Structured Programming using C

RCP2SFCES101

Unit-5

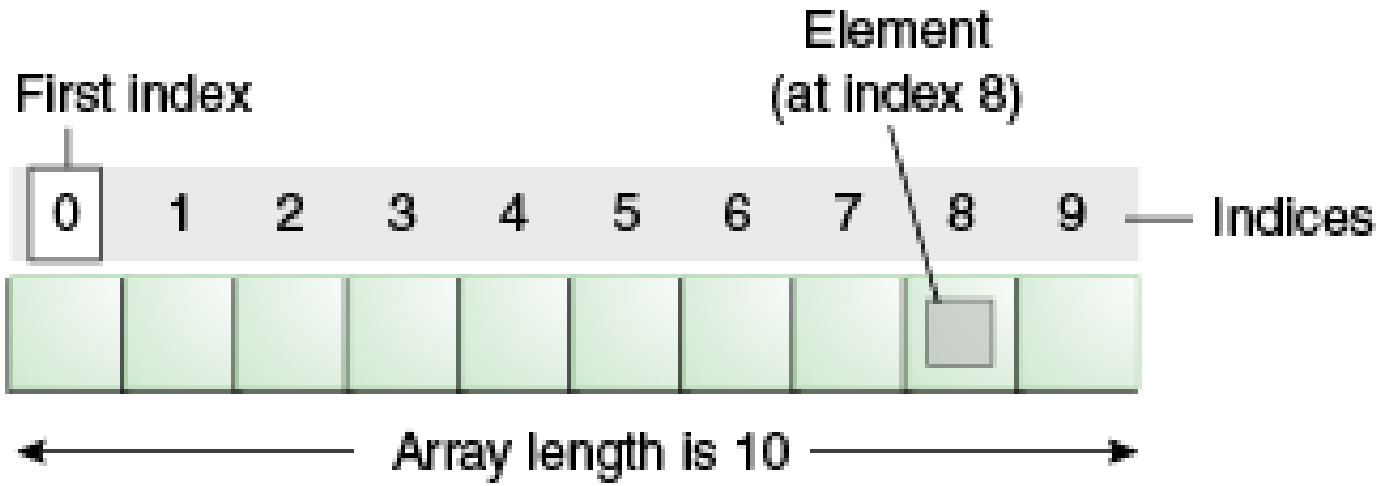
Arrays, String, Structure

Contents

Arrays

Arrays

- An array is collection of elements of similar type that shares a common name.
- The elements of an array are stored in a contiguous memory location.
- We can store only a fixed set of elements in a Java array.
- Array are index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.



Types of Array

There are two types of array.

1. One Dimensional Array (1D array)
2. Multidimensional Array (2D array)

One Dimensional Array (1D array)

A one-dimensional array stores elements in a linear format.

Declaration of an 1D array:

Syntax:

```
data-type arrName[size];
```

Here,

data_type: specifies the type of element that will be contained in array, such as int, float, or char etc.

arrName: specifies name of array.

size: indicates maximum number of elements that can be stored in an array.

Example:

```
int a[5];      // Declares an array of 5 integers
float b[10];   // Declares an array of 10 floats
```

One Dimensional Array (1D array) contd..

Initialization of an 1D array:

Arrays can be initialized at the time of declaration.

Syntax:

data-type arrName[size] = {list of values};

Example:

int a[5] = {20, 10, 50, 40, 30};

a[0]	a[1]	a[2]	a[3]	a[4]
20	**10**	**50**	**40**	**30**
1000	1002	1004	1006	1008

One Dimensional Array (1D array) contd..

Accessing Elements of 1D array:

Array elements are accessed using their **index**. Indexing starts at 0.

Example:

```
int a[5] = {20, 10, 50, 40, 30};

printf("%d", a[2]);      // Will output 50 (Third Element)
```

a[0]	a[1]	a[2]	a[3]	a[4]
20	**10**	**50**	**40**	**30**
1000	1002	1004	1006	1008

Example: 1D array

```c
//Program to initialize array dynamically and display its elements

#include<stdio.h>
int main()
{
    int a[5];

    printf("Enter array elements: ");
    for(int i=0; i<5; i++)
    {
        scanf("%d",&a[i]);   //reading array elements
    }

    printf("Array elements are: ");
    for(int i=0; i<5; i++)
    {
        printf("%d\n",a[i]); //displaying array elements
    }

    return 0;
}
```

Output:

```
Enter array elements:
10 30 20 50 40
Array elements are:
10
30
20
50
40
```

Multidimensional Array (2D array)

A multidimensional array can store data in a tabular format (like matrices).

