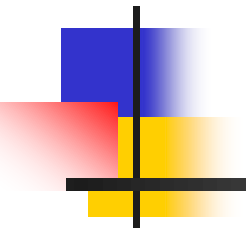


C Programming Lab 8

String

Character, Number, Symbol

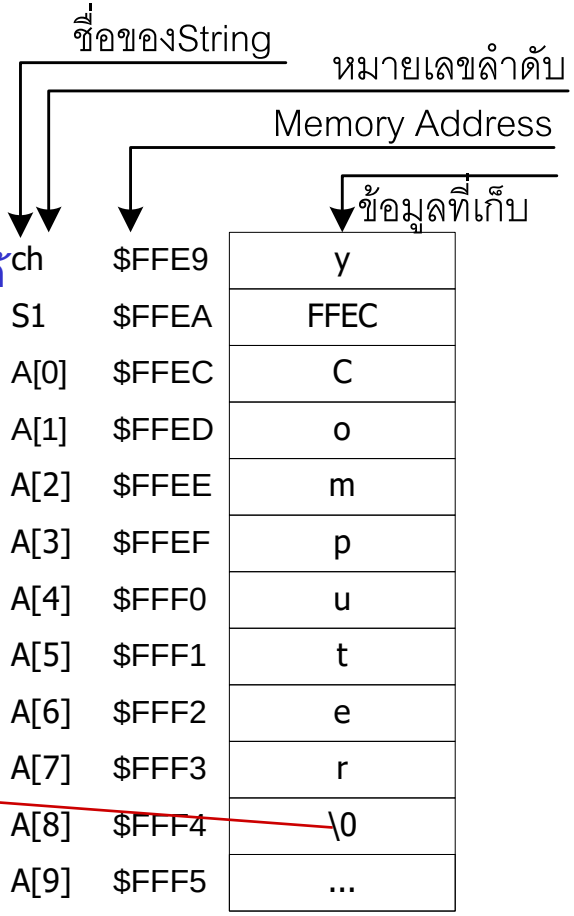
- ในภาษาซี มีตัวแปรที่ใช้เก็บตัวอักษร (Character) กำหนดเป็น char ใช้เนื้อที่เก็บข้อมูลขนาด 1 byte ซึ่งมีค่าเป็นตัวเลขอยู่ระหว่าง 0 - 255 แต่นิยมใช้สัญลักษณ์แทนตำแหน่งของตัวเลข
- วิธีการอ้างถึงค่า Character ในภาษาซี
  - เขียน เป็นตัวเลข หรือ รูปสัญลักษณ์ของตัวอักษรนั้น โดยใช้เครื่องหมาย Single Quotation ' เปิดและปิดคร่อมรูปสัญลักษณ์ของตัวอักษร เช่น

    65='A', 97='a', 48='0', 64='@' ,42='*'
  - ใช้ตัวควบคุมเป็น %c ในการอ่าน หรือแสดงผล 1 ตัวอักษร
- การนำเอา Character หลายๆมาเรียงต่อกัน (Array) นิยมเรียกว่า String เช่น

  char A[10] = "Computer" ; //จองตัวแปรสำหรับเก็บตัวอักษรไว้ไม่เกิน 10 ช่อง
  - การเขียนค่าคงที่ string ให้เขียนข้อความอยู่ระหว่างเครื่องหมาย " "
  - ภาษาซีจะเพิ่มตัวอักษรเรียกว่า null character (ลำดับที่ 0 ของ ASCII นิยมเขียนอ้างอิงด้วย'\0') ต่อท้ายข้อความให้เองโดยอัตโนมัติ (ไม่มีรูปสัญลักษณ์)
  - ต้องจองเผื่อไว้ 1 ช่องสำหรับให้ภาษาซีใช้เก็บตัวปิดข้อความ(null) กรณีนี้จะเก็บตัวอักษรได้ไม่เกิน 9 ตัว
  - string เป็น pointer (ชี้ไปยังกลุ่มของตัวอักษร) ไม่สามารถใช้เป็นชนิดข้อมูล character ซึ่งเป็นค่าของ ASCII ได้ ต้องอ้าง [index] ก่อนจึงจะสามารถใช้เป็น character ทีละตัวได้

