**Lab Assignment 1(Array) by Gavisht Singh**

****

1) Develop a Menu driven program to demonstrate the following operations of

Arrays——MENU——-

1.CREATE

2.DISPLAY

3.INSERT

4.DELETE

5.LINEAR SEARCH

6.EXIT

**Solution: -**

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100;

void displayMenu() {

cout << "------ MENU ------ "<<endl;

   cout << "1. CREATE"<<endl;

   cout << "2. DISPLAY""<<endl;

   cout << "3. INSERT""<<endl;

   cout << "4. DELETE""<<endl;

   cout << "5. LINEAR SEARCH""<<endl;

   cout << "6. EXIT""<<endl;

   cout << "Enter your choice: ";

}

int main() {

    int arr[MAX\_SIZE];

    int currentSize = 0;

    int choice, element, position, i;

    do {

        displayMenu();

cin >> choice;

        switch (choice) {

            case 1:

               cout << "Enter the number of elements to create " ;

               cin >> currentSize;

                if (currentSize > MAX\_SIZE || currentSize < 0) {

                   cout << "Invalid size. Please enter a size between 0 and " << MAX\_SIZE <<endl;

                    currentSize = 0;

                } else {

                    for (i = 0; i < currentSize; i++) {

                       cin >> arr[i];

                    }

                   cout << "Array created successfully"<<endl;

                }

                break;

            case 2:

                if (currentSize == 0) {

                   cout << "Array is empty. Nothing to display "<<endl;

                } else {

                   cout << "Array elements are: ";

                    for (i = 0; i < currentSize; i++) {

                       cout << arr[i] << " ";

                    }

                   cout <<endl;

                }

                break;

            case 3:

                if (currentSize >= MAX\_SIZE) {

                   cout << "Array is full. Cannot insert. "<<endl;

                } else {

                   cout << "Enter the element to insert: ";

                   cin >> element;

                   cout << "Enter the position to insert (1 to " << currentSize + 1 << "): ";

                   cin >> position;

                    if (position < 1 || position > currentSize + 1) {

                       cout << "Invalid position!"<<endl;

                    } else {

                        for (i = currentSize; i >= position; i--) {

                            arr[i] = arr[i - 1];

                        }

                        arr[position - 1] = element;

                        currentSize++;

                        std::cout << "Element inserted successfully."<<endl;

                    }

                }

                break;

            case 4:

                if (currentSize == 0) {

                    cout << "Array is empty. Cannot delete."<<endl;

                } else {

                 cout << "Enter the position of the element to delete (1 to " << currentSize << "): ";

                   cin >> position;

                    if (position < 1 || position > currentSize) {

cout << "Invalid position! "<<endl;

                    } else {

                        for (i = position - 1; i < currentSize - 1; i++) {

                            arr[i] = arr[i + 1];

                        }

                        currentSize--;

                       cout << "Element deleted successfully."<<endl;

                    }

                }

                break;

            case 5:

                if (currentSize == 0) {

                   cout << "Array is empty. Cannot search. "<<endl;

                } else {

                   cout << "Enter the element to search: ";

                   cin >> element;

                    for (i = 0; i < currentSize; i++) {

                        if (arr[i] == element) {

                           cout << "Element " << element << " found at position " << i + 1 << endl;

                            break;

                        }

else{

cout<<”Element not found in the array.”<<endl;

                    }

                }

                break;

            case 6:

               cout << "Exiting the program "<<endl;

                break;

            default:

               cout << "Invalid choice! Please enter a number between 1 and 6. "<<endl;

        }

    } while (choice != 6);

    return 0;

}

2) Design the logic to remove the duplicate elements from an Array and after the

deletion the array should contain the unique elements.