Declaration of an 2D array:

Syntax:

```
data-type arrName[rows][cols];
```

Here,

data_type: specifies the type of element that will be contained in array, such as int, float, or char etc.

arrName: specifies name of array.

rows: indicates maximum number of rows.

cols: indicates maximum number of columns

Example:

```
int a[3][2];    // Declares an array of integers having 3 rows and 2 columns
```

Multidimensional Array (2D array) contd...

Initialization of an 2D array:

Arrays can be initialized at the time of declaration.

Syntax:

data-type arrName[rows][cols] = {list of values};

Example:

int a[3][2] = {20, 10, 50, 40, 30, 60};

OR

int a[3][2] = {{20, 10},
{50, 40},
{30, 60}};

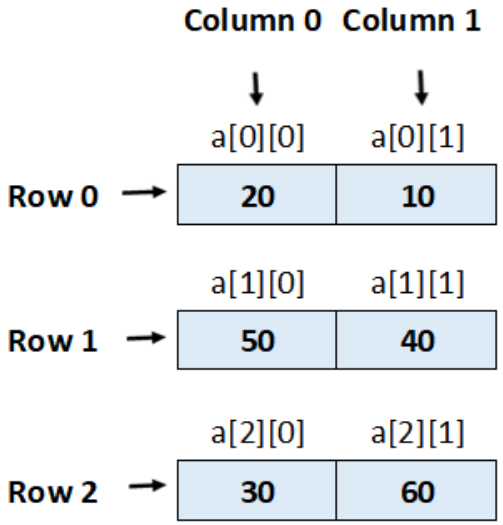


Figure: Representation of 2D array in memory

Multidimensional Dimensional Array (2D array) contd..

Accessing Elements of 2D array:

Elements of 2D array can be accessed using their row and column indices.

Example:

```
int a[3][2] = {{20, 10},
               {50, 40},
               {30, 60}};

a[2][1] = 10;            // Assigns 10 to the element at 2nd row, 1st column
printf("%d", a[2][1]);   // Prints element at 2nd row, 1st column
```

Example: 2D array

```c
//Program to demonstrate 2D array

#include<stdio.h>
int main()
{
    int a[3][2]={{1,3},{4,2},{6,5}};

    printf("Array elements are:\n");
    for(int i=0; i<3; i++)
    {
        //displaying array elements
        for(int j=0; j<2; j++)
        printf("%d ",a[i][j]);

        printf("\n");
    }

    return 0;
}
```

Output:

Array elements are:
1 3
4 2
6 5

Passing Arrays to Function

- Like simple variable, it is possible to pass the values of array to function.
- To pass 1D array to function it is sufficient to list name of the array without any subscripts, and size of the array as arguments

Example: Passing Arrays to Function

```c
//Program to calculate sum of array elements using user defined function

#include <stdio.h>

// Function to calculate sum of array elements
int calculateSum(int arr[], int size)
{
    int sum = 0;
    for (int i = 0; i < size; i++)
    {
        sum = sum + arr[i];
    }
    return sum;
}

int main()
{
    int arr[5] = {5, 10, 15, 20, 25};  // Array initialization

    // Call function and print result
    int result = calculateSum(arr, 5);
    printf("Sum of array elements: %d\n", result);

    return 0;
}
```

String

String

- In C programming, strings are a sequence of characters terminated by a null character '\0'
- C does not support strings as a data type.
- However, C allows us to represent strings as array of characters
- **Example:** char city[10] = "Shirpur";

  OR

  char city[ ] = "Shirpur";

S	h	i	r	p	u	r	'\0'
0	1	2	3	4	5	6	7

String Declaration and Initialization

String Declaration

- **Syntax:**

  char string_name[size];

- **Example:**

  char city[10];

String Initialization

- **Syntax:**

  char string_name[size] = value;

- **Example:**

  char city[10] = "Shirpur";

  OR

  char city[ ] = {'S', 'h', 'i', 'r', 'p', 'u', 'r', '\0' };

String Input and Output- Using scanf() and printf()

```c
//Example: String Input Output using scanf() and printf()

#include<stdio.h>
int main()
{
    char name[10];

    printf("Enter name: ");
    scanf("%s", name);

    printf("%s",name);
    return 0;
}
```

Output:

```
E:\RCPIT Docs\Academic Documents\AY_2024-25_ODD
Enter name: RCPIT
RCPIT
--------------------------------
```

```
E:\RCPIT Docs\Academic Documents\AY_2024-25
Enter name: Hello World
Hello
```

- Stops reading input at whitespace.
- Limitation: Cannot handle multi-word strings.

