PROGRAMMING IN C

NOTES

By

Prof. Prabhu Kichadi

9880437187

UNIT – III

CONTENTS

Derived data types in C: Arrays: One Dimensional arrays - Declaration, Initialization and Memory representation; Two Dimensional arrays - Declaration, Initialization and Memory representation. Strings: Declaring & Initializing string variables; String handling functions - strlen, strcmp, strcpy and strcat; Character handling functions - toascii, toupper, tolower, isalpha, isnumeric etc.

Derived Data Types:

Data types that are derived from fundamental data types are called derived data types, some of the properties of existing primitive data types are added.

Derived data types do not create new data types. Instead, they add some functionality to the existing data types.

Derived data types are derived from the primitive data types by adding some extra relationships with the various elements of the primary data types. The derived data type can be used to represent a single value or multiple values.

Examples:

1. Arrays
2. Pointers
3. Functions
4. Files.

Arrays:

"Array is derived data type; array is an indexed collection of homogeneous elements stored in contiguous memory locations."

- Arrays are derived data types in C.
- Using single variable, we can store 'n' number of elements(values).
- Array elements are accessed using index(indices).
- Array index always starts from 0 to n-1.
- All array elements must be homogeneous (Same Data type).

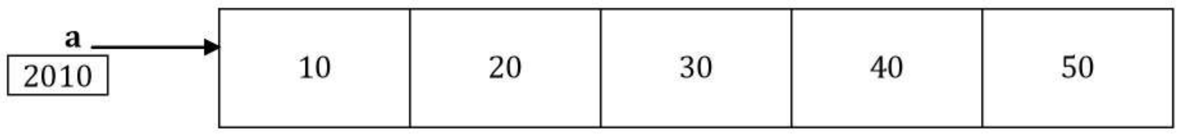
Array Declaration: [How arrays are declared?]

Syntax:

```
datatype  arrayname[size/bounds];
```

datatype: it must be valid data type related keyword which represents type of elements we store in an array.
arrayname: It must be a valid identifier; it indicates the name of the array variable.
Size: It must be integer number that specifies the number of elements that array can hold. It represents the capacity of the array.
[ ] - Subscript operator/Array Operator
Ex1:

int a[5];
"a is an array of 5 integers."



index->	**0**	**1**	**2**	**3**	**4**
element->	**a[0]**	**a[1]**	**a[2]**	**a[3]**	**a[4]**
Address->	**2010**	**2014**	**2018**	**2022**	**2026**
Address Access->	**&a[0]**	**&a[1]**	**&a[2]**	**&a[3]**	**&a[4]**

Array Initialization:

- Various methods we can see for array initialization.

1. Using Array Initializer List.: This is the way of initializing array elements at the time of array declaration.

Ex:

```
int a[5] = {1,2,3,4,5};
```

- We may ignore the array size also.

```
int a[ ] = {1,2,3,4,5};
```

it means array elements are initialized like this

```
a[0] = 1;
a[1] = 2;
a[2] = 3;
a[3] = 4;
a[4] = 5;
```

2. Element-wise initialization.

- After array declaration, we can access each element and initialize.

Ex:

```
int a[5];
a[0] = 10;
a[1] = 20;
a[2] = 30;
a[3] = 40;
a[4] = 50;
```

3. using for loop with printf( ) & scanf( ) .

- Index starts from 0 to n-1 hence we can iterate indices through loops.

Ex:

```
int a[10];
int i;
```

```
        for(i=0; i<10  i++)
        {
                printf("\nEnter Element : ");
                scanf("%d", &a[i]);
        }
```

Array Accessing:

- Array elements can be accessed using for loop and element wise.

Using arrayname, subscript operator we can access array elements for processing and printing.

```
#include<stdio.h>
int main()
{

        //Declare an array with specified size.
        int  a[5],  i;


        //Read Array Elements
        printf("\nEnter 5 Elements \n");
        for(i=0;  i<5;   i++)
        {
                printf("\nEnter Element : ");
                scanf("%d", &a[i]);
        }

        //Process array elements [for manipulation/access/Printing]
        printf("\nArray Elements\n");
        for(i=0;  i<5;  i++)
        {
                printf("%d\t", a[i]);
        }
        return 0;
}
```

Output:

```
Enter 5 Elements
Enter Element : 10
Enter Element : 20
Enter Element : 30
Enter Element : 40
Enter Element : 50

Array Elements
10      20      30      40      50
```

Classification of arrays:

Arrays are classified into 3 categories,
1. Single Dimensional Arrays or 1-Dimensional Arrays
2. Two-Dimensional Array or 2-Dimensional Arrays
3. Multi-Dimensional Arrays.

