Length of String

Lecturer: Jakkrit TeCho, Ph.D.

Ref : Assoc. Prof. Rangsit Sirirangsi

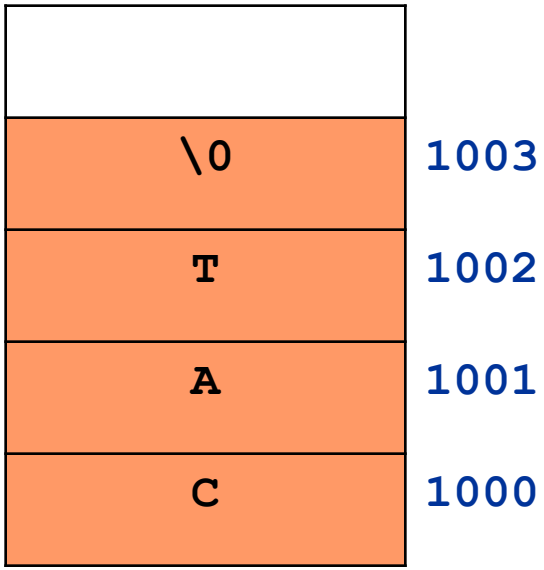
Pitchayanida Khumwichai

Declaration : String

- การประกาศตัวแปรแบบสตริงสามารถทำได้ 2 วิธี ดังนี้

```
char string[ ]= {'C','A','T','\0');
```

หรือ `char string[ ]= "CAT";` /*แบบนี้จะได้รับความนิยมมาก */

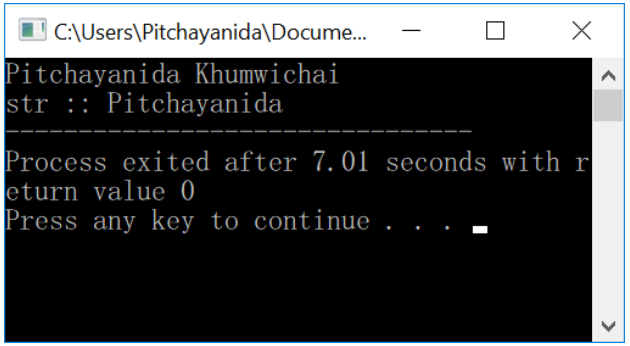


scanf() : Problem with String

- นอกจากนี้แล้วชื่อของตัวแปรแบบสตริงก็จะเป็นแอดเดรสเริ่มต้น เช่นเดียวกับอะเรย์ ดังนั้นเมื่อถูกใช้ในฟังก์ชัน scanf() จึงไม่จำเป็นต้องใส่เครื่องหมาย '&'(Ampersand) ไว้หน้าตัวแปรนั้น
- การรับค่าโดยใช้ scanf() จะมีปัญหาในการใช้งาน เนื่องจากฟังก์ชันจะใช้ช่องว่าง เป็นตัวกำหนดการสิ้นสุดการทำงาน ดังนั้นหากต้องการรับค่าในรูปของชื่อ และนามสกุลที่มีช่องว่างตรงกลางจะไม่สามารถทำได้อย่างสมบูรณ์

```
#include <stdio.h>
main()  {
        char str [25] ;
        scanf_s (“%s”, str, 25);
        printf(“\n%s”, str);
}
```

Output: This is a test
This



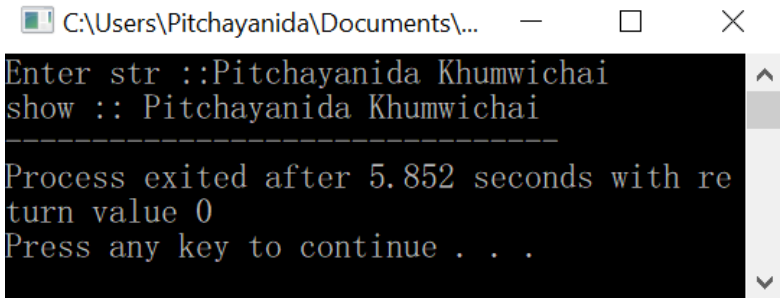
gets() & puts(): Function

- รับค่าจากแป้นพิมพ์และจะสิ้นสุดโดยการกด Return key เท่านั้น ส่วนฟังก์ชัน puts() จะทำหน้าที่ในการพิมพ์สตริงออกที่หน้าจอ