String Input and Output- Using gets() and puts()

```c
//Example: String Input Output using gets() and puts()

#include<stdio.h>
int main()
{
    char name[10];

    printf("Enter name: ");
    gets(name);

    puts(name);
    return 0;
}
```

Output:

```
E:\RCPIT Docs\Academic Documents\AY_2024-25_ODD Ser
Enter name: Hello World
Hello World
```

- Reads the entire line but is unsafe due to buffer overflow risks.

String Functions

C provides the following string manipulation functions that are present in <**string.h**> library:

Function	Description
strlen(s)	Returns the length of the string (excluding '\0').
strcpy(dest, src)	Copies **src** to **dest**.
strcat(dest, src)	Concatenates **src** to **dest**.
strcmp(s1, s2)	Compares two strings lexicographically.
strrev(s)	Reverses a string

Example: strlen()

```c
#include<stdio.h>
#include<string.h>
int main()
{
    char name[10];

    printf("Enter any String: ");
    gets(name);

    int n = strlen(name);
    printf("Length: %d",n);

    return 0;
}
```

Output

```
E:\RCPIT Docs\Academic Documents\AY_2024-25_ODD Semest
Enter any String: Rcpit
Length: 5
--------------------------------
```

Example: strcpy()

```c
#include<stdio.h>
#include<string.h>
int main()
{
    char src[10]="RCPIT", dest[10];

    strcpy(dest, src);

    printf("Destination: %s", dest);

    return 0;
}
```

Output

```
E:\RCPIT Docs\Academic Documents\A
Destination: RCPIT
--------------------------------
```

Example: strcat()

```c
#include<stdio.h>
#include<string.h>
int main()
{
    char s1[10] = "Hello ";
    char s2[10] = "World";

    strcat(s1, s2);
    printf("s1: %s", s1);

    return 0;
}
```

Output

```
E:\RCPIT Docs\Academic Documents\AY_2024-25_ODD Semester\SP
s1: Hello World
--------------------------------
```

Example: strcmp()

```c
#include<stdio.h>
#include<string.h>
int main()
{
    char s1[10] = "abc";
    char s2[10] = "abcd";

    int n = strcmp(s1, s2);

    if(n < 0)
    printf("s1 is smaller than s2");
    else if (n > 0)
    printf("s1 is larger than s2");
    else
    printf("s1 is equals to s2");

    return 0;
}
```

Output

```
E:\RCPIT Docs\Academic Documents\AY_2024-25_ODD Semes
s1 is smaller than s2
--------------------------------
```

Example: strrev()

```c
#include<stdio.h>
#include<string.h>
int main()
{
    char s1[10] = "Welcome";

    strrev(s1);
    printf("s1: %s", s1);

    return 0;
}
```

Output

```
E:\RCPIT Docs\Academic Documents\AY_2024-25_ODD Semest
s1: emocleW
--------------------------------
```

User defined function for string handling

```c
/* Program to calculate length of string using user defined function
(without using strlen()) */

#include<stdio.h>
int length(char str[])
{
    char ch;
    int i = 0, count = 0;

    while((ch = str[i])!='\0')
    {
        count++;
        i++;
    }
    return count;
}
int main()
{
    char str[]="Welcome";

    int len = length(str);
    printf("Length: %d", len);

    return 0;
}
```

Output:

Length: 7

User defined function for string handling

```c
/*Program to reverse string using user defined function
(without using strrev()) */

#include<stdio.h>
#include<string.h>
void reverseString(char str[])
{
    int start = 0;
    int end = strlen(str)-1;

    while(start < end)
    {
        char temp = str[start];
        str[start] = str[end];
        str[end] = temp;

        start++;
        end--;
    }
}
int main()
{
    char str[]="Welcome";
    reverseString(str);
    printf("Reversed string: %s\n", str);
    return 0;
}
```

Output:

Reversed string: emocleW

Structure

Structure

- A structure is a user-defined data type that allows you to combine data items of different types under one name.
- It is used to group related data together, which makes it easier to handle complex data.
- Structures are defined using the keyword struct.

Defining A Structure

Syntax:

```
struct tag_name
{
    data_type    member1;
    data_type    member2;
    data_type    member3;
    - - - - - - -
    - - - - - - -
};
```

Example:

```
struct Student
{
    int     rollNo;
    char    name[20];
    float   percentage;
};
```

Here,

struct is the keyword to define a structure.

tag_name is the name of the structure.

member1, member2, etc., are the structure members.

