Character Strings and String Functions

Assembled for 204112
2015 S2
by Kittipitch Kuptavanich

String

- String คือ Array ของตัวอักษร (char: 1 byte)

This	is	a	character	string.	\0

- โดยที่จะมีตัวอักษรพิเศษ เรียกว่า NULL (นัล) character ('\0') ที่ตำแหน่งสุดท้าย (หากไม่มีจะไม่ถือว่าเป็น String ที่ถูกต้อง)
- เช่น "Good Morning!" – ขนาดเท่าไร?
  - 13 ตัวอักษร
  - ต้องใช้ array ขนาด 14 เป็นอย่างน้อย

String Declaration

```
char message[81];       // declare
message[] = "com";      // assign value
```

OR

```
message[81] = "com";
```

OR

```
message[81] ={'c', 'o', 'm', '\0'};
```

- การแสดงค่า string โดยใช้เครื่องหมายคำพูดกำกับที่ตำแหน่งหัวและท้าย string ("") เรียกว่า string literal

String Declaration [2]

- หากต้องการพิมพ์ตัวอักษรพิเศษอื่น ๆ เช่น เครื่องหมายคำพูด หรือ tab เข้าไปใน string เราจำเป็นต้องมีการทำ escape character เช่น \' แทนเครื่องหมาย '

```
message[81] = "Bob\'s computer";
```

Escape Characters

- `\\` Literal backslash
- `\"` Double quote
- `\'` Single quote
- `\n` Newline (line feed)
- `\r` Carriage return
- `\b` Backspace
- `\t` Horizontal tab
- Etc.


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

String Input & Output

- การอ่าน input string:
  - ~~gets()~~ `fgets(char *BUF, int N, stdin)`
  - `scanf()`
  - `getchar()`
- การแสดงผล output string
  - `puts()`
  - `printf()`
  - `putchar()`

อ่านอักขระไม่เกิน N - 1 อักระ (ต้องเผื่อที่ให้ `\0`)


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

String Input & Output [2]

- `scanf() - %c`
  `getchar()` อ่าน characters เข้ามาหนึ่งตัวอักษร (อ่านทีละตัวอักษร)

  ```
  int c;
  c = getchar();   หรือ scanf("%c",&c);
  ```

- `scanf() - %s` อ่าน characters เข้ามาจนเจอ blank space หรือ newline (บรรทัดใหม่) – (อ่านทีละคำ)

  ```
  scanf("%s", message); //ไม่ต้องใส่ &
  ```

- `fgets()` อ่าน characters เข้ามาจนเจอ newline (บรรทัดใหม่) – (อ่านทีละบรรทัด) และจะติดอักขระ `'\n'` มาด้วย (ไม่ควรใช้ `gets()` เพราะเสี่ยง buffer overflow เนื่องจากขนาด array ปลายทางอาจเล็กกว่าความยาวบรรทัด)
- `printf()` และ `puts()` สามารถใช้แทนกันได้

  ```
  printf("%s\n", message); ≡ puts(message);
  ```



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

String Input & Output [3]

Program 9.1

```
1  #include <stdio.h>
2  int main()
3  {
4    #define MSIZE 81
5    char message[MSIZE]; /* enough storage for 80 characters plus '\0' */
6
7    printf("Enter a string:\n");
8    gets(message);
9    printf("The string just entered is:\n");
10   puts(message);
11
12   return 0;
13 }
```

Output

```
Enter a string:
This is a test input
The string just entered is:
This is a test input
```

String Encoding

String array	Stored codes	Expression	Value
t	116	string2[0]	116
h	104	string2[1]	104
i	105	string2[2]	105
s	115		
	32		
i	105		
s	115		
	32	.	.
a	97	.	.
	32	.	.
s	115		
t	116		
r	114		
i	105		
n	110		
g	103	string2[15]	103
\0	0	string2[16]	0

Figure 9.4 The ASCII codes used to store this is a string

- สังเกตว่า ตัว null character '\0' มีค่า เป็น 0

String Input & Output [4]

- ตัวอย่างการอ่าน string ทีละตัวอักษรจนจบบรรทัด โดยฟังก์ชัน getchar()

สังเกตว่า ต้องอ่านค่ามาเก็บไว้ที่ c ก่อนจึงนำมาเปรียบเทียบว่าเป็น '\n' หรือไม่

```
09 #define LSIZE 81
...
13
14     char message[LSIZE];     /* 80 characters plus '\0' */
15     char c;
16     int i;
17     printf("Enter a string:\n");
18     i = 0;
19     while (i < (LSIZE - 1) && (c = getchar()) != '\n') {
20         message[i] = c;      /* store the character entered */
21         i++;
22     }
23     message[i] = '\0';       /* terminate the string */
24     printf("The string just entered is: \n");
25     puts(message);
```