**Solution: -**

#include <iostream>

using namespace std;

int main() {

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

    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Original array: ";

    for (int i = 0; i < n; i++) {

        cout << arr[i] << " ";

    }

    cout << endl;

    if (n > 1) {

        int newSize = 0;

bool isDuplicate=false;

        for (int i = 0; i < n; i++) {

            for (int j = 0; j < newSize; j++) {

                if (arr[i] == arr[j]) {

                    isDuplicate = true;

                    break;

                }

            }

            if (!isDuplicate) {

                arr[newSize] = arr[i];

                newSize++;

            }

        }

        n = newSize;

    }

    cout << "Array after removing duplicates: ";

    for (int i = 0; i < n; i++) {

        cout << arr[i] << " ";

    }

    cout << endl;

    return 0;

}

3)Predict the Output of the following program

int main()

{

int i;

int arr[5] = {1};

for (i = 0; i < 5; i++)

printf("%d",arr[i]);

return 0;

}  
**Solution: -**

10000

4) Implement the logic to

a. Reverse the elements of an array

b. Find the matrix multiplication

c. Find the Transpose of a Matrix

**Solution**: -

a) #include <iostream>

using namespace std;

void printArray(int arr[], int size) {

    for (int i = 0; i < size; ++i) {

        cout << arr[i] << " ";

    }

    cout << endl;

}

int main() {

    int arr[] = {1, 2, 3, 4, 5, 6, 7};

    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Original array: ";

    printArray(arr, n);

    int start = 0;

    int end = n - 1;

    while (start < end) {

        swap(arr[start],arr[end]);

        start++;

        end--;

    }

    cout << "Reversed array: ";

    printArray(arr, n);

    return 0;

}

b) #include <iostream>

using namespace std;

int main() {

    int mat1[2][3] = {

        {1, 2, 3},

        {4, 5, 6}

    };

    int mat2[3][2] = {

        {7, 8},

        {9, 10},

        {11, 12}

    };

    int res[2][2];

    for (int i = 0; i < 2; ++i) {

        for (int j = 0; j < 2; ++j) {

            res[i][j] = 0;

        }

    }

    for (int i = 0; i < 2; i++) {

        for (int j = 0; j < 2; j++) {

            for (int k = 0; k < 3; k++) {

                res[i][j] = res[i][j] + mat1[i][k] \* mat2[k][j];

            }

        }

    }

    cout << endl << "Result of multiplication:" << endl;

    for (int i = 0; i < 2; i++) {

        for (int j = 0; j < 2; j++) {

            cout << res[i][j] << "\t";

        }

        cout << endl;

    }

    return 0;

}

c)

#include <iostream>

using namespace std;

int main() {

    int mat[3][4] = {

        {1, 2, 3, 4},

        {5, 6, 7, 8},

        {9, 10, 11, 12}

    };

    int transpose[4][3];

    for (int i = 0; i < 3; i++) {

        for (int j = 0; j < 4; j++) {

            transpose[j][i] = mat[i][j];

        }

    }

      for (int i = 0; i < 4; i++) {

        for (int j = 0; j < 3; j++) {

            cout << transpose[i][j] << "\t";

        }

        cout << endl;

    }

    return 0;

}

5) Write a program to find sum of every row and every column in a two-dimensional

array.

**Solution**: -

#include <iostream>

using namespace std;

int main() {

    int matrix[3][4] = {

        {1, 2, 3, 4},

        {5, 6, 7, 8},

        {9, 10, 11, 12}

    };

    int rowSum, colSum;

    for (int i = 0; i < 3; i++) {

        rowSum = 0;

        for (int j = 0; j < 4; j++) {

            rowSum = rowSum + matrix[i][j];

        }

        cout << "Sum of Row " << i + 1 << ": " << rowSum << endl;

    }

    for (int j = 0; j < 4; j++) {

        colSum = 0;

        for (int i = 0; i < 3; i++) {

            colSum = colSum + matrix[i][j];

        }

        cout << "Sum of Column " << j + 1 << ": " << colSum << endl;

    }

    return 0;

}