FORMAT: puts (varname or MSG);

gets (varname);

```
#include <string.h>
main()
{
   char  str[25] ;
   fgets(str, 25, stdin);
   printf("%s", str);          /* จะคล้ายกับฟังก์ชัน puts(str); */
}
```



ฟังก์ชันมาตรฐานที่ใช้กับสตริง

- เนื่องจากสตริงเป็นอะเรย์ของตัวอักขระ ดังนั้นในการจัดการข้อมูลในลักษณะนี้จำเป็นจะต้องกระทำคราวละตัวอักขระ ไม่ว่าจะเป็นการกำหนดค่า การเปรียบเทียบระหว่างค่าสตริง การหาความยาวของสตริง หรือแม้แต่การนำสตริงมาเชื่อมต่อกัน ซึ่งการกระทำเหล่านี้หากผู้ใช้จำเป็นจะต้องเขียนฟังก์ชันขึ้นมาเพื่อใช้เองก็จะทำให้สูญเสียเวลามาก ดังนั้นภาษาซีจึงได้สร้างฟังก์ชันมาตรฐานที่เกี่ยวข้องกับการจัดการข้อมูลแบบสตริงรวมไว้ในไฟล์ <string.h> ซึ่งฟังก์ชันที่ถูกเรียกใช้งานบ่อย ๆ จะมีดังนี้

strcpy(string1, string2);

- ฟังก์ชัน strcpy(string1, string2); เป็นฟังก์ชันที่ใช้ในกรณีที่ต้องการสำเนาข้อมูลจากตัวแปรสติงตัวที่สองที่อยู่ด้านขวามือ ไปยังตัวแปรสตริงตัวแรกที่อยู่ด้านซ้ายมือภายในวงเล็บ โดยการทำงานจะสิ้นสุดต่อเมื่อพบกับอักขระ nul

```
#include <stdio.h>
#include <string.h>
main()  {
        char str[] = "High Level";
        printf("Original string is %s ", str);
        strcpy_s( str, “Low Level”);
        printf("\nNow string is %s ", str);
}
```

Output: Original string is High Level
Now string is Low Level

Strings functions

- ฟังก์ชัน strlen() : string length เป็นฟังก์ชันมาตรฐานที่ใช้สำหรับหาค่าความยาวของตัวอักษร ที่อยู่ภายในตัวแปรแบบสตริง ฟังก์ชันนี้จะไม่นับรวมถึงอักขระ nul
- ฟังก์ชัน strcat() : string concat เป็นฟังก์ชันมาตรฐานที่ใช้สำหรับนำเอาสตริง 2 ข้อความมาเชื่อมต่อเข้าด้วยกัน

```
#include<stdio.h>
#include<string.h>
main() {
   char str[25]="BoyII";
   strcat_s(str, "men");
   printf("Length = %d and string is %s", strlen(str), str);
}
```

Output: Length = 8 and string is BoyIImen

Strings functions

- ฟังก์ชัน strcmp() : string compare ใช้สำหรับการเปรียบเทียบตัวแปรแบบสตริง 2 ตัวโดยปกติมักจะใช้ร่วมกับคำสั่ง if เพื่อแสดงผลการเปรียบเทียบเชิงตรรกะระหว่างสตริงสองตัว เช่น มากกว่า น้อยกว่า หรือ เท่ากับ เช่น

ผลการเปรียบเทียบ < 0	สตริงตัวแรกน้อยกว่าสตริงตัวที่สอง
ผลการเปรียบเทียบ $= 0$	สตริงตัวแรกเท่ากับสตริงตัวที่สอง
ผลการเปรียบเทียบ > 0	สตริงตัวแรกมากกว่าสตริงตัวที่สอง

2 Dimensional Array

```c
int main() {
    int arr[2][4] = {
        {1,2,3,4},
        {5,6,7,8}
    };

    for (int i=0; i<2; i++) {
        for (int j=0; j<4; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
}
```

Output:

```
1 2 3 4
5 6 7 8
```

(0,0)	(0,1)	(0,2)	(0,3)
1	2	3	4
5	6	7	8
(1,0)	(1,1)	(1,2)	(1,3)

