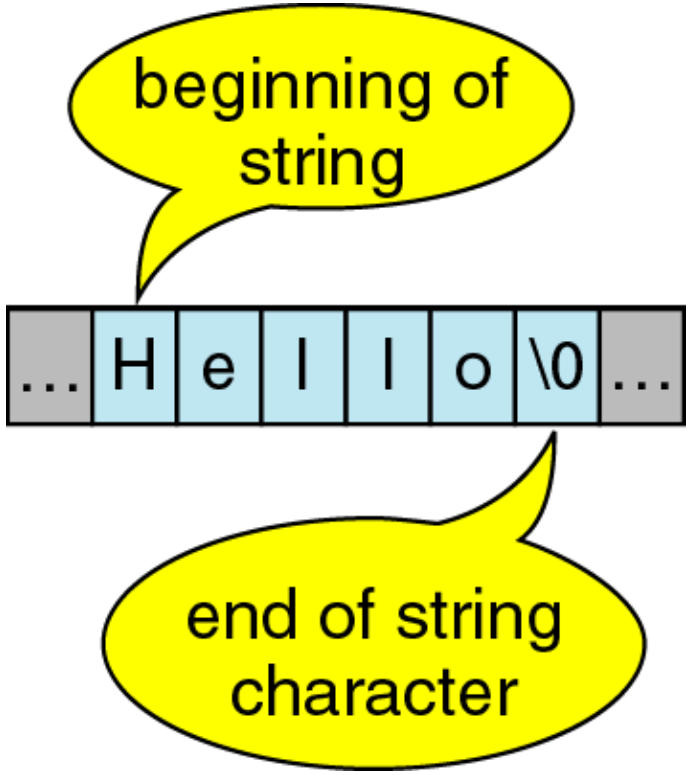
Strings

Outline

- **Strings**
  - **Representation in C**
  - **String Literals**
  - **String Variables**
  - **String Input/Output**
    - **printf, scanf, gets, fgets, puts, fputs**
  - **String Functions**
    - **strlen, strcpy, strncpy, strcmp, strncmp, strcat, strncat, strchr, strrchr, strstr, strspn, strcspn, strtok**
  - **Reading from/Printing to Strings**
    - **sprintf, sscanf**

Strings in C

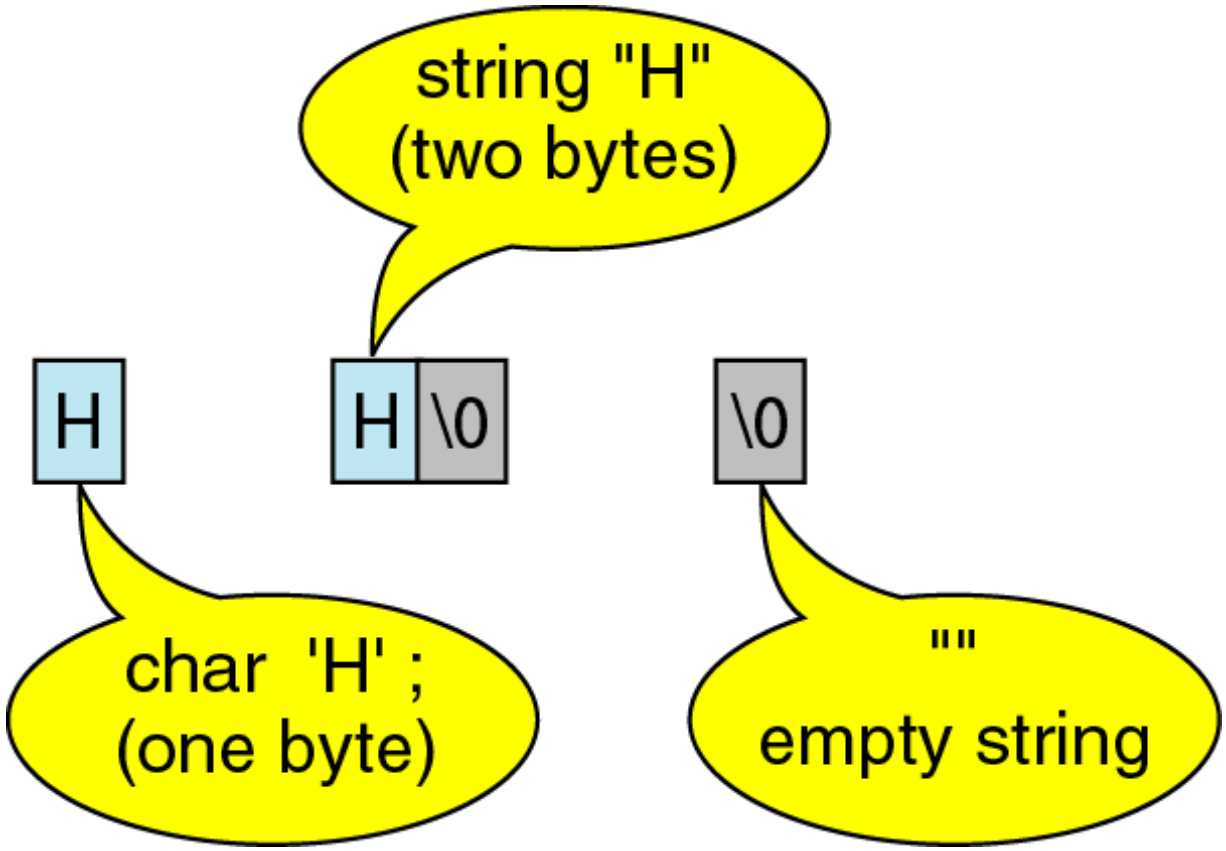
- **No explicit type, instead strings are maintained as arrays of characters**
- **Representing strings in C**
  - **stored in arrays of characters**
  - **array can be of any length**
  - **end of string is indicated by a *delimiter*, the zero character '\0'**



