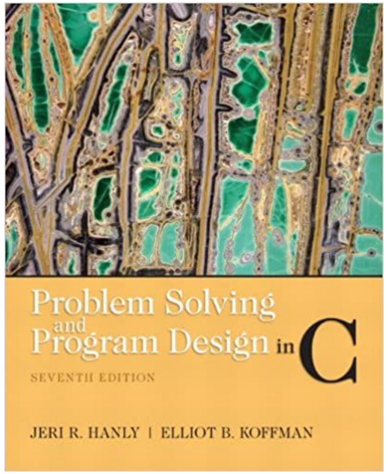
BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Department of Computer Science



Introduction to Computers and Programming (Comp 133)

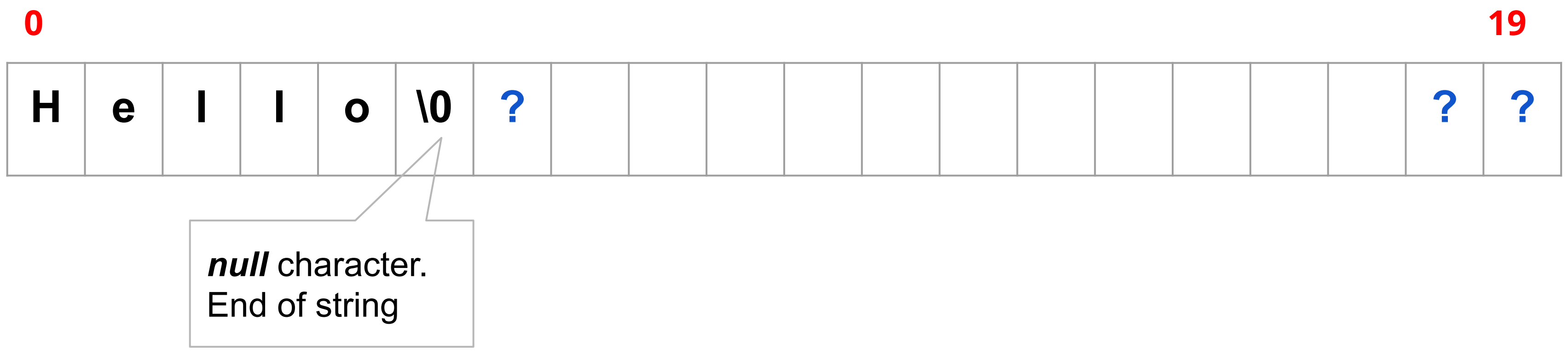
References :
Book : Problem Solving and Program Design in C (7th Edition) 7th Edition
Slides : Dr. Radi Jarrar , Dr. Abdallah Karakra , Dr. Majdi Mafarja.

Strings

Chapter 8

Strings

- String in C is implemented as an array.
- Declaring a string variable same as declaring an array of type **char**.
  - **char string_var[20];**
  - **string_var will hold strings from 0 to 19 characters long.**



Chapter 8

- Strings

String

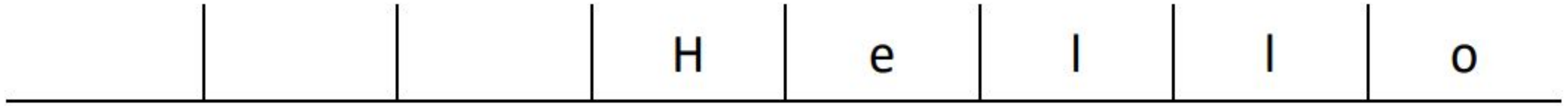
- String constant is a list of characters within double quotes e.g. **"Hello"** with the **'\0'** character being automatically appended at the end by the compiler.
  - char s[6] = "Hello"   as opposed to   char s[6] = { 'H', 'e', 'l', 'l', 'o', '\0' } ;

'H'	**'e'**	**'l'**	**'l'**	**'o'**	**'\0'**

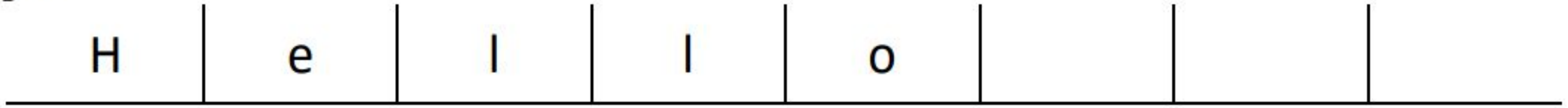
- To print out the contents of a string using **printf()** or **puts().**
  - **printf("%s", s) ; puts(s) ;**
- Strings can be read in using **scanf()** or **gets()**
  - **scanf("%s", s) ; // No need to use & with string**
  - **gets (s) ;**

String example

```
char str[6]="Hello";
printf("%8s\n",str); // %8s would print the string right align
```



```
char str[6]="Hello";
printf("%-8s\n", str); //  %-8s would print the string left
align
```



String Common Errors

- char my_char='A'; // correct
- char my_char="A"; // error
- char my_char [4]="A"; // correct

```
char one_string[4];
 one_string = "Hi";
```

```
error: assignment to expression with array type
```

Chapter 8

- String Library Functions
  - Assignment and Substring

String Library Functions

- The library **string.h** provides functions for ***substring***, ***concatenation***, string ***length***, string ***comparison*** ,***assignment functions.***
- The data type of the value returned by each string-building function is the pointer type **char ***
- **Functions :**
  - ***strcpy , strncpy***
  - ***strlen***
  - ***strcat, strncat***
  - ***strcmp, strncmp***
  - ***strtok***
  - ***size_t***

String Library Functions

TABLE 8.1 Some String Library Functions from string.h

Function	Purpose: Example	Parameters	Result Type

`strcpy`	Makes a copy of `source`, a string, in the character array accessed by `dest`: `strcpy(s1, "hello");`	`char *dest`
		`const char *source`
			`char *` h e l l o \0 ? ? ...

`strncpy`	Makes a copy of up to `n` characters from `source` in `dest`: `strncpy(s2, "inevitable", 5)` stores the first five characters of the source in `s1` and does NOT add a null character.	`char *dest`
		`const char *source`
		`size_t† n`
			`char *` i n e v i ? ? ...

`strcat`	Appends `source` to the end of `dest`: `strcat(s1, "and more");`	`char *dest`
		`const char *source`
			`char *` h e l l o a n d m o r e \0

`strncat`	Appends up to `n` characters of `source` to the end of `dest`, adding the null character if necessary: `strncat(s1, "and more", 5);`	`char *dest`
		`const char *source`
		`size_t† n`
			`char *` h e l l o a n d m \0 ?

`strcmp`	Compares `s1` and `s2` alphabetically; returns a negative value if `s1` should precede `s2`, a zero if the strings are equal, and a positive value if `s2` should precede `s1` in an alphabetized list: `if (strcmp(name1, name2) == 0)...`	`const char *s1`
		`const char *s2`
			`int`

`strncmp`	Compares the first `n` characters of `s1` and `s2` returning positive, zero, and negative values as does `strcmp`: `if (strncmp(n1, n2, 12) == 0) ...`	`const char *s1`
		`const char *s2`
		`size_t† n`
			`int`

`strlen`	Returns the number of characters in `s`, not counting the terminating null: `strlen("What")` returns `4`.	`const char *s`	`size_t`

`strtok`	Breaks parameter string source into tokens by finding groups of characters separated by any of the delimiter characters in delim. First call must provide both source and delim. Subsequent calls using NULL as the source string find additional tokens in original source. Alters source by replacing first delimiter following a token by '\0'. When no more delimiters remain, returns rest of source. For example, if `s1` is `"Jan.12,.1842"`, `strtok(s1,".,")` returns `"Jan"`, then `strtok (NULL,".,")` returns `"12"` and `strtok(NULL,".,")` returns `"1842"`. The memory in the right column shows the altered `s1` after the three calls to `strtok`. Return values are pointers to substrings of `s1` rather than copies.	`const char *source`
		`const char *delim`
			`char *` J a n \0 1 2 \0 1 8 4 2 \0

† `size_t` is an unsigned integer

String Library Functions

- ***strcpy*** function copies characters from **Source** to **Destination** up to and including the terminating null character and **returns Destination.**
- **Syntax : strcpy(Destination ,Source);**