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

int main()
{
  int i, enter = 0;
  char name[25];
  char list[5][25] = { "John","Nok","Marry","Patty","Osaka" };  /*2Array*/
  puts("Welcome to our party , enter your name please: ");
  fgets(name, 25, stdin);
  for (i = 0; i < 5; i++) {
    if (strcmp(&list[i][0], name) == 0) {
      enter = 1;

      if (enter == 1) {
        printf("Your're our member , please enter ");
      }
      else {
        printf("Guard! remove this person ");
      }
    }
  }
}
```

Output: **Welcome to our party,enter your name please: Dang**
Guard! removed this person

```
char list[5][25] = { "John","Nok","Marry","Patty","Osaka" };
```

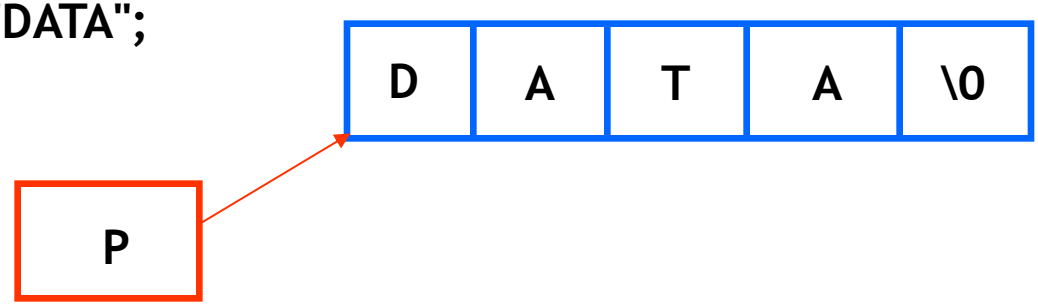
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	J	o	h	n	\0																				
1	N	o	k	\0																					
2	M	a	r	r	y	\0																			
3	P	a	t	t	y	\0																			
4	O	s	a	k	a	\0																			

list[0][0] => ‘J’
list[3][2] => ‘r’
list[0] => “John”
list[3] => “Patty”

Pointer To String

- จะมีลักษณะการทำงานคล้ายกับพอยน์เตอร์ที่ใช้กับตัวแปรแบบอะเรย์นั่นคือ ตัวแปรพอยน์เตอร์ที่ใช้จะถูกกำหนดให้ทำการรับค่าจากแอดเดรสเริ่มต้นของตัวแปรสตริง ส่วนความแตกต่างจะได้แก่ พอยน์เตอร์ที่ใช้กับสตริงจะสามารถใช้กับข้อความในรูปของสตริงได้โดยตรง ดังนั้นข้อความที่อยู่ภายในเครื่องหมายคำพูดใด ๆ สามารถใช้พอยน์เตอร์ประเภทนี้สำหรับรับค่าแอดเดรสเริ่มต้นของสตริงดังกล่าวได้

```
char *p;
   p = "DATA";
```



Pointer To String (cont.)

```
#include <stdio.h>
#include <string.h>
main()
{
   char *p;
   p = "DATA";
   while (*p)                    /* จะเหมือนกับ (*p!= '\0') */
   printf("%c", *p++);
}
```

- การตรวจสอบเงื่อนไขของ While Loop จะสิ้นสุดลง เมื่อพอยน์เตอร์ถูกเลื่อนตำแหน่งไปจนกระทั่งพบกับอักขระ nul ที่อยู่ตำแหน่งท้ายสุด ค่าอักขระดังกล่าวเมื่อปรากฏอยู่ในเงื่อนไขจะมีผลทำให้เงื่อนไขกลายเป็นเท็จ (0) ซึ่งจะเป็นการบังคับให้โปรแกรมหลุดออก Loop โดยอัตโนมัติ ดังนั้นการระบุเงื่อนไข

  while (*p) จึงมีความหมายเช่นเดียวกับ while (*p != '\0')

Pointer To String (cont.)

