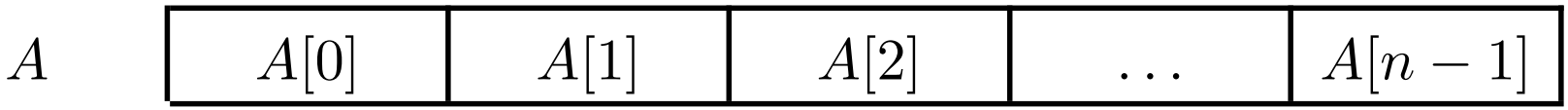
Simple Arrays and Strings

Computing Lab
https://www.isical.ac.in/~dfslab

Indian Statistical Institute

What is an array?

- **Sequences in mathematics:** $A = (A)_i = A_1, A_2, A_3, \ldots$
- **Arrays in C:**



  - Sequence of n *contiguous* memory locations
  - *Length* of the array $= n$
  - *Elements* of the array $\equiv$ each of the n memory locations
  - Elements numbered 0 through $n - 1$

Syntax

```
char charArray[128], c;    // charArray : array of 128 chars
int intArray[64], i, j;    // intArray : array of 64 ints
...
charArray[i] = c; // 0 <= i <= 127
intArray[0] = i; j = intArray[63];
```

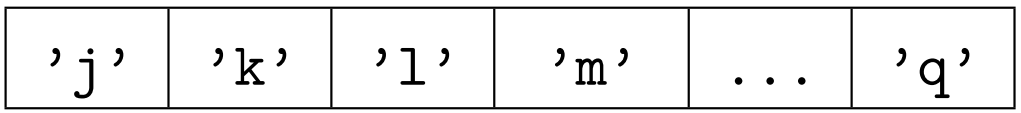
Strings

Definition

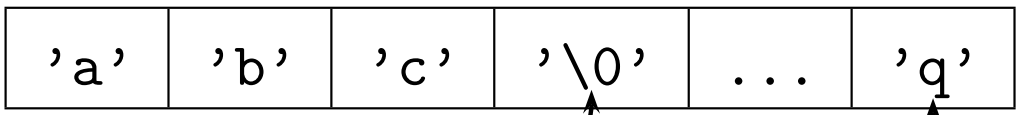
Strings are character arrays, but the end of the string is marked by the first occurrence of `'\0'` in the array (not the last element of the array)

Example:

```
1   char str0[8] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h' };
2   char str1[8] = { 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q' };
3   char str2[8] = { 'z', 'y', 'x', 'w', 'v', 'u', 't', 's' };
4
5   str1[0] = 'a'; str1[1] = 'b'; str1[2] = 'c'; str1[3] = '\0';
6   /* str1 now holds the string "abc" */
```



NOT a string



end of the string

end of the array

Review questions

1. Try
   `printf("%s\n%s\n%s\n", str0, str1, str2);`
   at lines 4 and 7 in the example code given above.
2. At lines 4 and 7, try
   `for (i=0; i<8; i++) printf("%c\n", str0[i]);`
   Repeat for `str1` and `str2`.
3. Print `str0`, `str1` and `str2` after replacing line 5 by
   (a) `strcpy(str1, "abc");`
   (b) `strncpy(str1, "abc", j);` for `j` $\in \{0, 1, 2, \ldots, 10\}$.
   (c) `strncpy(str1, "abcdefgh...xyz", j);` for `j` $\in \{0, 1, 2, \ldots, 26\}$.

Defining / initialising strings

```
char *str1 = "Style 1";
char str2a[] = "Style 2A";
char str2b[16] = "Style 2B";
```

`str2a` contains 8 + 1 bytes

`str2b` contains 16 bytes

`strlen(str2b)` is 8

Permitted operations

```
str1 = "Another string"; // change str1 itself
str1++;                  // move str1 1 char forward (to 'n')
str2a[i] = 'X';          // change elements of string;
                         //  0 <= i < sizeof(str2a)
strcat(str2b, str1);     // strcpy also works
```

Detailed discussion of `Style 1` after pointers are introduced

Defining / initialising strings

```
char *str1 = "Style 1";
char str2a[] = "Style 2A";
char str2b[16] = "Style 2B";
```

`str2a` contains 8 + 1 bytes

`str2b` contains 16 bytes

`strlen(str2b)` is 8

NOT permitted

```
str1[i] = 'X';              // changing the elements of the string
strcat(str1, str2b);        // strcpy will also NOT work
str2a++; str2b++;           // changing the array variable itself
str2a = "Another string";   // reassigning the array variable
str2b = "Another string";
```

Detailed discussion of `Style 1` after pointers are introduced

Some useful string library functions

- At the beginning of your program, write `#include<string.h>`
- Some useful functions

`$ man 3 string`

```
size_t   strlen(s);
-----------------------------------------------------
int      strcmp(s1, s2);
int      strncmp(s1, s2, n);
-----------------------------------------------------
char     *strcpy(destination, source);
char     *strncpy(destination, source, n);
-----------------------------------------------------
char     *strdup(s);
char     *strndup(s, n);
-----------------------------------------------------
char     *strcat(destination, source);
char     *strncat(destination, source, n);
```

- LATER: `*strchr(s, int)`, `*strrchr(s, int)`, `strstr(s, s)`

Programming problems I

1 *Run-length encoding and decoding.*

(a) Write a program to convert a given string s to s', its *run-length encoded* form. This means that s' will contain the same sequence of distinct characters as s, but any $m > 1$ consecutive occurrences of a character will be replaced by a single occurrence of the character immediately followed by the integer m (in base 10). For example, $aaabccd$ should be converted to $a3bc2d$.
The string s should be read from the terminal. It may contain letters (no digits), blanks and tabs (`'\t'`), but no newline. The length of the string s will not be known to you in advance.

(b) Add code to your program so that it prints the character that occurs consecutively the maximum number of times. For the example above, your program should print a. If the maximum number of consecutive occurrences is the same for two or more characters, you may print any one.

Programming problems II

(c) Modify your program so that it *decodes* a given run-length encoded string s' to its original form s. For example, given $a3bc2d$, your program should print $aaabccd$. Note that, given $abcd$, your program should print $abcd$ itself. The input format will be the same as for part (a).

NOTE: You do **not** need to use an array for this problem.

2. Write a program that reads text typed at the terminal, and counts the number of occurrences of *each of* the letters a–z in the text. For this problem, you should not distinguish between uppercase and lowercase letters. The input text may span multiple lines, and may contain digits, punctuation marks, blanks, tabs, and other printable characters, which you should ignore.

Programming problems III

3. Write a program that takes a single string as input, *reverses the string in place* and prints the reversed string to the terminal. You are told that the input string will contain at most 80 characters, and will not contain any whitespace (blanks, tabs or newlines).
4. Given an array of at most 100 integers, print the longest sequence of
   (a) elements that appear in ascending order;
   (b) *consecutive* elements that appear in ascending order.