1. Single Dimensional Arrays:
- 1-Dimensional array having only array size, collection of homogeneous elements stored in contiguous memory locations.

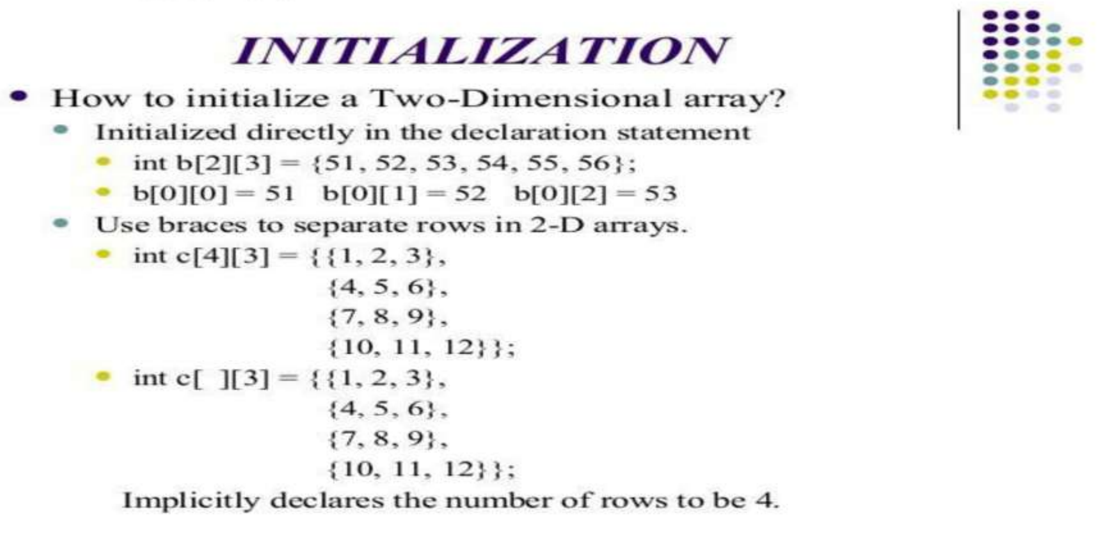
Syntax: datatype arrayname[arraysize];

Ex: int a[5];
int -> type of elements to be stored in an array.
a -> array name must be a valid identifier.
[ ] -> subscript operator.

float salary[10];

2. Two Dimensional Arrays:
- two-dimensional array is array of arrays.
- two-dimensional array is an array which contains data elements in matrix format.
- element are stored in rows & columns format.

Syntax: datatype arrayname[rowsize][colsize];
Ex:
int a[2][3];
a is a 2-D array which contains 2 rows and 3 columns.
Using below syntax, we can read and write (Access) array elements.



INITIALIZATION

- How to initialize a Two-Dimensional array?
  - Initialized directly in the declaration statement
    - int b[2][3] = {51, 52, 53, 54, 55, 56};
    - b[0][0] = 51 b[0][1] = 52 b[0][2] = 53
  - Use braces to separate rows in 2-D arrays.
    - int c[4][3] = {{1, 2, 3},
      {4, 5, 6},
      {7, 8, 9},
      {10, 11, 12}};
    - int c[ ][3] = {{1, 2, 3},
      {4, 5, 6},
      {7, 8, 9},
      {10, 11, 12}};

    Implicitly declares the number of rows to be 4.

Ex1: Program to read matrix and display matrix

```
#include<stdio.h>
int main()
{
  int row,col;

  printf("\nEnter Number of rows : ");
  scanf("%d", &row);

  printf("\nEnter Number of columns : ");
  scanf("%d", &col);

  int a[row][col],i,j;

  printf("\nEnter (%d X %d) Elements ....\n");
  for(i=0; i<row; i++)
  {
    for(j=0; j<col; j++)
    {
      printf("\nEnter (%d, %d) Element : ", i, j);
      scanf("%d", &a[i][j]);
    }
  }
  printf("\nMatrix Elements ....\n");
  for(i=0; i<row; i++)
  {
    for(j=0; j<col; j++)
    {
      printf("%d\t", a[i][j]);
    }
    printf("\n\n");
  }

  return 0;
}
```

Output:

```
Enter Number of rows :  2
Enter Number of columns : 2
Enter 2 X 2 Elements....
Enter (0, 0) Element : 1
Enter (0, 1) Element : 2
Enter (1, 0) Element : 3
Enter (1, 1) Element : 4
Matrix Elements....
1       2
3       4
```

