả về của hàm là một con trỏ chỉ đến phần tử đầu tiên của chuỗi s1 có chứa chuỗi s2 hoặc giá trị NULL nếu chuỗi s2 không có trong chuỗi s1.

- **Cú pháp**: char *strstr(const char *s1, const char *s2)
- **Ví dụ:**

```
279 char* strstr(char s1[],char s2[]){
280     int len_s1 = strlen(s1);
281     int len_s2 = strlen(s2);
282     for(int i=0;i<=len_s1-len_s2;++i){
283         if(s1[i]==s2[0]){
284             int j=1, tmp = 1;
285             while(j<len_s2){
286                 if(s1[i+j]!=s2[j]){
287                     tmp =0;
288                     break;
289                 }
290                 ++j;
291             }
292             if(tmp) return s1+i;
293         }
294     }
295     return NULL;
296 }
297
298 int main(){
299     char s1[1000], s2[1000];
300     gets(s1);
301     gets(s2);
302     if(strstr(s1,s2)==NULL)
303         printf("Khong tim thay chuoi s2 trong s1");
304     else printf("%s",strstr(s1,s2));
305 }
```

- **Ví dụ:**

```
307  #include <stdio.h>
308  #include <string.h>
309
310  int main(){
311      char s1[1000],s2[1000];
312      gets(s1);
313      gets(s2);
314      if(strstr(s1,s2)==NULL)
315          printf("Khong tim thay chuoi s2 trong s1");
316      else printf("%s",strstr(s1,s2));
317  }
318
```

7. Hàm strupr()

- **Thư viện**: string.h
- **Mục đích**: Dùng để chuyển đổi chuỗi chữ thường thành chuỗi chữ hoa, kết quả trả về của hàm là một con trỏ chỉ đến địa chỉ chuỗi được chuyển đổi.
- **Xây dựng hàm:**

```
372 char* strupr(char s1[]){
373     int len_s1 = strlen(s1);
374     for(int i=0;i<len_s1;++i){
375         if(s1[i]>='a'&&s1[i]<='z'){
376             s1[i] -=32;
377         }
378     }
379     return s1;
380 }
381
382 int main(){
383     char s1[1000];
384     gets(s1);
385     printf("%s",strupr(s1));
386 }
```

- **Cú pháp**: char *strupr(char *s)

- Ví dụ:

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

int main(){
    char s1[1000];
    gets(s1);
    printf("%s",strupr(s1));
}
```

8. Hàm strlwr()

- **Thư viện**: string.h
- **Mục đích**: Muốn chuyển đổi chuỗi chữ hoa thành chuỗi toàn chữ thường, ta sử dụng hàm **strlwr**(), các tham số của hàm tương tự như hàm strupr().
- **Xây dựng hàm:**

```
407 char* strlwr(char s1[]){
408     int len_s1 = strlen(s1);
409     for(int i=0;i<len_s1;++i){
410         if(s1[i]>='A'&&s1[i]<='Z'){
411             s1[i] +=32;
412         }
413     }
414     return s1;
415 }
416
417 int main(){
418     char s1[1000];
419     gets(s1);
420     printf("%s",strlwr(s1));
421 }
```

- **Cú pháp:** char *strlwr(char *s)

- **Ví dụ:**

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

int main(){
    char s1[1000];
    gets(s1);
    printf("%s",strlwr(s1));
}
```

9. Hàm strrev()

- **Thư viện:** string.h
- **Mục đích:** Hàm **strrev**(string) trả về một chuỗi được đảo ngược
- **Xây dựng hàm:**

459 }

- **Cú pháp:** char *strrev(char *s)

```
445 char* strrev(char s1[]){
446     int len_s1 = strlen(s1);
447     for(int i=0;i<len_s1/2;++i){
448         char tmp = s1[i];
449         s1[i] = s1[len_s1-i-1];
450         s1[len_s1-i-1] = tmp;
451     }
452     return s1;
453 }
454
455 int main(){
456     char s1[1000];
457     gets(s1);
458     printf("%s",strrev(s1));
459 }
```

- **Ví dụ:**

```
462 #include <stdio.h>
463 #include <string.h>
464
465 int main(){
466     char s1[1000];
467     gets(s1);
468     printf("%s",strrev(s1));
469 }
```

10. Hàm tolower() và hàm isupper();

	toupper()	islower()
Thư viện	ctype.h	ctype.h
Mục đích	Chuyển đổi các chữ cái hoa thành chữ cái thường	Kiểm tra chữ cái truyền vào có phải chữ cái hoa không
Cú pháp	int tolower(int c);	int isupper(int c);

- **Xây dựng hàm:**

```
int main(){
    char c;
    scanf("%c",&c);
    if(c>='A'&&c<='Z'){
        c += 32;
    }
    printf("%c",c);
}
```

- **Ví dụ:**

```
488  #include <ctype.h>
489
490  int main(){
491      char c;
492      scanf("%c",&c);
493      if(isupper(c)){
494          c = tolower(c);
495      }
496      printf("%c",c);
497  }
```

11.Hàm toupper() và hàm islower();

	toupper()	islower()
Thư viện	ctype.h	ctype.h
Mục đích	Chuyển đổi các chữ cái hoa thành chữ cái thường	Kiểm tra chữ cái truyền vào có phải chữ cái hoa không
Cú pháp	int toupper(int c);	int islower(int c);

- **Xây dựng hàm:**

```
505 int main(){
506     char c;
507     scanf("%c",&c);
508     if(c>='a'&&c<='z'){
509         c -= 32;
510     }
511     printf("%c",c);
512 }
```

- **Ví dụ:**

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

int main(){
    char c;
    scanf("%c",&c);
    if(islower(c)){
        c = toupper(c);
    }
    printf("%c",c);
}
```