Kỹ Thuật Lập Trình

Khang Q.H. Vo (*M.Sc.*)
FIT - IUH

CHUỖI KÝ TỰ - STRINGS

NỘI DUNG

1. Khai báo chuỗi
2. Nhập, xuất chuỗi
3. Các hàm thư viện ***string.h***
4. Một số hàm người dung định nghĩa
5. Bài tập

Declare/ Initialize a String

- **Static strings**: stored in data segment or stack segment

```
char  s1[21]; /* for a string of 20 characters*/
```

Initialize a string: NULL byte is automatically inserted.

```
char name[31] = "I am a student";
char name2[31] = {'H', 'e ', 'l', 'l', 'o', '\0' };
```

- **Dynamic strings:** Stored in the heap

```
char* S;
S = (char*) malloc( lengthOfString+1);
S = (char*) calloc( lengthOfString+1, sizeof(char));
```

Data Stored in a strings

- Each character in a string is stored as it's ASCII code.

```
/* string01.c-xem noi dung luu tru 1 chuoi */
#include <stdio.h>
#include <conio.h>
int main(){
    char S1[15]="ABC";
    char S2[15]= {'a','b','c','\0'};
    int i ;
    printf("Data luu tru cho S1:\n");
    for (i=0;i<15;i++) printf("%d ", S1[i]);
    printf("\n");
    printf("Data luu tru cho S2:\n");
    for (i=0;i<15;i++) printf("%d ", S2[i]);
    getch();
    return 0;
}
```

S1[i]: The character at the position i in the string S1

```
G:\GiangDay\FU\PFC\PFC_Lab\stri...
Data luu tru cho S1:
65 66 67 0 0 0 0 0 0 0 0 0 0 0 0
Data luu tru cho S2:
97 98 99 0 0 0 0 0 0 0 0 0 0 0 0
```

Output Strings – Test yourself

```c
/* thu nghiem chuoi */
#include <stdio.h>
#include <conio.h>
int main()
{    char S[11]="Hello";
     printf(S);
     getch();
     return 0;
}
```

```c
/* thu nghiem chuoi */
#include <stdio.h>
#include <conio.h>
int main()
{    char S[11]="Hello";
     printf("%s", S);
     getch();
     return 0;
}
```

```c
/* thu nghiem chuoi */
#include <stdio.h>
#include <conio.h>
int main()
{    char S[11]="Hello";
     printf("%s\n", S);
     getch();
     return 0;
}
```

```c
/* thu nghiem chuoi */
#include <stdio.h>
#include <conio.h>
int main()
{    char S[11]="Hello";
     puts(S);
     getch();
     return 0;
}
```

Observe the prompt symbol on the result screen .

Input Strings

- Library: stdio.h
- Function *scanf()* with type conversion %s
- Function *gets(string)*
- Each function has it's own advantages and weaknesses.

Input Strings: scanf(…)

The **%s** conversion specifier

- reads all characters until the first whitespace character,
- stores the characters read in memory locations starting with the address passed to **scanf**,
- Automatically stores the null byte in the memory byte following the last character accepted and
- leaves the delimiting **whitespace** plus any subsequent characters in the input buffer → ignores any leading whitespace characters (default).
- Option specifiers are used to change default characteristics of the function **scanf** on strings.

Input Strings: scanf(…)

```
char name[31];
scanf("%s", name );
```

Enter: **My name is Arnold**

Default

name																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
M	y	\0																												

```
char name[31];
scanf("%10s", name );
```

Enter: **Schwartzenegger**

name																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
S	c	h	w	a	r	t	z	e	n	\0																				

Input Strings: scanf(…)

How to accept blanks in a input string?

➔ **%[^\n]** conversion specifier

- reads all characters <u>until the newline</u> (**'\n'**),
- stores the characters read in memory locations starting with the address passed to **scanf**,
- stores the null byte in the byte following that where **scanf** stored the last character and
- leaves the delimiting character (here, **'\n'**) in the input buffer.

Input Strings: scanf(…)

How to accept blanks in a input string?