Ex1: Write a C Program to add two matrices.(IMP & Lab Experiment)

```
#include<stdio.h>
int main()
{
  int row,col;
  int a[10][10],b[10][10],res[10][10],i,j;


  printf("\nEnter Number of rows : ");
  scanf("%d", &row);

  printf("\nEnter Number of columns : ");
  scanf("%d", &col);


  printf("\nEnter First Matrix Elements ....\n");
  for(i=0; i<row; i++)
  {
    for(j=0; j<col; j++)
    {
      printf("\nEnter (%d, %d) Element : ", i, j);
      scanf("%d", &a[i][j]);
    }
  }

  printf("\nEnter Second Matrix Elements ....\n");
  for(i=0; i<row; i++)
  {
    for(j=0; j<col; j++)
    {
      printf("\nEnter (%d, %d) Element : ", i, j);
      scanf("%d", &b[i][j]);
    }
  }

  printf("\nResultant Matrix Elements  \n");
  for(i=0; i<row; i++)
  {
    for(j=0; j<col; j++)
    {
      res[i][j] = a[i][j] + b[i][j];
      printf("%d\t", res[i][j]);
    }
    printf("\n\n");
  }

  return 0;
}
```

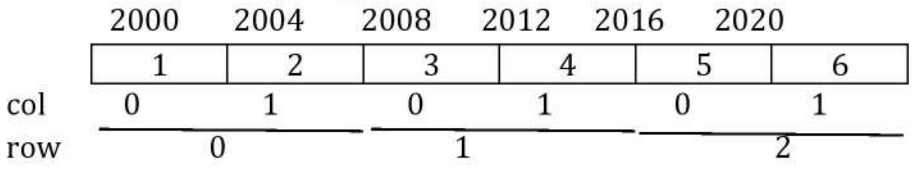
Output:

```
Enter Number of rows : 2

Enter Number of columns : 2

Enter First Matrix Elements ....

Enter (0, 0) Element : 1

Enter (0, 1) Element : 2

Enter (1, 0) Element : 3

Enter (1, 1) Element : 4

Enter Second Matrix Elements ....

Enter (0, 0) Element : 1

Enter (0, 1) Element : 2

Enter (1, 0) Element : 3

Enter (1, 1) Element : 4

Resultant Matrix Elements
2      4

6      8
```

2-D Array Memory Representation:

int a[3][2];

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$



Practice Programs.

1. Write a program to read array elements and display.
2. Write a C program to find sum of all array elements.
3. write a C program to find sum of all even numbers from a given array.
4. write a C program to find sum of all odd numbers from a given array.
5. write a C program to find sum of all even & odd numbers from a given array.
6. write a C program to find maximum of all even numbers from a given array.
7. write a C program to find maximum of all odd numbers from a given array.
8. write a C program to find minimum of all even numbers from a given array.
9. write a C program to find minimum of all odd numbers from a given array.
10. Write a function that accepts array of integers to find maximum and minimum element in an array.

Strings:

String Constant/Literal: Group of characters enclosed in a double quote, ends with a null ('\0') character. String is a Character array in C.
\0 – slash zero.
- Anything that we enclose in a double quote is a string in C.

Ex:
"Welcome to c programming",
"Ravi",
"KLE BCA Athani"
"12345"
"&*&*&*tyty134343"

- In C space is a character.

String Declaration:

- In C, string is declared as using char array.

Syntax:

```
char stringname[length];
```

char – char is a keyword we must use for string declaration, which represents we are declaring a variable for characters.

stringname – stringname is the name of the string variable, which is nothing character array name.

length – length specifies number of characters an array can contains.

Ex1:

char str[25];

- here str is a string variable which can hold maximum 25 characters in the str array.

char str2[50];

-here str2 is a char array which can hold up to 25 characters.

String Initialization OR String reading OR string Input:

- String initialization is nothing but reading characters into a string variable.
- Assigning character values to a string variable.

Ways to initialize string variable in C.

1. Using Index wise:

```
//string declaration
char name[25];

//string initialization
name[0] = ‘h’;
name[1] = ‘e’;
name[2] = ‘l’;
name[3] = ‘l’;
name[4] = ‘o’;
name[5] = ‘w’;
name[6] = ‘\0’;
```

2. Declare and Initialize at a time.

```
char name[ 20 ] = “hello”;
char name[ ] = “hello”;
```

3. Using Initializer List:

```
char name[25] = {‘h’,’e’,’l’,’l’,’o’,’\0’};
char name[ ] = {‘h’,’e’,’l’,’l’,’o’,’\0’};
```

4. using scanf( ) & %s - Not Recommnded

```
char name[20];
printf(“\nEnter Your Name : ”);
fflush(stdin);
scanf(“%s”, name);

printf(“\nYour Name : %s”,name);
```

Output1:

```
Enter Your Name : Ravi
Your Name : Ravi
```

Output2:

```
Enter Your Name : Ravi Shankar
Your Name : Ravi
```

Note: %s in scanf( ) works till whitespace, it will read the data until whitespace character only. Hence it is not recommended.