String in C

- ภาษาซี ไม่มีตัวแปรชนิด string โดยตรง แต่สามารถใช้แถวลำดับของตัวอักษร (character array) ซึ่งเป็นค่า Address เริ่มต้นของตัวแปร ร่วมกับคำสั่งพิเศษโดยเฉพาะเพื่อจัดการกับข้อมูลประเภทนี้
- ไม่สามารถใช้คำสั่งที่จัดการกับตัวเลขมาจัดการกับสตริงได้ เช่น (=, ==, >, <,>=,<=, !=) ต้องใช้คำสั่งพิเศษโดยเฉพาะ (#include <string.h>)
- คำสั่งเกี่ยวกับ string จะกระทำกับอักษรทีละตัวต่อเนื่องไปจนกว่าจะเจอ '\0'
- ภาษาซีถือว่าข้อมูลสตริงที่เก็บอยู่จะเริ่มตั้งแต่อักษรตัวแรกที่เก็บอยู่ใน Address นั้น จนถึงตัวอักษรที่เป็นจบสตริง
- ตัวอักษร '\0' ถูกเติมให้โดยอัตโนมัติ
  - char str[10] = "Computer", *S1 , ch ;
- ถ้าต้องการจองอาร์เรย์ของ string หลายๆตัว
  - char name[200][16] , grade[200][3];



*S1 คือ pointer ที่ชี้ไปยัง string

จองตัวแปร 200 ตัว ตัวละ 3 char

จองตัวแปร A เป็นตัวอักษรไว้ 10 ช่อง ใช้เก็บข้อความได้ 9 ตัวอักษร (เผื่อ'\0' อีกหนึ่งตัว)

ใช้ = กำหนดค่าเริ่มต้นให้กับ string ได้เฉพาะตอนจองตัวแปรเท่านั้น

Read String

- การอ่านข้อมูลโดยใช้คำสั่ง scanf() จะใช้เว้นวรรค(white space) เป็นตัวคั่นระหว่างข้อมูลแต่ละตัว โดยโปรแกรมจะไม่อ่านเว้นวรรค
- ตัวแปร string ที่จองในภาษาซีจะเป็น Address (ไม่ใช่ Value) ดังนั้นการอ่านข้อมูลชนิด string จากคีย์บอร์ด สามารถใช้คำสั่ง scanf( ) เก็บไว้ในตัวแปรได้ทันที โดยไม่ต้องแปลงเป็น Address อีก แต่ต้องใช้ control string เป็น "%s"

```
char   s1[80];
     scanf("%s",s1);
```

อ่านข้อความ(string) ด้วย "%s" ไม่ต้องใส่ & นำหน้าตัวแปร

สมมุติป้อนค่า " CPE 100 "

ค่าที่อ่านได้คือ s1= "CPE" (ไม่มีเว้นวรรค แต่มีรหัส \0 คือ NULL ต่อท้ายเป็นตัวจบสตริง)
เหลือข้อมูลค้างอยู่ใน stdin ยังไม่ได้อ่านคือ " 100 \n"
(\n คือ newline ASCII=10)

- ถ้าต้องการอ่านทั้งบรรทัดให้ใช้
  - `char *gets(char *s) ;` //สามารถ return address ของ s ได้ด้วย

```
gets(s1);
```

ใช้คำสั่ง gets(s1) แทน scanf("%s",s1);
ค่าที่อ่านได้คือ s1= " CPE 100 "
การอ่านด้วย gets() จะไม่เหลือค่า\n ค้างอยู่ใน stdin

- ต้องระวังเรื่องความยาวของข้อความที่ป้อนไม่ให้เกินจำนวนที่จองไว้ (ต้องลบออก 1 สำหรับ '\0') เพราะอาจทำให้เกิดความผิดพลาดได้
- การแสดงผลข้อมูล String แสดงได้โดยใช้ control string เป็น "%s" และโปรแกรมจะแสดงผลตั้งแต่ตัวอักษรตัวแรกจนกระทั่งพบรหัส '\0' จึงหยุดแสดง