➔**%[^\n]** conversion specifier.

```
scanf("%[^\n]", name );
```

```
My name is Arnold
```

stores

name																													
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
M	y		n	a	m	e		i	s		A	r	n	o	l	d	\0													

```
scanf("%10[^\n]", name );
```

```
My name is Arnold
```

stores

name																													
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
M	y		n	a	m	e		i	s	\0																				

Input Strings: scanf(…)

Some character specifiers used in the function scanf(): Set of character are or not accepted.

Specifier	Description
%[abcd]	Searches the input field for any of the characters a, b, c, and d
%[^abcd]	Searches the input field for any characters ***except*** a, b, c, and d
%[0-9]	To catch all decimal digits
%[A-Z]	Catches all uppercase letters
%[0-9A-Za-z]	Catches all decimal digits and all letters
%[A-FT-Z]	Catches all uppercase letters from A to F and from T to Z

Input Strings: gets(…)

gets is a standard library function (stdio.h) that

- accepts an empty string
- uses the **'\n'** as the delimiter
- throws away the delimiter after accepting the string
- Automatically appends the null byte to the end of the set stored

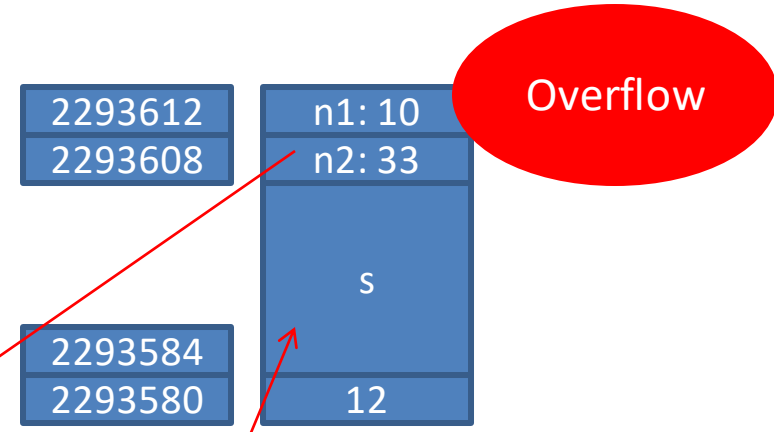
The prototype for **gets** is

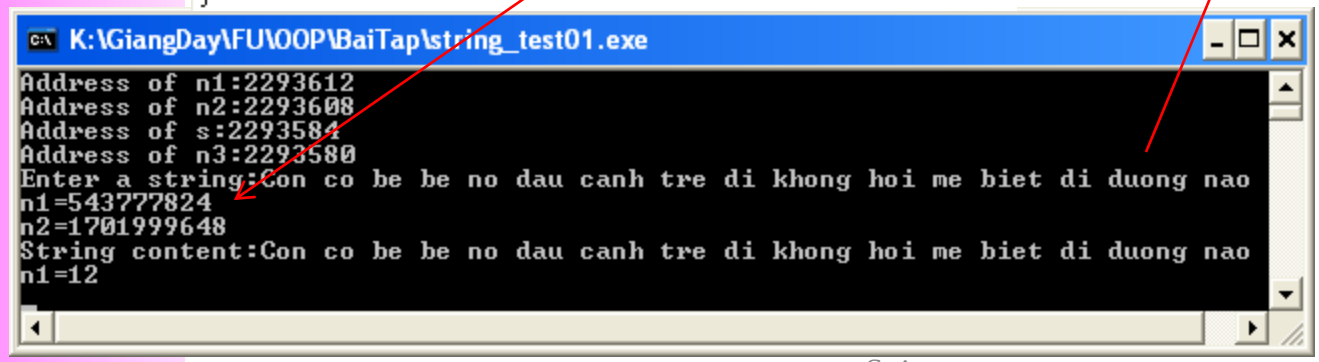
char* gets(char [ ]);

(***gets*** *is dangerous. It can fill beyond the memory that allocated for the string*)

Input Strings: gets(…)

```c
#include <stdio.h>
int main()
{   int n1=10;
    int n2= 33;
    char s[11];
    int n3=12;
    printf("Address of n1:%u\n", &n1);
    printf("Address of n2:%u\n", &n2);
    printf("Address of s:%u\n", s);
    printf("Address of n3:%u\n", &n3);
    printf("Enter a string:");
    gets(s);
    printf("n1=%d\n", n1);
    printf("n2=%d\n", n2);
    printf("String content:%s\n", s);
    printf("n1=%d\n", n3);
    getchar();
    return 0;
}
```