```
#include <stdio.h>
#include <string.h>
main() {
    char a[15] , *p;
    strcpy_s(a, "SILICON");
    puts(a);
    p = a;
    strcpy_s(p, "CONTROL");
    puts(a);
 }
```

Output: SILICON
CONTROL

- จากตัวอย่างจะพบว่ามีการกำหนดค่าแอดเดรสเริ่มต้นของอะเรย์ a ให้กับพอยน์เตอร์ p ดังนั้นพอยน์เตอร์ p จะชี้อยู่ที่ตำแหน่งแอดเดรสของสตริง a ด้วย เมื่อมีการเปลี่ยนค่าในตำแหน่งแอดเดรสดังกล่าว การแสดงผลลัพธ์ที่ผ่านพอยน์เตอร์ p ก็จะเปลี่ยนแปลงค่าตามการเปลี่ยนแปลงที่เกิดจากการกระทำที่เกิดขึ้นจากสตริง a ด้วย

การผ่านค่าตัวแปรแบบสตริงไปยังฟังก์ชัน

- ในกรณีที่ต้องการเรียกใช้ฟังก์ชันเพื่อทำการเปลี่ยนค่าของตัวแปรแบบสตริง การผ่านค่าไปยังฟังก์ชันจะสามารถทำได้ทั้งแบบโดยตรงและการผ่านค่าด้วยแอดเดรสของตัวแปร วิธีแรกจะไม่ค่อยได้รับความนิยมในการใช้ ทั้งนี้เนื่องจากการผ่านค่าโดยตรงจะทำการเก็บค่าตัวแปรไว้ในแสตก (Stack-เป็นกลไกการทำงานแบบหนึ่งที่ใช้ในการจัดการหน่วยความจำ) ก่อนที่จะทำการสำเนาข้อมูลไปยังตัวแปรภายในฟังก์ชัน การทำงานในลักษณะนี้สู้จะไม่ค่อยสะดวกนัก
- การส่งผ่านค่าตำแหน่งแอดเดรสยังฟังก์ชัน จะช่วยให้ผู้ใช้ทำงานกับข้อมูลที่มีขนาดใหญ่ได้สะดวกยิ่งขึ้น เช่น สตริงที่ประกอบไปด้วยอักขระเป็นจำนวนมาก หากทำการผ่านค่าทีละค่าอาจจะสูญเสียเวลาในการทำงานมาก แต่หากทำการผ่านค่าด้วยวิธีการผ่านค่าแอดเดรสของสตริงไปยังฟังก์ชัน ก็จะช่วยแก้ปัญหาในส่วนนี้ได้

Pointer Arithmatic (cont.)

```
#include <stdio.h>
#include <string.h>
void  modified ( char *);
main() {
    char test[] = "Minicomp";
    printf("The original string is %s\n", test);
    modified( test);
    printf("Now modified string to %s\n",test);
}
void  modified ( char *p)  {
    char temp[] = "Microcomputer";
    strcpy( p, temp);
}
```

Output: The original string is Minicomp
Now modified string to Microcomputer

Pointer Arithmatic (cont.)

