Specialitatea: Programare și testarea produselor program

Disciplina*: Programarea procedurală*  
Lucrare individuală nr. 1

*Executat:* Pricop Maxim  
*Grupa:* P-2413  
*Verificat:* Natalia Gairunova

**Sarcina 1 – Varianta 26**

**Condiție:**

Se consideră tabloul unidimensional T[0..n] cu elemente de tip String.   
Să se compună un program care va rearanja caracterele fiecărui element al vectorului T în ordine alfabetică prin metoda selecției. Tabloul modificat se va afișa pe ecran.

**Rezolvare:**

unsigned n;

cout << "Introdu numarul de elemente din array: ";

cin >> n;

string T[n];

// Input la elementele array-ului de catre utilizator

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

    cout << "Elementul " << i << ": ";

    cin >> T[i];

}

// Iterarea prin toate elementele array-ului

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

    // Efectuarea selection sort pentru fiecare caracter a elementului din array

    unsigned lungimeString = T[i].length();

    for (unsigned indexCaracter = 0; indexCaracter < lungimeString - 1; indexCaracter++) {

        unsigned indexMin = indexCaracter;

        // Cauta cel mai mic caracter din string, incepand de la indexul caracterului pana la capat

        for (unsigned indexUrmator = indexCaracter + 1; indexUrmator < lungimeString; indexUrmator++) {

            if (T[i][indexUrmator] < T[i][indexMin]) indexMin = indexUrmator;

        }

        // Schimbarea caracterelor intre ele

        char temp = T[i][indexCaracter];

        T[i][indexCaracter] = T[i][indexMin];

        T[i][indexMin] = temp;