5. using gets() & puts( ) function from string.h

gets( ) : To read string data from standard input device.[Input Function]
puts() : To print string data to standard output device.[Output Function]

gets ( ) :
syntax:

```
char *gets(char *str)
```

- gets (chararray) function is a standard library string input function, it is used to read a string from keyboard at runtime until Enter key.
- We need to pass a char array name to this function; it will read the string and stores it in given char array.

puts ( ) :
syntax:

```
int puts(char[]);
```

- puts(chararray) function is a standard library string unformatted output function, it is used to write/print a string data to standard output device (Console).
- We need to pass a char array name to this function; it will print the string on output screen.

Ex:

```
char name[20];
printf("\nEnter Your Name : ");
fflush(stdin);
gets(name);

printf("\nYour Name : ");
puts(name);
```

Output1:

```
Enter Your Name : Ravi
Your Name : Ravi
```

Output2:

```
Enter Your Name : Ravi Shankar
Your Name : Ravi Shankar
```

String Handling:

- It may need to often manipulate strings according to the need of a problem. Most, if not all, of the time string manipulation can be done manually but, this makes programming complex and large.
- To solve this, C supports a large number of string handling functions in the standard library "string.h".

String manipulations: Operations related to strings like comparison of strings, finding the length of the string, concatenating(join) of strings, reversing strings, converting to uppercase, etc.

String library functions: string.h header file functions

Function Name	Syntax	Functionality
strlen( )	int strlen(char s1[]);	Returns total number of characters in a given string. (Length of the string)
strcpy()	int strcpy(char s1[ ], char s2[ ]);	copies a string to another
strcat( )	char * strcat(char s1[ ], char s2[ ]);	concatenates(joins) two strings
strcmp( )	int strcmp(char s1[ ], char s2[ ]);	compares two strings
strlwr( )	char * strlwr(char s1[ ]);	converts string to lowercase
strupr( )	char * strupr(char s1[ ]);	converts string to uppercase
strrev( )	char * strrev(char s1[]);	reverses a given string

Programs on strings:

1.Write a c Program to find the length of a string (Without using library function)

```
#include<stdio.h>
#include<string.h>
int main()
{
  char s1[25];
  int n;

  printf("\nEnter a String : ");
  fflush(stdin);
  gets(s1);

  n = strlen(s1);

  printf("\nLength of String : %d",n);
  return 0;
}
```

Output:
```
Enter a String : hello kle
Length of String : 9
```

2.Write a c Program to find the length of a string (without using library function.)

```c
#include<stdio.h>
int main()
{
  char s1[25];
  int i, count;

  printf("\nEnter a String : ");
  fflush(stdin);
  gets(s1);

  i=0;
  count=0;

  while(s1[i] != '\0')
  {
    i++;
    count++;
  }
  printf("\nLength of String : %d",count);
  return 0;
}
```

Output:
```
Enter a String : hello kle
Length of String : 9
```

3. Write a C program to concatenate two strings. (using library function strcat)

```c
#include<stdio.h>
#include<string.h>
int main()
{
  char s1[50],s2[25];

  printf("\nEnter First String : ");
  fflush(stdin);
  gets(s1);

  printf("\nEnter Second String : ");
  fflush(stdin);
  gets(s2);

  strcat(s1,s2);

  printf("\nConcatenated String : %s",s1);
  return 0;
}
```

Output:

```
Enter First String : hello
Enter Second String : bca
Concatenated String : hello kle
```

4. Write a C program to concatenate two strings without using library function.

```
#include<stdio.h>
int main()
{
  char s1[50],s2[25];
  int i,j;

  printf("\nEnter First String : ");
  fflush(stdin);
  gets(s1);

  printf("\nEnter Second String : ");
  fflush(stdin);
  gets(s2);

  i=0;
  while(s1[i] != '\0')
  {
    i++;
  }
  j=0;
  while(s2[j] != '\0')
  {
    s1[i] = s2[j];
    i++;
    j++;
  }
  s1[i] = '\0';

  printf("\nConcatenated String : %s",s1);
```

```
    return 0;
}
```

Output:

```
Enter First String : hello
Enter Second String : bca
Concatenated String : hello kle
```