Over Length of String

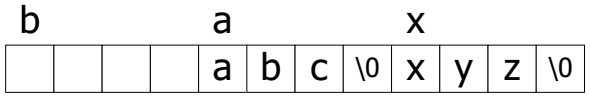
- ภาษาซีไม่มีการป้องกัน การทำงานกับสตริงที่เกินความยาวมากกว่าที่จองไว้
- การป้อนข้อมูลมากเกินกว่าที่จองไว้จะทำให้เกิดปัญหากับตัวแปรตัวอื่น หรืออาจทำให้โปรแกรมหยุดทำงานได้

```
{ char x[4] = "xyz";
  char a[4] , b[4];
```

สมมุติว่าโปรแกรมจองตัวแปรติดกันในลักษณะนี้

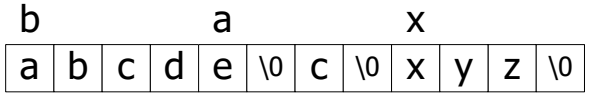

```
gets(a);  สมมุติ ป้อนค่า "abc[Enter]"
```

ใช้คำสั่ง gets(a) หรือ scanf("%s",a);




```
gets(b); สมมุติ ป้อนค่า " abcde[Enter]"
```

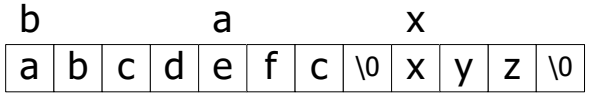
ป้อนค่าเกินความยาวที่เก็บได้




```
printf("x=%s, a=%s, b=%s\n",x,a,b);
```

ค่าที่แสดงผลคือ
x=xyz, a=e, b=abcde

```
b[5] = 'f';
```




```
printf("x=%s, a=%s, b=%s\n",x,a,b);
}
```

ค่าที่แสดงผลคือ
x=xyz, a=efc, b=abcdefc

String Operations

- string ในภาษาซีคือ pointer (address เริ่มต้นของข้อมูล) ดังนั้นจึงไม่สามารถนำ ข้อมูลทั้งหมดมากำหนดค่า(=) หรือเปรียบเทียบกันโดยตรง โดยใช้ operator (==, !=, >, >=, <, <=) เพราะจะกลายเป็นการเปรียบเทียบ address ของหน่วยความจำแทน จึงต้องมีคำสั่งที่สั่งให้ทำงานกับข้อความที่เก็บอยู่ใน address เหล่านั้นโดยเฉพาะ
- NULL pointer หมายถึง pointer ที่ไม่ได้ชี้ไปยัง address ใด มีค่าเป็น 0
- null string หมายถึง string ที่ไม่มีข้อความเก็บอยู่ จะมีข้อมูลเป็น "" หรือ "\0"
- #include <stdio.h> มีคำสั่งเกี่ยวกับ string // อ่านค่า, แสดงผล
  - อ่านข้อมูลทั้งบรรทัดจนกว่าจะกด Enter (gets)
  - อ่านข้อมูลจากตัวแปร string แทนคีย์บอร์ด (sscanf)
  - พิมพ์ผลไปที่ตัวแปร string แทนจอภาพ (sprintf)
- #include <string.h> // จัดการข้อความ
  - การคัดลอก/กำหนดค่า ให้ตัวแปร string (strcpy)
  - การหาความยาว(นับจำนวนตัวอักษร)ของข้อความ (strlen)
  - การนำ string ตัวที่ 2 มาต่อท้ายตัวที่ 1 (strcat)
  - การเปรียบเทียบข้อความ 2 ชุด (strcmp)
  - การค้นหาตำแหน่งของตัวอักษรที่อยู่ใน string (strchr)
  - การค้นหาตำแหน่งของข้อความที่อยู่ใน string (strstr)
  - การแยกสตริง (strtok)

String Operations

- #include <stdlib.h> // แปลงค่า
  - atoi(string); แปลง string เป็นค่าตัวเลขจำนวนเต็ม int
  - atol(string); แปลง string เป็นค่าตัวเลขจำนวนเต็ม long int
  - atof(string); แปลง string เป็นค่าตัวเลขจำนวนจริง float
  - strtod(string, *endptr); แปลง string เป็นเลขจำนวนจริง double มี *endptr ชี้ตำแหน่งที่แปลงไม่สำเร็จ
  - คำสั่งแปลงตัวเลขจำนวนเต็มเป็น string คือ itoa(value,string,radix) และ ltoa(value,string,radix) ไม่ใช่ ANSI C (run ได้เฉพาะใน C บางตัวเท่านั้น) ส่วนคำสั่งแปลงจากตัวเลขจำนวนจริง เป็น string ไม่มี
  - อาจใช้คำสั่ง sprintf(string, "format", num) เพื่อพิมพ์ค่า num ลงในตัวแปร string เพื่อแปลงตัวเลข num ให้เป็น string
- #include <ctype.h> // จัดการตัวอักษร 1 ตัว
  - isalnum, isalpha, isascii, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit ใช้ตรวจสอบตัวอักษรว่าอยู่ในกลุ่มที่กำหนดหรือไม่ // เทียบได้กับ strchr()
  - tolower(c), toupper(c) ใช้แปลงตัวอักษร 1ตัว ให้เป็นตัวพิมพ์เล็ก-ใหญ่