    }

}

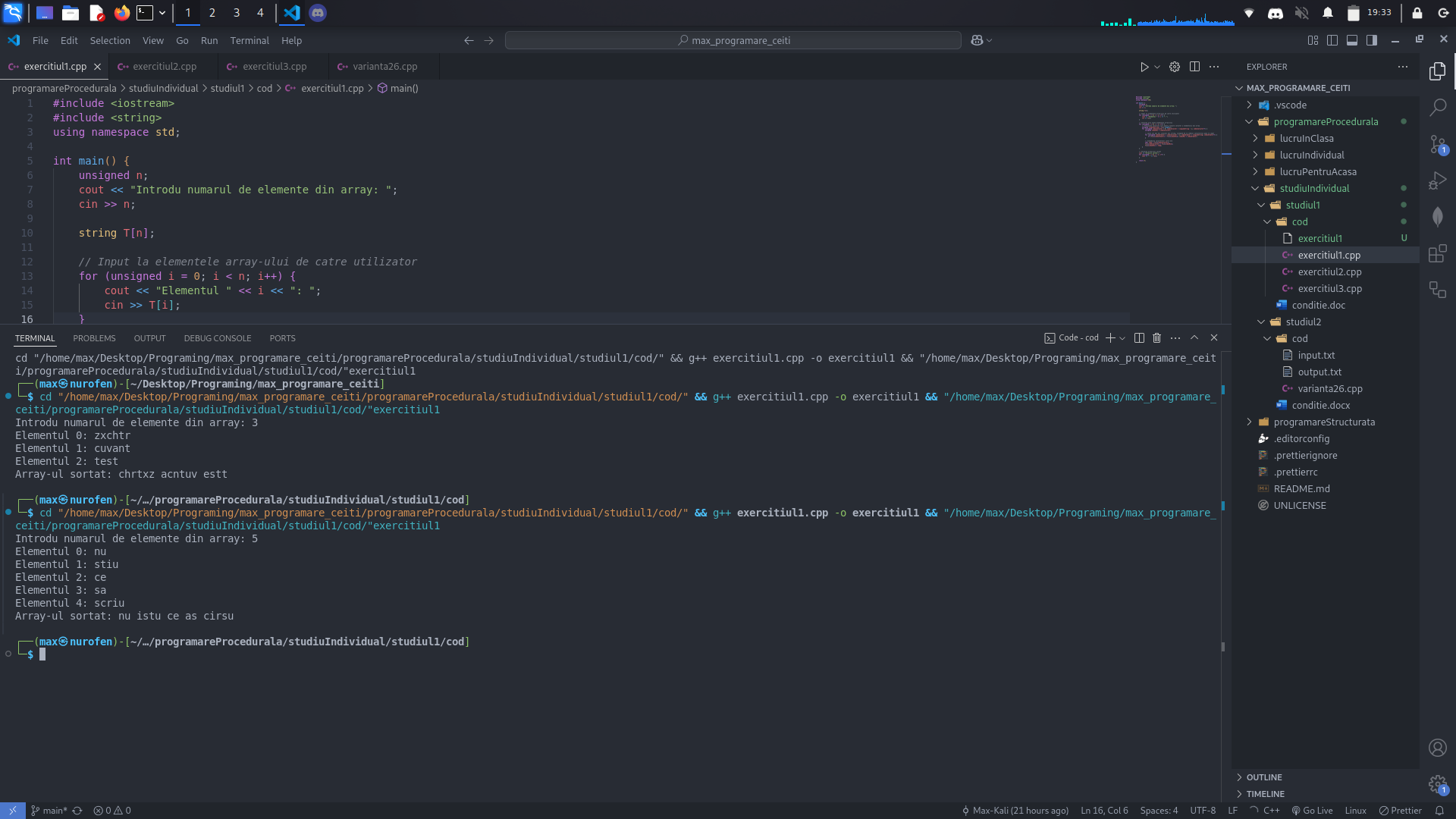
// Afisarea array-ului sortat