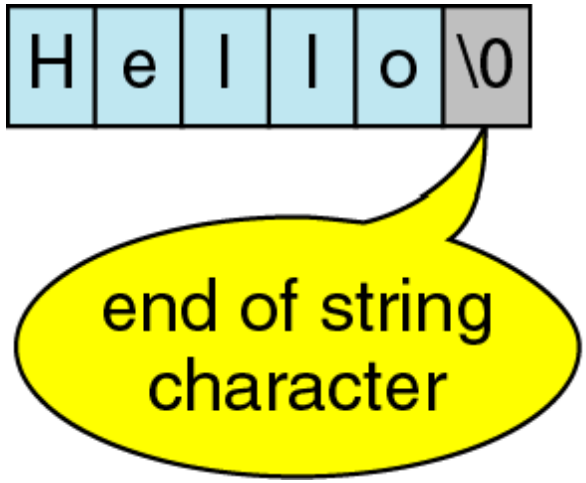
String Literals

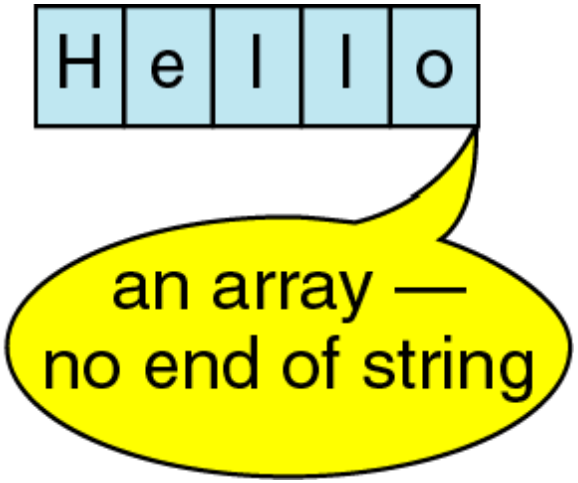
- String literal values are represented by sequences of characters between double quotes)
- Examples
  - "hello"
  - "" - empty string
- "H" versus 'H'
  - 'H' is a single character value (stored in 1 byte) as the ASCII value for H (72)
  - "H" is an array with two characters, the first is H, the second is the character value \0