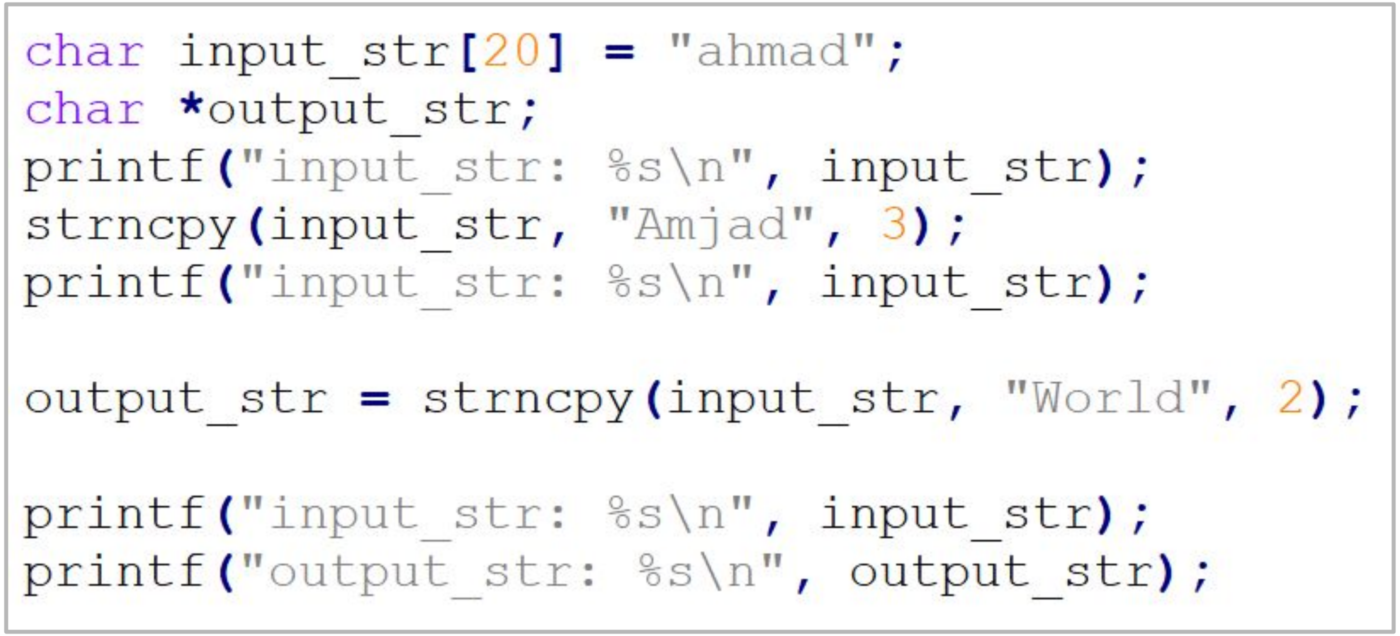
```
char input_str[20];
char *output_str;

 strcpy(input_str, "Hello");
 printf("input_str: %s\n", input_str);

 output_str = strcpy(input_str, "World");

 printf("input_str: %s\n", input_str);
 printf("output_str: %s\n", output_str);
```

Output

```
input_str: Hello
input_str: World
output_str: World
```

String Library Functions

- ***strcnpy*** Makes a copy of up to **n** characters from **src** to **dest** and including the terminating **null** character if length of **src is less than n**.
- **Syntax : strncpy** (**dest**, **src**, **n**)



```
char input_str[20] = "ahmad";
char *output_str;
printf("input_str: %s\n", input_str);
strncpy(input_str, "Amjad", 3);
printf("input_str: %s\n", input_str);

output_str = strncpy(input_str, "World", 2);

printf("input_str: %s\n", input_str);
printf("output_str: %s\n", output_str);
```

Output

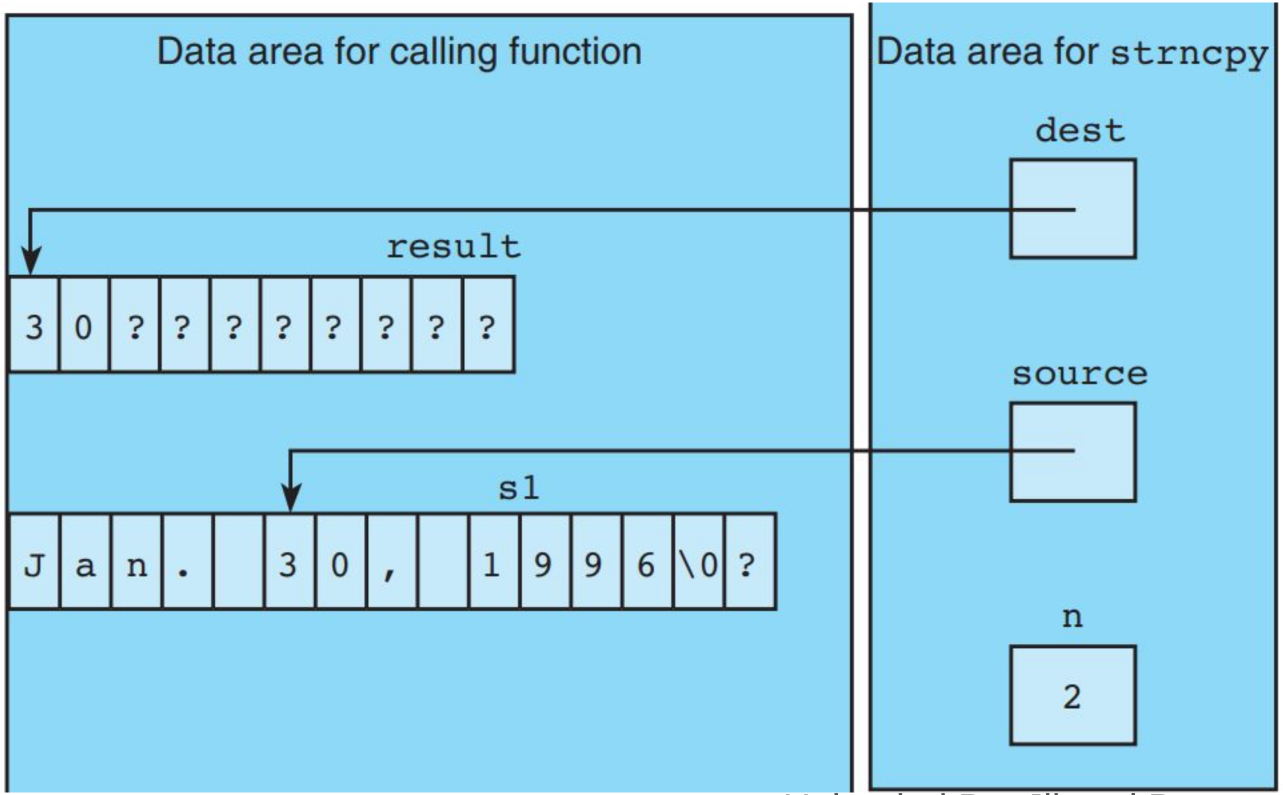
```
input_str: ahmad
input_str: Amjad
input_str: Wojad
output_str: Wojad
```

String Library Functions

- **Strcnpy : Eg. strncpy(result, &s1[5], 2);**
- **result[2] = '\0';**

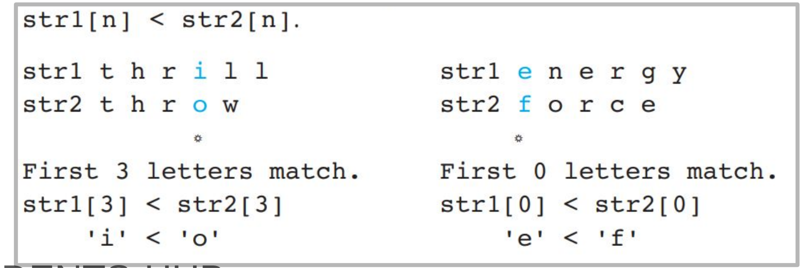
To know end of substring

strcpy always copies characters beginning with the initial character of a source string and continuing until a '\0' has been encountered (and copied).