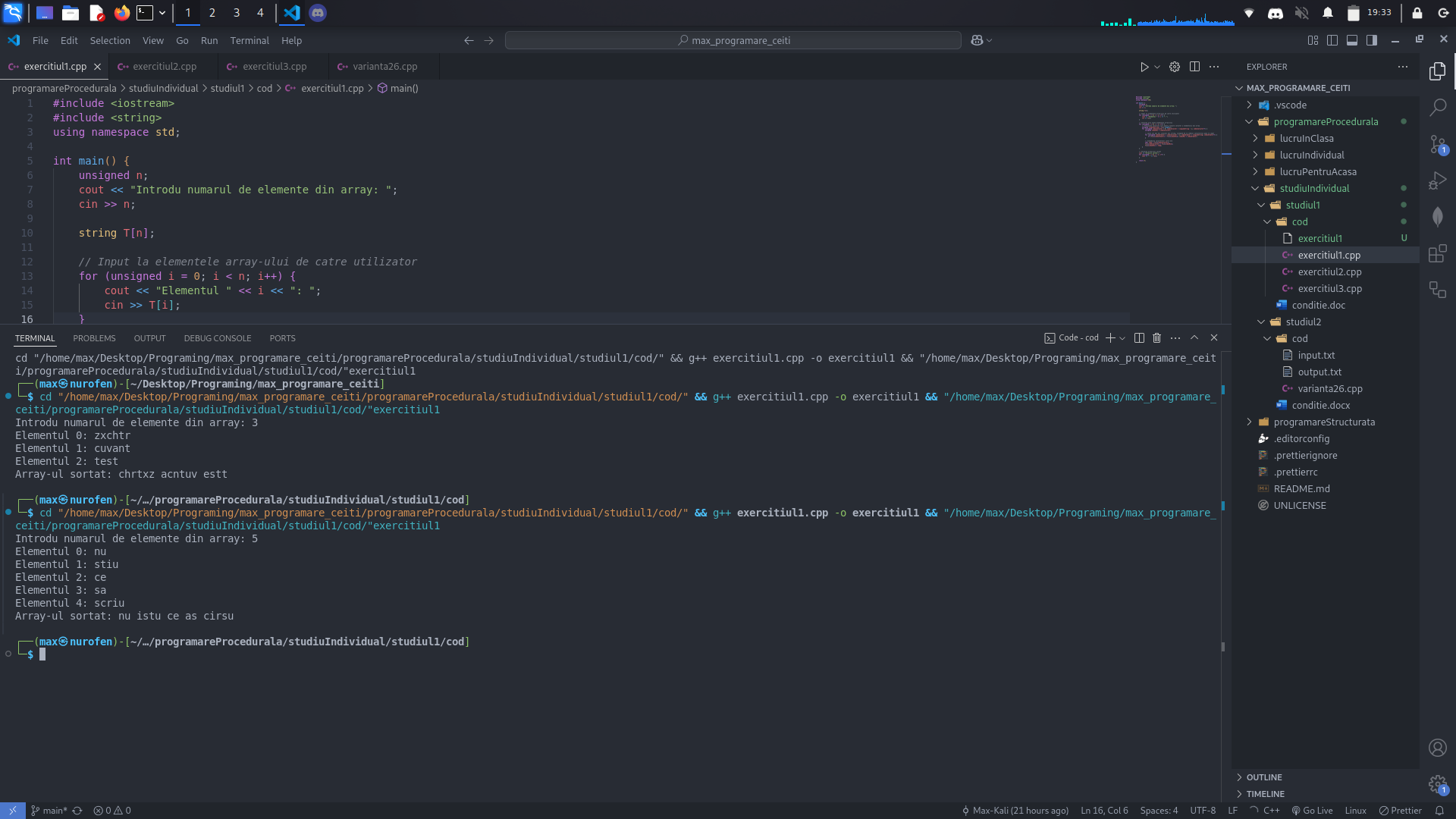
cout << "Array-ul sortat:";