5. Write a C program to compare two strings using library function.

```
#include<stdio.h>
#include<string.h>
int main()
{
    char s1[50],s2[25];
    int n;

    printf("\nEnter First String : ");
    fflush(stdin);
    gets(s1);
    printf("\nEnter Second String : ");
    fflush(stdin);
    gets(s2);

    n = strcmp(s1, s2);

    if(n == 0)
    {
        printf("\nStrings are equal");
    }
    else
    {
        printf("\nStrings are not equal");
    }
    return 0;
}
```

Output:

```
Enter First String : hello
Enter Second String : hello
Strings are equal
```

Output:

```
Enter First String : hello
Enter Second String : Hello
Strings are not equal
```

6. Write a C program to compare two strings without using library function.

```
#include<stdio.h>
#include<string.h>
int main()
{
  char s1[50], s2[25];
  int n1,n2, i,j,flag=0;

  printf("\nEnter First String : ");
  fflush(stdin);
  gets(s1);
  printf("\nEnter Second String : ");
  fflush(stdin);
  gets(s2);
  n1 = strlen(s1);
  n2 = strlen(s2);

  if(n1 == n2)
  {
    i=0;
    j=0;
    while(s1[i] != '\0')
    {
      if(s1[i] != s2[j])
      {
        flag = 1;
        break;
      }
      else
      {
        flag = 0;
      }
      i++;
      j++;
    }
    if(flag == 1)
    {
      printf("\nStrings are not equal");
    }
    else
    {
      printf("\nStrings are equal");
    }
  }
  else
  {
    printf("\nStrings are not equal");
  }
  return 0;
}
```

Output:

```
Enter First String : hello
Enter Second String : hello
Strings are equal
```

Output:

```
Enter First String : hello
Enter Second String : Hello
Strings are not equal
```

7. Write a C program to copy one string into another, using library function.

```
#include<stdio.h>
#include<string.h>
int main()
{
  char s1[50],s2[50];
  int i;
  printf("\nEnter a String : ");
  fflush(stdin);
  gets(s1);
  strcpy(s2,s1);
  printf("\nCopied String : %s",s2);
  return 0;
}
```

Output:

```
Enter a String : hello
Copied String : hello
```

8. Write a C program to copy one string into another, without using library function.

```
#include<stdio.h>
int main()
{
  char s1[50], s2[50];
  int i;
  printf("\nEnter a String : ");
  fflush(stdin);
  gets(s1);
  i=0;
  while(s1[i] != '\0')
  {
        s2[i] = s1[i];
        i++;
    }
    s2[i] = '\0';
    printf("\nCopied String : %s",s2);
    return 0;
}
```

Output:

```
Enter a String : hello
Copied String : hello
```

9. Write a C program reverse a given string, using library function.

```
#include<stdio.h>
#include<string.h>
int main()
{
    char s1[50],  *s2;
    int i;
    printf("\nEnter a String : ");
    fflush(stdin);
    gets(s1);

    s2 = strrev(s1);

    printf("\nReverse String : %s",s2);
    return 0;
}
```

Output:

```
Enter a String : madam
Reverse String : madam
```

Output:

```
Enter a String : hello
Reverse String : olleh
```

Character Handling Functions:

Character functions need **ctype.h** header file to be included in the pogrom. Different character functions provided by C Language are:

Character library functions: ctype.h header file functions

Function Name	Syntax	Functionality
toscii( )	**int toascii(int ch);**	The *toascii*() function shall convert its argument into a 7-bit ASCII character.
toupper( )	**int toupper(int ch);**	The toupper() function is used to convert lowercase alphabet to uppercase.
tolower( )	**int tolower(int ch);**	The tolower() function is used to convert lowercase alphabet to uppercase.
isalpha( )	**int isalpha(char ch)**	isalpha(c) is a function in C which can be used to check if the passed character is an alphabet or not. It returns a non-zero value if it's an alphabet else it returns 0
isdigit()	**int isdigit(char ch)**	The isdigit(c) is a function in C which can be used to check if the passed character is a digit or not. It returns a non-zero value if it's a digit else it returns 0.
isalnum()	**int isalnum(char ch)**	The function isalnum() is used to check that the character is alphanumeric or not. It returns non-zero value, if the character is alphanumeric means letter or number otherwise, returns zero
isspace()	**int isspace(char ch)**	If an argument (character) passed to the isspace() function is a white-space character, it returns non-zero integer. If not, it returns 0.

isupper()	**int isupper(char ch)**	If an argument (character) passed to the isupper() function is a Upper Case character, it returns non-zero integer. If not, it returns 0.
islower()	**int islower(char ch)**	If an argument (character) passed to the islower() function is a Lower Case character, it returns non-zero integer. If not, it returns 0.

Example Programs: