MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

Departamentul Informatică şi Ingineria Sistemelor

Nume Prenume student

Grupa: IA-201

Raport

pentru lucrarea de laborator Nr.1

la cursul de "Structuri de date și Algoritmi"

Verificat:

Burlacu Natalia, *doctor, conf. univ.*

Departamentul Informatică şi IS,

Facultatea FCIM, UTM

Chișinău – 2023

Sarcină:

1. A rezolva în C problemele ce urmează, utilizând funcții elaborate de dvs. Algoritmul funcțiilor va fi programat, reieșind din conținutul problemei. A prezenta soluția problemei prin abordare procedurală în două versiuni:

A. cu utilizarea metodei de transmitere a parametricelor funcțiilor prin valoare;
B. cu utilizarea metodei de transmitere a parametricelor funcțiilor prin adresă / pointeri (parametrul formal va fi un pointer către valoarea obiectului corespunzător)
C. A desena schema-bloc corespunzătoare problemei rezolvate.

2. A modifica conținutul probleme dvs. reieșind din posibilitățile care lipsesc, dar care pot fi aduse ca plus valoare în condiția problemei existente. A formula și prezenta în scris condiția modificată; a rezolva în C problema dvs. în varianta modificată, utilizând funcții elaborate de dvs. − Reieșind din faptul că în fiecare problemă din vr. I sunt de valorificat câte două metode de sortare în vr.II, a problemei propuse de dvs,, a se valorifica în mod obligatoriu metodele: Merge Sort & Insert Sort.

Condiția problemei:

22. Sunt date 2 array-uri 1-D cu elemente de tip *integer,* citite de la tastieră. Lungimea array-urilor va fi diferită. A scrie următoarele funcții C (cu apelare ulterioară ale acestora în main) care:

I. Să sorteze elementele primului array în ordine crescătoare aplicând metoda Merge Sort;
II. Să sorteze elementele celui de-al 2-lea array în ordine crescătoare aplicând metoda Insert Sort;
III. Să găsească mediana celor 2 array-uri deja sortate; să afișeze valoarea medianei celor 2 array-uri.

De exemplu:
Input:
A1 este:

90	240	300
0	1	2

A2 este:

10	13	14	20	25
8	5	2	3	4

Output:
Mediana array-urilor A1 & A2 = 22.50

IV. În cazul în care ambele array-uri au aceiași lungime ambele array-uri vor fi sortate în ordine descrescătoare, aplicând metoda Bubble Sort.
V. După fiecare manipulare să fie prevăzută afișarea rezultatelor ca și concluzie (De ex: Dacă sunt astfel de elemente, programul le afișează (elementul și indexul acesteia, alte date stipulate în condiția problemei); nu sunt astfel de elemente –programul informează despre absența lor), etc.

Figura 1.1 – Condiția probleme

1. **Codul programului, având comentarii relevante în el și schema Bloc;**

Codul:

----------Versiunea cu transmiterea parametrilor functiilor prin valoare----------

```
#include <stdio.h>

//Functia pentru citirea de la tastatura
//a elementelor unui vector
void scan(int arr[50], int n) {
	for (int i = 0; i < n; i++) {
		scanf_s("%i", &arr[i]);
	}
}

//Functia pentru afisarea vectorului
void print(int arr[50], int n) {
	for (int i = 0; i < n; i++) {
		printf("%i ", arr[i]);
	}
	printf("\n");
}

//Functia care schimba cu locul valorile
//a doua variabile intre ele
void swap(int* a, int* b) {
	int aux = *a;
	*a = *b;
	*b = aux;
}

//Bubble Sort ordine descrescatoare
void bubbleSort(int a[50], int n) {
	for (int i = 0; i < n - 1; i++) {
		for (int j = 0; j < n - i - 1; j++) {
			if (a[j] < a[j + 1]) {
				swap(&a[j], &a[j + 1]);
			}
		}
	}
}

//Insertion Sort
void insertSort(int a[50], int n) {
	int j;
	for (int i = 0; i < n; i++) {
		int key = a[i];
		j = i - 1;
		while (j >= 0 && a[j] > key) {
			a[j + 1] = a[j];
			j--;
		}
		a[j + 1] = key;
	}
}

void merge(int arr[], int l, int m, int r)
```

```
{
	int i, j, k;
	int n1 = m - l + 1;
	int n2 = r - m;

	//Cream 2 vectori temporati
	int L[50], R[50];

	//Copiem datele in vectorii temporari L si R
	for (i = 0; i < n1; i++)
		L[i] = arr[l + i];
	for (j = 0; j < n2; j++)
		R[j] = arr[m + 1 + j];

	//Unim subArray-urile inapoi in vectorul arr[l ..r]
	i = 0; //Indexul initial pt subArray-ul din stanga
	j = 0; //Indexul initial pt subArray-ul din dreapta
	k = l; //Indexul initial pt subArray-ul unit
	while (i < n1 && j < n2) {
		if (L[i] <= R[j]) {
			arr[k] = L[i];
			i++;
		}
		else {
			arr[k] = R[j];
			j++;
		}
		k++;
	}

	//Copiem elementele ramase din recursia stanga
	while (i < n1) {
		arr[k] = L[i];
		i++;
		k++;
	}

	//Copiem elementele ramase din recursia dreapta
	while (j < n2) {
		arr[k] = R[j];
		j++;
		k++;
	}
}

void mergeSort(int arr[], int l, int r)
{
	if (l < r) {

		int m = l + (r - l) / 2; //Mijlocul vectorului

		mergeSort(arr, l, m);
		mergeSort(arr, m + 1, r);
		merge(arr, l, m, r);
	}
}

//Calculeaza mediana
float findMed(int a[100], int n) {
	//Verificam daca numarul de
	//elemente este par sau impar
	int m = n / 2; //Elementul de mijloc
	if (n % 2 != 0) {
```

```
			return (float)a[m]; //Returneaza elementul din mijloc
		}
		else { //Returneaza media aritmetica a celor 2 elemente din mijloc
			return (float)((a[m - 1] + a[m]) / 2.0);
		}
}

int main() {

		int a1[50], a2[50], medArr[100], n1, n2;
		//Citim numarul de elemente pt vactori
		printf("Dati numarul de elemente pentru 2 vectori:\n");
		scanf_s("%i", &n1);
		scanf_s("%i", &n2);
		printf("\n");

		//Citim elementele vectorilor
		printf("Dati elementele celor doi vectori:\n");
		scan(a1, n1);
		scan(a2, n2);
		printf("\n");

		//Verificam:
		//			Daca dimensiunile celor 2 vectori sunt la fel,
		//		ei vor fi sortati descrescator folosind metoda Bubble Sort.
		//			Daca dimensiunile sunt diferite, atunci primul vector va fi
		//		sortat dupa metoda Merge Sort iar al doilea dupa metoda
		//		Insertion sort.

		if (n1 == n2) {
			//Apelam functia bubbleSort pentru ambii vectori
			printf("Ambii vectori sunt de aceeasi lungime,\nrespectiv ei vor fi sortati
descrescator:\n");
			bubbleSort(a1, n1);
			bubbleSort(a2, n2);

			//Afisam la ecran vectorii sortati
			print(a1, n1);
			print(a2, n2);
			printf("\n");
		}
		else {

			printf("Primul vector sortat dupa metoda Megre Sort:\n");
			mergeSort(a1, 0, n1);		//Sortam 1 array cu Merge Sort
			for (int i = 0; i < n1; i++)
				a1[i] = a1[i + 1];

			print(a1, n1);		//Afisam elementele vectorului sortat

			printf("Al doilea vector sortat dupa metoda Insertion Sort:\n");
			insertSort(a2, n2); //Sortam 2-lea arry cu Insert Sort
			print(a2, n2); //Afisam vectorul sortat
			printf("\n");
		}

		//Unim vectorii in vectorul "medArr" pentru a-l sorta in continuare si pt a afla
mediana
		int i, j = 0;
		for (i = 0; i < n1; i++) { //Primul array
			medArr[i] = a1[i];
		}
		for (i = n1; i < n1 + n2; i++) { //2-lea Array
```

```
			medArr[i] = a2[j];
			j++;
		}

		printf("Vectorul primit:\n");
		print(medArr, n1 + n2); //Afisam vectorul
		printf("\n");

		printf("Sortam vectorul primit din cele doua:\n");
		insertSort(medArr, n1 + n2); //Sortam vectorul cu insert Sort
		print(medArr, n1 + n2); //Afisam vectorul
		//Afisam mediana
		printf("Mediana vectorului a1 si a2 este: %f\n", findMed(medArr, n1 + n2));

		return 0;
}
```