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

    cout << " " << T[i];

}

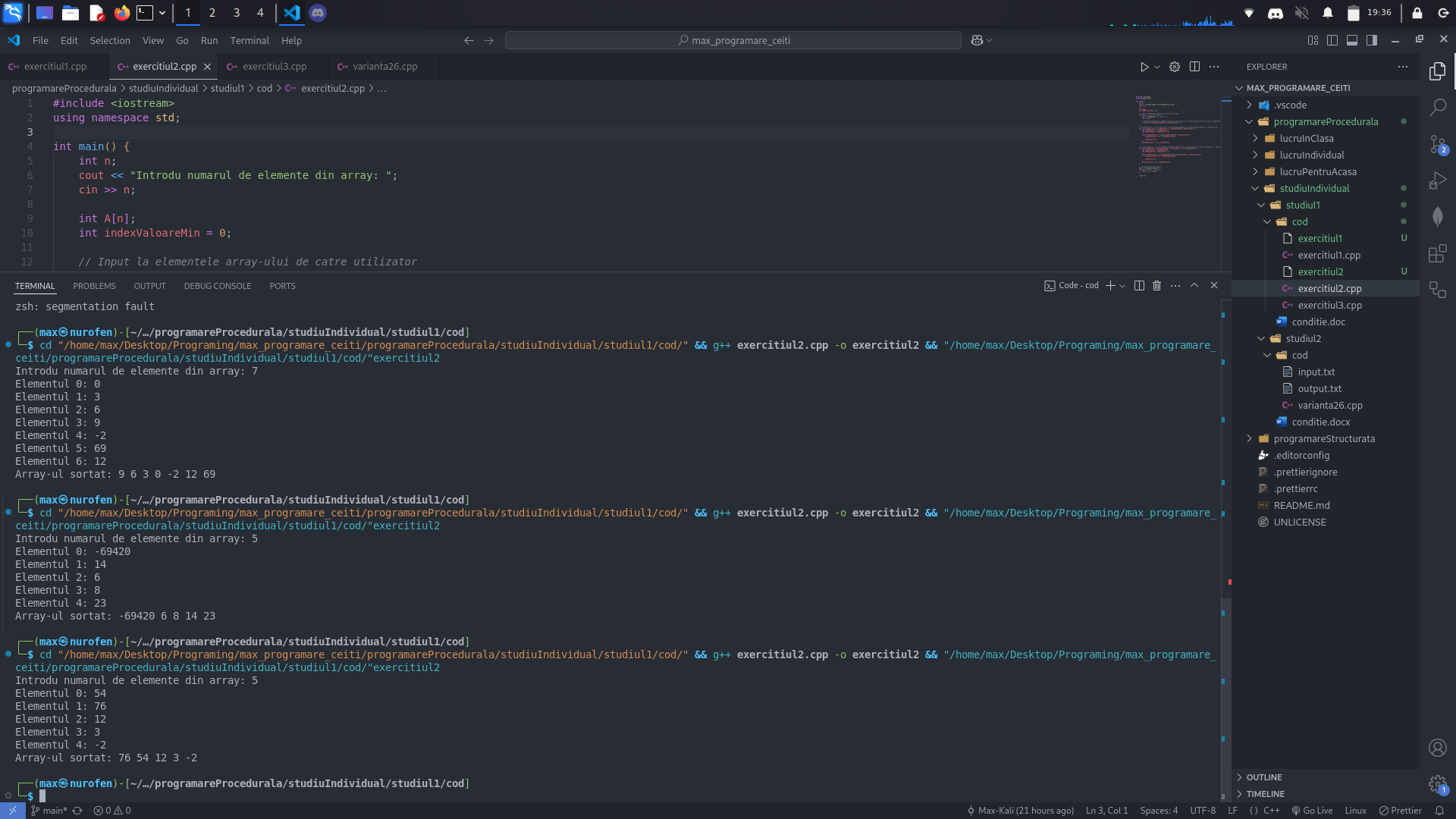
