Strings
(מחרוזות)

Example

```
int main(void)                    0 1 2 3 4 5 6 7 8 9 10 11...
{
    char str[] = "I'm a full string";

    printf("%s\n", str);          I'm a full string

    str[7] = 'o';
    str[8] = 'o';
    printf("%s\n", str);          I'm a fool string

    str[10] = '\0';
    printf("%s\n", str);          I'm a fool

    str[10] = 's';
    printf("%s\n", str);          I'm a foolsstring

    return 0;
}
```


Example - Comparing Strings

```
int i;
char A[101], B[101];

printf("Enter first string\n");
scanf("%100s",A);
printf("Enter second string\n");
scanf("%100s",B);

for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
     if(A[i]!=B[i])  {
        printf("A is different from B!\n");
        return 0;
     }
printf("A and B are the same!\n");
return 0;
```


Compare – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
       printf("A is different from B!\n");
       return 0;
    }

printf("A and B are the same!\n");
return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'w'	'\0'		

i

0

Compare – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
       printf("A is different from B!\n");
       return 0;
    }

printf("A and B are the same!\n");
return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'w'	'\0'		

i

0

Compare – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
       printf("A is different from B!\n");
       return 0;
    }

printf("A and B are the same!\n");
return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'w'	'\0'		

i

1

Compare – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }

  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'w'	'\0'		

i

1

Compare – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }

  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'w'	'\0'		

i

2

Compare – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }

  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'w'	'\0'		

i

2

Compare – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }

  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'w'	'\0'		

i

2

Compare – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }

  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'w'	'\0'		

i

2

Equal strings – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }

  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	's'	'\0'		

i

0

Equal strings – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	's'	'\0'		

i

0

Equal strings – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	's'	'\0'		

i

1

Equal strings – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	's'	'\0'		

i

1

Equal strings – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	's'	'\0'		

i

2

Equal strings – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	's'	'\0'		

i

2

Equal strings – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	's'	'\0'		

i

3

Equal strings – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	's'	'\0'		

i

3

Different length – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'\0'			

i

0

Different length – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'\0'			

i

0

Different length – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'\0'			

i

1

Different length – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'\0'			

i

1

Different length – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }
  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'\0'			

i

2

Different length – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }

  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'\0'			

i

2

Different length – step by step

```
for(i=0; A[i]!='\0' || B[i]!='\0'; i++)
    if(A[i]!=B[i])
    {
      printf("A is different from B!\n");
      return 0;
    }

  printf("A and B are the same!\n");
  return 0;
```

A

'Y'	'e'	's'	'\0'		

B

'Y'	'e'	'\0'			

i

2

Exercise @ class

- implement the function replace
  void replace(char str[], char what, char with);
- The function scans the string and replaces every occurrence of the first char ("what") with the second one (with)
- write a program to test the above function
  - the program reads a string from the user and two characters, then call the function with the input, and print the result.
- example
  - input: "papa", 'p', 'm'
  - output: "mama"

Example 1

- Implement the library function strrchr (from string.h).
  - Input - string *str*, char *c*.
  - Output - the index of the last occurrence of *c* in *str*.
- Write a program that accepts a string and a char from the user and displays the index of its last occurrence.

my_strrchr

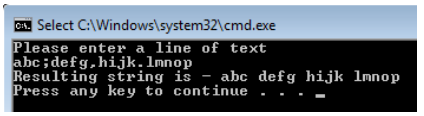
```
int my_strrchr(char str[], char c) {
        int i;

        for (i=strlen(str)-1; i>=0; i--)
                if (str[i] == c)
                        return i;

        return -1;
}
```


Using my_strrchr

```
char str[101];
char c;
int last_index;

printf("Please enter a string\n");
scanf("%100s",str);

printf("Please enter a character\n");
scanf(" %c", &c);

last_index = my_strrchr(str,c);
if(last_index > -1)
    printf("Last occurrence of %c in %s is in position %d\n", c, str, last_index);
else
    printf("The letter %c was not found in %s\n",c,str);
```