----------Versiunea cu transmiterea parametrilor funcțiilor prin pointeri----------

```
#include <stdio.h>

//Functia pentru citirea de la tastatura
//a elementelor unui vector
void scan(int* arr, int* n) {
	for (int i = 0; i < *n; i++) {
		scanf_s("%i", arr + i);
	}
}

//Functia pentru afisarea vectorului
void print(int* arr, int* n) {
	for (int i = 0; i < *n; i++) {
		printf("%i ", *(arr + i));
	}
	printf("\n");
}

//Functia care schimba cu locul valorile
//a doua variabile intre ele
void swap(int* a, int* b) {
	int aux = *a;
	*a = *b;
	*b = aux;
}

//Bubble Sort ordine descrescatoare
void bubbleSort(int* a, int* n) {
	for (int i = 0; i < *n - 1; i++) {
		for (int j = 0; j < *n - i - 1; j++) {
			if (*(a + j) < *(a + j + 1)) {
				swap(a + j, a + j + 1);
			}
		}
	}
}

//Insertion Sort
void insertSort(int* a, int* n) {
	int j;
```

```
	for (int i = 0; i < *n; i++) {
		int key = *(a + i);
		j = i - 1;
		while (j >= 0 && *(a + j) > key) {
			*(a + j + 1) = *(a + j);
			j--;
		}
		*(a + j + 1) = key;
	}
}

void merge(int* arr, int* l, int* m, int* r)
{
	int i, j, k;
	int n1 = *m - *l + 1;
	int n2 = *r - *m;

	//Cream 2 vectori temporati
	int L[50], R[50];

	//Copiem datele in vectorii temporari L si R
	for (i = 0; i < n1; i++)
		L[i] = arr[*l + i];
	for (j = 0; j < n2; j++)
		R[j] = arr[*m + 1 + j];

	//Unim subArray-urile inapoi in vectorul arr[l ..r]
	i = 0; //Indexul initial pt subArray-ul din stanga
	j = 0; //Indexul initial pt subArray-ul din dreapta
	k = *l; //Indexul initial pt subArray-ul unit
	while (i < n1 && j < n2) {
		if (L[i] <= R[j]) {
			arr[k] = L[i];
			i++;
		}
		else {
			arr[k] = R[j];
			j++;
		}
		k++;
	}

	//Copiem elementele ramase din recursia stanga
	while (i < n1) {
		arr[k] = L[i];
		i++;
		k++;
	}

	//Copiem elementele ramase din recursia dreapta
	while (j < n2) {
		arr[k] = R[j];
		j++;
		k++;
	}
}


void mergeSort(int* arr, int* l, int* rs)
{
	if (*l < *rs) {

		int m = *l + (*rs - *l) / 2; //Mijlocul vectorului
```

```
		int m2 = m + 1;

		mergeSort(arr, l, &m);
		mergeSort(arr, &m2, rs);
		merge(arr, l, &m, rs);
	}
}

float findMed(int* a, int* n) {
	//Verificam daca numarul de
	//elemente este par sau impar
	int m = *n / 2; //Elementul de mijloc
	if (*n % 2 != 0) {
		return (float)a[m];
	}
	else {
		return (float)((a[m - 1] + a[m]) / 2.0);
	}
}

int main() {

	int a1[50], a2[50], medArr[100], n1, n2, l = 0;
	//Citim numarul de elemente pt vactori
	printf("Dati numarul de elemente pentru 2 vectori:\nn1 = ");
	scanf_s("%i", &n1);
	printf("n2 = ");
	scanf_s("%i", &n2);
	printf("\n");

	//Citim elementele vectorilor
	printf("Dati elementele celor doi vectori:\nPrimul vector:\n");
	scan(&a1, &n1);
	printf("Al doilea vector:\n");
	scan(&a2, &n2);
	printf("\n");

	//Verificam:
	//		Daca dimensiunile celor 2 vectori sunt la fel,
	//	ei vor fi sortati descrescator folosind metoda Bubble Sort.
	//		Daca dimensiunile sunt diferite, atunci primul vector va fi
	//	sortat dupa metoda Merge Sort iar al doilea dupa metoda
	//	Insertion sort.

	if (n1 == n2) {
		//Apelam functia bubbleSort pentru ambii vectori
		printf("Ambii vectori sunt de aceeasi lungime,\nrespectiv ei vor fi sortati
descrescator cu Bubble Sort:\n");
		bubbleSort(&a1, &n1);
		bubbleSort(&a2, &n2);

		//Afisam la ecran vectorii sortati
		print(&a1, &n1);
		print(&a2, &n2);
		printf("\n");
	}
	else {

		printf("Primul vector sortat dupa metoda Megre Sort:\n");
		mergeSort(a1, &l, &n1);	//Sortam 1 array cu Merge Sort

		for (int i = 0; i < n1; i++)
			a1[i] = a1[i + 1];
```

```
		print(a1, &n1);		//Afisam elementele vectorului sortat

		printf("Al doilea vector sortat dupa metoda Insertion Sort:\n");
		insertSort(a2, &n2); //Sortam 2-lea arry cu Insert Sort
		print(a2, &n2); //Afisam vectorul sortat
		printf("\n");
	}

	//Unim vectorii in vectorul "medArr" pentru a-l sorta in continuare si pt a afla
mediana
	int i, j = 0;

	for (i = 0; i < n1; i++) { //Primul array
		medArr[i] = a1[i];
	}
	for (i = n1; i < n1 + n2; i++) { //2-lea Array
		medArr[i] = a2[j];
		j++;
	}

	int n3 = n1 + n2;
	printf("Vectorul primit dupa unirea vectorilor initiali:\n");
	print(medArr, &n3); //Afisam vectorul
	printf("\n");

	printf("Sortam vectorul primit din cele doua:\n");
	bubbleSort(medArr, &n3); //Sortam vectorul cu insert Sort
	print(medArr, &n3); //Afisam vectorul

	printf("\nMediana vectorului a1 si a2 este: %f\n", findMed(medArr, &n3)); //Afisam
mediana

	return 0;
}
```