#include <stdlib.h>

- แปลง string เป็นตัวเลข
  - int atoi(char *str); //แปลง str เป็นจำนวนเต็ม
  - double atof(char *str); //แปลง str เป็นจำนวนจริง
  - double strtod(char *str, char **end);//แปลง str เป็นจำนวนจริง+pointer

```
char  str[80] = "100x", *end;
double num;
        num = strtod(str, &end);
        if (strcmp(end,"")!=0)
            printf("remaining :%s:",end);
```

จะได้ num = 100 เหลือ end = "x"

  - sscanf(str,"%d",&i); //อ่านจำนวนเต็ม จาก str (แทนการอ่านจาก stdin)
  - sscanf(str,"%lf",&num); //อ่านจำนวนจริง จาก str
- แปลงตัวเลขเป็นสตริง
  - char *itoa(int num, char *str, int radix) //แปลงเลขฐาน radixเป็น str
  - sprintf(str, "%d", i); //พิมพ์ตัวเลขตาม format ลงใน str (แทนจอภาพ)
  - sprintf(str,"%.15g", num);

#include <ctype.h>

- การเปลี่ยนรูปตัวอักษรทั้งหมดใน str ให้กลายเป็นตัวพิมพ์เล็ก หรือ ตัวพิมพ์ใหญ่
  - char *strlwr(char *str) // แปลงสตริงทั้งหมดให้เป็นตัวพิมพ์เล็ก
  - char *strupr(char *str) // แปลงสตริงทั้งหมดให้เป็นตัวพิมพ์ใหญ่
- เปลี่ยนตัวอักษร 1 character ให้เป็นตัวพิมพ์เล็ก หรือ ตัวพิมพ์ใหญ่
  - int tolower(int ch) // แปลงตัวอักษร ch เป็นตัวพิมพ์เล็ก
  - int toupper(int ch) // แปลงตัวอักษร ch เป็นตัวพิมพ์ใหญ่
- ใช้ตรวจสอบตัวอักษรว่าอยู่ในกลุ่มที่กำหนดหรือไม่ return 0 if false
  - int isalpha(int ch) // ตรวจสอบ ch เป็น 'A'..'Z' หรือ 'a'..'z'
  - int isdigit(int ch) // ตรวจสอบ ch เป็น '0'..'9'
  - int islower(int ch) // ตรวจสอบ ch เป็น 'a'..'z'
  - int isupper(int ch) // ตรวจสอบ ch เป็น 'A'..'Z'
  - int isspace(int ch) // ตรวจสอบ ch เป็น space ,tab, newline'\n'

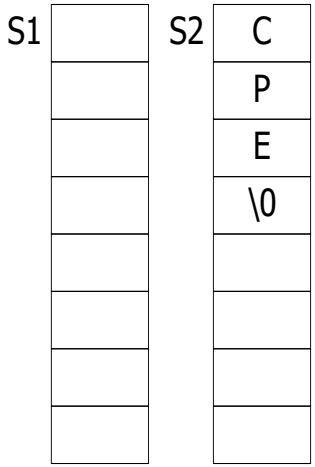
Copy String

- การกำหนดค่าให้ตัวแปร string ใช้คำสั่ง string copy
  - char *strcpy(char *dest, char *src);