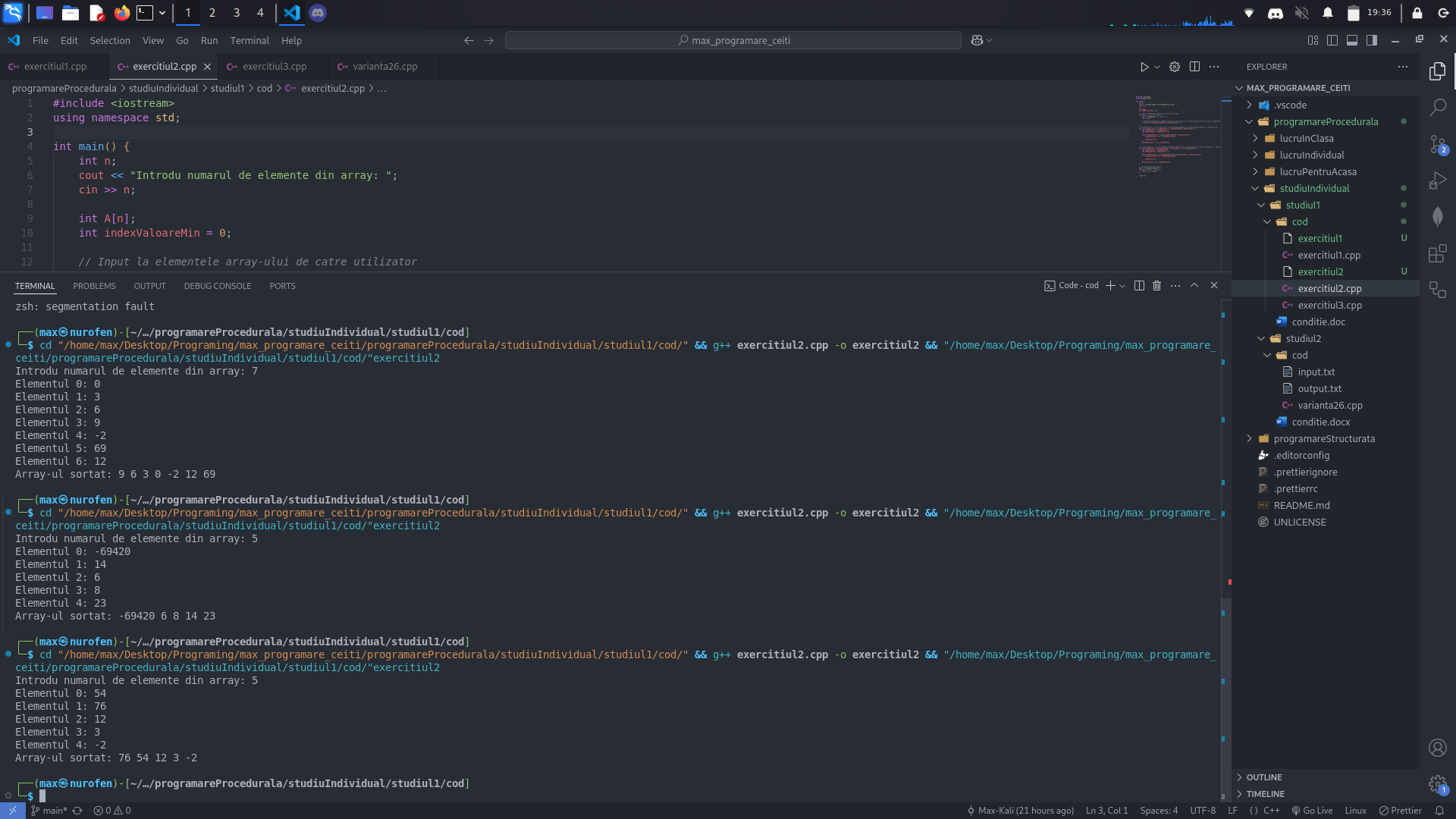
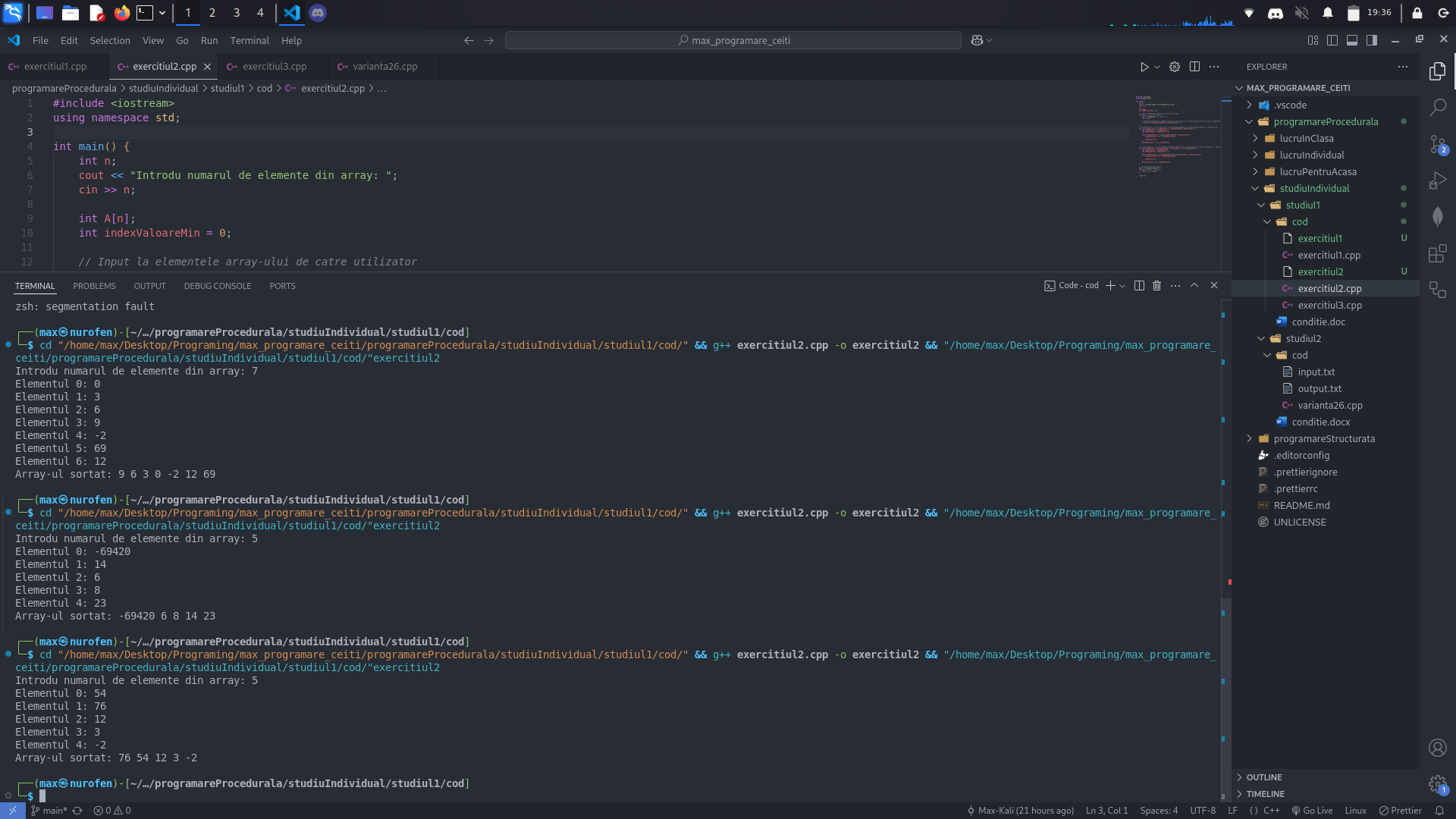
**Rezultat:**



