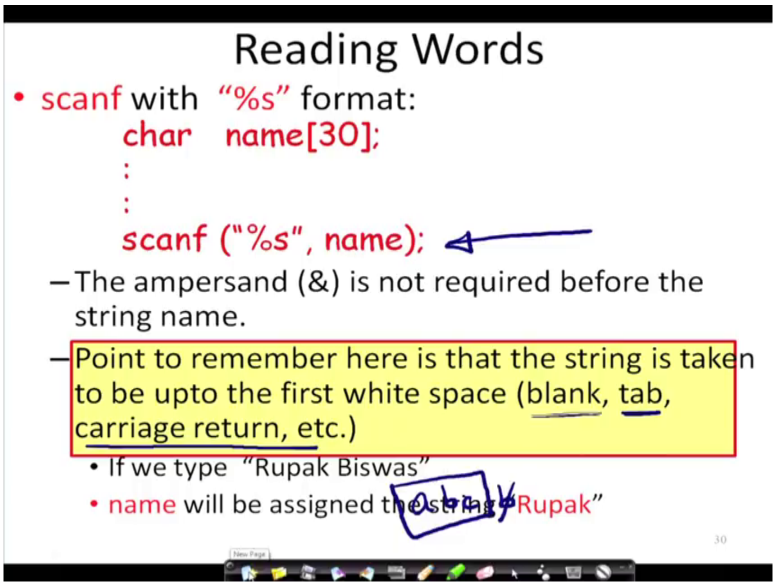
Problem Solving through Programming in C

Prof. Anupam Basu.
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 32
Character Arrays and Strings

We were discussing about strings and in today's discussion, let me recapitulate what we or the last part of what we discussed in the earlier lecture.

(Refer Slide Time: 00:29)



Reading Words

- scanf with "%s" format:

```
char  name[30];
  :
  :
scanf ("%s", name);
```

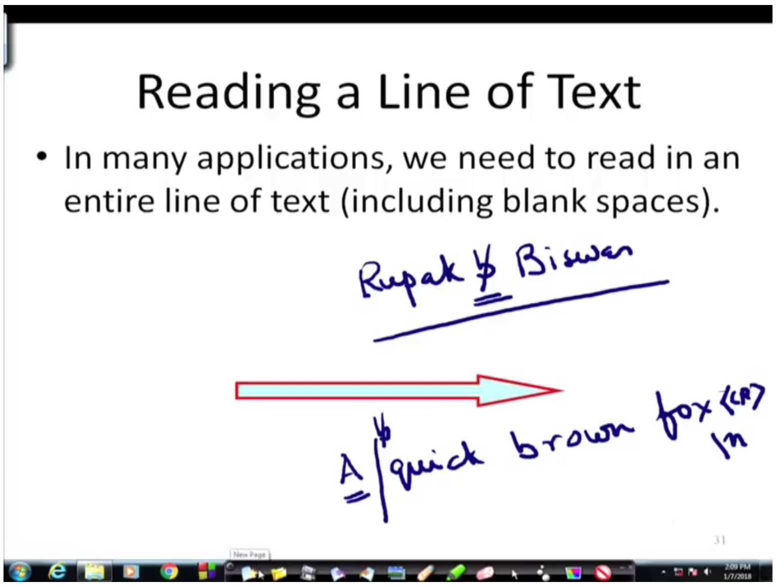
  - The ampersand (&) is not required before the string name.
  - Point to remember here is that the string is taken to be upto the first white space (blank, tab, carriage return, etc.)
    - If we type "Rupak Biswas"
    - name will be assigned the string "Rupak"

How can we read words from this from an input say? So, there we had introduced, this percentage is format with that, with percentage is if we read a name, then it has to be noted that the ampersand is not needed in this case right the ampersand is not needed when I am reading a string when I am reading a string I do not need that ampersand.

Now, the point to remember is that whenever I am reading this in this way the string will be read until there is some blank white space this is also called as white space or a tab sign or a carriage return or enter is pressed. So, if I type on a b c and then I type blank, then this a b c will be taken as the word. So, when I perform scan if in this form in name then a b c will go in the name alright. So, this is what we had seen last time.

So, that is why this example was given if we type Rupak Biswas since after Rupak, there is a blank we will stop at that point of time. The name will be assigned to Rupak ok name will be assigned to the string Rupak name will be assign the string Rupak.

(Refer Slide Time: 02:25)



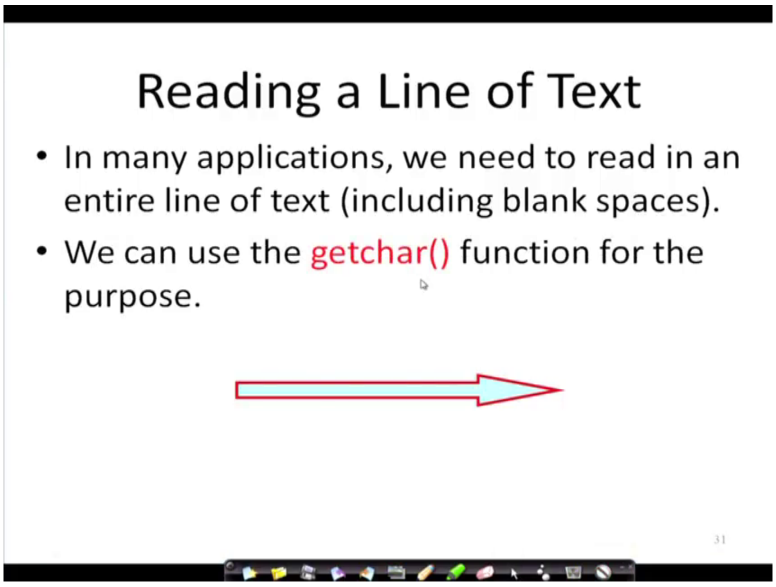
Reading a Line of Text

- In many applications, we need to read in an entire line of text (including blank spaces).

On the other hand when we read a line of text if I suppose I want to read say this Rupak Biswas, I want to read this blank also this entire thing the entire sentence, I want to read a quick brown fox I want to read.

So, I want to read this characters as well as this blank space is everything then what will be my delimiter? My delimiter will be the carriage return the return carriage return or that is often designated as backslash n. So, till I get a backslash n, I will go on reading this that is how I want I can read a line.

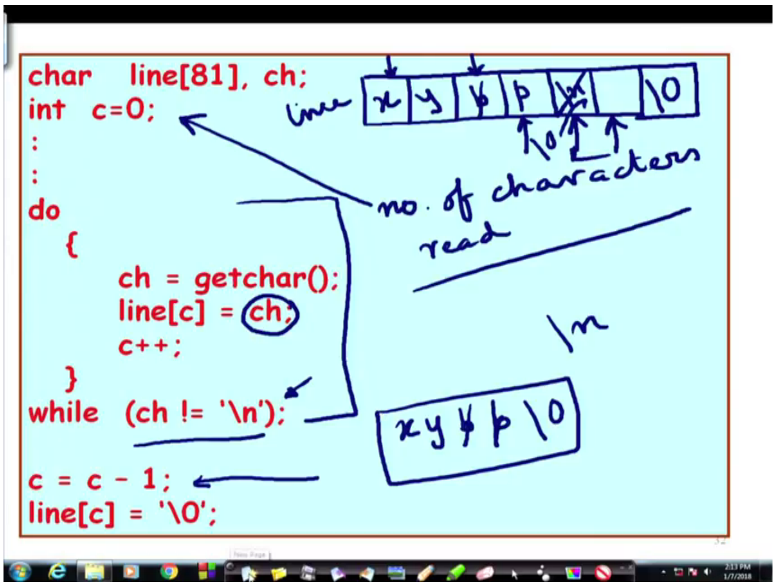
(Refer Slide Time: 03:26)



Reading a Line of Text

- In many applications, we need to read in an entire line of text (including blank spaces).
- We can use the getchar() function for the purpose.