  เป็นการกำหนดค่าให้สตริง dest มีค่าเท่ากับตัวแปร str

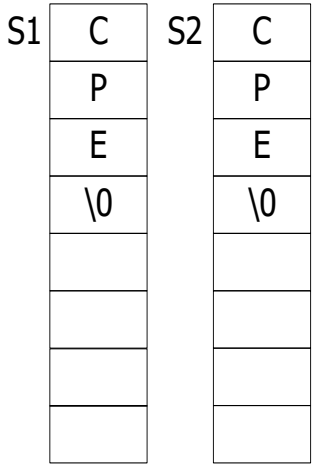
  ค่าที่ return เป็น char * หมายถึง address ของคำตอบ(dest)

ตัวอย่างที่ผิด	ตัวอย่างที่ถูก
s2 = "CPE"; s1 = s2;	strcpy(s2,"CPE"); strcpy(s1, s2);



- Example

```
if (total >=90) strcpy(grade,"A");
   else if (total >=80) strcpy(grade,"B+");
      else if (total >=70) strcpy(grade,"B");
         else if (total >=60) strcpy(grade,"C+");
            else if (total >=50) strcpy(grade,"C");
               else if (total >=40) strcpy(grade,"D+")
                  else if (total >=30) strcpy(grade,"D");
                     else strcpy(grade,"F");
```



Copy String

- คัดลอกบางส่วนของสตริง ตั้งแต่ตัวแรก จำนวน n ตัว

```
char *strcpycount(char *dest, char *src, int count)
{ int  i;
      for (i=0; i<count; i++)
          dest[i] = src[i] ;
      dest[i] = '\0';
      return dest;
}
```

- คัดลอกบางส่วนของสตริง ในตำแหน่งที่กำหนด start .. stop

```
char *strpart (char *dest, char *src, int  start, int stop)
{ int  i;
      for (i=start; i<=stop; i++)
          dest[i-start] = src[i] ;
      dest[i-start] = '\0';
      return dest;
}
```

Length of String

- การนับจำนวนตัวอักษรใน string (ความยาวของ string) ใช้ฟังก์ชัน string length ซึ่งจะ return ค่าตัวเลขความยาวของ stringที่นับได้
- Syntax

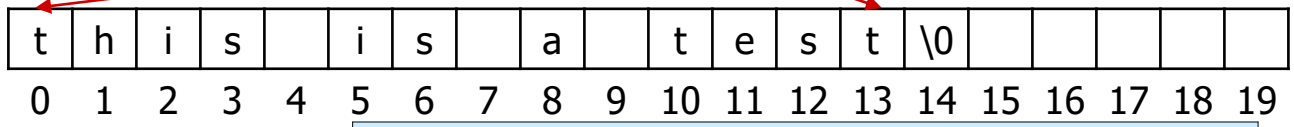
```
unsigned strlen(char *str);
```

unsigned เป็นการเขียนย่อของ unsigned int

  - พารามิเตอร์ (char *str) หมายถึง ให้ส่ง address ของข้อมูลชนิด string เพื่อนำไปใช้ในคำสั่ง
- Example

```
char A[20] = "this is a test";
int len;
len = strlen(A);
```

ข้อมูลจะเริ่มเก็บตั้งแต่ A[0] จนถึง A[13]



จะได้ len = 14 (นับรวมเว้นวรรค แต่ไม่นับรหัส \0)

Concatenate String

- การเชื่อมต่อ string 2 ตัวเข้าด้วยกันให้ใช้คำสั่ง string concatenation

  ไม่สามารถใช้ name = firstname + " " + lastname;

  - char *strcat(char *str1, char *str2);

    ค่าที่ได้จะเป็นการนำเอา str2 มาต่อท้าย str1

    (ตัวแปร str1 จะมีค่าเปลี่ยนไป)

  เช่น

  - char str1[10]="CPE", str2[10]="100";

    strcat(str1,str2);

    str1 = "CPE100" str2 = "100"

S1	C	S2	1
	P		0
	E		0
	\0		\0

  - char name[40],first[15]="Pipat",last[25]="Supasirisun";

    strcpy(name, first); name = "Pipat"

    strcat(name,"□"); name = "Pipat "

    strcat(name, last); name = "Pipat Supasirisun"

S1	C	S2	1
	P		0
	E		0
	1		\0
	0		
	0		
	\0		

- สามารถใช้ sprintf() พิมพ์ข้อความลงใน string ตัวใหม่
  - sprintf(name,"%s□%s",first,last);

Compare String

- การเปรียบเทียบค่า string 2 ตัว ว่าเหมือนกันหรือไม่ ใช้คำสั่ง string compare
  - int strcmp(char *str1, char *str2);

