-String-

Programming Fundamentals

String- Definition

- A string is **a sequence of characters that is treated as a single data item (null terminated)**
- String constant- any group of characters inside “ ” (double quotes)

e.g. “Programming Fundamentals”

Note: Use \ to escape “” inside a string

e.g. “\”Programming Fundamentals\””

- Character strings are often used to build meaningful and readable programs.

DECLARING AND INITIALIZING STRING VARIABLES

- C does not support strings as datatypes but allows representing them as character arrays.
- In C, string variable is any valid C variable name declared as an array of characters.
- Syntax: **char string_name[size];**
- size: determines the max number of characters in the string_name.
- E.g.: 1. char subjects[10];   //subjects is a array of 10 ele where each ele is a char; string of 10 characters (max)

  2. char students[30];

- When compiler assigns a character string to a character array,

it appends null character (‘\0’) at the end of the string.

So, size should be equal to the maximum number of characters in the string plus one.

- Like other arrays, character arrays can also be initialized at compile-time:
- E.g. char [8] = “Program”;

  char city [8]={‘P’, ’r’, ’o’, ’g’ , ’r’, ’a’ , ’m’, ‘\0’};

Note:

a. We can initialize character array without specifying #elements.

    char subject[]={‘P’, ’r’, ’o’, ’g’ , ’r’, ’a’ , ’m’, ‘\0’};

Size of the array is automatically taken based on initialization.

b. We can declare a character array with size greater than the string size in initialization.

    char subject[100]={‘P’, ’r’, ’o’, ’g’ , ’r’, ’a’ , ’m’, ‘\0’};

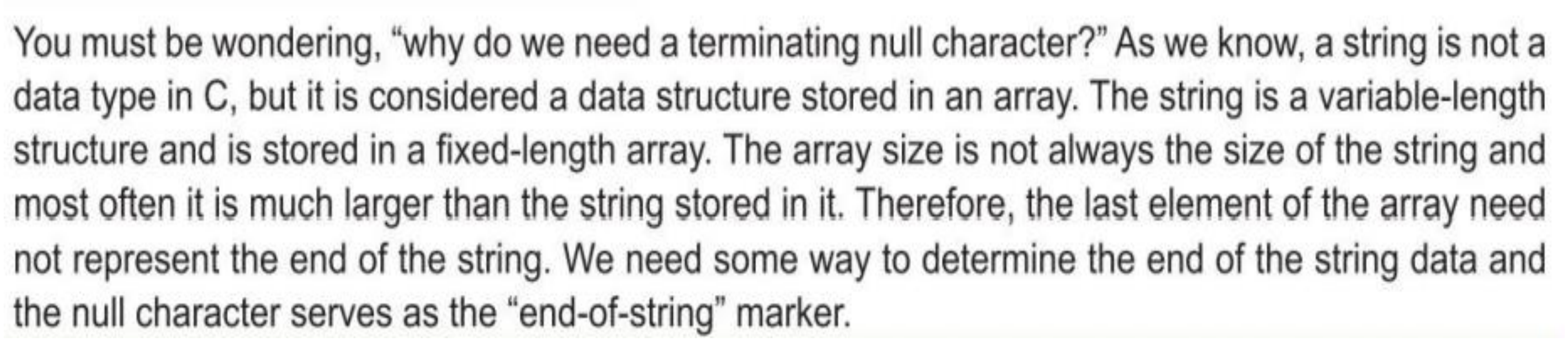
Compiler places the desired characters, ends it with a NULL character followed by NULL to all other positions/indices.

c. An array name can not be used on LHS of an assignment operator.

e.g. char name[10]=“Marlyn”; char copy[10]=name; //illegal

- Reason behind NULL (\0) character:

Serves as "end-of-string" marker to indicate end of a string (data-structure) inside character array (data-type)



You must be wondering, "why do we need a terminating null character?" As we know, a string is not a data type in C, but it is considered a data structure stored in an array. The string is a variable-length structure and is stored in a fixed-length array. The array size is not always the size of the string and most often it is much larger than the string stored in it. Therefore, the last element of the array need not represent the end of the string. We need some way to determine the end of the string data and the null character serves as the "end-of-string" marker.

READING STRINGS FROM TERMINAL

A. Using scanf Function

B. Using getchar and gets Functions

A. Using scanf Function

char names[10];

scanf("%s",names); //this is a book (writing array-name means base-address; no need to provide &)

printf("%s",names); //this

Note:

1. scanf stops reading input on the first whitespace character (blank, tab, new line etc.)
2. scanf automatically terminates the character array with a NULL character (array should be able to hold the input string plus the null character)

3. Can specify field with as: scanf(“%**ws**”, char_array_name);

If field width >= #characters in input, then entire string is stored in string variable

If field width < #characters in input, extra characters are truncated and left unread.