So, in order to do that we can use the gate care function for the purpose by gate care function we will be getting the characters one after another, and I will go on getting the functions till the I mean I will be going on getting the characters, till I come to a character that is carriage return designating that there is an that is an end of the line. (Refer Slide Time: 04:56)



```
char  line[81], ch;
int  c=0;
:
:
do
  {
      ch = getchar();
      line[c] = ch;
      c++;
  }
while  (ch != '\n');

c = c - 1;
line[c] = '\0';
```

So, for example, here a line has been defined to be of maximum length 81, typically as I said a line consist of 80 characters and you must be understanding now realizing now that why I put 81, the reason is here I need a space for backslash 0. And I put c to be 0 c is the number of characters c is here designating the number of characters read, all right.

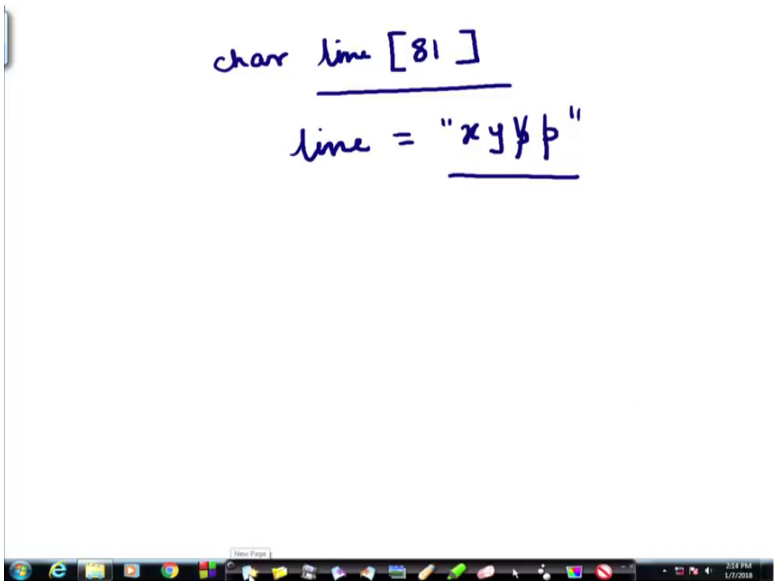
Now, here we are doing it through a do while statement, you can see that we can do a we are doing it through a do while. So, I am reading one character here say x and that I am reading it though to get char, and that character that has been read is coming to the variable array line all right line c c was initially 0.

So, here I put x, then I go up then character I go on I read it and then found that their character is not ampersand. I go back again and read another character say x y increase c. So, c is now pointing here, c is now pointing here and I again go back now suppose a blank has been typed in. So, I come here get char is blank. So, that comes here and then c is pointing to the next position here, and then I go again here and suppose I type in p. So, the c is now pointing here, and then after that I read the character and the character was ampersand I mean the character was backslash n; that means, the end of the character.

So, I put backslash n here, whatever it was there and c is pointing here. Now I have after insertion of backslash n, I come and find here that well what I have entered is backslash n. Now backslash n should not be there instead of that we should have backslash 0 therefore, I will come back I will decrement c by ones position. So, it will come here and I will replace this with backslash 0 right here I will bring in backslash 0 will come here.

So, the pattern that will be stored do will be x y blank p backslash 0. So, that is what this thing will look like in that way I am reading a line that is how we are reading a line. So, we have seen that we can read characters as a character array.

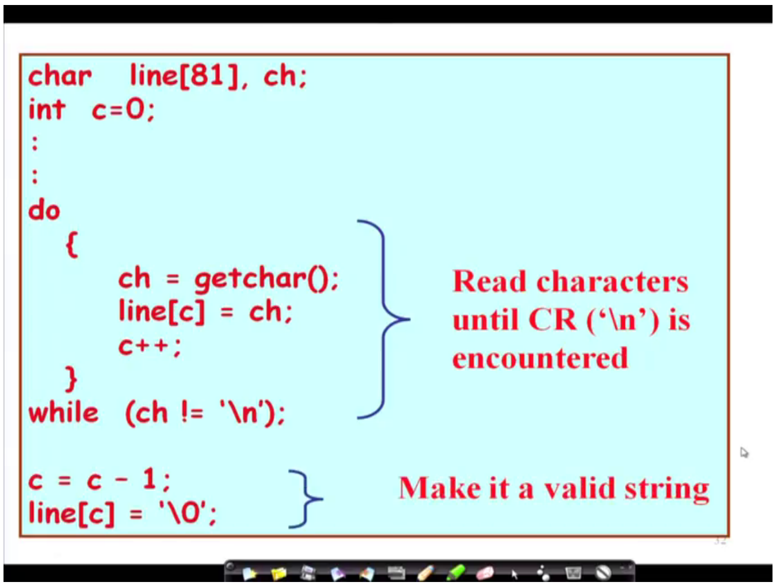
(Refer Slide Time: 07:22)



```
char line[81]
     line = "xyȼp"
```

Char some line in this way line 81. Now this line I can read character by character or I can type in line to be x y blank p that is also will do the same thing, otherwise, I could have copied it character by character both of them are equivalent, all right.

(Refer Slide Time: 08:02)



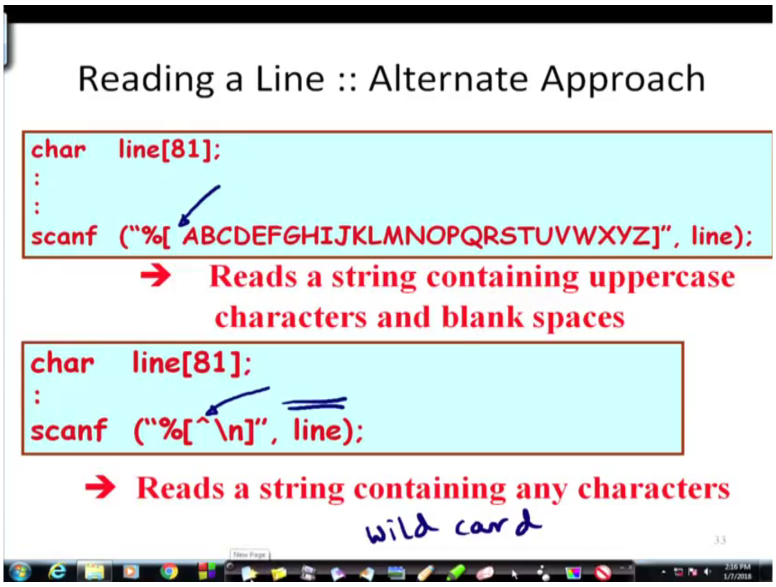
```
char   line[81], ch;
int  c=0;
:
:
do
   {
        ch = getchar();
        line[c] = ch;
        c++;
   }
while  (ch != '\n');

c = c - 1;
line[c] = '\0';
```

Read characters until CR ('\n') is encountered

Make it a valid string

So, here we have read the character until the char return or backslash n is encountered and then we make it a valid string by replacing backslash n by backslash 0.

(Refer Slide Time: 08:15)



Reading a Line :: Alternate Approach

```
char   line[81];
:
:
scanf  ("%[ ABCDEFGHIJKLMNOPQRSTUVWXYZ]", line);
```

➔ Reads a string containing uppercase characters and blank spaces

```
char   line[81];
:
scanf  ("%[^\n]", line);
```

➔ Reads a string containing any characters

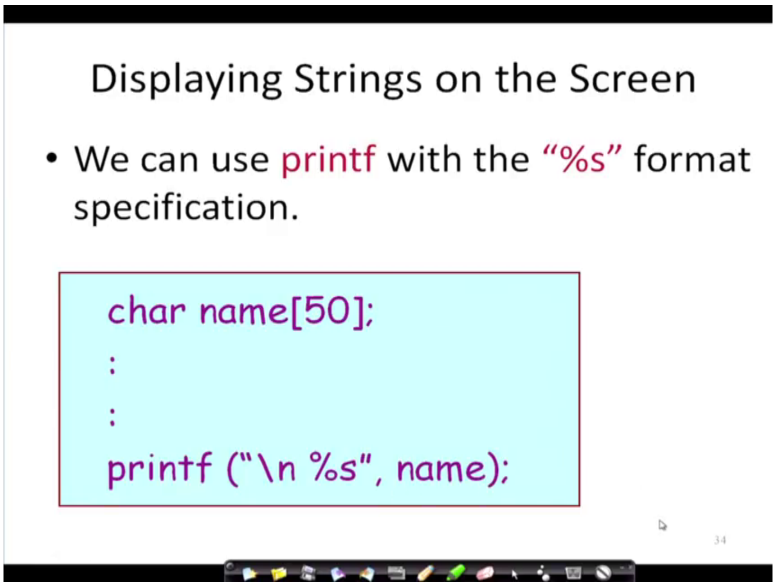
wild card

An alternative approach of reading a line can be this that I just specify the format, here till now what we have seen is that our formats could be percentage s percentage d percentage f like that. I can also specify my format in this way as you can see here, what it means is anything I am sorry what happened; anything that is a b c d or whatever that this entire thing and here there is a blank, I do not know what is happening here, here you can see there is a blank here.