Example 2

- Implement the function my_strcspn:
  - Input – two strings *str1*, *str2*
  - Output – the index in *str1* of the first instance of **any** of the characters contained in str2
- Write a program that accepts a string from the user and replaces all punctuation signs (,.;:!?) with spaces



my_strcspn

```
int my_strchr(char s[], char c) {
    int i = 0;

    while (s[i] != '\0') {
        if (s[i] == c)
            return i;

        i++;
    }

    return -1;
}
```

```
int my_strcspn(char str[], char find[]) {
    int i = 0;

    while (str[i] != '\0')
    {
            if (my_strchr(find, str[i]) > -1)
                return i;
            i++;
    }
    return -1;
}
```


main

```
char s[MAX_LENGTH+1];
int index;

printf("Please enter a line of text\n");
scanf("%100s", s);

index = my_strcspn(s, ".,;:!?");
while (index > -1)  {
    s[index] = ' ';
    index = my_strcspn(s, ".,;:!?");
}

printf("Resulting string is - %s\n", s);
```


Example 3

Write a program that gets a string from the user and checks whether it is a palindrome.

A palindrome example: *abbcbba*

(Hint: use strlen…)

Palindrome

```
int len,i;
char str[101];

printf("Enter a string\n");
scanf("%100s",str);
len = strlen(str);
for (i = 0; I < len/2; i++)
  if (str[i] != str[len-i-1]) {
    printf("The string is not a palindrome!\n");
    return 0;
  }
printf("The string is a palindrome!\n");
```


Example 4: Recursive strcmp

- Implement the function my_strcmp that compares two string **recursively**:
  - Input – two strings *str1*, *str2*
  - Output – 0 if the string are identical, the lexicographical order otherwise
- Use a helper recursive function that holds the current index to be compared in the strings

Solution

```
int myStrcmpRec(const char s[], const char t[], int i)
{
   if (s[i] != t[i])
      return s[i] - t[i];
   if (s[i] == '\0')
      return 0;
   return my_strcmp(s, t, i+1);
}

int my_strcmp(const char s[], const char t[]) {
      return myStrcmpRec(s,t,0);
}
```


Example 5: string2uint

- implement the function string2uint
  unsigned int string2uint(char str[]);
- The function scans a string that holds an positive integer and returns the corresponding unsigned integer
- Assume legal input
- How do we transform a digit-char to a digit-integer? Answer: -'0'

Example 5: string2uint (Cont.)

- write a program to test the above function
  - read a string that represents an integer from the user
  - call the function with the input
  - print the result.
- example
  - input: "1304", output: 1304
  - input: "0560", output: 560

Solution

```
unsigned int string2uint(char str[])
{
      int i = 0, res = 0;
      while (str[i] != '\0') {
            res = res * 10 + (str[i] - '0');
            ++i;
      }
      return res;
}
```


Solution – main (cont.)

```
char str[SIZE+1];
unsigned int uint;
printf("Please enter a string: ");
scanf("%100s", str);

uint = string2uint(str);

printf("the number is %u\n",uint);
```


Holding Multiple Strings

- How would we hold multiple strings?
- Why? Students names, for example
- An array on strings!
- → A 2D array of characters:

```
char str [NUM][MAX];
int i;
for (i = 0; i < NUM; ++i) {
     printf("enter string %d:\n",i);
     scanf("%s",str[i]);
}
```

Solution to class exercise

```
void replace(char str[], char replace_what,
                         char replace_with)
{
    int i;

    for (i = 0; str[i] != '\0'; ++i)
    {
        if (str[i] == replace_what)
            str[i] = replace_with;
    }
}
```


Solution (cont.)

```
#define STRING_LEN 100

int main(void)
{
    char str[STRING_LEN + 1];
    char replace_what, replace_with;

    printf("Please enter a string (no spaces)\n");
    scanf("%100s", str);

    printf("Letter to replace: ");
    scanf(" %c", &replace_what);

    printf("Letter to replace with: ");
    scanf(" %c", &replace_with);

    replace(str, replace_what, replace_with);

    printf("The result: %s\n", str);

    return 0;
}
```