Shri Vile Parle Kelavani Mandal's

SHRI BHAGUBHAI MAFATLAL POLYTECHNIC

Computer Engineering Department

Arrays and String

Course: Programming in C

Course Code: PRC238912

Name of Staff: Mr. Pratik H. Shah

SEMESTER : II

DIVISION : A

Course Outcome – 4

Use primary, derived and user defined data types in application programs

Student will be able to

- Use Array as derived data types in developing applications
- Implement various string handling function

Introduction to Arrays

1. An array is a collection of data that holds fixed number of values of same type.
2. An array is a variable that can store multiple values.

Example

if you want to store 100 integers, you can create an array for it.

Types of an Array

1. Single Dimensional Array
2. Multidimensional Array

How to declare an array?

1. To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows

datatype array_name[array_size];

Here, we declared an array, mark, of floating-point type and size 5. Meaning, it can hold 5 floating-point values.

float mark[5];

How to initialize an array?

1. It is possible to initialize an array during declaration.
2. You can initialize an array in C either one by one or using a single statement as follows −

   int mark[5] = {19, 10, 8, 17, 9};

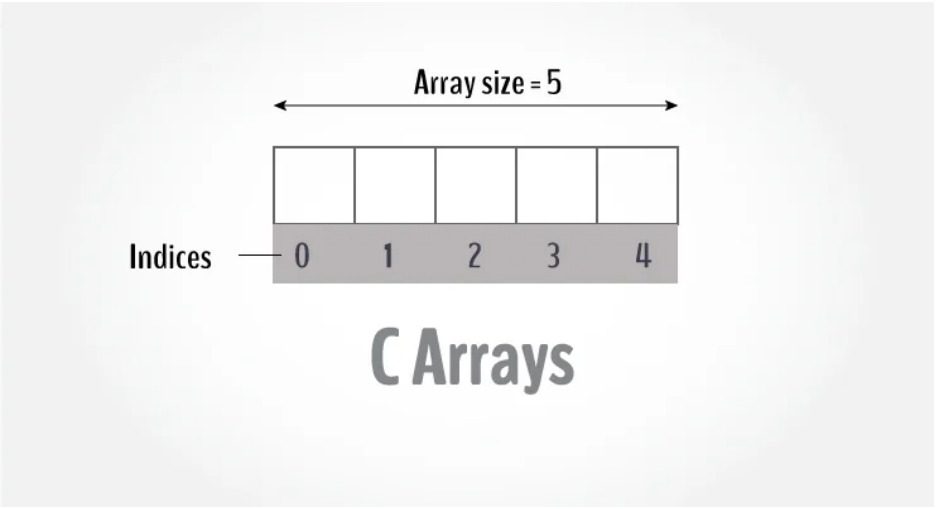
3. The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].

   int mark[] = {19, 10, 8, 17, 9};

   Here, we haven't specified the size. However, the compiler knows its size is 5 as we are initializing it with 5 elements.

Access Array Elements

1. You can access elements of an array by indices.
2. Suppose you declared an array mark as above.
3. The first element is mark[0], the second element is mark[1] and so on.



mark[0]	mark[1]	mark[2]	mark[3]	mark[4]
19	10	8	17	9

mark[0] is equal to 19
mark[1] is equal to 10
mark[2] is equal to 8
mark[3] is equal to 17
mark[4] is equal to 9

Few keynotes:

1. Arrays have **0 as the first index**, not 1. In this example, mark[0] is the first element.
2. If the size of an array is n, to access the **last element, the n-1** index is used. In this example, mark[4]
3. Suppose the starting **address of mark[0] is 2120d**. Then, the address of the mark[1] will be 2122d. Similarly, the address of mark[2] will be 2124d and so on.
4. This is because the **size of int is 2 bytes**.