So; that means, what is allowed? Anything including bank blank capital A to Z everything is allowed as members of the variable line. Similarly here the specification is that it can be char at this is a wild card it is called; that means, we can put in anything preceding backslash n. So, this means it is a wild card; that means, anything can come here as a character, that can be a member of the variable line. So, that is also another way of specifying it.

Now, these are specifics to the language c of course.

(Refer Slide Time: 10:13)



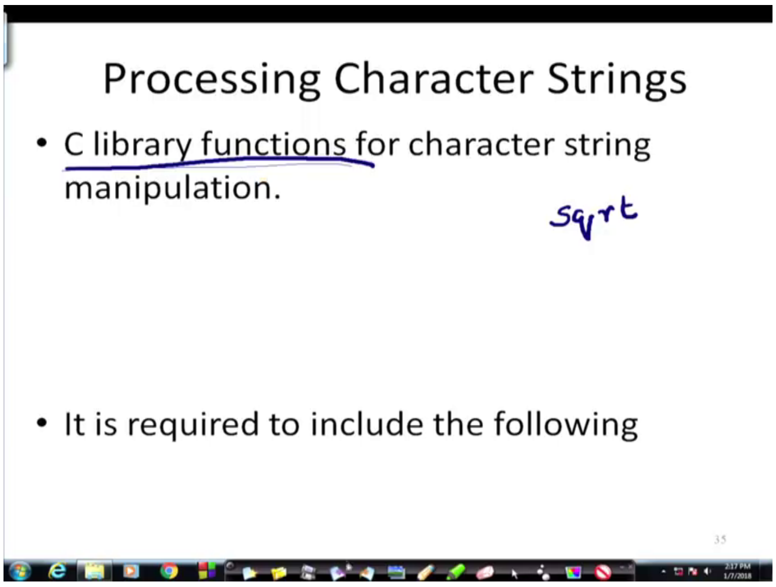
Displaying Strings on the Screen

- We can use printf with the "%s" format specification.

```
char name[50];
:
:
printf ("\n %s", name);
```

Now it is easier relatively simpler to display strings on the screen we can simply do a string, we can display the string with percentage x s and followed by the string name that is simpler now we come to a very important aspect how do we process character strings.

(Refer Slide Time: 10:30)



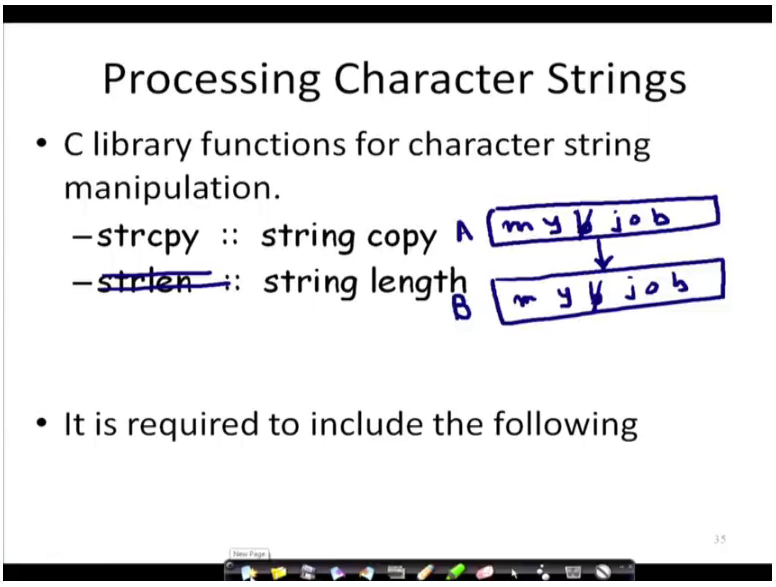
Processing Character Strings

- C library functions for character string manipulation.

- It is required to include the following

Now, for that we have got several c library functions we will soon come to functions and we have already seen different types of functions like square root, maybe did you see and we have seen we have seen the standard functions like scan f, print f all those things. Similarly we have got c libraries built in library functions, for character string manipulation how are they looking like? They are we have to in order to do that.

(Refer Slide Time: 11:21)



Processing Character Strings

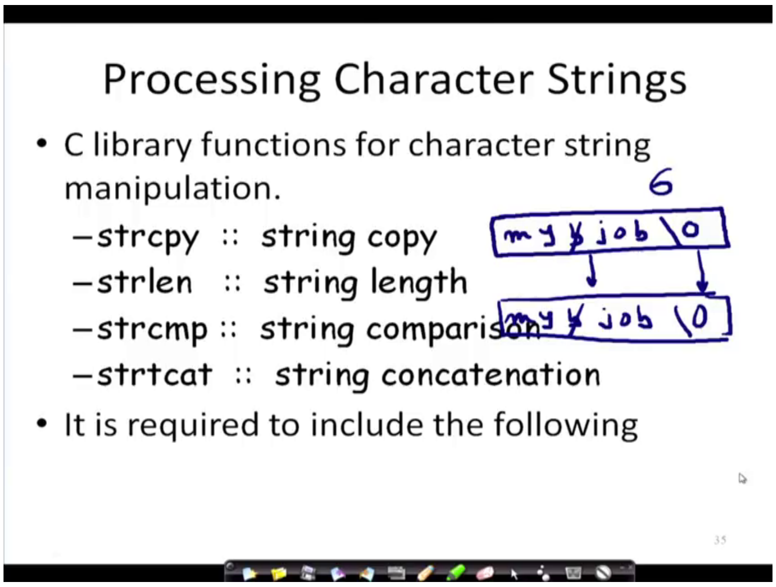
- C library functions for character string manipulation.
  - strcpy :: string copy
  - ~~strlen~~ :: string length

- It is required to include the following

Say one is s t r c p y; that means, string copy; that means, if I have a string here say string here is my sorry let me put a blank here, my space job. Suppose these are string A and I want to copy it to another string B. So, B will also therefore, have my blank job.

So, that is that can be done by string copy function s t r c p y how can that be done? We also have a similar thing like string length s t r l e n which means that I have got a string say my job again, blank job and automatically there is a backslash 0 at the end.

(Refer Slide Time: 12:24)



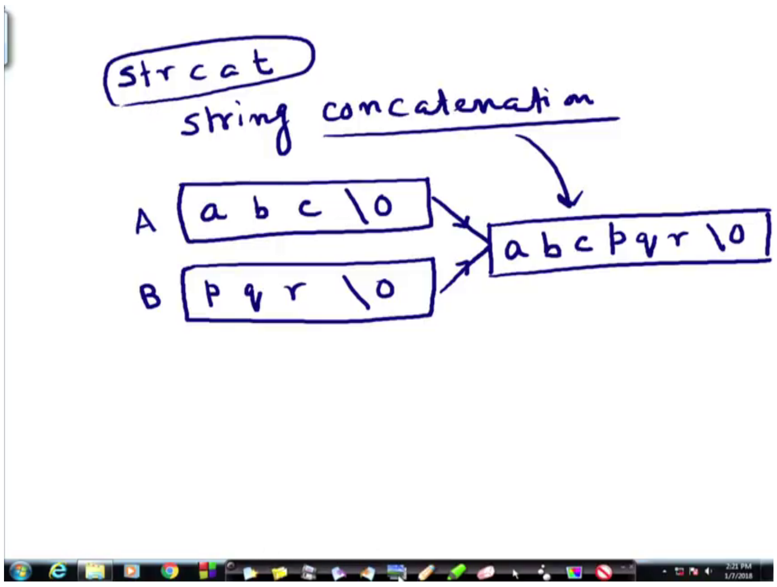
Processing Character Strings

- C library functions for character string manipulation.
  - strcpy :: string copy
  - strlen :: string length
  - strcmp :: string comparison
  - strtcat :: string concatenation
- It is required to include the following

So, when I copy it, then this will also be with s t r c p y we will have my blank job backslash n will come here sorry backslash 0 will come here, but s t r l e n means string length I will count how many elements are there in the string; can you tell me how many elements are there in the string? 1 2 3 blank is a valid character 1, 2, 3 4, 5, 6. So, the string length will be returned that will returned is 6, similarly we have another function very popular function string comparison s t r c m p; that means, I have got two strings my job and here my job, and I compared them if sorry if they are the same then I will have a one ok.