**Sarcina 2 – Varianta 26**

**Condiție:**

Se consideră tabloul unidimensional A[0..n] cu elemente numere întregi.  
Să se compună un program care va rearanja toate elementele până la ultimul element minimal din tabloul A în ordine descrescătoare iar cele ce urmează după aceasta în ordine crescătoare. Tabloul modificat se va afișa pe ecran.

**Rezultat:**

**Rezolvare:**

int n;

cout << "Introdu numarul de elemente din array: ";

cin >> n;

int A[n];

int indexValoareMin = 0;

// Input la elementele array-ului de catre utilizator

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

    cout << "Elementul " << i << ": ";

    cin >> A[i];

    // Stocarea indexului cu elementul minim din array. Mai mic sau egal pentru a stoca ultima lui aparinta in array

    if (A[i] <= A[indexValoareMin]) indexValoareMin = i;

}

// Sortarea partii 1 (de la A[0] pana la A[indexValoareaMin]) in ordine descrescatoare cu insertion sort

for (int indexCurent = 1; indexCurent <= indexValoareMin; indexCurent++) {

    int elementCurent = A[indexCurent];

    int indexInserare = indexCurent - 1;

    while (indexInserare >= 0 && A[indexInserare] < elementCurent) {

        A[indexInserare + 1] = A[indexInserare];

        indexInserare--;

    }

    A[indexInserare + 1] = elementCurent;

}

// Sortarea partii 2 (de la A[indexValoareaMin] pana la finalul array-ului) in ordine crescatoare cu insertion sort

for (int indexCurent = indexValoareMin + 1; indexCurent < n; indexCurent++) {

    int elementCurent = A[indexCurent];

    int indexInserare = indexCurent - 1;

    while (indexInserare >= indexValoareMin && A[indexInserare] > elementCurent) {

        A[indexInserare + 1] = A[indexInserare];

        indexInserare--;

    }

    A[indexInserare + 1] = elementCurent;

}

// Afisarea array-ului sortat

cout << "Array-ul sortat:";