String Library Functions

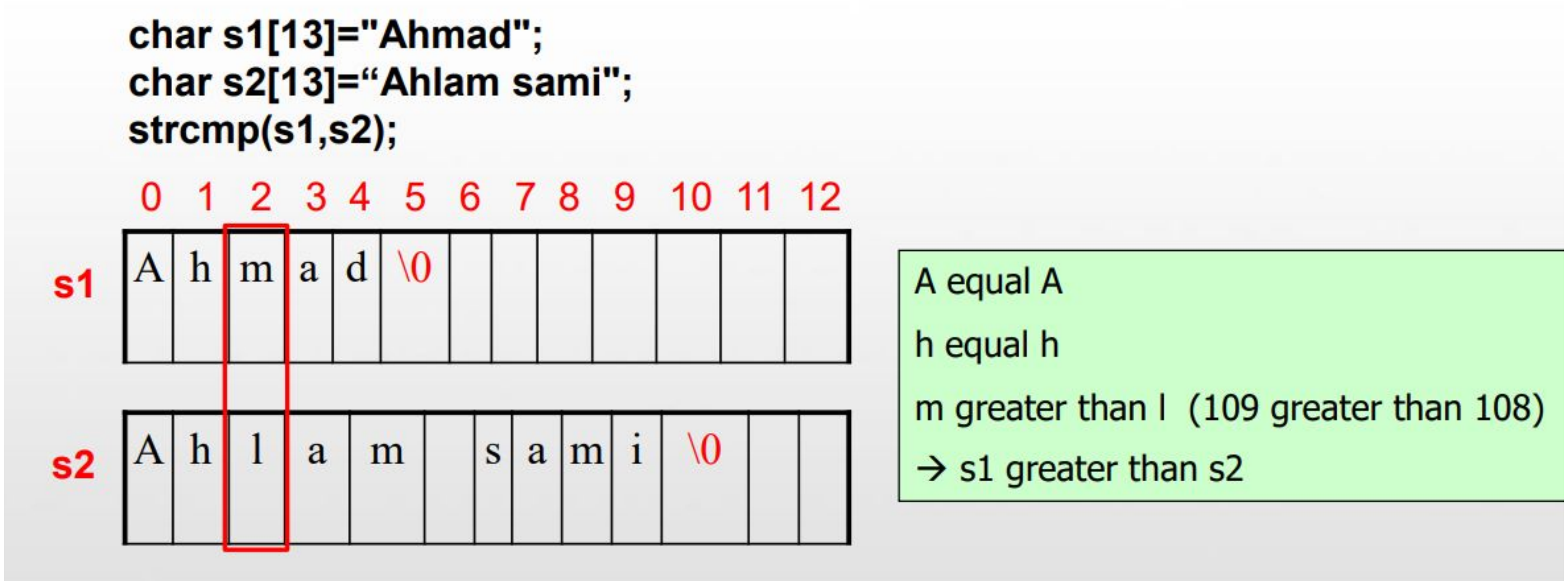
- ***strcmp*** Compare string1 and string2 to determine alphabetic order
    - **Syntax : strcmp (string 1, string2)**
- int **value** = strcmp (string1,string2);
- if Return **value** < 0 then it indicates string1 is less than string2
- if Return **value** > 0 then it indicates string1 is greater than string2
- if Return **value** = 0 then it indicates string1 is equal to string2
- **Note : Strcmp uses ASCII values to compare between two strings.**



```
str1[n] < str2[n].

str1 t h r i l l              str1 e n e r g y
str2 t h r o w                str2 f o r c e
           *                           *
First 3 letters match.        First 0 letters match.
str1[3] < str2[3]             str1[0] < str2[0]
    'i' < 'o'                     'e' < 'f'
```

String Library Functions

- ***strcmp***



```
char s1[13]="Ahmad";
char s2[13]="Ahlam sami";
strcmp(s1,s2);
```

A equal A

h equal h

m greater than l (109 greater than 108)

→ s1 greater than s2

String Library Functions

- **Syntax : strcmp (string 1, string2)**

```
char string1[20];
char string2[20];

strcpy(string1, "Ahmed");
strcpy(string2, "Ahmed");
printf("Return Value is : %d\n", strcmp( string1, string2));//0

strcpy(string1, "ahmed");
strcpy(string2, "ahmad");
printf("Return Value is : %d\n", strcmp( string1, string2));//4

strcpy(string1, "Ahmed");
strcpy(string2, "Mohammad");
printf("Return Value is : %d\n", strcmp( string1, string2));//-12
```

String1 = string2

String1 > string2

String1 < string2

Output

```
Return Value is : 0
Return Value is : 4
Return Value is : -12
```

String Library Functions

- **strncmp : Compare first n characters of two strings**
- **Syntax : strncmp (string 1, string2 , n)**

Output

```
Return Value is : 4

Return Value is : 0
```

```
strcpy(string1, "ahmed");
strcpy(string2, "ahmad");
printf("Return Value is : %d\n", strcmp( string1, string2));//4

 strcpy(string1, "ahmed");
strcpy(string2, "ahmad");
printf("Return Value is : %d\n", strncmp( string1, string2,3));//0
```

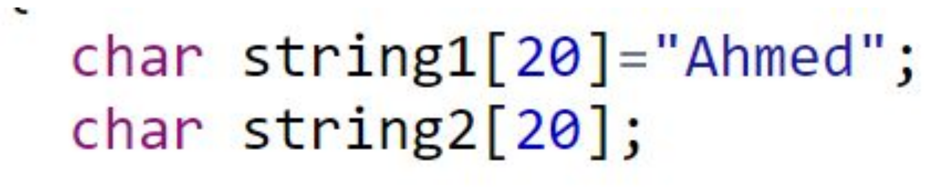
Compare first three characters **ahm**

String Library Functions

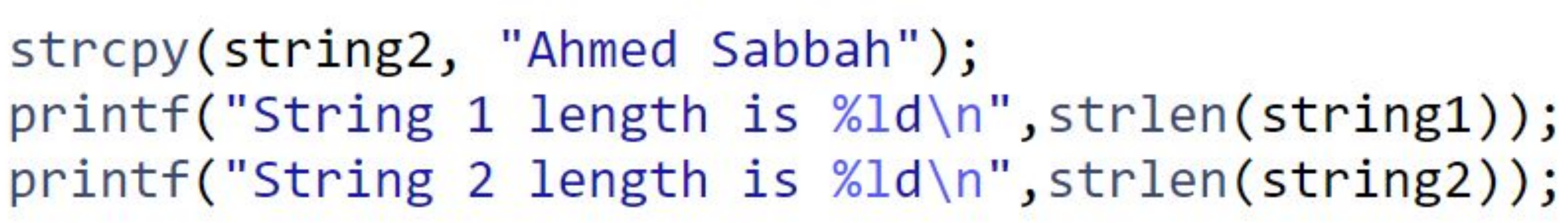
- **strlen : Determine the length of a string**
- **Syntax : strlen (string)**

Output

```
String 1 length is 5

String 1 length is 12
```



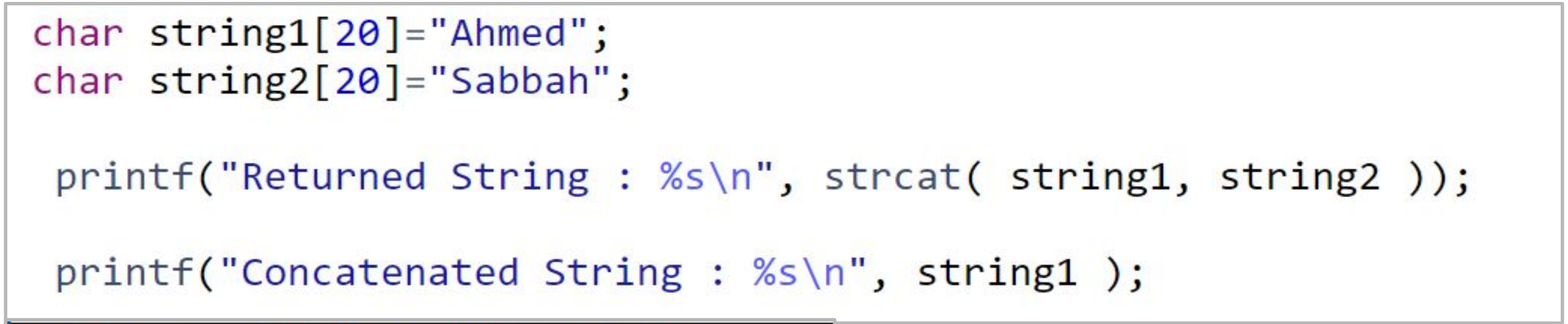
```
char string1[20]="Ahmed";
char string2[20];
```



