|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Programming in C** | **Semester:** | **II** |
| **Date of Performance:** | **07 / 02 / 2025** | **DIV/ Batch No:** | **C4-1** |
| **Student Name:** | **Dhruv Pankhania** | **Roll No:** | **16010124216** |

**Experiment No: 4**

**Title: Write a program in C to demonstrate use of arrays**

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| 1.Write a program in C to read n number of values in an array and display them in reverse order.  2. Program to sort the 1D array in the ascending or descending order  3. Write a program in C to count the total number of duplicate elements in an array. |

|  |
| --- |
| **COs to be achieved:** |
| **CO3: Apply the concept of arrays and string** |

|  |
| --- |
| **Theory:** |
| An array in C is a fixed-size collection of similar data items stored in contiguous memory locations. It can be used to store the collection of primitive data types such as int, char, float, etc., and also derived and user-defined data types such as pointers, structures, etc.  arrays in c  **C Array Declaration**  In C, we have to declare the array like any other variable before using it. We can declare an array by specifying its name, the type of its elements, and the size of its dimensions. When we declare an array in C, the compiler allocates the memory block of the specified size to the array name.  **Syntax of Array Declaration**  **data\_type array\_name [size];**  Lightbox  **// C Program to illustrate the array declaration**  #include <stdio.h>  int main()  {  // declaring array of integers  int arr\_int[5];  // declaring array of characters  char arr\_char[5];  return 0;  }  c array initialization Array Initialization with Declaration without Size data\_type array\_name[] = {1,2,3,4,5}; Array Initialization after Declaration (Using Loops) for (int i = 0; i < N; i++)  {  array\_name[i] = valuei;  }  Types of Array in C  There are two types of arrays based on the number of dimensions it has. They are as follows:   * One Dimensional Arrays (1D Array)   Syntax of 1D Array in C  array\_name [size];   * Two-Dimensional Array in C   array\_name[size1] [size2]; |

|  |
| --- |
| **Problem Statements:** |
| 1.Write a program in C to read n number of values in an array and display them in reverse order.  2. Program to sort the 1D array in the ascending or descending order  3. Write a program in C to count the total number of duplicate elements in an array. |

|  |
| --- |
| **Code :** |
| 1.  #include<stdio.h>  *int* main(){  *int* sizei;  printf("Enter the size of the array you want to create: ");        scanf("%d", &sizei);  *int* arr[sizei], revArr[sizei];    printf("Enter the Numbers for the array: ");    for(*int* i = 0; i<sizei; i++){    scanf("%d", &arr[i]);    }  printf("The entered array is: \n");    for(*int* j = 0; j<sizei; j++){      printf("%d \t", arr[j]);    }    for(*int* i =sizei-1; i>=0; i--){      revArr[i] = arr[sizei-i-1];    }    printf("\nThe Reversed array is: \n");    for(*int* j = 0; j<sizei; j++){      printf("%d \t", revArr[j]);    }    return 0;  }  2.  #include<stdio.h>  *int* main(){  *int* sizei;  printf("Enter the size of the array you want to create: ");    scanf("%d", &sizei);  *int* arr[sizei], ascendingArr[sizei], decendingArr[sizei];    printf("Enter the Numbers for the array: ");    for(*int* i = 0; i<sizei; i++){    scanf("%d", &arr[i]);      ascendingArr[i] = arr[i];      decendingArr[i] = arr[i];    }  printf("The entered array is: \n");    for(*int* j = 0; j<sizei; j++){      printf("%d \t", arr[j]);    }  for(*int* i = 0; i<sizei; i++){  for(*int* j = i; j<sizei; j++){  *int* temp = 0;  if(ascendingArr[i]>ascendingArr[j]){    temp = ascendingArr[i];    ascendingArr[i] = ascendingArr[j];    ascendingArr[j] = temp;  }  temp = 0;  if(decendingArr[i]<decendingArr[j]){    temp = decendingArr[i];    decendingArr[i] = decendingArr[j];    decendingArr[j] = temp;  }  }  }  printf("\nThe Ascending array is: \n");    for(*int* j = 0; j<sizei; j++){      printf("%d \t", ascendingArr[j]);    }  printf("\nThe Descending array is: \n");    for(*int* j = 0; j<sizei; j++){      printf("%d \t", decendingArr[j]);    }  return 0;  }  3.  #include<stdio.h>  *int* main(){  *int* size, counter =0;  printf("Enter the size of the array you want to create: ");        scanf("%d", &size);  *int* arr[size];    printf("Enter the Numbers for the array: ");    for(*int* i = 0; i<size; i++){    scanf("%d", &arr[i]);    }  for(*int* i = 0; i<size; i++){  for(*int* j = i; j<size; j++){  *int* temp = 0;  if(arr[i]>arr[j]){    temp = arr[i];    arr[i] = arr[j];    arr[j] = temp;  }  }  }    for(*int* i = 0; i<size; i++){    for(*int* j = i+1; j<size; j++){      if(arr[i]==arr[j]){        counter++;      };    }    }    printf("The number of duplicates in the given array is: %d", counter);    return 0;  }  3.  #include <stdio.h>  *int* main() {  *int* size, counter = 0;        printf("Enter the size of the array you want to create: ");      scanf("%d", &size);  *int* arr[size];        printf("Enter the numbers for the array: ");      for (*int* i = 0; i < size; i++) {          scanf("%d", &arr[i]);      }      for (*int* i = 0; i < size - 1; i++) {          for (*int* j = i + 1; j < size; j++) {              if (arr[i] > arr[j]) {  *int* temp = arr[i];                  arr[i] = arr[j];                  arr[j] = temp;              }          }      }   for (*int* i = 0; i < size - 1; i++) {          if (arr[i] == arr[i + 1]) {  *int* count = 1;              while (i < size - 1 && arr[i] == arr[i + 1]) {                  count++;                  i++;}              counter += count;          }      }      printf("The total count of duplicate elements is: %d\n", counter);      return 0;  } |