Declaring Structure variables

There are two ways to declare structure variables:

1. Declaration after Structure Definition:

Example:

```
// Defining structure
struct Student
{
    int    rollNo;
    char    name[20];
    float    percentage;
 };

// Declaring structure variables
struct Student s1, s2;
```

Declaring Structure variables

2. Declaration While Defining the Structure:

Example:

```
// Defining structure
struct Student
{
    int    rollNo;
    char    name[20];
    float    percentage;
}s1, s2;        // Declaring structure variables
```

Accessing Structure Members

We can access structure members using the dot operator (.) if you have a variable of the structure type.

```
// Declaring structure variables
struct Student s1, s2;

// Accessing structure Members
```

Example:

```
s1.rollNo = 10;
strcpy(s1.name, "ABC");
s1.percentage = 78.5f;

s2.rollNo = 20;
strcpy(s2.name, "XYZ");
s2.percentage = 82.7f;
```

Structure Example:

Program to Define structure, to declare its variable & to access structure members

```c
#include<stdio.h>
struct Student
{
    int rollNo;
    char name[20];
    float percentage;
};

int main()
{
    struct Student s1;

    printf("Enter rollNo, name and percentage of student: ");
    scanf("%d %s %f",&s1.rollNo, s1.name, &s1.percentage);

    printf("\nStudent Details: ");
    printf("\nRoll Number: %d", s1.rollNo);
    printf("\nName: %s", s1.name);
    printf("\nPercentage: %f", s1.percentage);

    return 0;
}
```

Structure Example:

Output:

```
Enter rollNo, name and percentage of student:
101 AAA 87.5

Student Details:
Roll Number: 101
Name: AAA
Percentage: 87.500000
--------------------------------
```

Array of Structure

- An array having structure as its base type is known as an array of structure.
- To create an array of structure, first structure is declared and then array of structure is declared just like an ordinary array.

Example:

```
// Defining structure
struct Student
{
    int     rollNo;
    char    name[20];
    float   percentage;

// Declaring array of structure
struct Student s1[5];
```

Program: Array of Structure

```c
#include<stdio.h>

//defining structure
struct Student
{
    int rollNo;
    char name[20];
    float percentage;
};

int main()
{
    struct Student s[2];    //declaring array of structure

    for(int i=0;i<2;i++)
    {
        printf("Enter rollNo, name and percentage of student: ");
        scanf("%d %s %f",&s[i].rollNo, s[i].name, &s[i].percentage);
    }

    for(int i=0;i<2;i++)
    {
        printf("\nRoll Number: %d", s[i].rollNo);
        printf("\nName: %s", s[i].name);
        printf("\nPercentage: %f", s[i].percentage);
    }
    return 0;
}
```

Program: Array of Structure

```
E:\RCPIT Docs\Academic Documents\AY_2024-25_ODD Semester\SPC\SPC Practical Programs\StuctArray.exe
Enter rollNo, name and percentage of student:
101
AAA
78.54
Enter rollNo, name and percentage of student:
102
BBB
65.78


Roll Number: 101
Name: AAA
Percentage: 78.540001

Roll Number: 102
Name: BBB
Percentage: 65.779999
--------------------------------
```

Structure within structure

Structure within structure means nesting of structures.

Example:

```
struct Student
{
    int rollNo;
    char name[20];
    float percentage;

    struct
    {
        int dd, mm, yyyy;
    }dob;
}s1;
```

Members in inner structure can be accessed as:

s1.dob.dd

s1.dob.mm

s1.dob.yyy

Structure within structure contd..

We can also use tag name to define inner structure

Example:

```
struct DOB
{
    int dd, mm, yyyy;
};

struct Student
{
    int rollNo;
    char name[20];
    float percentage;
    struct DOB b_date;
}s1;
```

References

References

- "MASTERING C" by K.R.Venugopal and Sudeep R.Prasad , Tata McGraw-Hill Publications
- "Programming in ANSI C", by E. Balaguruswamy, Tata McGraw-Hill Education.
- "Let Us C", by Yashwant Kanetkar, BPB Publication.
- "Programming in C", by Pradeep Day and Manas Gosh, Oxford University Press.
- https://www.javatpoint.com/c-programming-language-tutorial
- https://www.tutorialspoint.com/cprogramming/index.htm
- https://www.geeksforgeeks.org/c-programming-language/
- https://www.youtube.com/playlist?list=PLdo5W4Nhv31a8UcMN9-35ghv8qyFWD9_S


Thank
you