Programming in C

Chapter: Arrays and Strings

Study Notes for Engineering / Polytechnic Students

3.1 Arrays

An array is a collection of elements of the same data type, stored in contiguous memory locations, and referred to by a single common name. Each element of an array is accessed using an index, also called a subscript. Because an array behaves like a single variable holding multiple values, it is often called a subscripted variable.

Advantages of Subscripted Variables / Arrays

- A single array name can represent many related values, avoiding the need to declare a separate variable for each value.
- Elements are stored in contiguous memory, allowing fast, direct, and random access to any element using its index.
- Arrays make it possible to process large sets of data efficiently using loops, instead of writing repetitive code for each value.
- They simplify sorting, searching, and other common data-processing operations by giving structured access to a sequence of values.
- Arrays form the basis for more advanced data structures, such as matrices, strings, stacks, and queues.

One Dimensional Arrays

A one dimensional array stores a linear list of elements. It is declared by specifying the data type, the array name, and the number of elements (the size) in square brackets.

```
data_type array_name[size];

// Declaration
int marks[5];

// Declaration with initialization
int marks[5] = {70, 65, 90, 55, 82};

// Size can be omitted when initializing directly
int marks[] = {70, 65, 90, 55, 82};
```

Two Dimensional Arrays

A two dimensional array stores data in a table-like structure of rows and columns, and is commonly used to represent matrices. It requires two subscripts — one for the row and one for the column.

```
data_type array_name[rows][columns];

// Declaration
int matrix[3][3];
```

```
// Declaration with initialization
int matrix[2][3] = { {1, 2, 3}, {4, 5, 6} };
```

Character Arrays

A character array stores a sequence of characters and is the fundamental way strings are represented in C. When initialized with a string constant, the compiler automatically appends a null character ('\0') at the end to mark where the string terminates.

```
char name[20];

// Initialization with a string literal
char name[20] = "Engineering";

// Initialization as a list of characters (null terminator added manually)
char name[] = {'C', ' ', 'L', 'a', 'n', 'g', '\0'};
```

Accessing Array Elements

In C, array indexing always begins at zero. This means that for an array of size n, valid indices run from 0 to n minus 1. An individual element is accessed by writing the array name followed by the required index in square brackets.

```
int marks[5] = {70, 65, 90, 55, 82};

printf("%d", marks[0];   // prints 70, the first element
printf("%d", marks[4];   // prints 82, the last element
marks[2] = 100;           // modifies the third element

// Accessing a two dimensional array element
printf("%d", matrix[1][2];   // row index 1, column index 2
```

Loops, especially the for loop, are typically used to access or process every element of an array systematically, since the loop counter can double as the array index. It is the programmer's responsibility to ensure that an index always stays within the valid range, since C does not perform automatic bounds checking, and accessing an out-of-range index leads to undefined behaviour.

3.2 Strings and String Handling

In C, a string is simply an array of characters terminated by a null character ('\0'). There is no separate built-in string data type; instead, strings are handled through character arrays and, together with pointers, through the standard string-handling library declared in the header file string.h.

Declaration and Initialization of String Variables

```
// Declaration only
char city[30];

// Declaration with initialization using a string literal
char city[30] = "Kolkata";

// Reading a string from the keyboard
scanf("%s", city);       // no & needed, since the array name is already an address
```

```
// Reading a string that may contain spaces
fgets(city, sizeof(city), stdin);
```

The compiler reserves one extra byte beyond the visible characters of a string literal to store the terminating null character, so an array meant to hold a string of length n must be declared with a size of at least n plus 1.

String Handling Functions from the Standard Library

The header file string.h provides several ready-made functions for common string operations, removing the need to write these routines manually every time.

- strlen(str): Returns the length of the string, that is, the number of characters before the terminating null character. It does not count the null character itself.
- strcpy(dest, src): Copies the string src, including its terminating null character, into the character array dest. The destination array must be large enough to hold the copied string.
- strcat(dest, src): Appends (concatenates) the string src to the end of the string dest, and places a single null character at the new end. The dest array must have enough spare space to accommodate the added characters.
- strcmp(str1, str2): Compares two strings character by character and returns zero if they are identical, a negative value if str1 is alphabetically less than str2, and a positive value if str1 is alphabetically greater than str2.

```
#include <string.h>

char a[20] = "Hello";
char b[20] = "World";

printf("%d", strlen(a));      // 5
strcpy(b, a);                 // b now holds "Hello"
strcat(a, b);                 // a now holds "HelloHello"
printf("%d", strcmp(a, b));   // non-zero, since a and b now differ
```

Extracting a Substring — Left, Right, and Middle

The standard library does not provide direct left(), right(), or mid() functions as found in some other languages; the same effect is achieved in C using a combination of strncpy(), array indexing, and pointer arithmetic, along with strlen() to determine the total length of the source string.