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

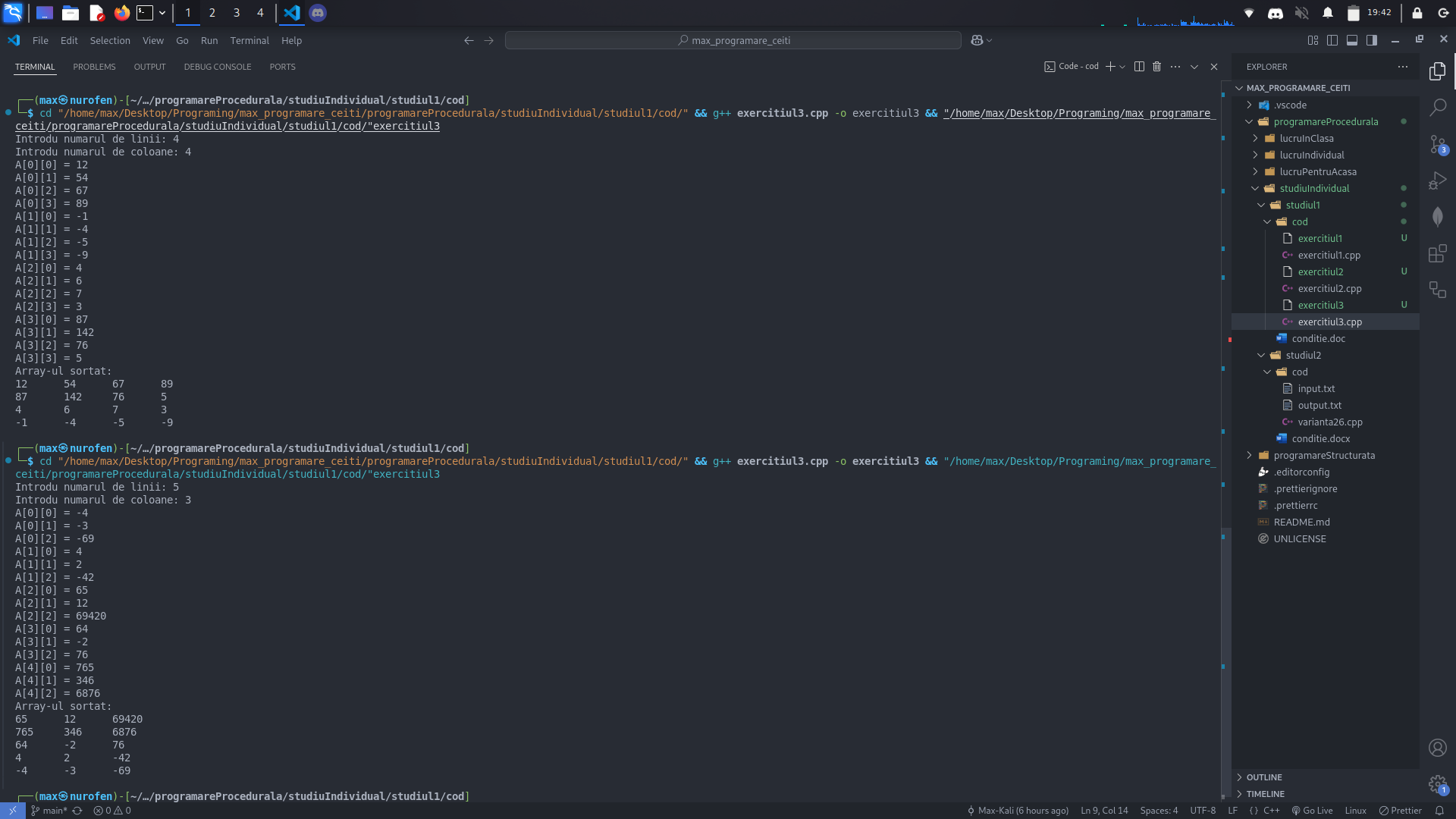
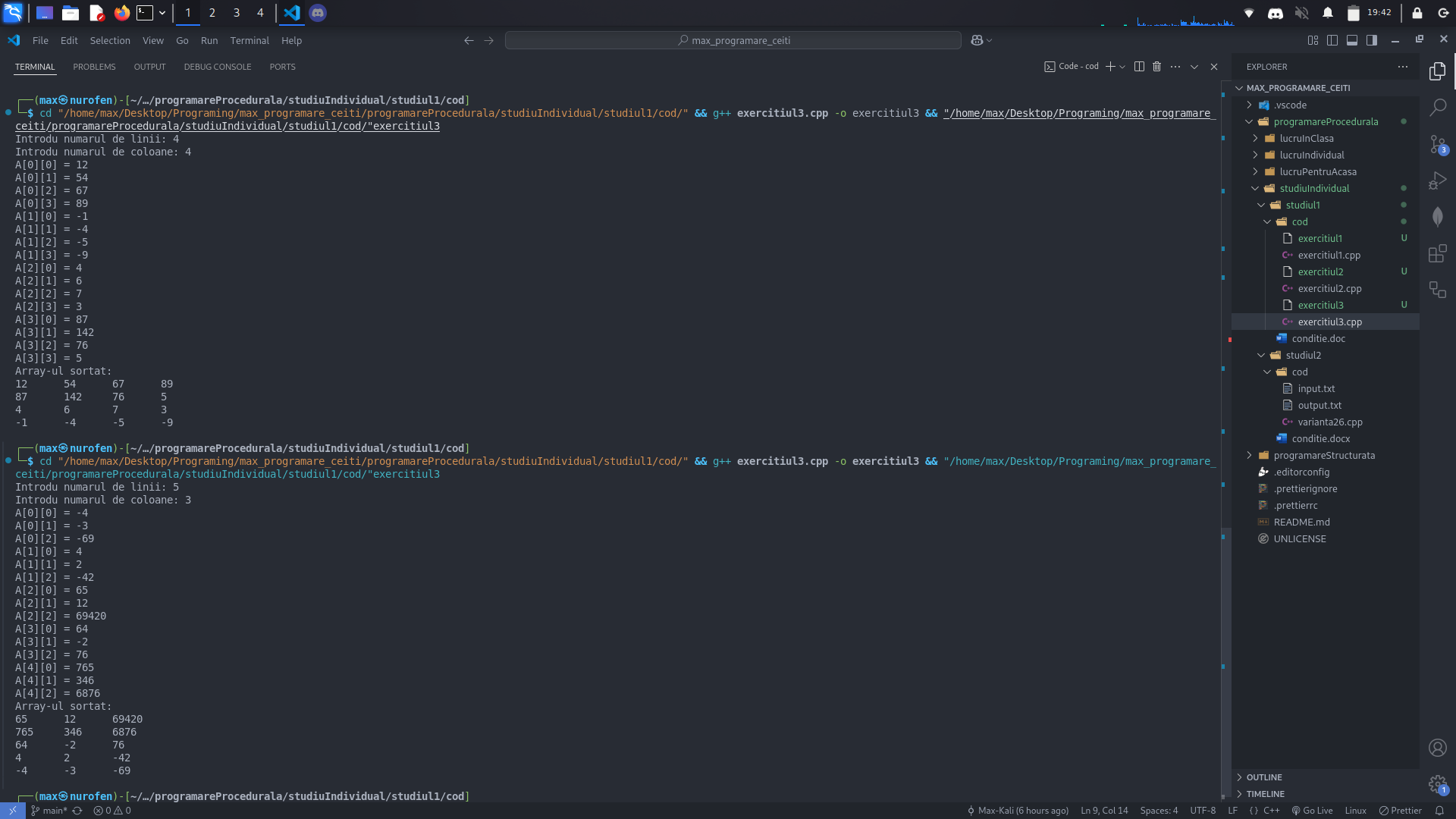
    cout << " " << A[i];