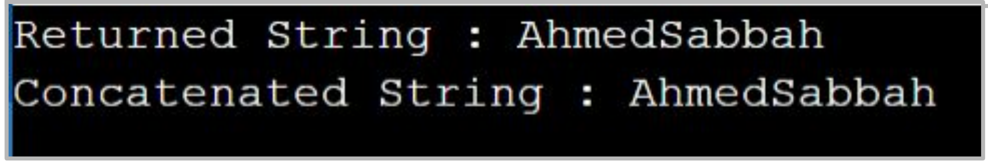
```
strcpy(string2, "Ahmed Sabbah");
printf("String 1 length is %ld\n",strlen(string1));
printf("String 2 length is %ld\n",strlen(string2));
```

String Library Functions

- **strcat : Concatenate string src to the string dest**
  - **Syntax : strcat (dest, src)**
- **strncat : Concatenate n characters from string src to the dest.**
  - **Syntax : strncat (dest, src,n) // n is integer**



```
char string1[20]="Ahmed";
char string2[20]="Sabbah";

 printf("Returned String : %s\n", strcat( string1, string2 ));

 printf("Concatenated String : %s\n", string1 );
```



```
Returned String : AhmedSabbah
Concatenated String : AhmedSabbah
```

String Library Functions

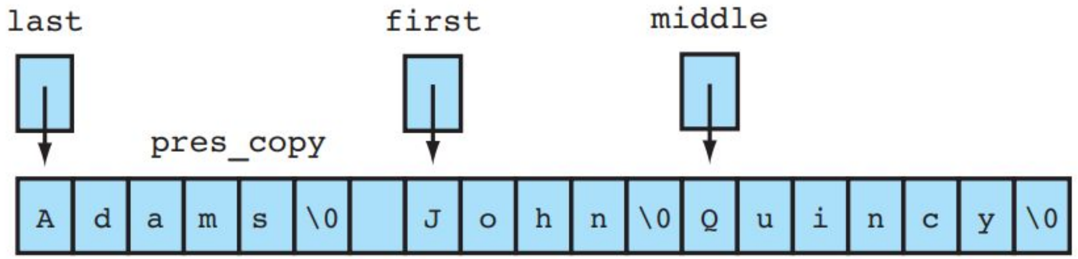
- **strtok :** This function **split** string into **tokens**, which are separated by any of the characters that are part of **delimiters**.

- **strtok** returns a pointer to the first token found in the string. A **NULL** pointer is returned if there are no tokens left to retrieve.
  - **Syntax : strtok(string , delim)**

String Library Functions

- **strtok**

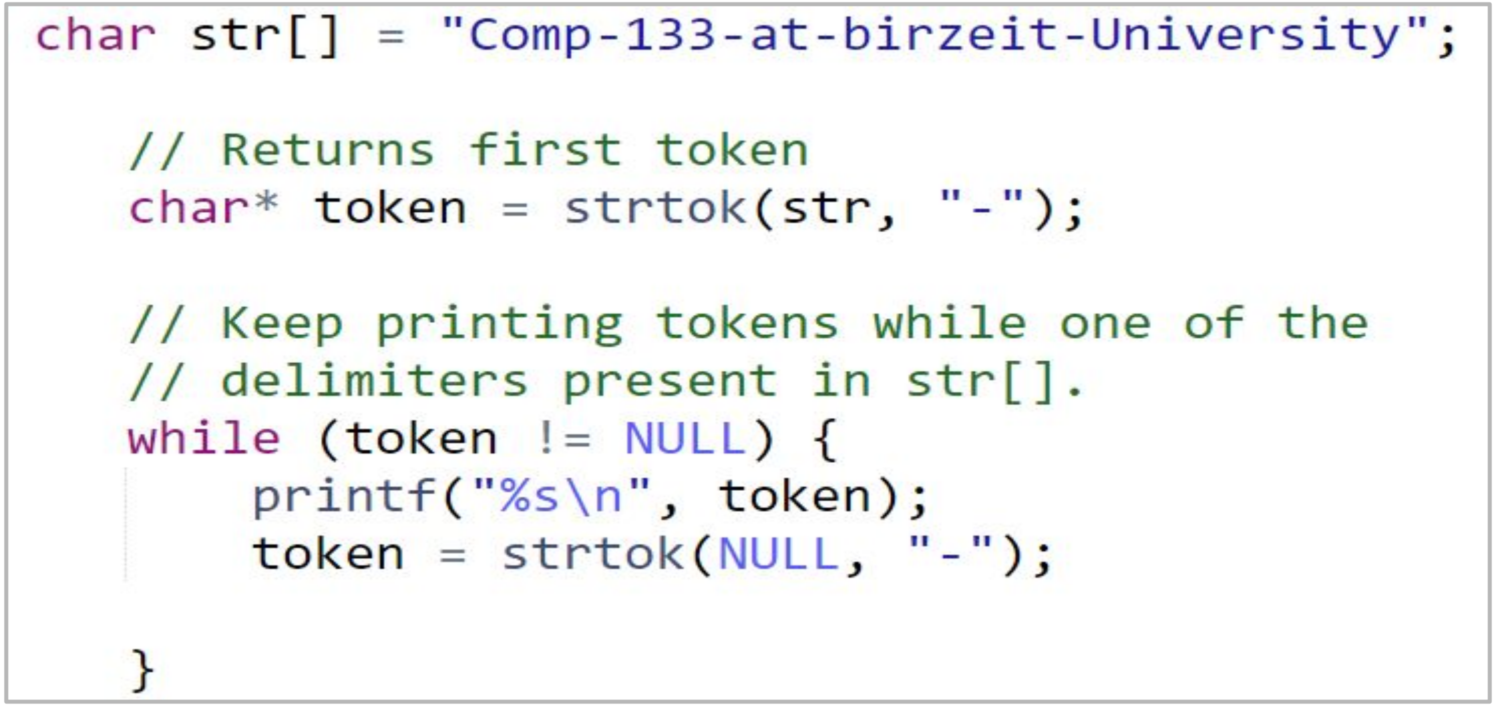
```
char *last, *first, *middle;
char pres[20] = "Adams, John Quincy";
char pres_copy[20];
strcpy(pres_copy, pres);
```



```
last = strtok(pres_copy, ", ");
first = strtok(NULL, ", ");
middle = strtok(NULL, ", ");
```

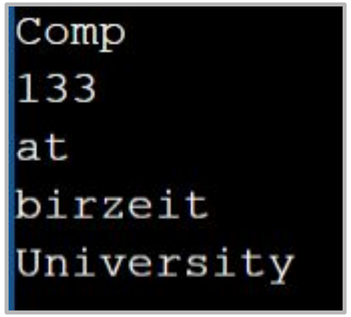
String Library Functions

- **strtok**



```c
char str[] = "Comp-133-at-birzeit-University";

    // Returns first token
    char* token = strtok(str, "-");

    // Keep printing tokens while one of the
    // delimiters present in str[].
    while (token != NULL) {
        printf("%s\n", token);
        token = strtok(NULL, "-");

    }
```