String Formatting

```
char happy[] = "Have a Happy Day"
printf("|%25s|",happy);
|         Have a Happy Day|

printf("|%-25s|",happy);
|Have a Happy Day         |

printf("|%25.12s|",happy);
|             Have a Happy|

printf("|%.12s|",happy);
|Have a Happy|
```


String Arrays

- เนื่องจาก String คือ Array ของ char
  - Array ของ String คือ Array 2 มิติ ของ char

```
char str_arr[____][____];
```

str_arr

b	a	t	\0		
c	a	t	\0		
b	e	e	r	\0	
c	a	n	d	y	\0

String Arrays [2]

- หากต้องการอ่านคำจากไฟล์ด้วยวิธี **Redirection**
  - โจทย์ระบุว่าบรรทัดแรกคือจำนวนคำ
  - โดยมีคำไม่เกิน **10** คำ
  - และมีความยาวแต่ละคำไม่เกิน **5** ตัวอักษร

word_list.txt
```
4
cat
bat
beer
candy
```

```c
#include <stdio.h>

#define WORD_SIZE 5      /* ความยาว 5 */
#define MAX_WORD  10     /* ไม่เกิน 10 ตัว */

int main()
{

    char word_list[MAX_WORD][WORD_SIZE + 1];
```


String Arrays [3]

read_string.c
```c
int main()
{

    char word_list[MAX_WORD][WORD_SIZE + 1];
    int word_count;
    int i;
    scanf("%d", &word_count);

    for (i = 0; i < word_count; i++){
        scanf("%s", word_list[i]);
    }

    for (i = 0; i < word_count; i++){
        puts(word_list[i]);
    }

    return 0;
}
```

word_list.txt
```
4
cat
bat
beer
candy
```

```
> read_string.exe < word_list.txt
cat
bat
beer
candy
```


String Arrays [4]

- ในหลาย ๆ กรณี **String** แต่ละตัวจะมีความยาวต่างกันไป ทำให้การประกาศตัวแปรในลักษณะนี้อาจไม่ยืดหยุ่นพอ
- ใช้วิธีสร้าง **Array** ของ **pointer** ชนิด **char** * ก่อน แล้วจึงจองหน่วยความจำแยกทีละคำภายหลัง

word_list

b	a	t	\0		
c	a	t	\0		
b	e	e	r	\0	
c	a	n	d	y	\0

```c
char *word_list[MAX_WORD];

for(i = 0; i < MAX_WORD; i++) {
    word_list[i] = NULL;
}
```


String Array [5]

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

#define WORD_SIZE 5      /* ความยาว 5 */
#define MAX_WORD  10     /* ไม่เกิน 10 ตัว */
```

```c
int main()
{

    char *word_list[MAX_WORD] = {};
    char temp[WORD_SIZE + 1];
    int word_count;
    int i;
    scanf("%d", &word_count);

    for (i = 0; i < word_count; i++){
        scanf("%s", temp);
        word_list[i] = (char *) malloc(strlen(temp) + 1);
        strcpy(word_list[i], temp);
    }

...
    return 0;
}
```

strlen() เป็น **String Function** ทำหน้าที่_____

strcpy() เป็น **String Function** ทำหน้าที่_____

#include <string.h>

String Library Functions

- มี 3 กลุ่มใหญ่
  - str_() ทำการเปลี่ยนแปลงทั้ง string
    - ทำทุกอักษรจนถึง NULL
  - strn_() ทำการเปลี่ยนแปลงเฉพาะบางส่วน
    - ทำตัวอักษร n ตัวตามที่ระบุมา ที่ไม่ใช่ NULL
  - mem_()
    - ทำตามกลุ่มอักษรที่กำหนดมา ไม่พิจารณาว่ามี NULL หรือไม่
    - ใช้น้อยกว่าสองกลุ่มแรก

#include <string.h>

strlen()

- Return ความยาว (length) ของ string (ต้องจบด้วย '\0')
  - strlen(char *)

```
07 #include <stdio.h>
08 #include <string.h>
09
10 int main()
11 {
12     char *t = "XXX";
13     printf("Length of <%s> is %d.\n", t, strlen(t));
14     return 0;
15 }
```

output:
Length of <XXX> is 3.

- Note: ความยาวของ string ไม่จำเป็นต้องเท่ากับขนาดของ array

#include <string.h>

strcpy()

```
11 int main()
12 {
13     char s1[100];
14     char s2[100];
15
16     strcpy(s1, "xxxxxxx 1");
17     strcpy(s2, "zzzzzzz 2");
18
19     puts("Original strings: ");
20     puts(s1);
21     puts(s2);
22     puts("------");
23
24     strcpy(s1, s2);
25     puts("New strings: ");
26     puts(s1);
27     puts(s2);
28
29     return 0;
30 }
```

- คัดลอก (copy)ข้อมูลจาก s2 ไปยัง s1
  - strcpy(char *s1, char* s2)

output:
```
xxxxxxx 1
zzzzzzz 2
------
New strings:
zzzzzzz 2
zzzzzzz 2
```


#include <string.h>

strcpy()[2]

- strcpy(char *s1, char* s2)
  - Function จะไม่ทำการเชคว่า s1 มีขนาดพอหรือไม่ที่จะ copy s2 ไปใส่
    - หาก array s1 มีขนาดไม่พอก็จะเกิด error
- strncpy(char *s1, char* s2, int n)
  - คัดลอกตัวอักษรแรก n ตัว จาก s2 ไปยัง s1

#include <string.h>

strcat()

- นำ string *s2* มาต่อท้าย (concatenate) string *s1*
  - `strcat(char *s1, char* s2)`

```
08 #include <stdio.h>
09 #include <string.h>
10 int main()
11 {
12     char s1[50];
13     char s2[20];
14
15     strcpy(s1, "Tweedledee ");
16     strcpy(s2, "Tweedledum");
17
18     strcat(s1, s2);
19     puts(s1);
20
21     return 0;
22 }
```

output:
`Tweedledee Tweedledum`

#include <string.h>

strcat()[2]

- `strcat(char *s1, char* s2)`
  - Function จะไม่ทำการเช็คว่า s1 มีขนาดพอหรือไม่ที่จะนำ s2 ไปต่อท้าย
    - หาก array s1 มีความยาวไม่พอก็จะเกิด error
- `strncat(char *s1, char* s2, int n)`
  - นำ ตัวอักษร n ตัวแรก จาก s2 ไปต่อท้าย s1

#include <string.h>

strcmp()

- เปรียบเทียบระหว่าง (compare) s1 และ s2
  - `strcmp(char *s1, char* s2)`

```
10 #define ANSWER "blue"
11
12 int main()
13 {
14     char t[100];
15     puts("What is the secret color?");
16     scanf("%s", t);
17
18     while (strcmp(t, ANSWER) != 0) {
19         puts("Wrong, try again.");
20         scanf("%s", t);
21     }
22
23     puts("Right!");
24
25     return 0;
26 }
```


#include <string.h>

strcmp()[2]

- `strcmp(char *s1, char* s2)`
  - ถ้า s1 มาก่อน s2 (ตามตัวอักษร)
    - `strcmp("bat", "cat")`
    - return ค่าลบ (< 0)
  - ถ้า s1 เหมือนกับ s2
    - `strcmp("cat", "cat")`
    - return 0
  - ถ้า s1 มาหลัง s2 (ตามตัวอักษร)
    - `strcmp("cat", "bat")`
    - return ค่าบวก (> 0)

#include <string.h>

strcmp()[3]

- stricmp(char *s1, char* s2)
  - เปรียบเทียบโดยไม่พิจารณา case
    - case insensitive (‘a’ มีค่าเท่ากับ ‘A’)
- strncmp(char *s1, char* s2, int n)
  - เปรียบเทียบระหว่าง n อักษรแรกจาก s1 และ s2
  - strncmp("caterpillar","category",3)
    - จะได้ค่า 0
- strnicmp() – n อักษรแรก + case insensitive

#include <string.h>

strchr()

- หาตำแหน่งแรกของอักษร (character) c ใน s1
  - char *strchr(char *s1, char c)
  - ค่าที่ return จะเป็น pointer ไปยังตำแหน่งของ c

```
int main()
{
    char *t = "MEAS:VOLT:DC?";
    char *p;
    p = t;
    puts(p);

    while ((p = strchr(p, ':')) !=  NULL) {
       puts(++p);
    }

    return 0;
}
```

output:
MEAS:VOLT:DC?
VOLT:DC?
DC?

#include <string.h>

strchr() [2]

- strrchr(char *s1, char c)
  - หาตำแหน่งแรกจากทางขวา (right) ของอักษร c ใน string s1
- strstr(char *s1, char* s2)
  - หาตำแหน่งแรกของ string s2 ใน string s1

#include <string.h>

strtok()

- ใช้ delim แบ่ง string s1 ออกเป็นส่วน ๆ (token)
- char *strtok(char *s1, constant char* delim)

```
int main()
{
    char str[80] = "hormones:the:series";
    char *s = ":";
    char *token;

    /* get the first token */
    token = strtok(str, s);

    /* walk through other tokens*/
    while (token != NULL) {
        printf("%s\n", token);
        token = strtok(NULL, s);
    }

    return 0;
}
```

output:
hormones
the
series

ให้ค้นหาต่อจากตำแหน่งก่อนหน้านี้

#include <string.h>

Lesser Used Functions

- strupr(char *s1)
  - เปลี่ยนตัวอักษรใน s1 ให้เป็นตัวพิมพ์ใหญ่ (upper case) ทั้งหมด
- strlwr(char *s1)
  - เปลี่ยนตัวอักษรใน s1 ให้เป็นตัวพิมพ์เล็ก (lower case) ทั้งหมด
- ~~strrev(char *s1)~~ non-standard C
  - กลับลำดับตัวอักษร (reverse) ใน s1 (จาก "bat" เป็น "tab")

#include <stdlib.h>

Conversions

- int atoi(char *s1)
  - เปลี่ยน ASCII string เป็น integer
  - atoi("1234")
- double atof(char *s1)
  - เปลี่ยน ASCII string เป็น เลขทศนิยมชนิด double (double-precision floating-point)
  - atof("12.34")
- ~~char [] itoa(int x, char*,int base)~~ non-standard C
  - เปลี่ยน integer เป็น ASCII string (non standard)
  - itoa(1234,s,10)

#include <ctype.h>

Character Classifications

- การแยกประเภทอักขระ (char)
- isalpha(char c)
  - ตรวจสอบดูว่าอักขระ c เป็นตัวอักษร (alphabet) หรือไม่ (return 0 กรณีไม่ใช่)
- isdigit(char c)
  - ตรวจสอบดูว่าอักขระ c เป็นตัวเลข (digit) หรือไม่ (return 0 กรณีไม่ใช่)
- isalnum(char c)
  - ตรวจสอบดูว่าอักขระ c เป็นตัวอักษร (alphabet) หรือตัวเลข (number) หรือไม่ (return 0 กรณีไม่ใช่)

#include <ctype.h>

Character Classification [2]

- isblank(char c)
  - ตรวจสอบดูว่าอักขระ c เป็นวรรค (blank space หรือ tab) หรือไม่ (return 0 กรณีไม่ใช่)
- isspace(char c)
  - ตรวจสอบดูว่าอักขระ c เป็นอักขระว่าง (space, \t, \n, \f ,\r) หรือไม่ (return 0 กรณีไม่ใช่)
- ispunct(char c)
  - ตรวจสอบดูว่าอักขระ c เป็นเครื่องหมายวรรคตอน (punctuation mark) หรือไม่ (return 0 กรณีไม่ใช่)
  - เครื่องหมายวรรคตอนได้แก่
    - !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

Example: Dynamic Decimal Places

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

#define MAX_FORMAT_LEN 10
// ไม่เกิน 100 ตำแหน่ง
#define PLACE_STRING_WIDTH 3

int main()
{
    double  i = 1.2345678912;
    int place;
    printf("Enter n: ");
    scanf("%d",&place);
    char place_s[PLACE_STRING_WIDTH];
    char formatH[MAX_FORMAT_LEN];

    strcpy(formatH,"%.");        //formatH = "%."

    itoa(place,place_s,10);      //เปลี่ยนจากตัวเลขเป็น char array เช่น 13 เป็น "13"
    strcat(formatH,place_s);     //formatH = "%.XX"

    strcat(formatH,"f");         //formatH = "%.XXf"
    printf(formatH,i);

    return 0;
}
```


Example: Dynamic Decimal Places [2]

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

int main()
{
    double  i = 1.2345678912;
    int place;
    printf("Enter n: ");
    scanf("%d",&place);

    char formatH[] = "%.000f";

    formatH[2] =  '0' + (place / 100);
    formatH[3] =  '0' + (place / 10 % 10);
    formatH[4] =  '0' + (place % 10);

    printf(formatH,i);

    return 0;
}
```

Without string.h