Reading a Line of Text:

char line [80];

scanf**(”%[^\n]**”, line); //this is a book

printf(“%s”, line); //this is a book

B.

i. getchar(): Use this function repeatedly to read successive single characters from the input and save it in character array.

- This can be achieved until \n is entered and then inserting \0 at the end of the string

ii. gets(): more convenient to read entire string.

- It keeps on reading characters until a new line is encountered and then appends \0 at the end.
- It does not skip whitespaces.

e.g. char line [80];

gets (line); //it is programming

printf (“%s”, line); // it is programming

WRITING STRINGS TO SCREEN

A. Using printf Function
B. Using putchar and
C. Using puts Functions

A. printf(): Format %s can be used to display an array of characters that is terminated by the \0.

```
    e.g. printf("%s", char_array_name);
```

B. i. putchar(): output the values of character variables.

```
e.g.    char name[6] = "PARIS" ;
        for (i=0, i<5;i++) putchar(name[i]);
```

ii. puts(): more convenient to read entire string.

- This prints the value of the string variable and then puts the cursor to next line.

e.g. char line [80];

```
        gets (line);
        puts (line);
```

ARITHMETIC OPERATIONS ON CHARACTERS

- We can manipulate char same way as we do for int.
- When a character variable/constant is used inside an expression, it is converted to its integer equivalent (ASCII)

e.g.:

```
1. x = 'a';
        printf("%d\n",x);
2. x = 'z'–1;
3. ch >= 'A' && ch <= 'Z'
```

stdlib.h: atoi()

- The C library supports a function that converts a string of digits into their integer values.
- Syntax: **atoi(“1988”);**

```
e.g.: char number[10] = “1988”;
      year = atoi(number);
```

PUTTING STRINGS TOGETHER (concatenate)

- string3 = string1 + string2; //not valid in C

The characters from string1 and string2 should be copied into the string3 one after the other.

Note: Size of string3 should be large enough.

COMPARISON OF TWO STRINGS

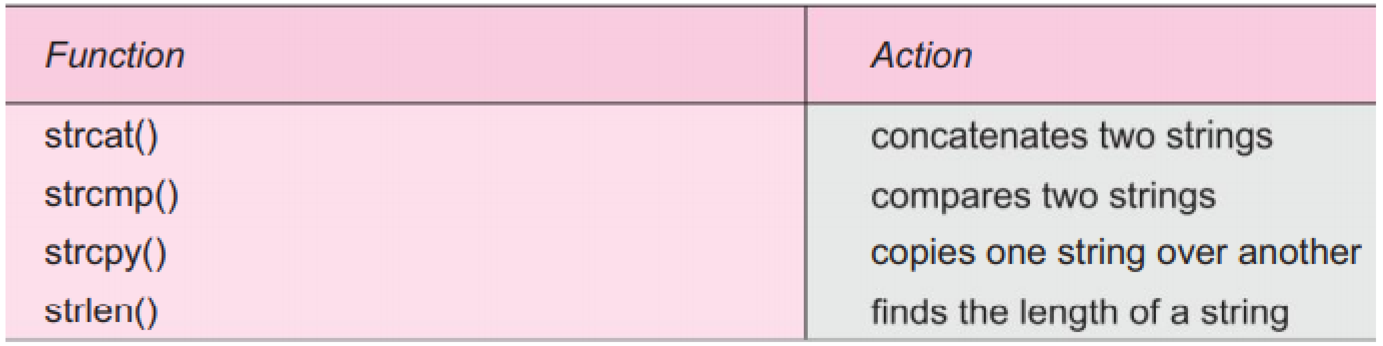
- if(name1 == name2)
- if(name == “ABC”)

If required, comparison is done on the string character by character until there is a mismatch or one of the strings terminates into a null character.

STRING-HANDLING FUNCTIONS

- Common functions on strings:

These functions are present in string.h library.



Function	*Action*
strcat()	concatenates two strings
strcmp()	compares two strings
strcpy()	copies one string over another
strlen()	finds the length of a string

TABLE OF STRINGS

```
char city[5][25] ={ “Chandigarh”,
                     “Madras”,
                    “Ahmedabad”,
                    “Hyderabad”,
                     “Bombay” }
```

To refer: 1st city/0th index city ➔ city[0]

2nd city/1st index city → city[1]