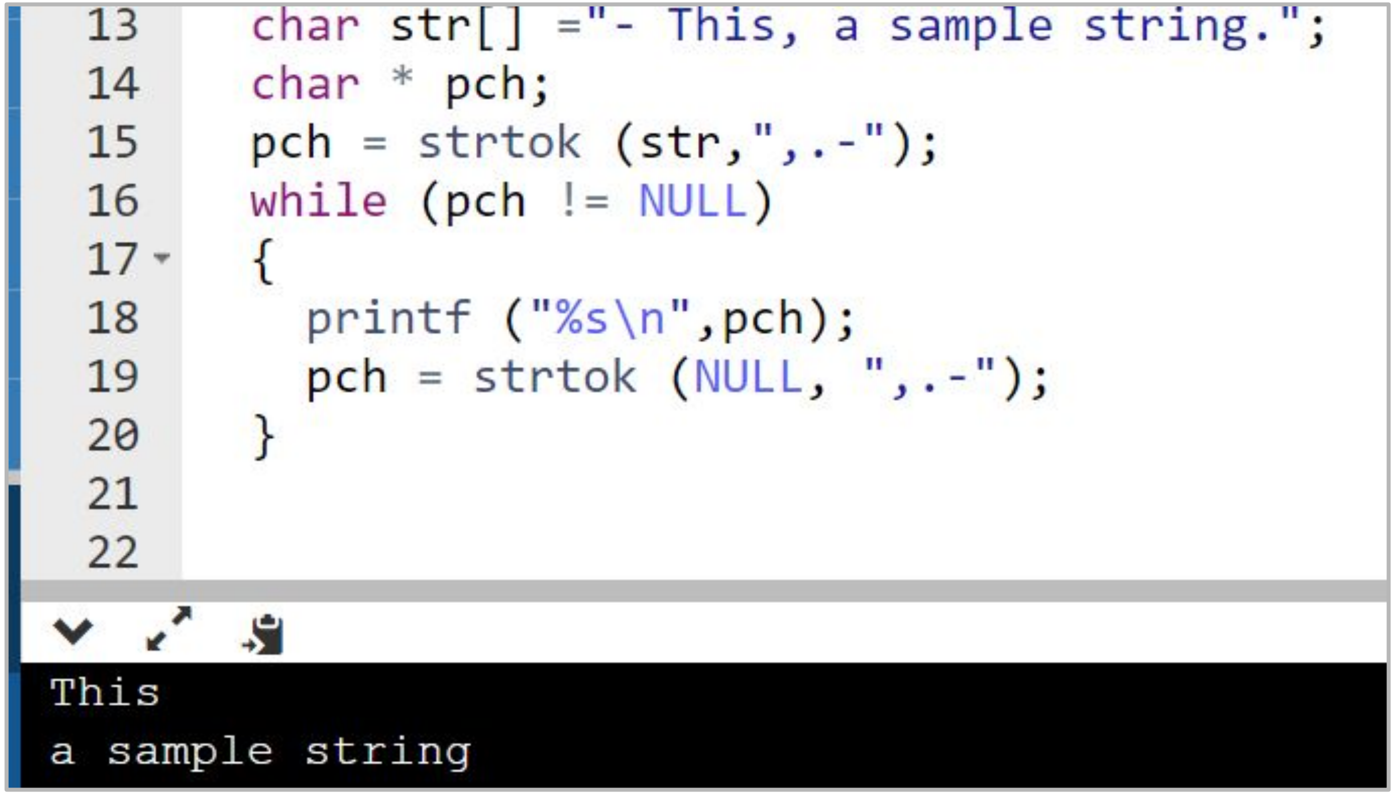
Output



```
Comp
133
at
birzeit
University
```

String Library Functions

- **strtok**

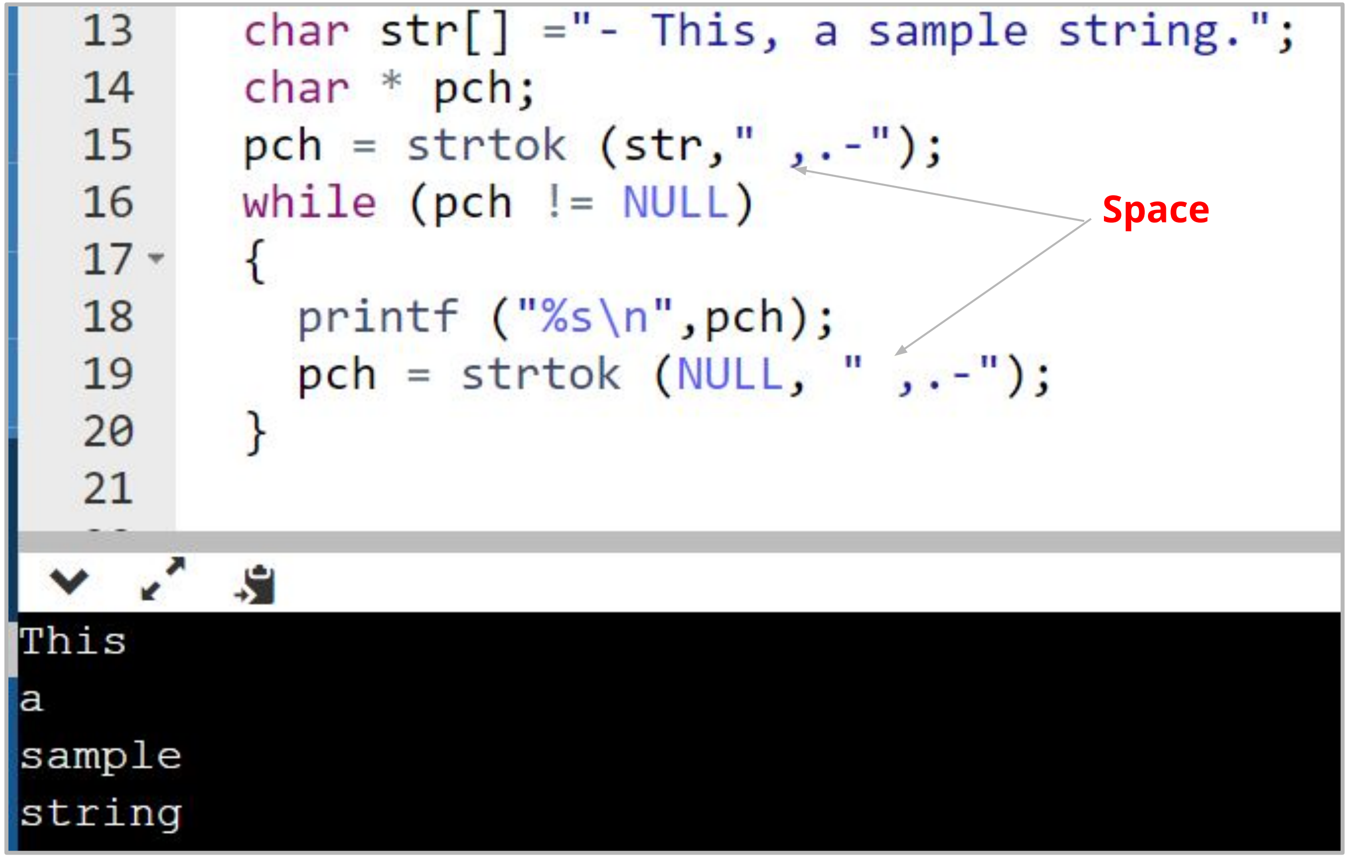


```
char str[] ="- This, a sample string.";
char * pch;
pch = strtok (str,",.-");
while (pch != NULL)
{
  printf ("%s\n",pch);
  pch = strtok (NULL, ",.-");
}
```

```
This
a sample string
```

String Library Functions

- **strtok**



```c
char str[] ="- This, a sample string.";
char * pch;
pch = strtok (str," ,.-");
while (pch != NULL)
{
  printf ("%s\n",pch);
  pch = strtok (NULL, " ,.-");
}
```

```
This
a
sample
string
```

Chapter 8

- Arrays of Strings

Arrays of Strings

- An array of strings is in fact a two dimensional array of characters
- **Row** index is used to access the **individual row strings** and where the **column** index is the **size of each string**,
    - Example : **char str_array[10] [30] ;**
    - **str_array** is an array of **10** strings each one has a maximum size of **29** characters the one extra for the terminating **null (\0)** character