```
#include <stdio.h>
#include <string.h>
void   modified ( char **);
main()   {
char *test = "Minicomp";
printf("The original is %s\n", test);
    modified( &test);
printf("Now modified string to %s\n",test);
}
void   modified ( char **p)   {
    strcpy( *p, "Microcomp");
```

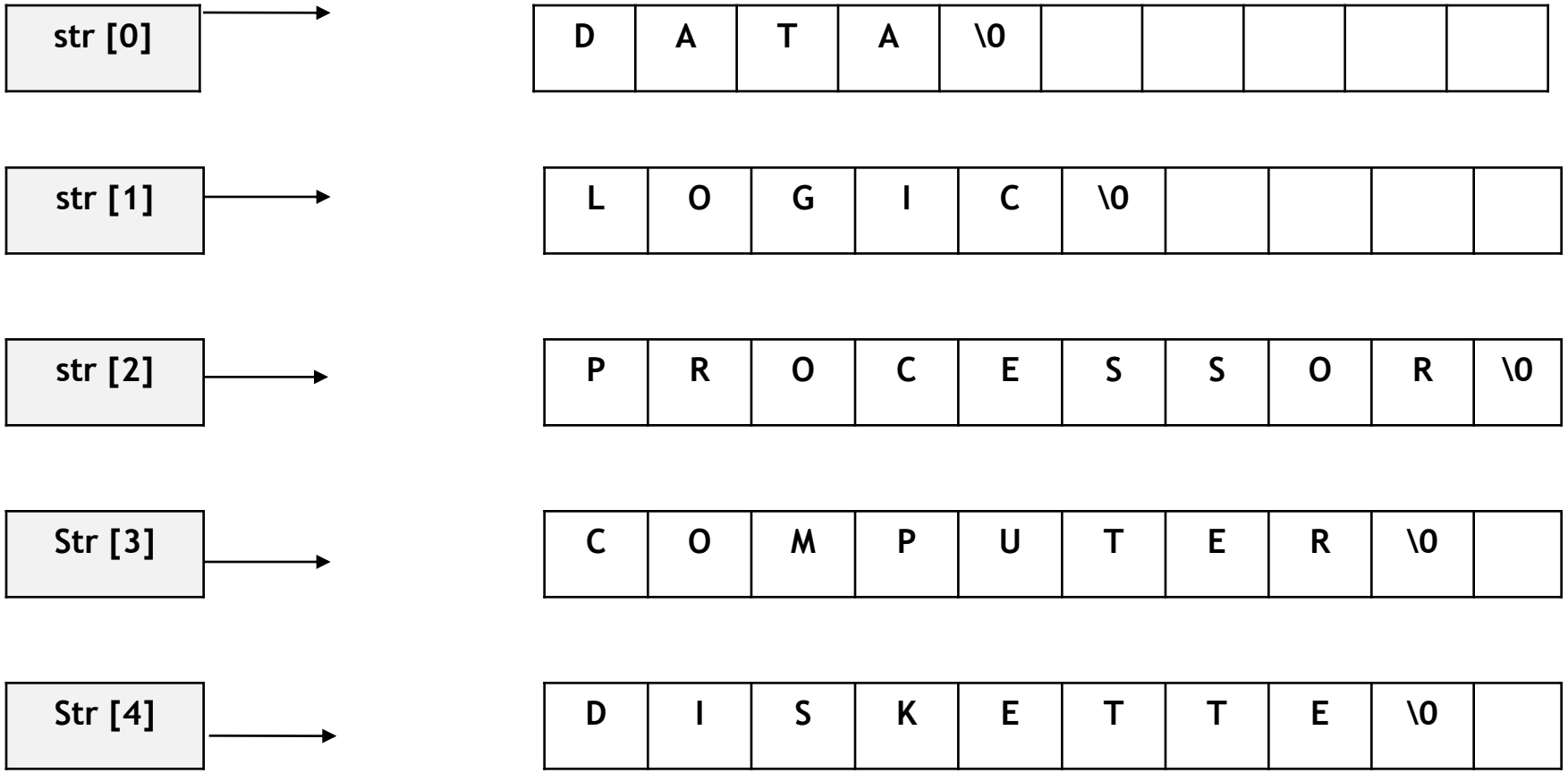
Output: The original string is Minicomp
Now modified string to Microcomputer

สตริงในรูปของอะเรย์ (Array of String)

- หรือบางครั้งที่เรียกว่า String tables นั่นคือการจัดการกับข้อมูลแบบสตริง โดยใช้ รูปแบบของตาราง ซึ่งประกอบไปด้วยจำนวนแถวและคอลัมน์ เช่นเดียวกับตัวแปรอะเรย์ แบบ 2 มิตินั่นเอง การดำเนินการที่เกี่ยวข้องกับสตริงลักษณะนี้ จะสามารถกระทำได้โดยใช้วิธีการเดียวกับการจัดการอะเรย์แบบ 2 มิติ
- การประกาศตัวแปรสตริงในรูปอะเรย์ จะสามารถทำได้สองวิธี คือ วิธีแรกจะเป็นการประกาศตัวแปรแบบอะเรย์ 2 มิติ เช่น
- *char str[5][10];*

Array of String

char str[][10]={ "DATA","LOGIC","PROCESSOR","COMPUTER","DISKETTE" };



```
#include <stdio.h>
#include <string.h>
void sort(char [][10]);
void display(char [][10]);

void sort(char str[][10])  {
 char temp[10];
 int i, j;
    for ( i = 0; i < 4; i++)
      for ( j = i +1 ; j < 4; j++)  {
           if (strcmp (str[i], str[j]) > 0 )
          {
                strcpy(temp,str[i]);
                strcpy(str[i],str[j]);
                strcpy(str[j],temp);
           }
        }
}
```

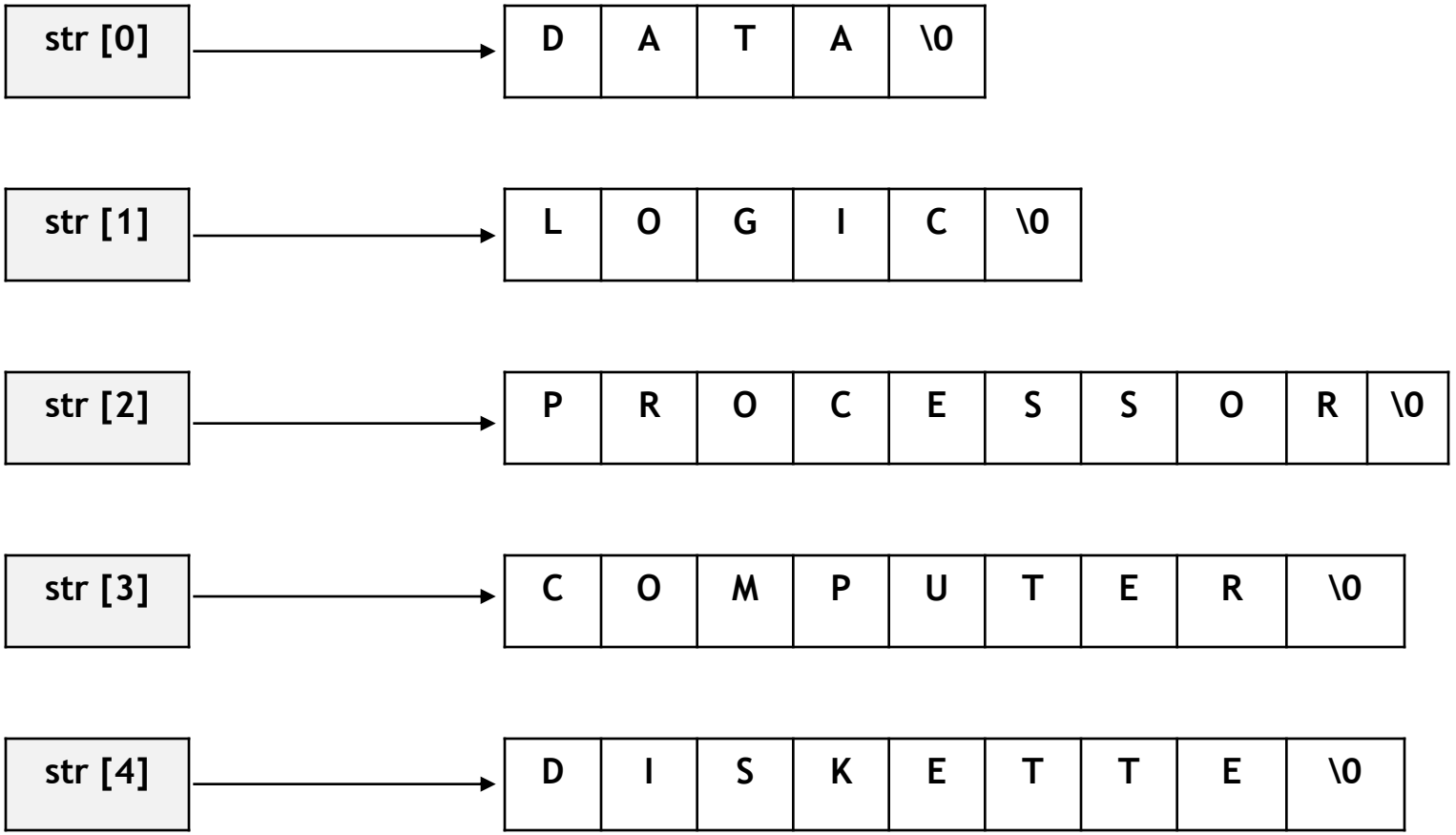
```
main()  {
   char ch[][10]={"data","computer","compiler","assembler"};
   sort(ch);
   display(ch);
}

void display(char st[][10])          {
 int i;
        for(i=0;i<4;i++)
          printf("%s\t",st[i]);
}
```

Output: assembler compiler computer data

String Pointer : Declaration

```
char *str[]={ "DATA", "LOGIC", "PROCESSOR", "COMPUTER", "DISKETTE" };
```



```
#include <conio.h>
#include <string.h>
void sort(char *[]);
void display(char **);
main()  {
   char *str[]={"data","computer","compiler","assembler"};
   sort(str);
   display(str);
}
void sort(char *str[])          {
 char *temp;
 int i, j;
    for ( i = 4; i < 4; i++)
       for ( j = i +1; j < 4; j ++)             {
            if (strcmp (str[i], str[j]) > 0 ) {
               temp  =  str[i];
                    str[i] =  str[j];
                 str[j] =  temp;
              }
       }
 }
```

```
void display(char **st)     {
        int i;
        for(i=0;i<4;i++)
          printf("%s  ",*(st+i));
}
```