- Left substring: Copy the required number of characters starting from index 0 of the source string using strncpy(), and then manually place a null character at the end of the destination array.
- Right substring: Calculate the starting index as (total length minus the number of characters required), and copy from that position to the end of the string.
- Middle substring: Copy the required number of characters starting from a chosen offset within the string, again followed by a manually placed null terminator.

```
char str[] = "Programming";
char result[20];
int len = strlen(str);         // 11

// Left 4 characters -> "Prog"
strncpy(result, str, 4);
result[4] = '\0';
```

```
// Right 4 characters -> "ming"
strncpy(result, str + (len - 4), 4);
result[4] = '\0';

// Middle 4 characters starting at offset 3 -> "gram"
strncpy(result, str + 3, 4);
result[4] = '\0';
```

Replacement of String Characters

Since C strings are simply character arrays, an individual character within a string can be replaced directly through array indexing, without needing any library function. Replacing an existing character does not change the length of the string, since one character is simply substituted for another at the same position.

```
char str[] = "Programming";
str[0] = 'p';        // "programming"

// Replacing every occurrence of one character with another
for (int i = 0; str[i] != '\0'; i++)
{
    if (str[i] == 'm')
        str[i] = 'M';
}
// result: "prograMMing"
```

Concatenation of Two Strings

Concatenation means joining one string to the end of another to form a single, combined string. In C, this is most commonly done using the strcat() function from string.h, which appends its second argument to the first.

```
char first[30] = "Web";
char second[] = "ByArun";

strcat(first, second);    // first now holds "WebByArun"
printf("%s", first);
```

Care must always be taken to ensure that the destination array passed to strcat() is large enough to hold both original strings together, along with the terminating null character; otherwise the operation may write beyond the bounds of the array and corrupt adjacent memory.

Multiple Choice Questions (15)

1. In C, array indices always start from:

a) 1

b) -1

c) 0

d) The array size

2. Which of the following is NOT an advantage of using arrays?

a) Fast, direct access to any element via its index

b) A single name can represent many related values

c) Arrays automatically check for out-of-bounds access

d) They make loop-based processing of data easier

3. Which declaration correctly creates a two dimensional integer array with 4 rows and 3 columns?

a) int arr[4, 3];

b) int arr(4)(3);

c) int arr[4][3];

d) int arr[4-3];

4. What is automatically added at the end of a character array initialized with a string literal?

a) A blank space

b) The null character '\0'

c) The character '\n'

d) Nothing is added

5. For the declaration int marks[] = {70, 65, 90, 55, 82};, what is the size of the array?

a) 4

b) 5

c) 6

d) Undefined

6. In the statement matrix[1][2], what do 1 and 2 represent?

a) Two different arrays

b) Row index and column index

c) Two column indices

d) The size of the array

7. What happens when a C program accesses an array index outside its valid range?

a) The compiler automatically stops it

b) It results in undefined behaviour

c) The index is automatically adjusted

d) The program pauses and asks for a valid index

8. In C, a string is internally represented as:

a) A built-in string data type

b) A character array terminated by a null character

c) An integer array

d) A pointer to a float

9. Which function is used to find the length of a string in C?

a) strcat()

b) strcmp()

c) strlen()

d) strcpy()

10. What does strcpy(dest, src) do?

a) Compares dest and src

b) Copies src into dest

c) Appends src to dest

d) Finds the length of src

11. What does strcmp(str1, str2) return when the two strings are identical?

a) 1

b) -1

c) 0

d) The length of str1

12. Which function is most commonly used to join (concatenate) two strings in C?

a) strlen()

b) strcmp()

c) strcat()

d) strncpy()

13. Which header file must be included to use functions like strcpy(), strcat(), and strcmp()?

a) stdio.h

b) stdlib.h

c) string.h

d) math.h

14. To extract a substring from the middle of a string, which approach is typically used in C?

a) The built-in mid() function

b) strncpy() with a chosen starting offset

c) strcmp() with two strings

d) It is not possible to extract a middle substring in C

15. Replacing a single character within a string at a fixed position:

a) Always increases the string's length

b) Always decreases the string's length

c) Does not change the string's length

d) Requires the strcat() function

Answer Key

1-c, 2-c, 3-c, 4-b, 5-b, 6-b, 7-b, 8-b, 9-c, 10-b, 11-c, 12-c, 13-c, 14-b, 15-c

Broad / Descriptive Questions (10)

1. What are the advantages of using arrays (subscripted variables) over declaring separate variables for each value? Explain with an example.

2. Explain the declaration and initialization of a one dimensional array in C, with a suitable example program.

3. Explain the declaration and initialization of a two dimensional array in C. How are its elements accessed? Give an example.

4. What is a character array? Explain how it differs from a numeric array, with reference to the null character.

5. Explain, with an example, how individual elements of a one dimensional and a two dimensional array are accessed in C.

6. What is a string in C? Explain how string variables are declared, initialized, and read from the keyboard, with suitable examples.

7. Explain any four string handling functions available in the string.h library, giving the purpose and a short example of each.

8. Explain, with example code, how the leftmost, rightmost, and middle substrings of a given string can be extracted in C.

9. Explain how an individual character within a string can be replaced in C. Write a program to replace all occurrences of a given character in a string.

10. Explain the concatenation of two strings using strcat(). Write a C program to concatenate two strings and then compare them using strcmp().