เป็นการเปรียบเทียบค่าของ str1 ว่ามากหรือน้อยกว่า str2 โดยดูลำดับตามตัวอักษร ASCII (A..Z มาก่อน a..z) ถ้าตัวหน้าเหมือนกัน จะดูตัวถัดไป

- ค่าที่ return จะเป็นตัวเลข ตามเงื่อนไขดังนี้
  - == 0 แสดงว่า str1 เหมือนกับ str2 (ทุกตัว) เช่น strcmp("AX","AX");
  - != 0 แสดงว่า str1 ไม่เท่ากับ str2 เช่น strcmp("AA","Aa");
  - <0 (-1) แสดงว่าลำดับของ str1 มาก่อน str2 เช่น strcmp("A","Z");
  - >0 (1) แสดงว่าลำดับของ str1 มากกว่า str2 เช่น strcmp("a","ABC");

ต้องดูตัวที่ 2 จึงจะสรุปได้

ดูตัวแรกก็สรุปได้แล้ว

```
int check_function(char *str);
{  int  found = 0;
    if (strcmp(str,"sin")==0) found = 1;
      else if (strcmp(str,"cos")==0) found = 1;
        else if (strcmp(str,"tan")==0) found = 1;
          else if (strcmp(str,"sqrt")==0) found = 1;
            else if   ..........................................
   return found ;
}
```

Compare String

```
int is_function (char *token)
{ char fname[15][10] =  // จองเนื้อที่ไว้ 15 ตัว ตัวละไม่เกิน 10 อักษร
    {"sin","cos","tan","asin","acos","atan","sqrt","pow","log","exp","abs"} ;
  int  fcount = 11,  i = 0 ;
  char str[20];
    strcpy(str,token);  // copy token ไปใช้เพื่อไม่ให้ค่าinputเปลี่ยน
    strlwr(str);  // แปลงเป็นอักษรตัวเล็ก เพื่อเทียบกับค่าคงที่
    while ( i<fcount && strcmp(fname[i],str) !=0 )
              i++;
    if (i < fcount)
        return 1;
    else
        return 0;
}
```

Search character in string

- การค้นหาตำแหน่งตัวอักษรที่อยู่ใน string ใช้คำสั่ง string character
  - char *strchr(const char *str, int ch);

ตัวอย่าง ค้นหาตำแหน่งของ 's' ใน "This is a test"

```
char s1[40]="This is a test" , *s2 ;
        s2 = strchr(s1,'s') ;  // s2 = "s is a test"
```

เจอ's'ที่ตำแหน่ง index = s2-s1;

ตัวอย่าง ตรวจสอบว่า char ch ใช่เครื่องหมาย "+-*/" หรือไม่

```
if (strchr("+-*/", ch) != NULL)
    printf("%c is arithmetic operator\n",ch);
```

ตัวอย่าง เติมช่องว่าง ข้างหน้าและหลัง ตัวอักษรที่กำหนด +-*/^()

```
void add_space(char *str)
{ char buff[255] = "", old[255]="";
  int i, j;
  for (i=0,j = strlen(str); i<j; i++)
    {  if (strchr("+-*/^()",str[i]) != NULL)
         sprintf(buff,"%s %c ",old,str[i]); //หลีกเลี่ยงการ sprintf ใส่ตัวเอง
       else
         sprintf(buff,"%s%c",old,str[i]);
       strcpy(old,buff);  }
    strcpy(str,buff);  // คำตอบ
}
```

if (ch=='+'||ch=='-'||ch=='*'||ch=='/'||ch=='^'||ch=='('||ch==')')

Search pattern in string

- การค้นหาว่ามีข้อความที่เราสนใจอยู่ใน string ที่กำหนดหรือไม่ ใช้คำสั่ง string in string
  - char *strstr(char *str1, char *str2);
- ค่าที่ return จะเป็น address แรกที่พบข้อความ str2 ใน str1 แต่ถ้าค้นหาไม่เจอจะ return ค่าเป็น NULL

```
char *p, s1[40]="This is a test", s2[40]="is";
int  i ;
```

- ตรวจสอบว่ามีค่า s2 (is) อยู่ใน s1 ใช่หรือไม่

```
if (strstr(s1,s2) != NULL) {..เจอ....}
```

- ค้นหาตำแหน่งของ "test" ที่เก็บในตัวแปร s1

```
p = strstr(s1,"test");
if (p!=NULL)
    i = p-s1;
```

เจอที่ address P = FFEA

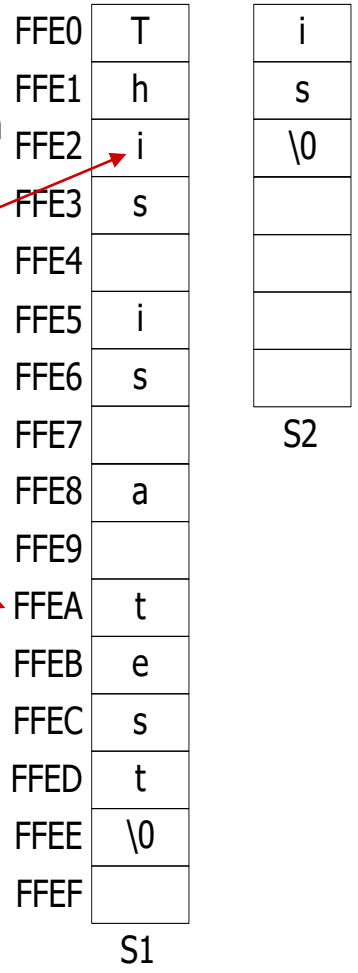
เจอ"test"ที่ตำแหน่ง index = p-s1;

- ตรวจสอบว่า s1 ขึ้นต้นด้วย "This" ใช่หรือไม่

```
if (strstr(s1,"This") == s1) {.....}
```

Address ของ This ใน S1

Address ของ S1



Split string to tokens

- คำสั่งการแยกสตริงออกเป็น token

  char *strtok(char *str, const char *delim)

  - เป็น sequence ของคำสั่งในการดึง token (string ย่อย) ออกจากสตริง str โดยกำหนดตัวอักษร delimiter เป็นตัวแบ่ง (ตัว delimiter จะถูกตัดทิ้ง)
  - delimiter อาจมีหลายตัวได้ เช่น "+-*/□" (จะใช้ '+','-','*','/','□' เป็นตัวแบ่ง)
  - ค่า return คือ pointer ที่ชี้ไปยังคำที่พบ

```
char  str[80]="□+-This--is□a*□sample/string",*token,word[10][40];
        token = strtok(str,"+-*/□");
```

จอง pointer สำหรับชี้ข้อมูล

token ชี้ไปยัง "This\0-is□a*□sample/string"

  - delimiter ที่อยู่ข้างหน้าจะถูกข้าม
  - delimiter ตัวแรกที่อยู่หลัง token จะถูกเปลี่ยนเป็น null (\0)

รูปแบบการค้นต่อเนื่องในรอบถัดๆ ไป ให้ใช้ NULL แทน

  - ใช้ในรอบต่อๆ ไป

```
while (token != NULL)
{ printf("%s\n",token);
  strcpy(word[count++],token);
  token = strtok(NULL, "+-*/□");
}
```

token ชี้ไปยัง "is\0a*□sample/string"
token ชี้ไปยัง "a\0□sample/string"
token ชี้ไปยัง "sample\0string"
token ชี้ไปยัง "string"

str="□+-This\0-is\0a\0□sample\0string"

Replace String

- ค้นหา oldstr ในสตริงstr ถ้าเจอให้แทนที่ด้วย newstr

```
char *strreplace(char *str, char *oldstr, char *newstr)
{ char *p, first[255], last[255];
  int  n;
      p = strstr(str,oldstr);
      if (p!=NULL)
      {  n = p-str;
        strcpycount(first,str,n);
        strpart(last,str,n+strlen(oldstr),strlen(str)-strlen(oldstr));
        sprintf(str,"%s%s%s",first,newstr,last);
      }
 return str;
}
```

Split String

- แบ่ง string str ออกเป็น 2 ส่วน ที่ตำแหน่ง n

```
void  strsprit(char *str,  char *first, char *last, int n)
{ int  i;
      for (i=0;i<n;i++)
         first[i] = str[i];
      first[i]='\0';
      for (i=n; i<strlen(str); i++)
         last[i-n] = str[i];
      last[i] = '\0';
}
```