Example 1: Array Input/output

```
#include <stdio.h>
void main()
{
  int values[5];
  int i;
  printf("Enter 5 integers: ");

     // taking input and storing it in an array
    for(i = 0; i < 5; i++)
    {
        scanf("%d", &values[i]);
    }

    printf("Displaying integers: ");

    // printing elements of an array
   for(i = 0; i < 5; i++)
  {
      printf("%d\n", values[i]);
   }
}
```

Output

```
Enter 5 integers: 1
-3
34
0
3

Displaying integers: 1
-3
34
0
3
```

Example 2: Calculate Average

```
#include <stdio.h>
void main()
{
   int marks[10], i, n, sum = 0;
   double average;
   printf("Enter number of elements: ");
   scanf("%d", &n);

  for(i=0; i < n; ++i)
  {
      printf("Enter number%d: ",i+1);
      scanf("%d", &marks[i]);
     sum += marks[i];
   }
   average = (double) sum / n;
   printf("Average = %.2lf", average);
}
```

Output

```
Enter number of elements: 5
Enter number1: 45
Enter number2: 35
Enter number3: 38
Enter number4: 31
Enter number5: 49
Average = 39.60
```

Access elements out of its bound!

Suppose you declared an array of 10 elements. Let's say,

int testArray[10];

You can **access** the array elements from **testArray[0] to testArray[9].**

Now let's say if you **try to access testArray[12].** The element is not available. This may cause unexpected output (undefined behavior). Sometimes you might get an error and some other time your program may run correctly.

Hence, you should never access elements of an array outside of its bound.

Multidimensional Arrays

1. In C programming, you can create an array of arrays. These arrays are known as multidimensional arrays

Example

float x[3][4];

- Here, **x is a two-dimensional (2d) array.**
- The array can hold 12 elements. Y
- You can think the array as a table with 3 rows and each row has 4 columns.

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Initializing a multidimensional array

```
// Different ways to initialize two-dimensional array

int c[2][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[2][3] = {1, 3, 0, -1, 5, 9};
```

Example 1: 2D Array Input/output

```
#include <stdio.h>
void main()
{
    int a[2][2], i,j;
   printf("Enter elements \n");
   for (i = 0; i < 2; i++)
   {
      for (j = 0; j < 2; j++)
      {
         printf("Enter a%d%d: ", i + 1, j + 1);
         scanf("%d", &a[i][j]);
      }
   }
   for (i = 0; i < 2; i++)
   {
      for (j = 0; j < 2; j++)
      {
         printf("\n a%d%d:%d ", i + 1, j + 1,a[i][j]);
      }
   }
}
```

Output

```
Enter elements
Enter a11: 2
Enter a12: 5
Enter a21: 1
Enter a22: 3

a11:2
a12:5
a21:1
a22:3
```

Pass arrays to a function in C

In C programming, you can pass an entire array to functions.

Example 1: Pass Individual Array Elements

```
#include <stdio.h>
void display(int age1, int age2)
 {
  printf("%d\n", age1);
  printf("%d\n", age2);
}

void main()
{
  int ageArray[] = {2, 8, 4, 12};
  display(ageArray[1], ageArray[2]);
}
```

Output

```
8
4
```

String

Introduction to String

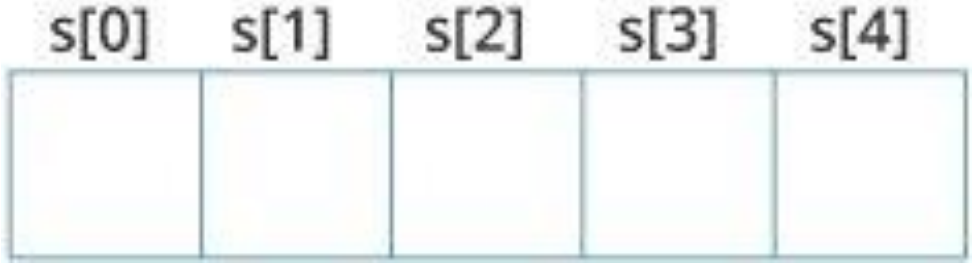
1. In C programming, a string is a sequence of characters terminated with a null character \0.
2. When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character \0 at the end by default.

Example

```
char c[] = "c string";
```

How to declare a string?

```
char s[5];
```



Here, we have declared a string of 5 characters.

How to initialize strings?

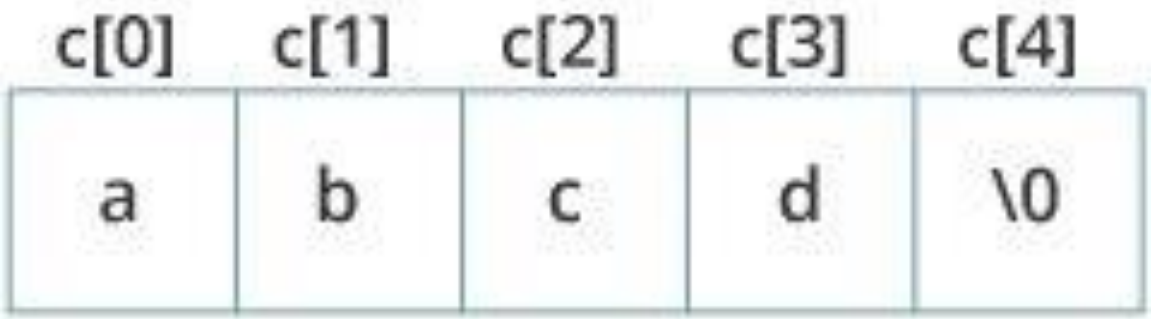
```
// Different ways to initialize String

char c[] = "abcd";

char c[50] = "abcd";

char c[] = {'a', 'b', 'c', 'd', '\0'};

char c[5] = {'a', 'b', 'c', 'd', '\0'};
```



Assigning Values to Strings

Arrays and strings are second-class citizens in C; they do not support the assignment operator once it is declared

```
char c[100];
c = "C programming";  // Error! array type is not assignable.
```

Note: Use the **strcpy()** to copy the string instead.

Read String from the user

You can use the **scanf()** function to read a string.

The scanf() function reads the sequence of characters until it encounters whitespace (space, newline, tab, etc.).

Example 1: scanf() to read a string

```
#include <stdio.h>
void main()
{
   char name[20];
   printf("Enter name: ");
   scanf("%s", name);
   printf("Your name is %s.", name);
}
```

Output

```
Enter name: Dennis Ritchie
Your name is Dennis.
```

EXPLANATION of PREVIOUS SLIDE PROGRAM

Even though Dennis Ritchie was entered in the above program, **only "Dennis" was stored in the name string.** It's because there was a space after Dennis.

Also notice that we have used the code name instead of &name with scanf().

scanf("%s", name);

This is because name is a char array, and we know that array names decay to pointers in C.

Thus, the name in scanf() already points to the address of the first element in the string, which is why we don't need to use &.

How to read a line of text?

1. You can use the **fgets()** function to **read** a line of string.
2. You can use **puts()** to **display** the string.

Example 2: fgets() and puts()

```
#include <stdio.h>
void main()
{
   char name[30];
   printf("Enter name: ");
   fgets(name, sizeof(name), stdin);  // read string
   printf("Name: ");
   puts(name);          // display string
}
```

Output

```
Enter name: Tom Hanks
Name: Tom Hanks
```

EXPLANATION of PREVIOUS SLIDE PROGRAM

1. Here, we have used fgets() function to read a string from the user.

2. fgets(name, sizeof(name), stdlin); // read string

3. The sizeof(name) results to 30. Hence, we can take a maximum of 30 characters as input which is the size of the name string.

4. To print the string, we have used puts(name);.

5. **Note:** The gets() function can also be to take input from the user. However, it is removed from the C standard.

6. It's because gets() allows you to input any length of characters. Hence, there might be a buffer overflow.

String Handling Function

1. You need to often manipulate strings according to the need of a problem.
2. Most, if not all, of the time string manipulation can be done manually but, this makes programming complex and large.

3. To solve this, C supports a large number of string handling functions in the standard library **"string.h".**

Few commonly used string handling functions

Function	Work of Function
strlen()	computes string's length
strcpy()	copies a string to another
strcat()	concatenates(joins) two strings
strcmp()	compares two strings
strlwr()	converts string to lowercase
strupr()	converts string to uppercase

C strlen()

1. The strlen() function calculates the length of a given string.
2. The strlen() function takes a string as an argument and returns its length.
3. It is defined in the <string.h> header file.
4. **Note:** strlen() function doesn't count the null character \0 while calculating the length.

Example: C strlen() function

```
#include <stdio.h>
#include <string.h>
void main()
{
   char a[20]="Program";

   printf("Length of string a = %d \n",strlen(a));
}
```

Output

```
Length of string a = 7
```

C strcpy()

1. The strcpy() function copies the string pointed by source (including the null character) to the destination.
2. The strcpy() function also returns the copied string.
3. It is defined in the <string.h> header file.
4. **Note:** When you use strcpy(), the size of the destination string should be large enough to store the copied string. Otherwise, it may result in undefined behavior.

Example: C strcpy() function

```
#include <stdio.h>
#include <string.h>
void main()
{
  char str1[20] = "C programming";
  char str2[20];
  strcpy(str2, str1);
  puts(str2);                 // C programming
}
```

Output

C programming

C strcat()

1. the strcat() function contcatenates (joins) two strings.
2. The strcat() function concatenates the destination string and the source string, and the result is stored in the destination string..
3. It is defined in the <string.h> header file.
4. **Note:** When we use strcat(), the size of the destination string should be large enough to store the resultant string. If not, we will get the segmentation fault error.

Example: C strcat() function

```
#include <stdio.h>
#include <string.h>
void main()
{
  char str1[100] = “Hello ", str2[] = “World";
  strcat(str1, str2);
  puts(str1);
  puts(str2);
}
```

Output

Hello World
World

C strcmp()

1. The strcmp() compares two strings character by character.
   If the strings are equal, the function returns 0.

strcmp() Parameters

The function takes two parameters:

str1 - a string

str2 - a string

The strcmp() function is defined in the string.h header file.

Return Value from strcmp()

Return Value	Remarks
0	if strings are equal
> 0	if the first non-matching character in str1 is greater (in ASCII) than that of str2.
< 0	if the first non-matching character in str1 is lower (in ASCII) than that of str2.

Example: C strcmp() function

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

void main()
{
   char str1[] = "abcd", str2[] = "abCd", str3[] = "abcd";
   int result;

      // comparing strings str1 and str2
   result = strcmp(str1, str2);
   printf("strcmp(str1, str2) = %d\n", result);

     // comparing strings str1 and str3
   result = strcmp(str1, str3);
   printf("strcmp(str1, str3) = %d\n", result);

}
```

Output

```
strcmp(str1, str2) = 1
strcmp(str1, str3) = 0
```

Exercise

1. Write a program in C to sort the numbers in Ascending order using Arrays.
2. Write a program in C to scan and print 2 dimensional matrix using Arrays
3. Write a program in C to Add 2 matrices of 3*3 order using Arrays.
4. Write a program in C to search the elements using Arrays
5. Write a program in C to find the determinant of 2*2 order using Arrays
6. Write a program in C to Find the frequency of a character in a string.
7. Write a program in C to Find the number of vowels, consonants, digits and white spaces.
8. Write a program in C to Reverse a string using recursion.
9. Write a program in C to Remove all characters in a string except alphabets.
10. Write a program in C to Sort elements in the lexicographical order (dictionary order)

Theory Questions

1. Define Array
2. Write the syntax of two dimensional array
3. Explain Array to function with example program
4. Define String
5. Write the syntax of any two string function
6. Explain the use of following string function with example program:
   a)strcat() b)strlen() c)strcmp() d)strcpy
7. Enlist all the string handling function

THANK YOU !!!!