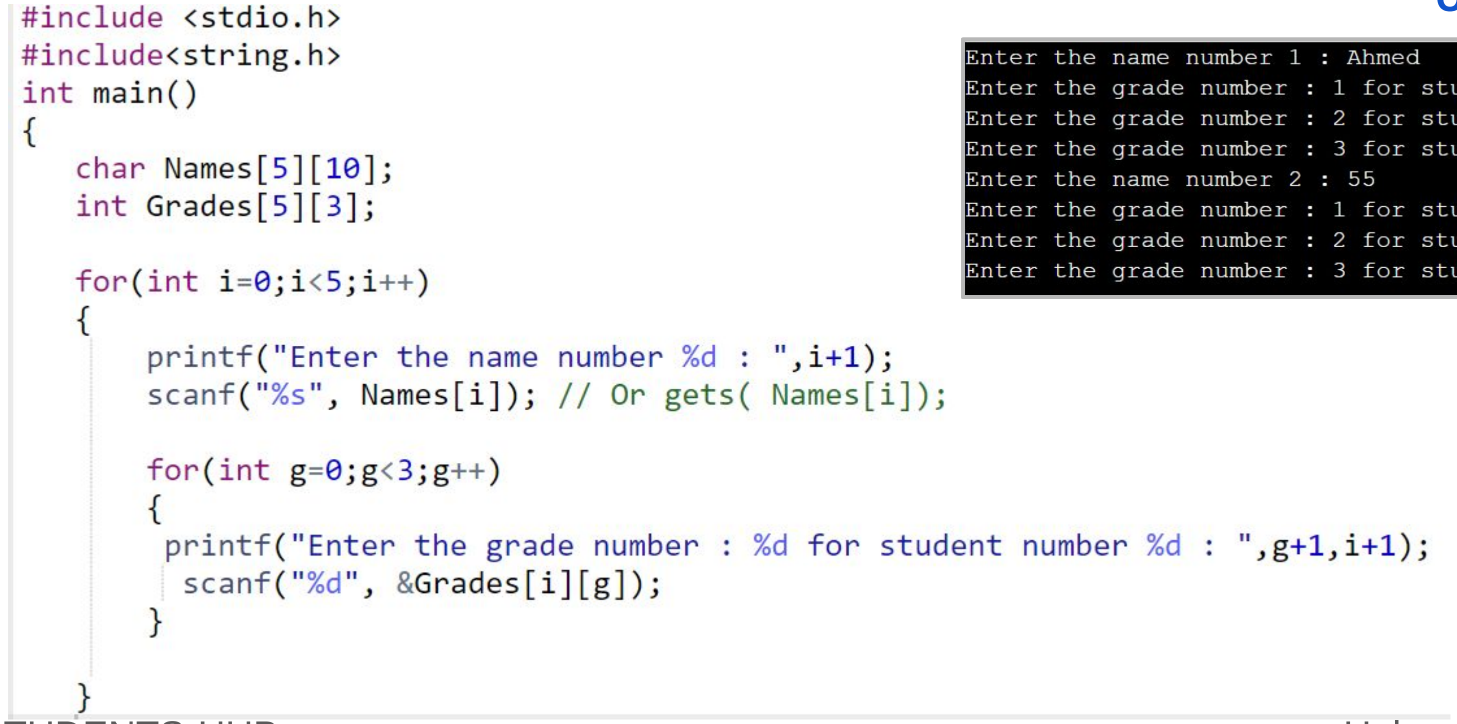
Arrays of Strings

- char week_days[7][13]={"Monday","Tuesday","Wednesday",...}

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	M	o	n	d	a	y	\0	?	?	?	?	?	?
1	T	u	e	s	d	a	y	\0	?	?	?	?	?
2	W	e	d	n	e	s	d	a	y	\0	?	?	?
3													
4													
5													
6													

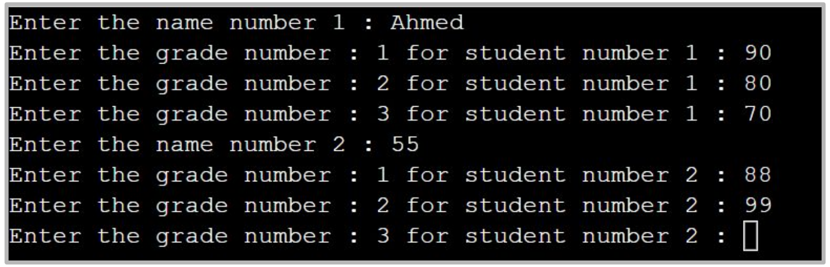
Arrays of Strings

- Write a program to read the names of 5 students and also their grades (three grades for each students),and save them



```c
#include <stdio.h>
#include<string.h>
int main()
{
    char Names[5][10];
    int Grades[5][3];

    for(int i=0;i<5;i++)
    {
        printf("Enter the name number %d : ",i+1);
        scanf("%s", Names[i]); // Or gets( Names[i]);

        for(int g=0;g<3;g++)
        {
         printf("Enter the grade number : %d for student number %d : ",g+1,i+1);
          scanf("%d", &Grades[i][g]);
        }

    }
```

Output

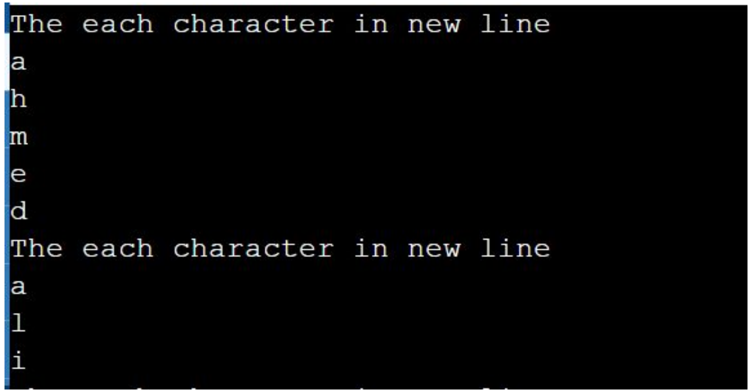


```
Enter the name number 1 : Ahmed
Enter the grade number : 1 for student number 1 : 90
Enter the grade number : 2 for student number 1 : 80
Enter the grade number : 3 for student number 1 : 70
Enter the name number 2 : 55
Enter the grade number : 1 for student number 2 : 88
Enter the grade number : 2 for student number 2 : 99
Enter the grade number : 3 for student number 2 :
```

Arrays of Strings

- Print out the previous example

First two Output



```
The each character in new line
a
h
m
e
d
The each character in new line
a
l
i
```

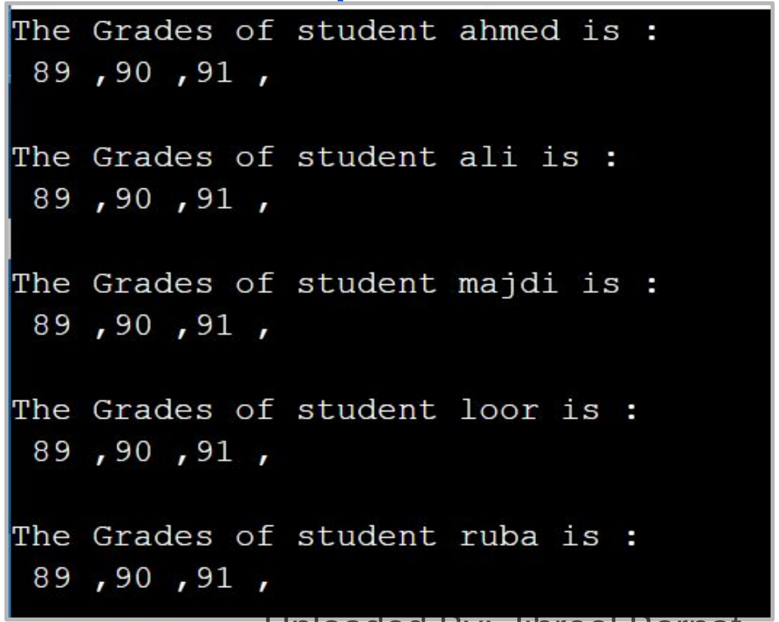
```
for(int i=0;i<5;i++)
{
    printf("The each character in new line \n");
  for(int j=0;Names[i][j] != '\0';j++)
    {

        putchar ( Names[i][j] ) ;// Or printf("%c",Names[i][j])
        putchar('\n');

    }
}
```

