**DR. D. Y. PATIL SCHOOL OF SCIENCE & TECHNOLOGY** 

**DR. D. Y. PATIL VIDYAPEETH, PUNE**

**(Deemed to be University)**

**(Accredited (3rd cycle) by NAAC with a CGPA of 3.64 on four-point scale at ‘A++’ Grade) (Declared as Category - I University by UGC Under Graded Autonomy Regulations, 2018) (An ISO 9001: 2015 and 14001:2015 Certified University and Green Education Campus)**

**Date: 19-1-24**

**Assignment No: 2**

**Problem Statement: . Write a program to read an integer 1D array containing 8 bits (0s, 1s) of an unsigned binary integer. The program should print the decimal value for the integer.**

**Algorithm:**

1. Start
2. Declare an array binaryArray of size 8 to store the 8-bit binary number.
3. Declare a variable decimalValue and initialize it to 0.
4. Read the 8-bit binary number from the user and store it in the binaryArray.
5. Loop through each bit of the binary number from the least significant bit (LSB) to the most significant bit (MSB).
6. For each bit, multiply its value by 2 raised to the power of its position, and add the result to decimalValue.
7. Print the decimalValue.
8. End

**Source Code:**

#include <iostream>

int binaryToDecimal(int binaryArray[], int size) {

int decimalValue = 0;

for (int i = size - 1; i >= 0; i--) {

decimalValue += binaryArray[i] \* (1 << (size - 1 - i));

}

return decimalValue;

}

int main() {

const int size = 8;

int binaryArray[size];

std::cout << "Enter an 8-bit binary number (separate each bit with a space): ";

for (int i = 0; i < size; i++) {

std::cin >> binaryArray[i];

}

for (int i = 0; i < size; i++) {

if (binaryArray[i] != 0 && binaryArray[i] != 1) {

std::cerr << "Invalid input. Please enter only 0s and 1s." << std::endl;

return 1; // Return error code

}

}

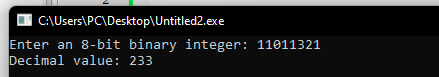
int decimalValue = binaryToDecimal(binaryArray, size);

std::cout << "Decimal value: " << decimalValue << std::endl;

return 0;

}

**Sample Output:**

**

**Problem Statement:**

**Currency notes are available in the following seven denominations: Rs. 1, Rs. 2, Rs. 5, Rs. 10, Rs. 20,Rs. 50, and Rs. 100. You will first read an integer array (named purse [7]) of seven elements, where the array elements represents the number of notes of each denomination (in increasing order of denomination) available with you. You are also given an amount of money to be paid using these notes. The amount will be less than Rs. 1000. You have to pay the amount using minimum number of notes. Print the number of notes of each denomination which you will use to pay the amount. If the amount cannot be paid using the available notes, print “Failed to Pay”.**

**Algorithm:**

1. Initialize an array denominations with the available denominations in increasing order: [1, 2, 5, 10, 20, 50, 100].
2. Read the integer array purse containing the number of notes for each denomination.
3. Read the amount to be paid (amount).
4. Initialize a variable remainingAmount to the total amount initially.
5. Initialize an array usedNotes to store the count of each denomination used for payment.
6. Iterate through the denominations in reverse order (starting from Rs. 100) and do the following:
7. Calculate the number of notes of the current denomination needed (notesNeeded) to pay the remaining amount.
8. Update usedNotes array with the count of notes used for the current denomination.
9. Update remainingAmount by subtracting the value of notes used.
10. If the remainingAmount becomes zero, print the contents of the usedNotes array. Otherwise, print "Failed to Pay."

**Source Code:**

#include <iostream>

int main() {

const int numDenominations = 7;

int purse[numDenominations];

std::cout << "Enter the number of notes for each denomination (Rs. 1 to Rs. 100):" << std::endl;

for (int i = 0; i < numDenominations; i++) {

std::cout << "Number of Rs. " << (i + 1) \* 10 << " notes: ";

std::cin >> purse[i];

}

int amount;

std::cout << "Enter the amount to be paid (less than Rs. 1000): ";

std::cin >> amount;

if (amount < 1 || amount >= 1000) {

std::cerr << "Invalid amount. Please enter an amount between Rs. 1 and Rs. 999." << std::endl;

return 1; // Return error code

}

int notesUsed[numDenominations] = {0};

for (int i = numDenominations - 1; i >= 0; i--) {

notesUsed[i] = std::min(amount / ((i + 1) \* 10), purse[i]);

amount -= notesUsed[i] \* ((i + 1) \* 10);

}

if (amount == 0) {

std::cout << "Number of notes used:" << std::endl;

for (int i = 0; i < numDenominations; i++) {

if (notesUsed[i] > 0) {

std::cout << "Rs. " << (i + 1) \* 10 << ": " << notesUsed[i] << " notes" << std::endl;