So, that is another function the fourth function is s t r c a t what is meant by concatenation? Concatenation s t r c a t is actually s t r c a t means string concatenation.

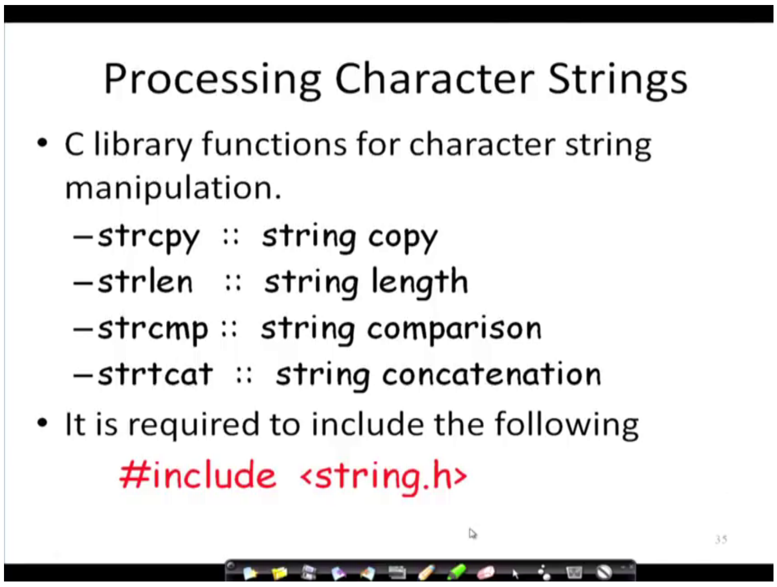
(Refer Slide Time: 14:08)



That means, if I have a string here say a b c ended with this, and there is another string say p q r ended with this, when I concatenate it that these two strings what I get is a joining of these two strings one after another. When I say that string b is concatenated with a then my pattern will be a b c note that this backslash 0 will not come p q r and backslash 0. This is known as concatenation all right of two strings. So, that is being given by a function s t r c a t.

Now. So, these are the very common functions for string operations. For our job if we need some other functions we can always write functions which will learn in a couple of lectures from now, but these are already available in the c library all right s t d i o dot h and also when we use that square root function .

(Refer Slide Time: 15:51)



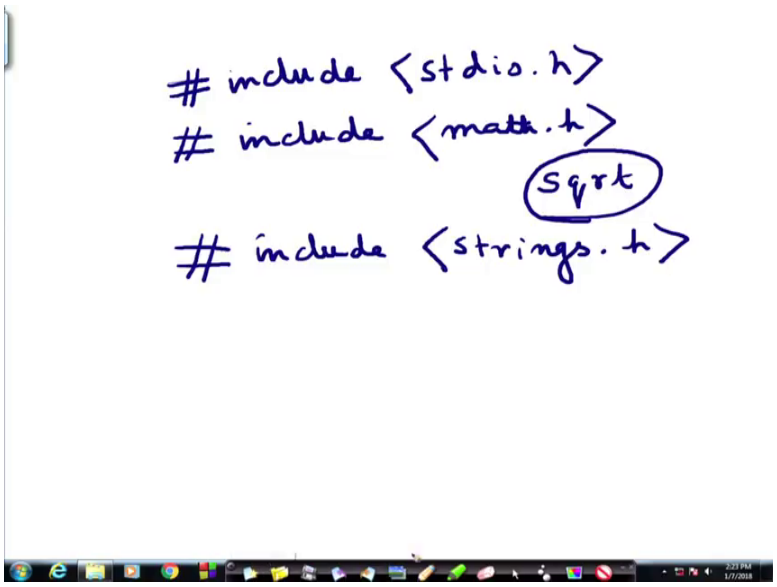
Processing Character Strings

- C library functions for character string manipulation.
  - strcpy :: string copy
  - strlen :: string length
  - strcmp :: string comparison
  - strtcat :: string concatenation
- It is required to include the following

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

And therefore, in order to do that we have to include st string dot h, if you recall we had said that if you recall we can include always you do that hash include.

(Refer Slide Time: 16:12)

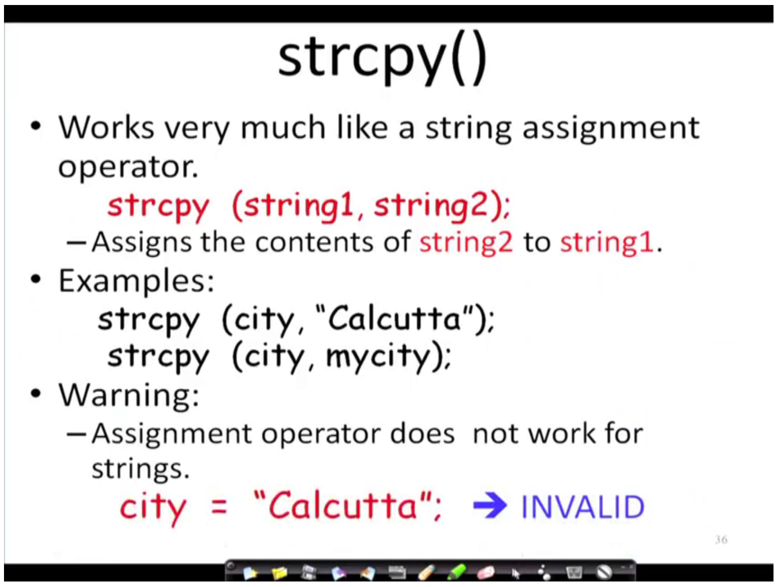


```
# include <stdio.h>
# include <math.h>
                 sqrt
# include <strings.h>
```

Earlier we had included math dot h; that means, all the mathematic library of all the mathematical functions math dot h when we use for example, s q r square root.

So, this is this is a function that is already inbuilt, inbuilt in this math library similarly for strings if I use this s t r c p y, s t r c a t, s t r c m, c m p compare, then I have to include strings dot h in my function alright. So, here are some examples.

(Refer Slide Time: 17:26)



strcpy()

- Works very much like a string assignment operator.

  strcpy (string1, string2);

  – Assigns the contents of string2 to string1.
- Examples:

  strcpy (city, "Calcutta");

  strcpy (city, mycity);
- Warning:
  – Assignment operator does not work for strings.

  city = "Calcutta"; → INVALID

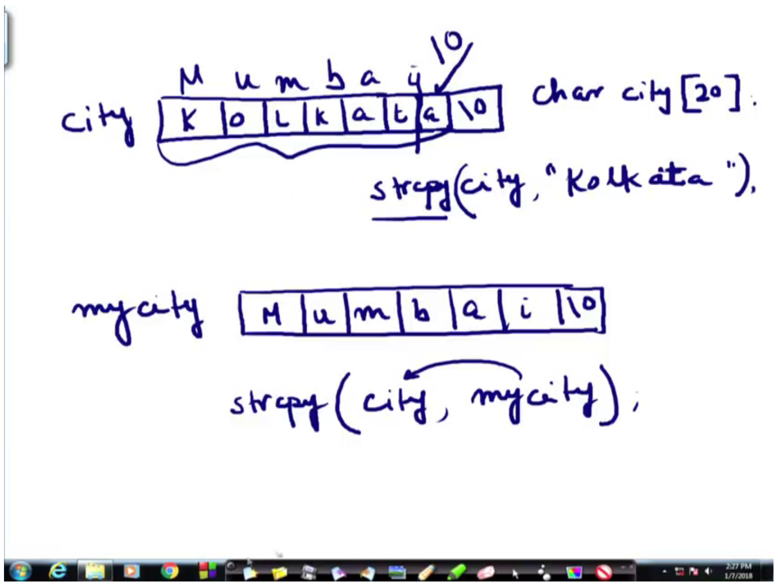
36

Very much like a string assignment operator string copy just like an assignment, for example, we have we are saying string copy string 1 string 2.