}

**Sarcina 3 – Varianta 26**

**Condiție:**

Se consideră tabloul bidimensional A[0..n, 0..m] cu elemente numere întregi.  
Să se compună un program care va rearanja liniile matricei A astfel încât elementele de pe ultima coloană să fie în ordine descrescătoare prin metoda selecției, fără a folosi o nouă matrice. Matricea modificată se va afișa pe ecran.

**Rezultat:**

**Rezolvare:**

unsigned n, m;

cout << "Introdu numarul de linii: ";

cin >> n;

cout << "Introdu numarul de coloane: ";

cin >> m;

int A[n][m];

// Input la elementele array-ului de catre utilizator

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

    for (unsigned j = 0; j < m; j++) {

        cout << "A[" << i << "][" << j << "] = ";

        cin >> A[i][j];

    }

}

// Sortarea liniilor in functie de elementul de pe ultima coloana (in ordine descrescatoare) cu selection sort

for (unsigned linieCurenta = 0; linieCurenta < n - 1; linieCurenta++) {

    unsigned linieMaxima = linieCurenta;

    // Cauta linia cu valoarea maxima pe ultima coloana

    for (int linieUrmatoare = linieCurenta + 1; linieUrmatoare < n; linieUrmatoare++) {

        if (A[linieUrmatoare][m - 1] > A[linieMaxima][m - 1]) linieMaxima = linieUrmatoare;

    }

    // Daca s-a gasit o linie cu un element mai mare pe ultima coloana, se schimba intre ele

    if (linieMaxima != linieCurenta) {

        for (unsigned coloana = 0; coloana < m; coloana++) {

            unsigned temp = A[linieCurenta][coloana];

            A[linieCurenta][coloana] = A[linieMaxima][coloana];

            A[linieMaxima][coloana] = temp;

        }

    }

}

// Afisarea array-ului sortat

cout << "Array-ul sortat:" << endl;

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

    for (int j = 0; j < m; j++) {

        cout << A[i][j] << "\t";

    }

    cout << endl;

}

**Concluzie**

Exercițiile realizate demonstrează importanța aprofundării conceptelor de manipulare a datelor în C++.

Prin rezolvarea unor exerciții, am evidențiat modul în care algoritmii clasici de sortare pot fi adaptați pentru a satisface cerințe specifice. Completarea lor a subliniat necesitatea unei gândiri logice riguroase și a unei atenții sporite la detalii, aspecte esențiale în dezvoltarea unor soluții stabile.

Rezultatele obținute ne arată că, prin folosirea metodelor de sortare și a tehnicilor de procesare a array-urilor și matricilor, se pot construi programe eficiente care gestionează corect datele. Aceste exerciții constituie o bază solidă ce poate fi extinsă ulterior la probleme de complexitate mai mare, demonstrând impactul pe termen lung al unor tehnici de programare bine încorporate.

**Literatura și Siteuri Utilizate**

Microsoft Visual Studio Code: https://code.visualstudio.com/

Microsoft Word: <https://www.microsoft.com/en/microsoft-365/word>