}

}

} else {

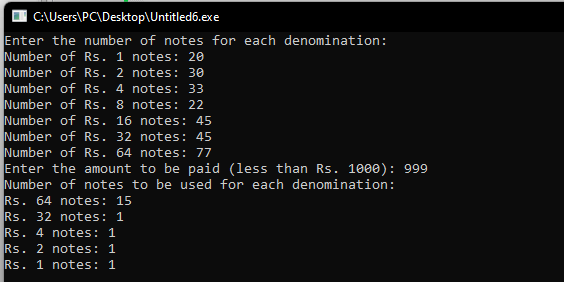
std::cout << "Failed to Pay" << std::endl;

}

return 0;

}

**Sample Output:**

****

**Problem Statement:**

**Write C functions to perform the following operations with two-dimensional arrays.**

**i) Reading any two dimensional array elements.**

**ii) Finding maximum and standard deviation of array elements.**

**iii) Printing the transpose of a matrix**

**Algorithm:**

Read Array Elements:

1. Initialize a two-dimensional array (arr).
2. Input the number of rows and columns (rows and cols) for the array.
3. Check if the input for rows and columns is valid.
4. Use nested loops to read elements into the array (arr).

Find Max and Standard Deviation:

* + 1. Initialize variables for maximum element (maxElement), sum (sum), sum of squared elements
    2. sumSquared), and element count (count).
    3. Use nested loops to iterate through the array:
    4. Update maxElement if the current element is greater.
    5. Accumulate elements in sum.
    6. Accumulate squared elements in sumSquared.
    7. Increment count.
    8. Calculate the mean (mean) as sum / count.
    9. Calculate the standard deviation (stdDev) using the formula sqrt((sumSquared / count) - pow(mean, 2)).

Print Transpose of Matrix:

* + 1. Use nested loops to iterate through the original matrix (arr).
    2. Print the transpose of the matrix by swapping row and column indices.

Main Function:

* + 1. Declare variables for rows, columns, array (arr), maximum element (maxElement), and standard
    2. deviation (stdDev).
    3. Call the readArray function to input array elements.
    4. Call the findMaxAndStdDev function to find the maximum element and standard deviation.
    5. Call the printTranspose function to print the transpose of the matrix.
    6. Display the maximum element and standard deviation.

**Source Code:**

#include <iostream>

#include <cmath>

const int MAX\_ROWS = 100;

const int MAX\_COLS = 100;

void readArray(int arr[MAX\_ROWS][MAX\_COLS], int rows, int cols) {

std::cout << "Enter elements of the array:" << std::endl;

for (int i = 0; i < rows; i++) {

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

std::cout << "Element at position (" << i + 1 << ", " << j + 1 << "): ";

std::cin >> arr[i][j];

}

}

}

void findMaxAndStdDev(int arr[MAX\_ROWS][MAX\_COLS], int rows, int cols, int& maxElement, double& stdDev) {

maxElement = arr[0][0];

double sum = 0.0, sumSquared = 0.0;

int count = 0;

for (int i = 0; i < rows; i++) {

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

if (arr[i][j] > maxElement) {

maxElement = arr[i][j];

}

sum += arr[i][j];

sumSquared += pow(arr[i][j], 2);

count++;

}

}

double mean = sum / count;

stdDev = sqrt((sumSquared / count) - pow(mean, 2));

}

void printTranspose(int arr[MAX\_ROWS][MAX\_COLS], int rows, int cols) {

std::cout << "Transpose of the matrix:" << std::endl;

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

for (int i = 0; i < rows; i++) {

std::cout << arr[i][j] << " ";

}

std::cout << std::endl;

}

}

int main() {

int rows, cols;

std::cout << "Enter the number of rows and columns of the array: ";

std::cin >> rows >> cols;

if (rows <= 0 || rows > MAX\_ROWS || cols <= 0 || cols > MAX\_COLS) {

std::cerr << "Invalid input for rows or columns." << std::endl;

return 1; // Return error code

}

int arr[MAX\_ROWS][MAX\_COLS];

int maxElement;

double stdDev;

readArray(arr, rows, cols);

findMaxAndStdDev(arr, rows, cols, maxElement, stdDev);

printTranspose(arr, rows, cols);

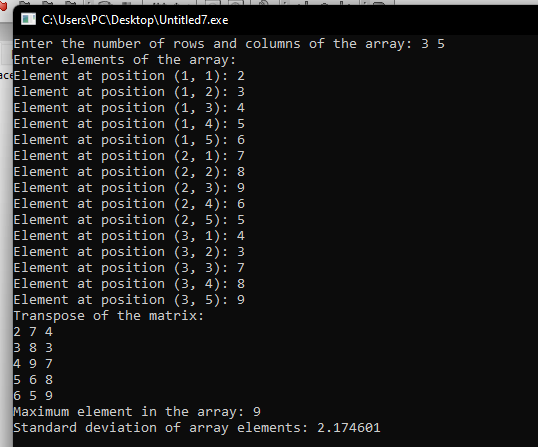
std::cout << "Maximum element in the array: " << maxElement << std::endl;

std::cout << "Standard deviation of array elements: " << stdDev << std::endl;

return 0;

}

**Sample Output:**

****

**GITHUB:**