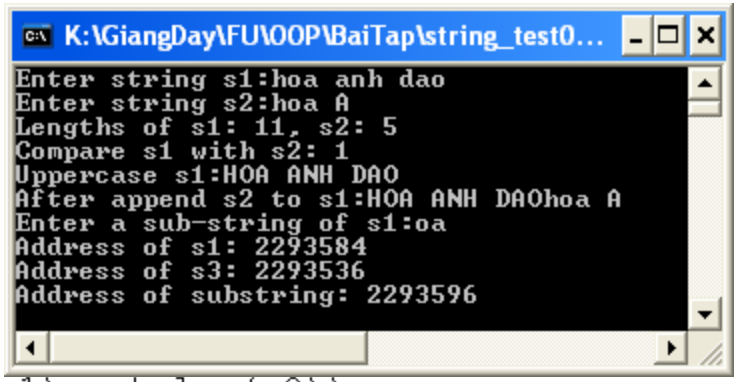


Others String Functions: string.h

Purpose	Function
Get the length of a string	int ***strlen*** (char s[])
Copy **s**ou**rc**e string to **dest**ination string	char* ***strcpy***(char dest[], char src[])
Compare two strings	int ***strcmp***(char s1[], char s2[]) → -1, 0, 1
Concatenate string src to the end of dest	char* ***strcat***(char dest[], char src[])
Convert a string to uppercase	char* ***strupr***(char s[])
Convert a string to lowercase	char* ***strlwr***(char s[])
Find the address of a substring	char* ***strstr*** (char src[], char subStr[]) → **NULL** if **subStr** does not exist in the **src**.

Others String Functions: string.h

```c
#include <stdio.h>
#include <string.h>
int main()
{   char s1[21];
    char s2[21];
    printf("Enter string s1:");
    gets(s1);
    printf("Enter string s2:");
    gets(s2);
    printf("Lengths of s1: %d, s2: %d\n", strlen(s1), strlen(s2));
    printf("Compare s1 with s2: %d\n", strcmp(s1,s2));
    strupr(s1);
    printf("Uppercase s1:%s\n", s1);
    strcat(s1, s2);
    printf("After append s2 to s1:%s\n", s1);
    char s3[10];
    printf("Enter a sub-string of s1:");
    gets(s3);
    char* ptr = strstr(s1, s3);
    printf("Address of s1: %u\n", s1);
    printf("Address of s3: %u\n", s3);
    printf("Address of substring: %u\n", ptr);
    getchar();
    return 0;
}
```



HOA ANH DAOhoa A

2293584

2293596

strstr() → NULL if the substring doesn't exist.

Some User-Defined String Functions

Purpose	Prototype
Trim blanks at the beginning of a string: " Hello" → "Hello"	char* **lTrim**(char s[])
Trim blanks at the end of a string: "Hello " → "Hello"	char* **rTrim**(char s[])
Trim extra blanks in a string: " I am a student " → "I am a student"	char* **trim** (char s[])
Convert a string to a name: " hoang thi hoa " → "Hoang Thi Hoa"	char* **nameStr**(char s[])

Some user-defined String Functions

0	1	2	3	4	5	6
			H	o	a	NULL
i=0	1	2	3			

0	1	2	3	4	5	6
H	o	a	NULL	o	a	NULL

```
char* lTrim (char s[])
{   int i=0;
    while (s[i]==' ') i++;
    if (i>0) strcpy(&s[0], &s[i]);
    return s;
}
```