|  |
| --- |
| **Output:** |
| 1.    2.    3. |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:**   1. Write a program in C to find the maximum and minimum elements in an array.   #include<stdio.h>  int main(){  int size, counter =0;  printf("Enter the size of the array you want to create: ");  scanf("%d", &size);  int arr[size];  printf("Enter the Numbers for the array: ");  for(int i = 0; i<size; i++){  scanf("%d", &arr[i]);  }  for(int i = 0; i<size; i++){  for(int j = i; j<size; j++){  int temp = 0;  if(arr[i]>arr[j]){  temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }  }  }  printf("The element with maximum value is: %d \n", arr[size-1]);  printf("The element with minimum value is: %d \n", arr[0]);  return 0;  }     1. Write a program in C for adding two matrices of the same size.   #include<stdio.h>  *int* main(){  *int* size1i, size1j;  printf("Enter the size of the first 2D array you want to create: ");        scanf("%d %d", &size1i, &size1j);  *int* arr1[size1i][size1j];    printf("Enter the Numbers for the array: ");    for(*int* i = 0; i<size1i; i++){      for(*int* j = 0; j<size1j; j++){        scanf("%d", &arr1[i][j]);      }    }  *int* size2i, size2j;  printf("Enter the size of the second 2D array you want to create: ");        scanf("%d %d", &size2i, &size2j);  *int* arr2[size2i][size2j];    printf("Enter the Numbers for the array: ");    for(*int* i = 0; i<size2i; i++){      for(*int* j = 0; j<size2j; j++){        scanf("%d", &arr2[i][j]);      }    }  *int* arrSum[2][2];    for(*int* i = 0; i<2; i++){      for(*int* j = 0; j<2; j++){        arrSum[i][j] = arr1[i][j] + arr2[i][j];          }}    printf("The First array is: \n");  for(*int* i = 0; i<2; i++){      for(*int* j = 0; j<2; j++){        printf("%d \t", arr1[i][j]);      }      printf("\n");    }    printf("The Second array is: \n");  for(*int* i = 0; i<2; i++){      for(*int* j = 0; j<2; j++){        printf("%d \t", arr2[i][j]);      }      printf("\n");    }    printf("The sum of the two arrays is: \n");  for(*int* i = 0; i<2; i++){      for(*int* j = 0; j<2; j++){        printf("%d \t", arrSum[i][j]);      }      printf("\n");    }    return 0;  } |

|  |
| --- |
| **Conclusion:** |
| We learned about arrays, which store multiple values of the same data type in a contiguous memory location. Arrays can be one-dimensional (1D) or multi-dimensional (2D, etc.), useful for organizing data efficiently. We performed various operations like insertion, deletion, searching, sorting, and traversal. We also explored sorting algorithms and methods to identify duplicates. |

|  |
| --- |
| **Signature of faculty in-charge with Date:** |