So, both string 2 will be copied in string 1. So, assigns the content of string 2 to string 1 assigns the. So, this is the source this is the destination. So, it is copied from here to here say string copy city Calcutta what will happen here; that means, I am trying to copy city is a string that is already defined alright I have defined city and I am copying this string now this is a string constant, this part is a string constant this is a string constant.

So, what I am doing is I am I have got a variable city, variable of type string.

(Refer Slide Time: 18:42)



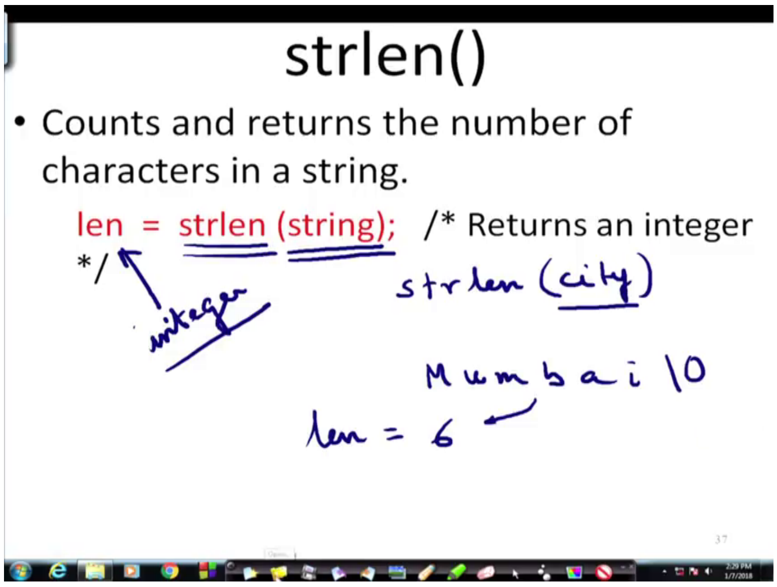
Now this how did you define it to be a type of type string it was char city say something like this char city 20. So, 20 characters can come there.

Now, when I write city s t r c copy city to Calcutta or right now we say Kolkata say we do Kolkata. Then in this variable Kolkata will be loaded all right a followed by a backslash 0 right. On the other hand suppose city is here and suppose another city another string variable is there, which is maybe Mumbai and I copy string copy city my city; that means, what will happen this string, this string my city will come into this variable city.

So, this Kolkata will be overwritten and will be replaced by Mumbai. Now Kolkata is of length 1 2 3 4 5 6 7 and Mumbai is 1 2 3 4 5 6 in that case what will happen? As I copy this whole string over here then here Mumbai m will come u will come m b a i and this backslash 0 will come here and so, the entire string will be kept here the last sorry up to this and with a backslash 0 here, and the remaining part of unused part of Kolkata will be lost.

Now, warning there is an warning here that is assignment operator does not work for string. So, I could not have done string 1 assign string 2 or as we are doing here city assigned, my city that would not do that assignment operator will not work in the case of strings. So, city assigned Calcutta is invalid.

(Refer Slide Time: 22:08)



strlen()

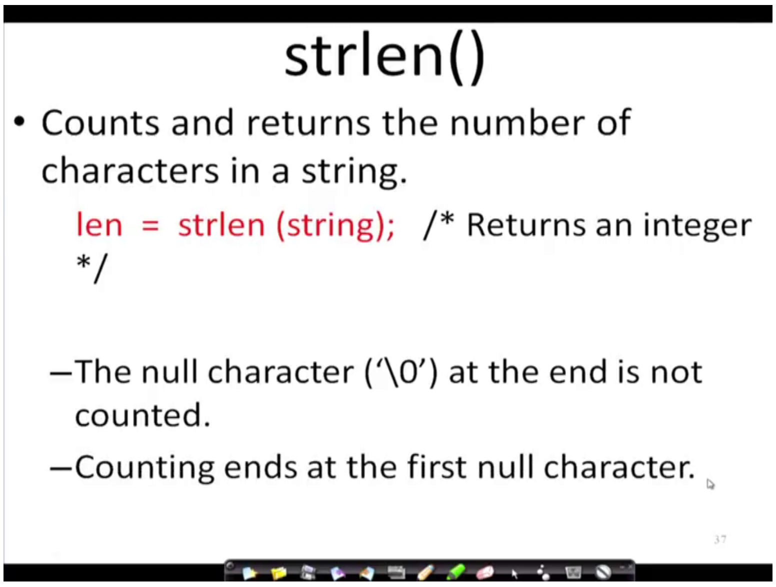
- Counts and returns the number of characters in a string.

```
len = strlen (string);   /* Returns an integer */
```

String length s t r l e n this is you I will ask you when we teach you function, to write a function for finding the string length although it is available in the standard c library. Counts and returns the number of characters in a string. So, len suppose len is a variable of type integer, len is of type integer len is of type integer and we say s t r l e n string.

So, string is some string all right some variable name s t r1e n city. So, city is a variable and whatever is the n suppose city is Mumbai, it will find out the length of the characters in this string and that will come in to len. So, len here we will get the value 6 when I do this function on this on the string.

(Refer Slide Time: 23:25)



strlen()

- Counts and returns the number of characters in a string.

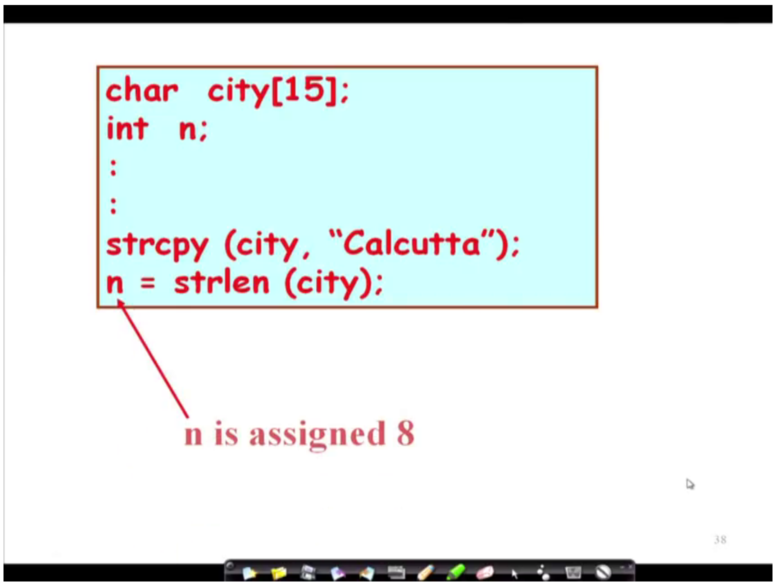
```
len = strlen (string);   /* Returns an integer */
```

  - The null character ('\0') at the end is not counted.
  - Counting ends at the first null character.

37

The null character at the end as I said is not counted because that tells you that you need not count any further; counting ends with the first null character.

(Refer Slide Time: 23:41)



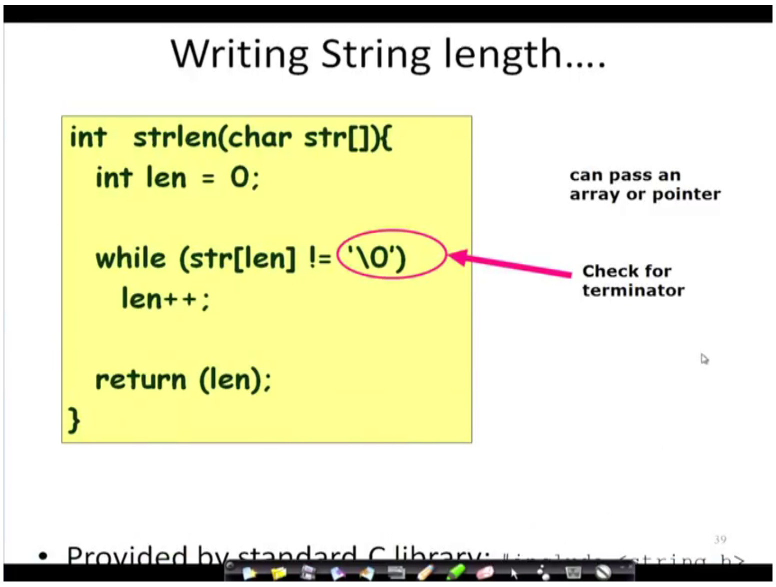
```
char  city[15];
int  n;
:
:
strcpy (city, "Calcutta");
n = strlen (city);
```

n is assigned 8

38

So, if I do Calcutta if I copy Calcutta the string, constant calculative city and I find out the length of city it will be 1 2 3 4 5 6 7 8 and backslash 0 will be left out. So, n is assigned 8 writing the string length.

(Refer Slide Time: 24:03)



Writing String length....

```
int  strlen(char str[]){
  int len = 0;

  while (str[len] !='\0')
    len++;

  return (len);
}
```

can pass an array or pointer
Check for terminator
• Provided by standard C library: #include <string.h>

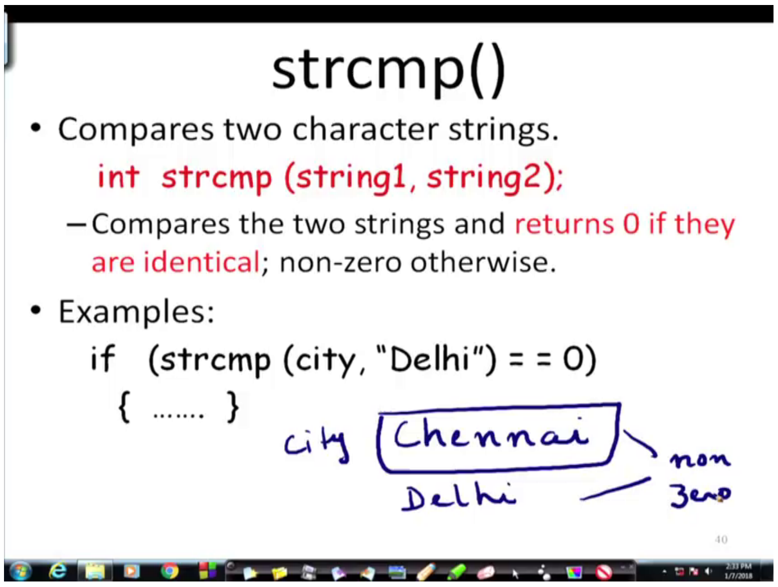
So, one problem that can be given to you that I am not talking about this function part right now.

Suppose I am trying to find out the string length of a character string I am trying to find how do you find out string length I mean its writing a program that is finding the string length you need not bother about the structure of this as yet, but let us look at the algorithm purely what is being done? Len some variable has been put to 0 and then while I am not encountering now s t r is an array of course, array of characters. As you can see its an idea characters as long as that is not equal to backslash 0; that means, not the end of the string I am going to going in going on incrementing this len, len was 0 len becomes one like that and then we return len after I completed then I return len .

Now, this is provided in the of course, I have to include the string dot h, but this thing this is what I am writing if, but actually this is a already available. So, I need not write it, I need to simply include hash include I call it hash include strings dot h that library. There similar program is already written and when I write strlen that is the program that is activated.

String compare is comparing two strings.

(Refer Slide Time: 25:53)



strcmp()

- Compares two character strings.

```
int strcmp (string1, string2);
```

  – Compares the two strings and returns 0 if they are identical; non-zero otherwise.

- Examples:

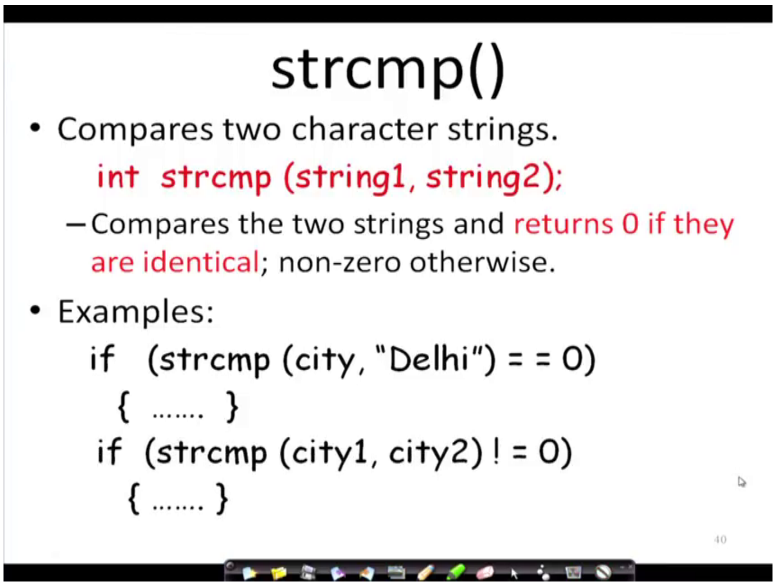
```
if (strcmp (city, "Delhi") = = 0)
  { ....... }
```

So, we are comparing two strings and returns 0 if they are identical what I just now said was just the opposite. If they are matching then it returns 0 and if they do not match a little non intuitive. So, keep it in mind that if the two strings match, then we will return a 0 otherwise will return a non-zero.

So, example is here if I compare city with the string Delhi is 0 then I do something. What does it mean city is a string already, suppose that is again Chennai suppose that is Chennai and I am comparing. So, city is a variable which has got this value, and I am comparing with Delhi of course, now they are not matching. So, it will not it will return non zero. But if the city was Delhi then these two have matched and I would have got a one here right.

Now; obviously, you can also just think and decide how this algorithm can be written that is 0073o, simple now you have learnt all the tidbits of writing such a program.

(Refer Slide Time: 27:22)



strcmp()

- Compares two character strings.

  int strcmp (string1, string2);

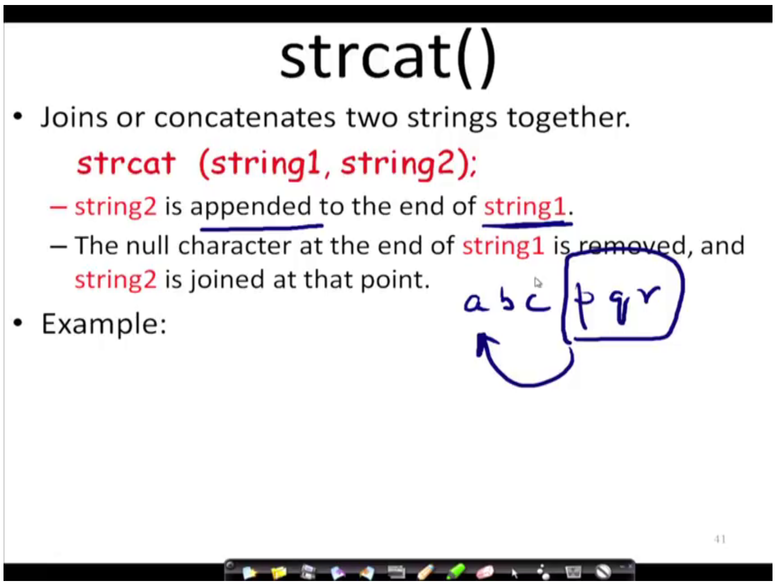
  – Compares the two strings and returns 0 if they are identical; non-zero otherwise.
- Examples:

```
if  (strcmp (city, "Delhi") = = 0)
  { ....... }
 if  (strcmp (city1, city2) ! = 0)
   { ....... }
```

40

So, if this is not equal to 0 then then we do this. So, similarly I can do city 1 city 2 now.