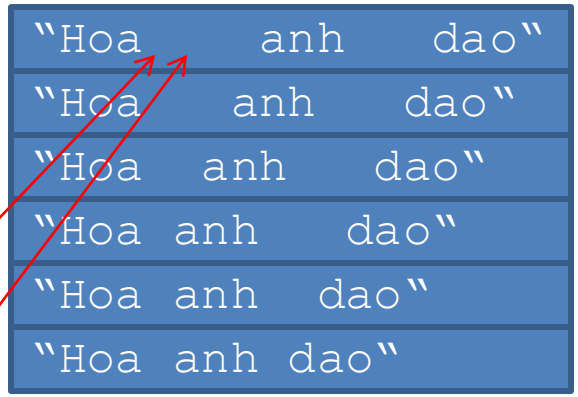
Some user-defined String Functions

0	1	2	3	4	5	6
H	o	a				NULL
		2	3	4	i=5	

0	1	2	3	4	5	6
H	o	a	NULL			NULL

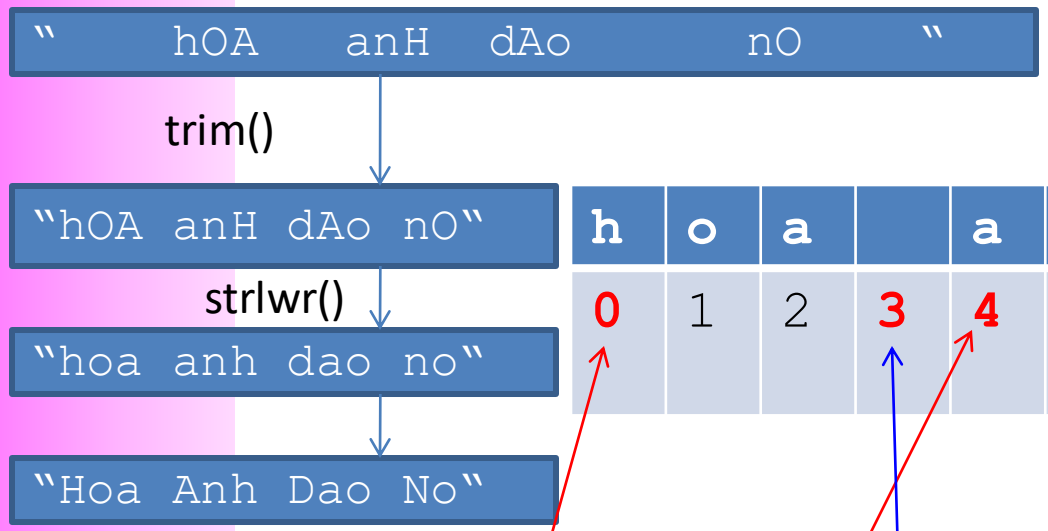
```
char* rTrim (char s[])
{   int i=strlen(s)-1;
    while (s[i]==' ') i--;
    s[i+1]= '\0';   /* NULL */
    return s;
}
```

Some user-defined String Functions




```
char* trim (char s[])
{   rTrim(lTrim(s));
    char *ptr = strstr(s, "  ");
    while (ptr!=NULL)   /* While two blanks exist */
     {   strcpy( ptr, ptr+1);     /* remove one blank */
         ptr = strstr(s, "  ");
     }
     return s;
}
```

Some user-defined String Functions



h	o	a		a	n	h		d	a	o		n	o
0	1	2	3	4	5	6	7	8	9	10	11	12	13

```
char* nameStr(char s[])
{   trim(s); /* trim all extra blanks */
    strlwr(s);  /* convert it to lowercase */
    int L = strlen(s);
    int i;
    for (i=0;i<L; i++)
      if (i==0 || (i>0 && s[i-1]==' ')) s[i] = toupper (s[i]);
    return s;
}
```

Some user-defined String Functions

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <ctype.h>
4  char* lTrim (char s[])
5  {   < your code >
9  }
10 char* rTrim (char s[])
11 {  < your code >
15 }
16 char* trim (char s[])
17 {  < your code >
23 }
24 char* nameStr(char s[])
25 {  < your code >
32 }
33
34 int main()
35 {   char s[21];
36     printf("Enter string s1:");
37     gets(s);
38     trim(s);
39     printf("After extra blanks are remove:");
40     puts(s);
41     nameStr(s);
42     printf("After convert it to a name:");
43     puts(s);
44     getchar();
45     return 0;
46 }
```

K:\GiangDay\FU\OOP\BaiTap\string_test02.exe

```
Enter string s1:    hoA      anH      dAo      nO      !
After extra blanks are remove:hoA anH dAo nO !
After convert it to a name:Hoa Anh Dao No !
```

Summary

String Input

- scanf
- gets
- Do yourself using getchar()

String Functions and Arrays of Strings

- Functions
  - strlen
  - strcpy
  - strcmp
  - strcat
  - strstr

Q&A