String Literals

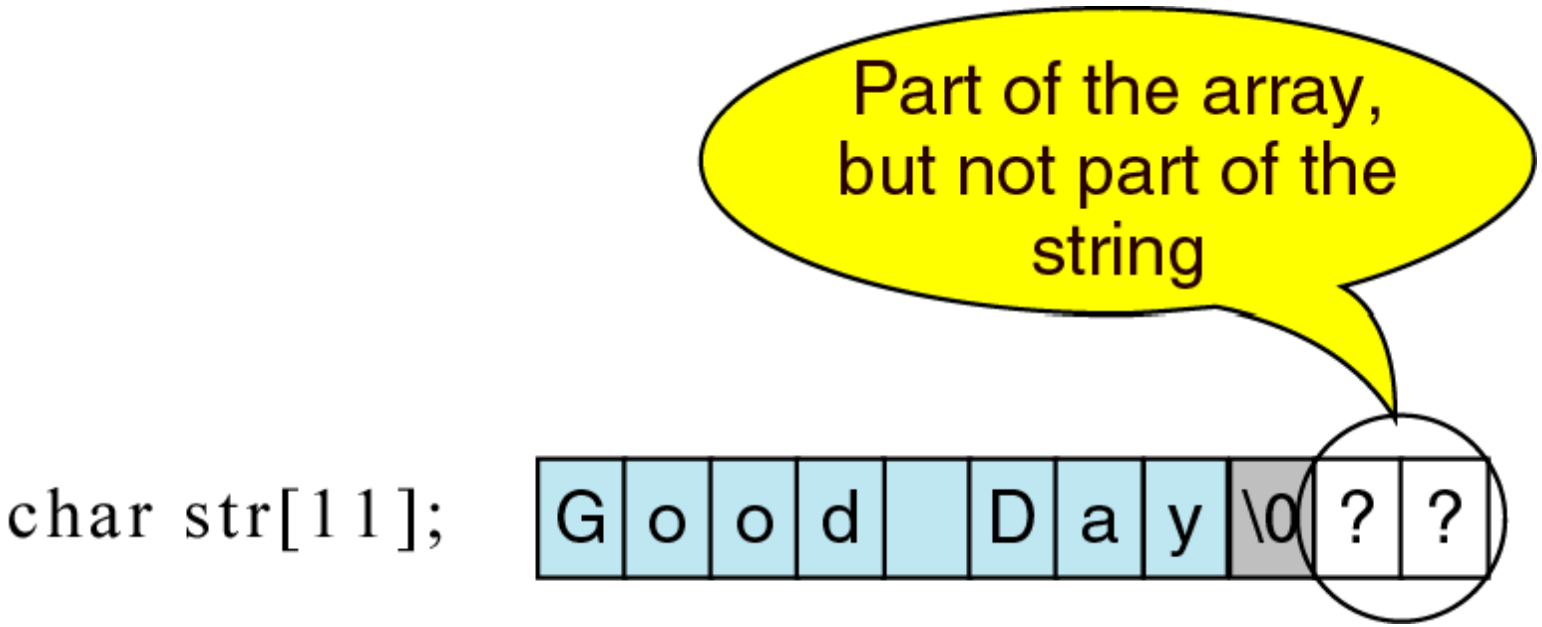


String Literals





String Literals



Referring to String Literals

- String literal is an array, can refer to a single character from the literal as a character
- Example:

  printf("%c","hello"[1]);

  outputs the character 'e'
- During compilation, C creates space for each string literal (# of characters in the literal + 1)
  - referring to the literal refers to that space (as if it is an array)

String Variables

- Allocate an array of a size large enough to hold the string (plus 1 extra value for the delimiter)
- Examples (with initialization):

```
char str1[6] = "Hello";
char str2[ ] = "Hello";
char str3[20] = "Hello";
char *str4 = "Hello";
char str5[6] = {'H','e','l','l','o','\0'};
```

- Note, each variable is considered a constant in that the space it is connected to cannot be changed (except str4)

```
str1 = str2; /* not allowed, but we can copy the
   contents of str2 to str1 (more later) */
```

Duplicate String Literals

- Each string literal in a C program is stored at a different location
- So even if the string literals contain the same string, they are not equal (in the == sense)
- Example:
  - char str1[6] = "hello";
  - char str2[6] = "hello";
  - but str1 does not equal str2 (they are stored at different locations)
  - if(str1 == str2) … is FALSE

Changing content of String Variables

- **Can change parts of a string variable**

```
char str1[6] = "hello";
str1[0] = 'y';
/* str1 is now "yello" */
str1[4] = '\0';
/* str1 is now "yell" */
```

- **Important to retain delimiter (replacing str1[5] in the original string with something other than '\0' makes a string that does not end)**
- **Have to stay within limits of array**

String Input

- **Use %s field specification in scanf to read string**
  - **ignores leading white space**
  - **reads characters until next white space encountered**
  - **C stores null (\0) char after last non-white space char**
  - **Reads into array (no & before name, array is a pointer)**
- **Example:**

  ```
  char Name[11];
  scanf("%s", Name);
  ```

- **Problem: no limit on number of characters read (need one for delimiter), if too many characters for array, problems may occur**

String Input (cont)