(Refer Slide Time: 27:31)



strcat()

- Joins or concatenates two strings together.

  strcat (string1, string2);

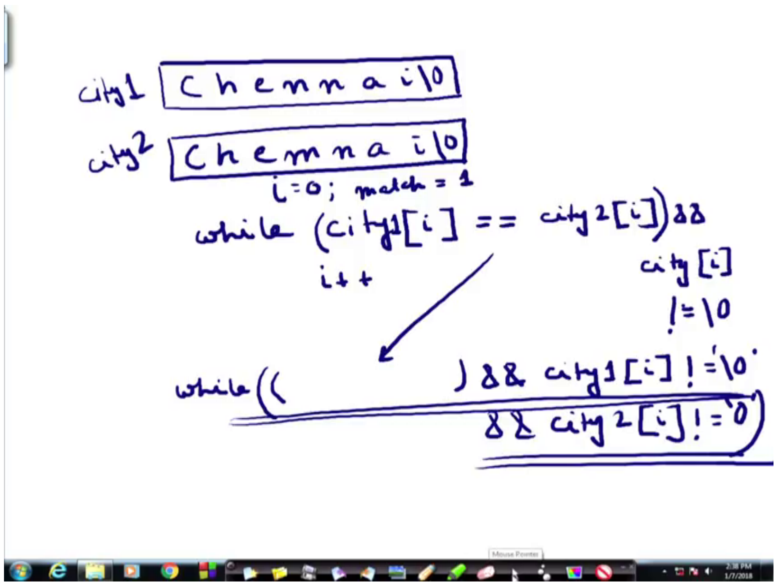
  – string2 is appended to the end of string1.

  – The null character at the end of string1 is removed, and string2 is joined at that point.
- Example:

41

So, before that you can simply think of how the algorithm will look like in case of this suppose I have got two strings.

(Refer Slide Time: 27:40)



One is say Chennai and the other string is of course, backslash 0 here instead of n there is a name here.

So, how will you do that? So, you will compare these two character by character. So, this is suppose city 1 and this is city 2. So, you can very easily compare while city 1 i of course, i has been assigned to 0 here is same as city 2 i what shall we do? We will go on incrementing I plus plus, but and if it is not equal we will come out.

So, this here I compare here I compare I come here and here I find that city 3 here, city 1 3 and city 2 3 are not the same. So, I will come out, but what happens if they are same. So, I need to put in if they are same I how long shall I go on no mismatch is there suppose this is also n while city is city 1 is not equal to city 2, while city 1 is equal to city 2 do I need another condition here yes and city i is not equal to backslash 0.

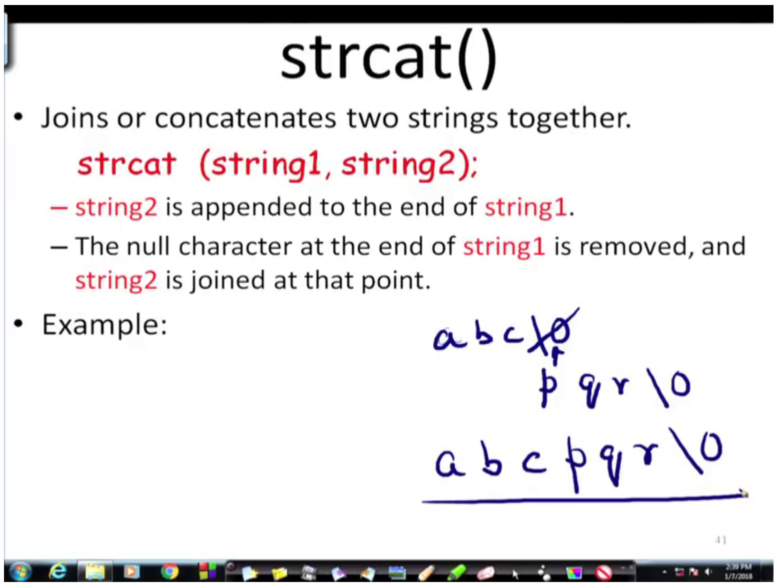
So, are you understanding this, while this condition which I have already written there. That should be true in order to proceed in order to proceed further for further checking, I should they are matching. So, I should proceed and, but and also the fact should be that none of these two cities names. So, two strings have reached the end is not equal to (Refer Time: 30:33) this and city 2 i if any one of them has matched has reached the end then my while condition will be violated. So, I will not continue any further.

So, I can keep a flag here that what should I say match is one. So, if I go on doing this match will be remaining 1, whenever I come out of this loop I will make match 0 that means, it has there has been a match because the convention is that if they are matching then it should be 0. So, you please also try to write this function right this program right we will ask you to make it a function a couple of lectures later. I hope this is clear this part please try to understand this condition try to understand this condition all right.

Next we have got the last one that is a s t r c a t. that is rather simple you will also be able to write the program for that it is two strings are just being kind of concatenated, but one thing that you must remember appending concatenating means joining, while appending means at adding one at the end of the other. So, when I write this a b c, p q r then p q r has been appended to a b c .

So, string 2 is appended to string one; that means, it is joined at the end of string 1. So, the null character at the end of string 1 is removed and string 2 is joined from that point.

(Refer Slide Time: 32:48)



strcat()

- Joins or concatenates two strings together.

  strcat (string1, string2);

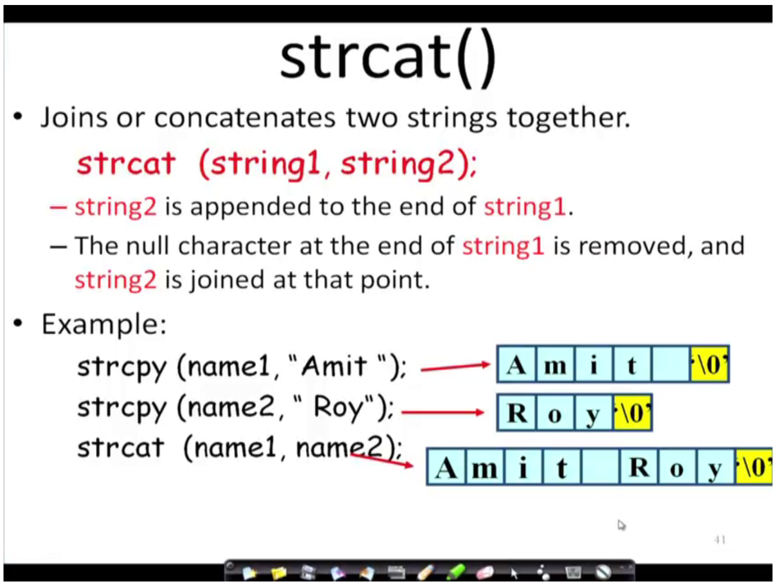
  – string2 is appended to the end of string1.

  – The null character at the end of string1 is removed, and string2 is joined at that point.

- Example:

as we said that there can be a b c backslash 0 and then when I append p q r to that, then p q r will p will replace this and backslash 0 will come here. So, ultimately we will have a b c, p q r backslash 0 all right.

(Refer Slide Time: 33:20)



strcat()

- Joins or concatenates two strings together.

  strcat (string1, string2);

  – string2 is appended to the end of string1.

  – The null character at the end of string1 is removed, and string2 is joined at that point.