----------Versiunea modificată----------

```
#include <stdio.h>

//Functia pentru citirea de la tastatura
//a elementelor unui vector
void scan(int* arr, int* n) {
	for (int i = 0; i < *n; i++) {
		scanf_s("%i", arr + i);
	}
}

//Functia pentru afisarea vectorului
void print(int* arr, int* n) {
	for (int i = 0; i < *n; i++) {
		printf("%i ", *(arr + i));
	}
	printf("\n");
}

//Functia care schimba cu locul valorile
//a doua variabile intre ele
```

```
void swap(int* a, int* b) {
	int aux = *a;
	*a = *b;
	*b = aux;
}

//Bubble Sort ordine descrescatoare
void bubbleSort(int* a, int* n) {
	for (int i = 0; i < *n - 1; i++) {
		for (int j = 0; j < *n - i - 1; j++) {
			if (*(a + j) < *(a + j + 1)) {
				swap(a + j, a + j + 1);
			}
		}
	}
}

//Insertion Sort
void insertSort(int* a, int* n) {
	int j;
	for (int i = 0; i < *n; i++) {
		int key = *(a + i);
		j = i - 1;
		while (j >= 0 && *(a + j) > key) {
			*(a + j + 1) = *(a + j);
			j--;
		}
		*(a + j + 1) = key;
	}
}

void merge(int* arr, int* l, int* m, int* r)
{
	int i, j, k;
	int n1 = *m - *l + 1;
	int n2 = *r - *m;

	//Cream 2 vectori temporati
	int L[50], R[50];

	//Copiem datele in vectorii temporari L si R
	for (i = 0; i < n1; i++)
		L[i] = arr[*l + i];
	for (j = 0; j < n2; j++)
		R[j] = arr[*m + 1 + j];

	//Unim subArray-urile inapoi in vectorul arr[l ..r]
	i = 0; //Indexul initial pt subArray-ul din stanga
	j = 0; //Indexul initial pt subArray-ul din dreapta
	k = *l; //Indexul initial pt subArray-ul unit
	while (i < n1 && j < n2) {
		if (L[i] <= R[j]) {
			arr[k] = L[i];
			i++;
		}
		else {
			arr[k] = R[j];
			j++;
		}
		k++;
	}

	//Copiem elementele ramase din recursia stanga
```

```
	while (i < n1) {
		arr[k] = L[i];
		i++;
		k++;
	}

	//Copiem elementele ramase din recursia dreapta
	while (j < n2) {
		arr[k] = R[j];
		j++;
		k++;
	}
}


void mergeSort(int* arr, int* l, int* rs)
{
	if (*l < *rs) {

		int m = *l + (*rs - *l) / 2; //Mijlocul vectorului
		int m2 = m + 1;

		mergeSort(arr, l, &m);
		mergeSort(arr, &m2, rs);
		merge(arr, l, &m, rs);
	}
}

//Gaseste mediana unui vector
float findMed(int* a, int* n) {
	//Verificam daca numarul de
	//elemente este par sau impar
	int m = *n / 2; //Elementul de mijloc
	if (*n % 2 != 0) {
		return (float)a[m];
	}
	else {
		return (float)((a[m - 1] + a[m]) / 2.0);
	}
}

void localExtrem(int* a, int* n, int* indMin, int* indMax) {
	int k1 = 0, k2 = 0;
	for (int i = 1; i < *n - 1; i++) {
		//Verifica daca ambii vecini ai lui a[i] sunt mai mari si
		//stocheaza indexul acelui element in indMin in caz contrar
		//indexul va fi stocat in indMax
		if (a[i] < a[i + 1] && a[i] < a[i - 1]) {
			indMin[k1] = i;
			k1++;
		}
		else if (a[i] > a[i + 1] && a[i] > a[i - 1]) {
			indMax[k2] = i;
			k2++;
		}
	}
}


int main() {

	int a1[50], a2[50], medArr[100], n1, n2, l = 0, iMin[50], iMax[50], i, j = 0, n3;
```

```
	//Citim numarul de elemente pt vectori-----------------------
	printf("Dati numarul de elemente pentru 2 vectori:\nn1 = ");