- Can use the width value in the field specification to limit the number of characters read:

```
char Name[11];
scanf("%10s",Name);
```

- Remember, you need one space for the \0
  - width should be one less than size of array
- Strings shorter than the field specification are read normally, but C always stops after reading 10 characters

String Input (cont)

- **scanf with %[^*ListofNonAllowableChars*]**
  - **ListofNonAllowableChars specifies set of characters that are not allowed (called non scan set)**
  - **Characters read as long as character does not fall in set**
  - **Stops when first non scan set character encountered**
  - **Any character may be specified except]**
- **Examples:**

  scanf("%[^\n]",Line); /* read until newline char */

String Input (cont)

- scanf with limit and read until enter

  scanf("%10[^\n]",Line); /* read until newline char or 10 characters if newline is not encountered within 10 chars */

  - If you want any other character input after the string input then you have to flush the rest of the input first like below:

    scanf("%10[^\n]%*[^\n]",s);
    scanf("%*c%c",&c);

String Output

- **Use %s field specification in printf:**

  characters in string printed until \0 encountered

  char Name[10] = "Rich";

  printf("%s",Name); /* outputs Rich */

- **Can use width value to print string in space:**

  printf("%10s",Name); /* outputs Rich */

- **Use - flag to left justify:**

  printf("%-10s",Name); /* outputs Rich */

Input a String

- *gets*
  - Get a string from user input
  - reads until enter is pressed

```
main(){
   char c[80];
   gets(c);
   printf("%s\n", c);
}
```

```
Input: TODAY IS MONDAY
Output: TODAY IS MONDAY
```

Input a String

- *fgets*
  - Get a string from user input
  - reads until enter is pressed or limit is reached

```
#include <stdio.h>
main(){
  char c[80];
  fgets(c,79,stdin);
  printf("%s\n", c);
}
```

```
Input: TODAY IS MONDAY
Output: TODAY IS MONDAY
```

Determining length of a string

- ***strlen***
  - **Returns the number of characters in *"Saturday"***

```
int length = strlen("Saturday");
  //answer is 8
```

Write down a program that will print *n*-th letter in a sentence entered by a user. *n* will be input to your program

Solution

```
main()
{
    char s[80];
    int n, length;
    printf("Enter a Sentence:");
    gets(s); // or fgets(s,79,stdin);
    length = strlen(s);
    printf("Total char in sentence is:%d\n", length);
    printf("Which position?");
    scanf("%d",&n);
    if(n < length)
         printf("The letter is:  %c", s[n]);
    else
         printf("No letter at such position");
}
```

Write down a program that will print letters of a sentence in a vertical line. Add delay as needed.

Solution

```
#include <windows.h>
main()
{
   char s[80];
   int n,length,i;
   printf("Enter a Sentence:");
   gets(s);
   length = strlen(s);
   for(i = 0; i < length; i++)
   {
        printf("    %c\n",s[i]);
        Sleep(500);
   }

}
```

- **Example:**
  - **char str1[6] = "hello";**
  - **char str2[6] = "hello";**
  - **if(str1 == str2) … does not evaluate to be TRUE**

Write down a function that compares two strings and returns 1 if they are same and returns 0 otherwise

Solution

```
int samestring(char s1[ ], char s2[ ]) {
  int i;

  /* Not same if not of same length */
  if (strlen(s1) != strlen(s2))
    return 0;
  /* look at each character in turn */
  for (i = 0; i < strlen(s1); i++)
    /* if a character differs, string not same */
    if (s1[i] != s2[i]) return 0;
  return 1;
}
```

Write down a program that searches for a letter in a sentence. Both letter and sentence will be input to your program. Print last position of the letter found in the sentence.

Solution

```
main()
{
    char s[80],t;
    int n,l,i,p;
    printf("Enter a Sentence:");
    gets(s);
    printf("Which letter? ");
    scanf("%c",&t);
    length = strlen(s);
    p = -1;
    for(i = 0; i < length; i++)
         if(s[i] == t)
                  p = i;
    if(p == -1)  printf("Sorry not found");
    else  printf("Found at position: %d", p);
}
```

Write down a program that prints how many times a letter appeared in a sentence. Both letter and sentence will be input to your program.

Solution

```
main()
{
    char s[80],t;
    int n,l,i,count;
    printf("Enter a Sentence:");
    gets(s);
    printf("Which letter? ");
    scanf("%c",&t);
    length = strlen(s);
    count = 0;
    for(i = 0; i < length; i++)
         if(s[i] == t)
             count++;
    if(count == 0) printf("Sorry not found");
    else   printf("Found %d times", count);
}
```

Write down a program that searches for a word in a sentence. Both word and sentence will be input to your program. Print first position of the word found in the sentence.

Solution

```
main()
{  char s[80],t[80];
   int i,p;
   printf("Enter a Sentence:");
   gets(s);
   printf("Which word? ");
   gets(t);
   p = -1;
   for(i = 0; i < strlen(s); i++)
        if(s[i] == t[0]){
             for(j = 1; j < strlen(t); j++)
                  if(s[i+j] != t[j])
                       break;
             if(j == strlen(t)){
                  p = i;
                  break;
             }
   }
   if(p == -1)  printf("Sorry not found");
   else  printf("Found at position: %d", p);
}
```

Write down a program that prints how many words, letters, vowels and consonants exist in a sentence. The sentence will be input to your program.

Solution

```
main(){
   char s[80],t;
   int w, v, c, l, i,length;
   printf("Enter a Sentence:");
   gets(s);
   length = strlen(s);
   w = v = c = 0;
   for(i = 0; i < length; i++){
        t = tolower(s[i]);
        if(t == ' ')    w++;
        else if ((t == 'a') || (t == 'e') ||
(t == 'i') || (t == 'o') || (t == 'u'))
      v++;
        else      c++;
}
   printf("Number of words: %d \n",w+1);
   printf("Number of letters: %d \n",v+c);
   printf("Number of vowels: %d \n",v);
   printf("Number of consonants: %d \n",c);
}
```

Write down a program that will take a word as input and will determine whether the word is palindrome or not. A palindrome is a word that reads the same backward as forward

Solution

```
main(){

    char s[80],t[80];
    int length, i, j;
    gets(s);
    length = strlen(s);
    j = 0;
    for(i = length-1; i >= 0; i--){
        t[j] = s[i];
        j++;
    }
    for(i = 0; i < length; i++){
        if(s[i] != t[i]){
            break;
        }
    }
    if(i == length)
        printf("Palindrome");
    else
        printf("No");
}
```

Solution (in-place check i.e. without using any extra array)

```
main(){

    char s[80];
    int length, i, j;
    gets(s);
    length = strlen(s);
    j = length-1;
    for(i = 0; i < j; i++,j--){
       if(s[i] != s[j]){
            break;
        }
    }
    if(i == j)
        printf("Palindrome");
    else
        printf("No");
}
```

Strings input output

- **gets(s)** – take a string as input and place it in array s
- **puts(s)** – show the content of the string s

```
#include <stdio.h>
#include <string.h>
int main() {
        char s[30];
        printf("Please enter a sentence: ");
        gets(s);
        puts("You have entered: ");
        puts(s);
        return 0;
}
```

Strings initialization at the time of declartion

```
#include <stdio.h>
#include <string.h>
int main() {
        char s[80]="To be or not to be that is the question";
        puts(s);
        return 0;
}
```

C offers following major library functions on strings

- strlen(s) – return the length of a string s
- strlwr(s)– convert the string s in lower case
- strupr(s) – convert the string s in upper case
- strrev(s) – reverse the content of the string s
- strcpy(s, t) – copy string t into another string s
- strncpy(s, t, n) - copy n characters of string t into another string s
- strcat(s, t) – append string t into the right side of the string s
- strncat(s, t, n) - append n characters of the string t onto the right side of the string s
- strcmp(s, t) – compare alphabetic order of two strings s and t

For detailed implementation see:

https://en.wikibooks.org/wiki/C_Programming/String_manipulation

strlen(s) – returns the length of a string s

Example

```
#include <stdio.h>
#include <string.h>
int main( ) {
        char str[20] = "BeginnersBook";
        int length;
        length = strlen(str);
        printf("Length of the string is : %d", length);
        return 0;
}
```

Output:

```
Length of the string is: 13
```

strlwr(s)– convert the string s in lower case

Example

```
#include <stdio.h>
#include <string.h>
int main( ) {
        char str[20] = "BeginnersBook";
        strlwr(str);
        printf("%s",str);
        return 0;
}
```

Output:

```
beginnersbook
```

strupr(s)– convert the string s in upper case

Example

```
#include <stdio.h>
#include <string.h>
int main( ) {
        char str[20] = "BeginnersBook";
        strupr(str);
        printf("%s",str);
        return 0;
}
```

Output:

```
BEGINNERSBOOK
```

strrev(s) – reverse the content of the string s

Example

```
#include <stdio.h>
#include <string.h>
int main( ) {
        char str[20] = "DRAWER";
        strrev(str);
        printf("%s",str);
        return 0;
}
```

Output:

```
REWARD
```

strcpy(s, t) – copy string t into another string s

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[30] = "Bad";
        char s2[30] = "Good";
        strcpy(s1, s2);
        printf("%s",s1);
        return 0;
}
```

Output:

```
Good
```

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[30] = "Bad";
        char s2[30] = "Good";
        strcpy(s2, s1);
        printf("%s",s2);
        return 0;
}
```

Output:

```
Bad
```

strncpy(s, t, n) - copy n characters of string t into another string s. Fills with null character if t doesn't have n characters

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[30] = "Coastal";
        char s2[30] = "Cry";
        strncpy(s1, s2,3);
        printf("%s",s1);
        return 0;
}
```

Output:

```
Crystal
```

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[30] = "Coastal";
        char s2[30] = "Cry";
        strncpy(s1, s2,4);
        printf("%s",s1);
        return 0;
}
```

Output:

```
Cry
```

strcat(s, t) – append string t into the right side of the string s

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[30] = "Hello ";
        char s2[30] = "World";
        strcat(s1, s2);
        printf("%s",s1);
        return 0;
}
```

Output:

```
Hello World
```

strncat(s, t, n) - append n characters of the string t onto the right side of the string s

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[30] = "";
        char s2[30] ="Happy ";
        strncat(s1, s2, 6);
        printf("%s",s1);
        return 0;
}
```

Output:

```
Happy
```

LITTLE QUIZ FOR YOU

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[30] = "Happy ";
        char s2[30] = "New Year!";
        char s3[30] = "";
        strcat(s1, s2);
        strncat(s3, s1,6);
        strcat(s3, s1);
        printf("%s",s3);
        return 0;
}
```

s1 = "Happy New Year!"
s3 = "Happy "
s3 = "Happy Happy New Year!"

Output:

```
Happy Happy New Year!
```

strcmp(s, t) – compare alphabetic order of two strings s and t

strcmp

- strcmp(s, t)

- Compares s and t alphabetically
- Returns a negative value if s precedes t alphabetically
- Returns a positive value if t precedes s alphabetically
- Returns 0 if they are same
- Note lowercase characters are greater than Uppercase

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[ ] = "cat";
        char s2[] = "cat";
        char s3[] = "dog";
        int x = strcmp(s1, s2);
        if(x == 0)
                printf("They are same") ;
        else if (x < 0)
                printf("s1 comes before s2") ;
        else if (x > 0)
                printf("s1 comes after s2") ;
        return 0;
}
```

Output:

```
They are same
```

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[ ] = "cat";
        char s2[] = "cat";
        char s3[] = "dog";
        int x = strcmp(s1, s3);
        if(x == 0)
                printf("They are same") ;
        else if (x < 0)
                printf("s1 comes before s3") ;
        else if (x > 0)
                printf("s1 comes after s3") ;
        return 0;
}
```

Output:

```
s1 comes before s3
```

strcmpi(s, t) – compare alphabetic order of two strings s and t ignoring case

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[ ] = "cat";
        char s2[] = “Cat";
        char s3[] = "dog";
        int x = strcmp(s1, s2) ;
        if(x == 0)
                printf("They are same") ;
        else if (x< 0)
                printf("s1 comes before s2") ;
        else if (x > 0)
                printf("s1 comes after s2") ;
        return 0;
}
```

Output:

```
s1 comes after s2
```

Example

```
#include <stdio.h>
#include <string.h>
int main() {
        char s1[ ] = "cat";
        char s2[] = “Cat";
        char s3[] = "dog";
        int x = strcmpi(s1, s2);
        if(x == 0)
                printf("They are same") ;
        else if (x < 0)
                printf("s1 comes before s2") ;
        else if (x > 0)
                printf("s1 comes after s2") ;
        return 0;
}
```

Output:

```
They are same
```

Program: Palindrome testing

Example: Palindrome testing

```
#include <stdio.h>
#include <string.h>
int main() {
        char s[80]="madam";
        char t[80];

        gets(s);
        strcpy(t,s);
        strrev(t);
        if(strcmpi(s,t) == 0)
                printf("\"%s\" is a palindrom", s);
        else
                printf("\"%s\" is NOT a palindrom", s);
        return 0;
}
```

Questions?