- Example:

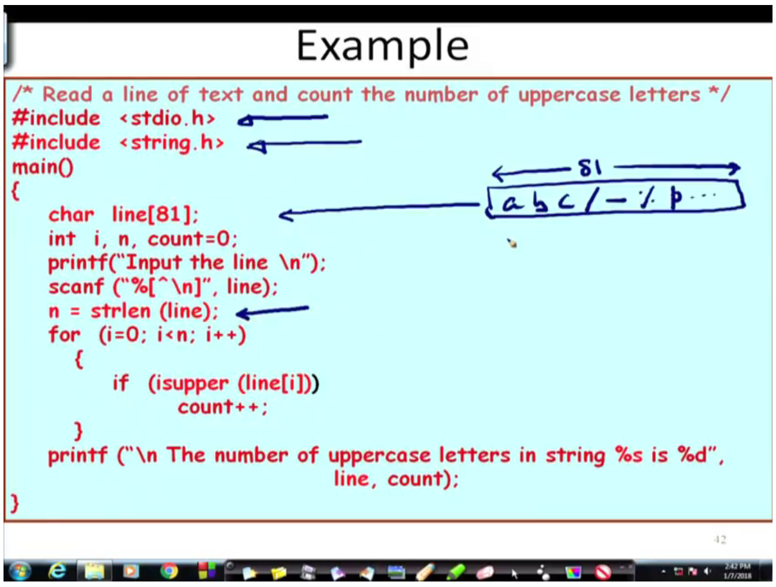
```
strcpy (name1, "Amit ");
strcpy (name2, " Roy");
strcat (name1, name2);
```

A m i t '\0'
R o y '\0'
A m i t R o y '\0'
41

So, string copy suppose I have name one a string assigned by string copy Amit; A m i t. So, it looks like this a m i t backslash here there is a backslash 0 and there is a blank here you see the string is not a m i t a m i t blank and another string is name two which is R o y, then when I append them concatenate them there it will be a m i t blank this blank and then R o y and this this blank has been replaced by r. So, r has gone here and we will get this.

So, this is string concatenation this will often come handy when you type in some character strings or type and lines or compute using some text strings text English sentence has been given and you are trying to find out where the verb is and all those then you need a lot of string operations like this.

(Refer Slide Time: 34:33)



Example

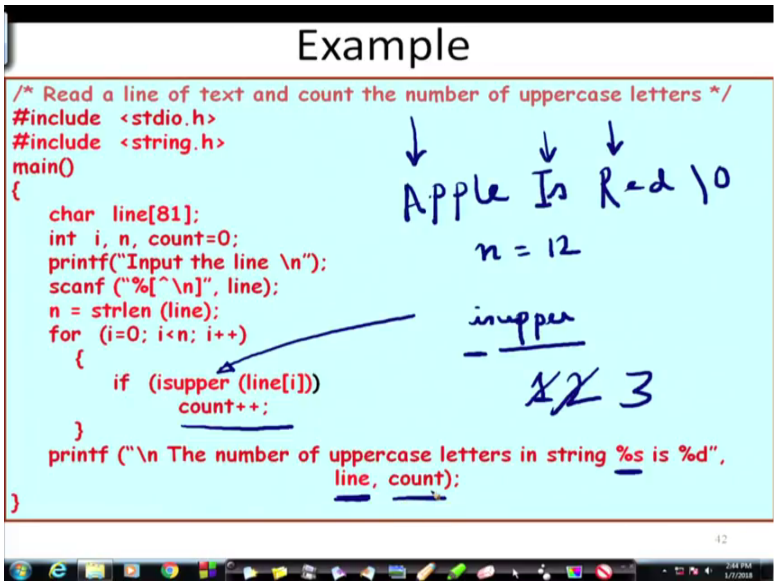
```
/* Read a line of text and count the number of uppercase letters */
#include <stdio.h>
#include <string.h>
main()
{
    char line[81];
    int i, n, count=0;
    printf("Input the line \n");
    scanf ("%[^\n]", line);
    n = strlen (line);
    for (i=0; i<n; i++)
      {
         if (isupper (line[i]))
             count++;
      }
    printf ("\n The number of uppercase letters in string %s is %d",
                        line, count);
}
```

So, here is an example here we are reading a line of text and counting the number of uppercase letters how many uppercase letters are there. So, what are the things we are including here? s t d i o dot h and old friend is here, string dot h is also there then look at this function I have defined a line to be of length 81 big one 81.

Now, I have got the variables I and current count is equal to 0, I am asking the user to input the line I am reading the line using this format which we just discussed; that means, anything can come here I mean anything can come here like a b c slash, dash alright percentage p whatever is coming except backslash n its coming here and then I am finding the length of the line.

Suppose the line that was actually typed in in this way through this scan f is say apple blank is red.

(Refer Slide Time: 35:52)



Example

```
/* Read a line of text and count the number of uppercase letters */
#include  <stdio.h>
#include  <string.h>
main()
{
    char  line[81];
    int  i, n, count=0;
    printf("Input the line \n");
    scanf ("%[^\n]", line);
    n = strlen (line);
    for  (i=0; i<n; i++)
      {
         if  (isupper (line[i]))
                count++;
      }
    printf ("\n The number of uppercase letters in string %s is %d",
                       line, count);
}
```

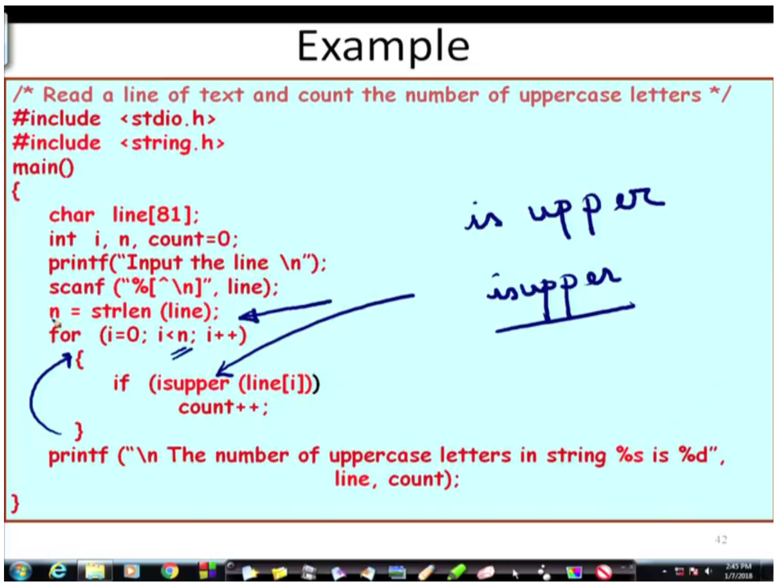
All right and then there is backslash here now this string length s t r l e n will find out the length of the string 1 2 3 4 5 6 7 8 9 10 11 12. So, 12 is n is becoming 12.

So, for i equal to 1 to n, now here is another new function that we are finding is upper if the character is upper case letter, is upper if the character that is being red that that is being passed here I put in some character and if the character is an upper case letter then I will count that. So, suppose here I make a little change I say this is I say that this is capital and I say that this is capital.

So, what is happening is it is reading character by character from here I 0 onwards, and checking whether this character is an upper character if that is so, count is becoming 1. So, count becomes one as a is upper case letter, then we go on in this loop i is being incremented until it comes to 12 less than 12. So, here is another one. So, I will get the count to be 2, it goes on and here I will get another one. So, its free print f the number of uppercase letters is percentage d is now look at this. The number of uppercase letters in string percentages s is percentage d. So, line will be printed as a line apple is red the number of uppercase letters in apple is red is 3.

So, here this program demonstrates a couple of things.

(Refer Slide Time: 38:14)



Example

```
/* Read a line of text and count the number of uppercase letters */
#include  <stdio.h>
#include  <string.h>
main()
{
    char  line[81];
    int  i, n, count=0;
    printf("Input the line \n");
    scanf ("%[^\n]", line);
    n = strlen (line);
    for  (i=0; i<n; i++)
    {
           if  (isupper (line[i]))
                count++;
    }
    printf ("\n The number of uppercase letters in string %s is %d",
                     line, count);
}
```

One is first of all this is a new thing that you have learned is upper, I am just writing it separately. So, that it is clear, but in actual c library the function is written in the form without this gap or without any special character in between, this is the first thing that we have learned. And how we have already seen it is an application of what we learnt how we can read a line using this wildcard format. And then we found out what is the length of the line by our newly learned function s t r l e n; and using that we have this is this looping we already know we have practiced it so often.

So, we now using this value using this for loop for so, many iterations. So, many repetitions that is determined by the value of n will check the entire string. So, this is an example of applying the string function in string operations, will go further with some more examples later.