

**COURSE PROJECT**

**IN**

**DISCIPLINE**

APPLIED INFORMATICS

**ON TOPIC**

**PROGRAM FOR PROCESSING THE TWO-DIMENSIONAL ARRAY A[N,N]**

Prepared by: Teacher:

Martin Sebahtin Hadzhiseid

Course: SSAI

Faculty. № 371222001

Group: 90А

Sofia, 2023

**Task text**

Compile a program for processing the two-dimensional array A[N, N], where the data are integers in the interval [-10;10]. The program is to perform the following actions:

• print the condition of the task;

• print the names of the program author;

• enter the input data;

• print the input data;

• a) form a one-dimensional array C, the elements of which are the elements of the array A, whose squares are greater than 50

• b) sort the resulting array by size;

• printing of the obtained results after processing

**Description of the used modules**

This part of the code uses nested loops to prompt the user to enter elements of a two-dimensional array A of size n x n.

The outer loop iterates over the rows of the array, and the inner loop iterates over the columns of the array.

Within the inner loop, the program uses a do-while loop to ensure that each input is within the range of [-10, 10]. This is accomplished using the logical NOT operator (!) and the logical AND operator (&&) to check if the current input value is not greater than 10 or less than -10. If the input is not within the specified range, the user is prompted to enter a new value until a valid input is received.

Finally, the input value is stored in the appropriate element of the array A using the index variables i and j.

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

{

for (j = 0; j < n; j++)

{

do

{

printf("Enter number in A[%d][%d] = ", i, j);

scanf\_s("%d", &A[i][j]);

} while (!((A[i][j] > -10) && (A[i][j] < 10)));

}

}

This part of the code is responsible for printing out the elements of the two-dimensional array A. It uses nested loops to iterate through each element in A, with the outer loop iterating through each row and the inner loop iterating through each column.

During each iteration, the current element at position A[i][j] is printed to the console using the printf() function with a %d format specifier to print integers.

After printing all the elements in a row, the program prints a newline character using printf("\n") to move to the next line and start printing the elements in the next row.

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

{

for (j = 0; j < n; j++)

{

printf("%d ", A[i][j]);

}

printf("\n");

}

This section of the code is responsible for creating a new array, C, and populating it with elements from the original array, A, whose squares are greater than 50.

The first step is to initialize a variable called "count" to 0, which will be used to keep track of the number of elements in the new array C.

Next, nested loops are used to iterate through each element in the original array A. For each element, the code checks if the square of the element is greater than 50 using the pow() function from the math library. If it is, the current element is added to the new array C at the index indicated by the count variable, and count is incremented by 1.

int count = 0;

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

{

for (j = 0; j < n; j++)

{

if (pow(A[i][j], 2) > 50)

{

C[count] = A[i][j];

count++;

}

}

}

This part of the code is implementing a bubble sort algorithm to sort the elements of the new array C in ascending order. The bubble sort algorithm works by repeatedly swapping adjacent elements if they are in the wrong order until the array is sorted.

The first loop iterates through the array from the first element to the second last element (count - 1) since the last element is already in its correct position after the previous iterations.

The second loop iterates through the unsorted part of the array from the first element to the second last unsorted element (count - i - 1), since the largest element will have already bubbled up to the end of the array after each iteration.

Inside the loop, the if statement checks if the current element is greater than the next element. If it is, then the two elements are swapped using a temporary variable called temp, which stores the value of the current element before it is overwritten with the value of the next element, and then the value of the next element is assigned to the current element, and the value of the temporary variable is assigned to the next element.

After the inner loop completes, the largest element in the unsorted part of the array will have bubbled up to the end of the array. This process is repeated until the array is fully sorted.

for (i = 0; i < count - 1; i++)

{

for (j = 0; j < count - i - 1; j++)

{

if (C[j] > C[j + 1])

{

int temp = C[j];

C[j] = C[j + 1];

C[j + 1] = temp;

}

}

}

**General description of how the program works**

This is a C program that prompts the user to enter the size and elements of a two-dimensional array A of size n x n. It then forms a one-dimensional array C by selecting the elements of A whose squares are greater than 50, and sorts the resulting array by size. The program makes use of nested loops to iterate through each element in A and a bubble sort algorithm to sort the elements of C.

The program starts by declaring constant variable k and variables i, j, n, and two-dimensional arrays A and C, all of size k. The user is then prompted to enter the size of the array and its elements using nested loops, and the input is validated using a do-while loop to ensure that each input is within the range of [-10, 10]. Next, a new array C is created by selecting the elements of A whose squares are greater than 50, and the number of elements in C is stored in the variable count. The elements of C are then sorted using a bubble sort algorithm. Finally, the program outputs the input data and t.

**Listing of the source code of the program**

#include<stdio.h>

#include <math.h>

const int k = 10;

int A[k][k], i, j, n, C[k];

int main()

{

printf("Author: Martin Hadzhiseid ");

printf("FNum: 371222001 ; Group: 90A \n");

printf("\n");

printf("Condition: Form a one-dimentional array C whose elements are the elements of the array A whose squares are greater than 50, and sort the resulting array by size\n");

printf("\n");

printf("Enter number of rows and columns in the array: n = ");

scanf\_s("%d", &n);

printf("Enter elements of the array: \n");

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

{

for (j = 0; j < n; j++)

{

do

{

printf("Enter number in A[%d][%d] = ", i, j);

scanf\_s("%d", &A[i][j]);

} while (!((A[i][j] > -10) && (A[i][j] < 10)));

}

}

printf("\n");

printf("Your array: \n");

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

{

for (j = 0; j < n; j++)

{

printf("%d ", A[i][j]);

}

printf("\n");

}

int count = 0;

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

{

for (j = 0; j < n; j++)

{

if (pow(A[i][j], 2) > 50)

{

C[count] = A[i][j];

count++;

}

}

}

for (i = 0; i < count - 1; i++)

{

for (j = 0; j < count - i - 1; j++)

{

if (C[j] > C[j + 1])

{

int temp = C[j];

C[j] = C[j + 1];

C[j + 1] = temp;

}

}

}

printf("New array:\n");

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

{

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

}

}

**Results of program implementation**