Arrays of Strings

- Print out the previous example

```
for(int i=0;i<5;i++)
{
    printf("\n\nThe Grades of student %s is :\n ",Names[i]);


  for(int j=0;j<3;j++)
    {

     printf("%d ,",Grades[i][j]) ;

    }
}
```

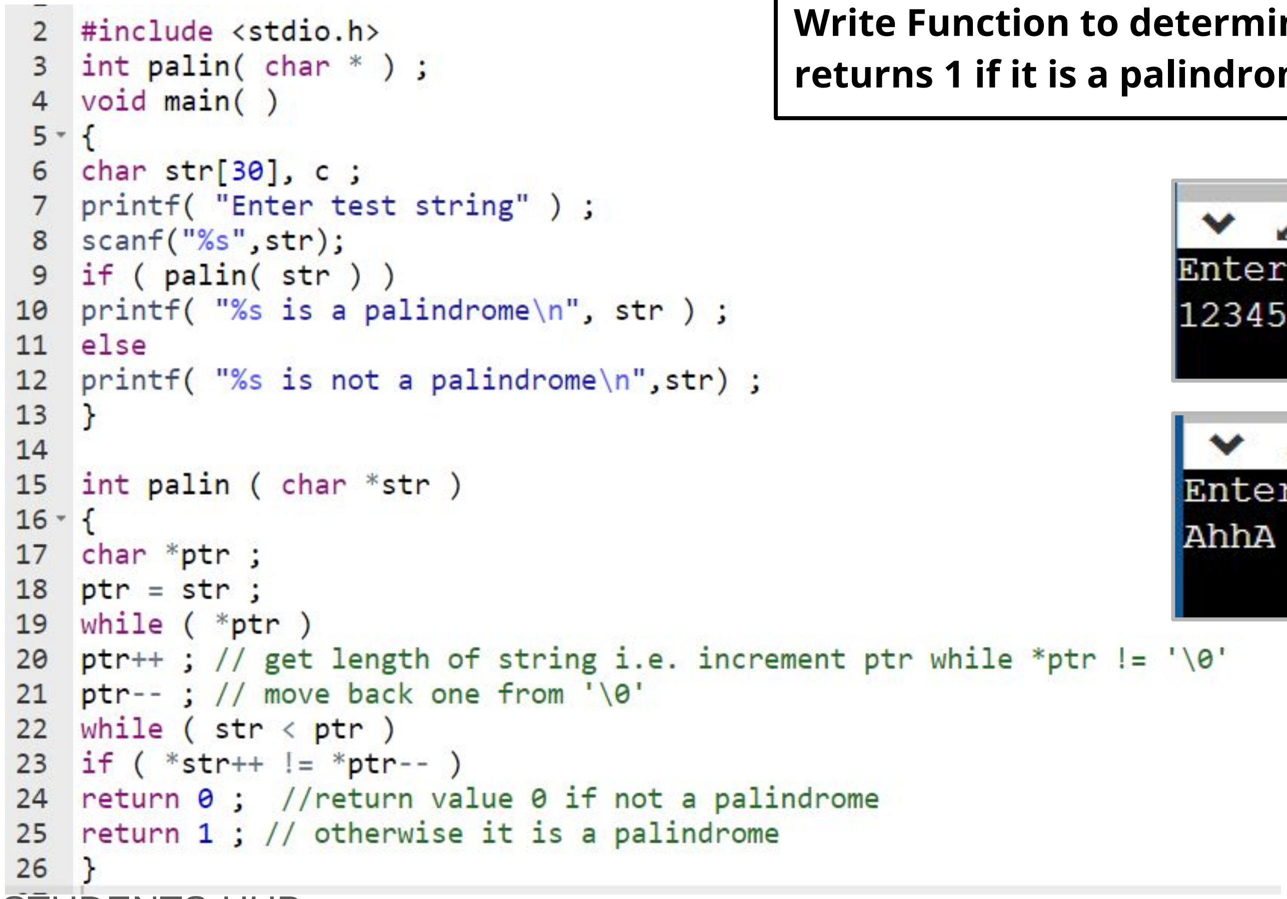
Output



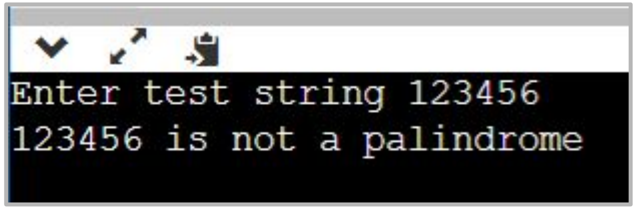
```
The Grades of student ahmed is :
 89 ,90 ,91 ,

The Grades of student ali is :
 89 ,90 ,91 ,

The Grades of student majdi is :
 89 ,90 ,91 ,

The Grades of student loor is :
 89 ,90 ,91 ,

The Grades of student ruba is :
 89 ,90 ,91 ,
```

Strings and pointers

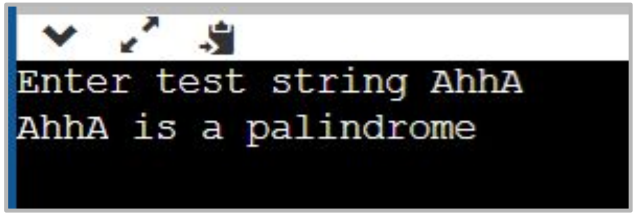
Write Function to determine if array is a palindrome. returns 1 if it is a palindrome, 0 otherwise.



```c
#include <stdio.h>
int palin( char * ) ;
void main( )
{
char str[30], c ;
printf( "Enter test string" ) ;
scanf("%s",str);
if ( palin( str ) )
printf( "%s is a palindrome\n", str ) ;
else
printf( "%s is not a palindrome\n",str) ;
}

int palin ( char *str )
{
char *ptr ;
ptr = str ;
while ( *ptr )
ptr++ ; // get length of string i.e. increment ptr while *ptr != '\0'
ptr-- ; // move back one from '\0'
while ( str < ptr )
if ( *str++ != *ptr-- )
return 0 ;  //return value 0 if not a palindrome
return 1 ; // otherwise it is a palindrome
}
```



```
Enter test string 123456
123456 is not a palindrome
```



```
Enter test string AhhA
AhhA is a palindrome
```

Chapter 8

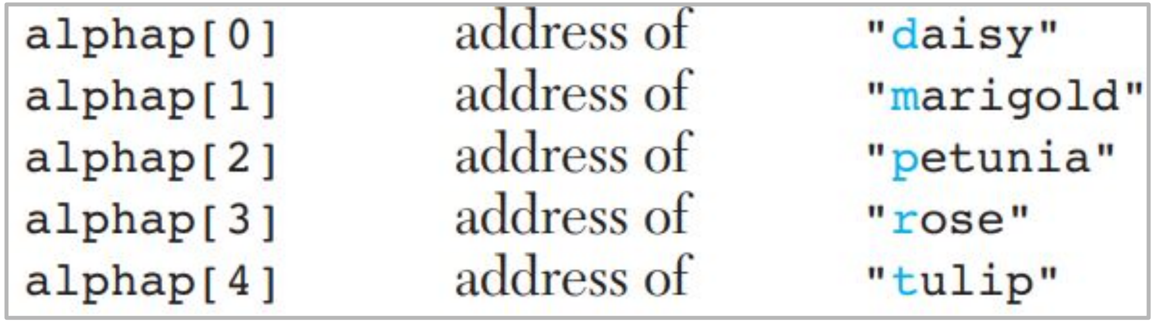
- Arrays of pointers

Arrays of Pointers

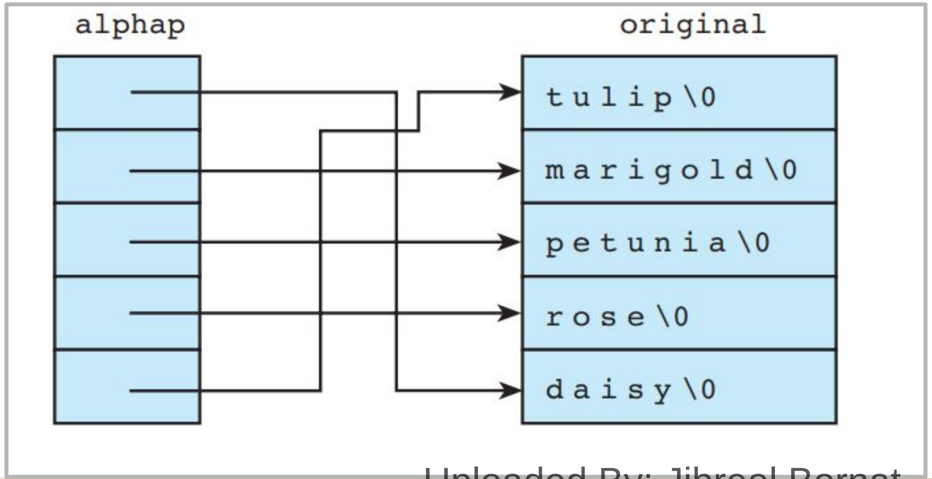
- In C declare arrays of pointers same as any other 'type'.
  - **int *x[10] ; //** declares an array of ten integer pointers
- Pointers point to a variable one
  - **int var;**
  - **x[2] = &var ;**
- To access the value pointed to by x[2]
  - ***x[2]=9 ;**

Arrays of Pointers

- **char *alphap[5];**



alphap[0]	address of	"daisy"
alphap[1]	address of	"marigold"
alphap[2]	address of	"petunia"
alphap[3]	address of	"rose"
alphap[4]	address of	"tulip"



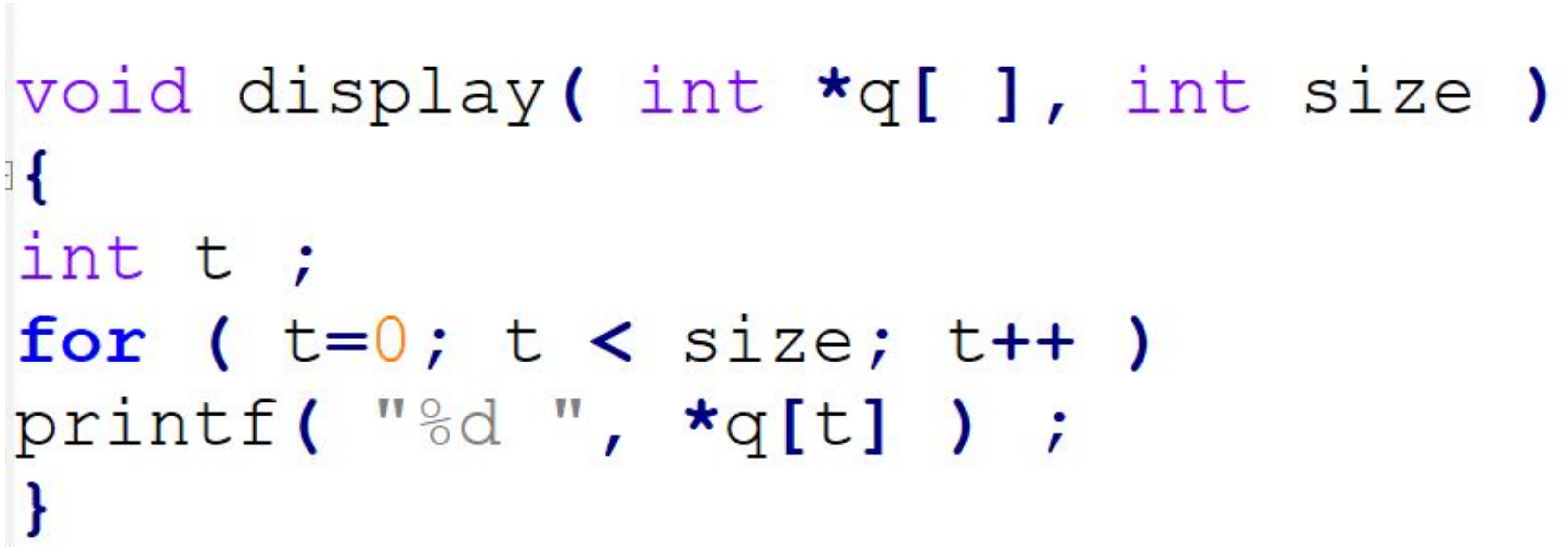
Arrays of Pointers

- **Arrays of String Constants**

```
char month[12][10] = {"January", "February", "March", "April",
                      "May", "June", "July", "August", "September",
                      "October", "November", "December"};
char *month[12] = {"January", "February", "March", "April", "May",
                   "June", "July", "August", "September",
                   "October", "November", "December"};
```

Arrays of Pointers

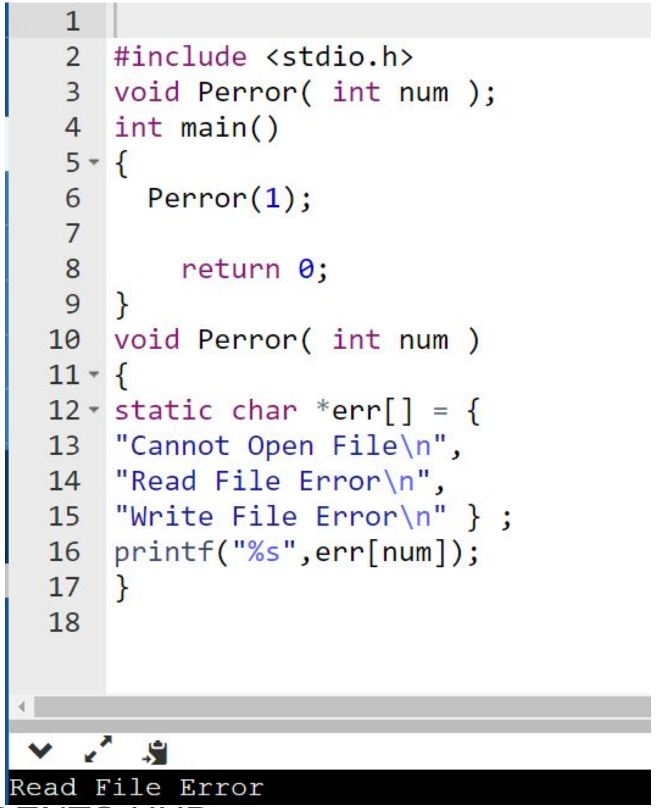
- Passing this array to a function



```
void display( int *q[ ], int size )
{
int t ;
for ( t=0; t < size; t++ )
printf( "%d ", *q[t] ) ;
}
```

Arrays of Pointers

- A common use of pointer arrays is to hold arrays of strings.



```c

#include <stdio.h>
void Perror( int num );
int main()
{
  Perror(1);

    return 0;
}
void Perror( int num )
{
static char *err[] = {
"Cannot Open File\n",
"Read File Error\n",
"Write File Error\n" } ;
printf("%s",err[num]);
}

```

```
Read File Error
```

#include < ctype.h>

Facility	Checks	Example

`isalpha`	if argument is a letter of the alphabet	`if (isalpha(ch))`
		`    printf("%c is a letter\n", ch);`

`isdigit`	if argument is one of the ten decimal digits	`dec_digit = isdigit(ch);`

`islower` `(isupper)`	if argument is a lowercase (or uppercase) letter of the alphabet	`if (islower(fst_let)) {`
		`    printf("\nError: sentence ");`
		`    printf("should begin with a ");`
		`    printf("capital letter.\n");`
		`}`

`ispunct`	if argument is a punctuation character, that is, a noncontrol character that is not a space, a letter of the alphabet, or a digit	`if (ispunct(ch))`
		`    printf("Punctuation mark: %c\n",`
		`           ch);`

`isspace`	if argument is a whitespace character such as a space, a newline, or a tab	`c = getchar();`
		`while (isspace(c) && c != EOF)`
		`    c = getchar();`

Facility	Converts	Example

`tolower` `(toupper)`	its lowercase (or uppercase) letter argument to the uppercase (or lower-case) equivalent and returns this equivalent as the value of the call	`if (islower(ch))`
		`    printf("Capital %c = %c\n",`
		`           ch, toupper(ch));`

Thank You.