	scanf_s("%i", &n1);
	printf("n2 = ");
	scanf_s("%i", &n2);
	printf("\n");
	//-----------------------------------------------------------

	//Citim elementele vectorilor--------------------------------
	printf("Dati elementele celor doi vectori:\nPrimul vector:\n");
	scan(&a1, &n1);
	printf("\nAl doilea vector:\n");
	scan(&a2, &n2);
	printf("\n");
	//-----------------------------------------------------------

	//Unim vectorii in vectorul "medArr"----------
	for (i = 0; i < n1; i++) { //Primul array
		medArr[i] = a1[i];
	}
	for (i = n1; i < n1 + n2; i++) { //2-lea Array
		medArr[i] = a2[j];
		j++;
	}
	//--------------------------------------------

	//Afisam vectorul care contine elementele celor doi vectori---
	n3 = n1 + n2;
	printf("Vectorul primit dupa unirea vectorilor initiali:\n");
	print(medArr, &n3); //Afisam vectorul
	printf("\n");
	//-----------------------------------------------------------

	//Aflam extremele locale ale vectorului------------------
	localExtrem(medArr, &n3, iMin, iMax);

	printf("Indexii extremelor locale minime:\n   ");
	for (i = 0; iMin[i] > -1000 && iMin[i] < 1000; i++) {
		printf("%i ", iMin[i]);
	}
	if (i == 0) {
		printf("Nu au fost gasite extreme minime locale\n");
	}

	printf("\nIndexii extremelor locale maxime:\n   ");
	for (i = 0; iMax[i] > -1000 && iMax[i] < 1000; i++) {
		printf("%i ", iMax[i]);
	}
	if (i == 0) {
		printf("Nu au fost gasite extreme maxime locale\n");
	}
	//-------------------------------------------------------

	//Verificam:
	//		Daca dimensiunile celor 2 vectori sunt la fel,
	//	ei vor fi sortati descrescator folosind metoda Bubble Sort.
	//		Daca dimensiunile sunt diferite, atunci primul vector va fi
	//	sortat dupa metoda Merge Sort iar al doilea dupa metoda
	//	Insertion sort.
	if (n1 == n2) {
		//Apelam functia bubbleSort pentru ambii vectori
		printf("\n\nAmbii vectori sunt de aceeasi lungime,\nrespectiv ei vor fi sortati
descrescator cu Bubble Sort:\n");
```

```
            bubbleSort(&a1, &n1);
            bubbleSort(&a2, &n2);

            //Afisam la ecran vectorii sortati
            print(&a1, &n1);
            print(&a2, &n2);
            printf("\n");
        }
        else {

            printf("\n\nPrimul vector sortat dupa metoda Megre Sort:\n");
            mergeSort(a1, &l, &n1);     //Sortam 1 array cu Merge Sort

            for (int i = 0; i < n1; i++)
                a1[i] = a1[i + 1];

            print(a1, &n1);      //Afisam elementele vectorului sortat
            printf("\nAl doilea vector sortat dupa metoda Insertion Sort:\n");
            insertSort(a2, &n2); //Sortam 2-lea arry cu Insert Sort
            print(a2, &n2); //Afisam vectorul sortat
            printf("\n");
        }

        printf("Sortam vectorul primit din cele doua cu Bubble Sort:\n");
        bubbleSort(medArr, &n3); //Sortam vectorul cu Bubble Sort
        print(medArr, &n3); //Afisam vectorul

        printf("\nMediana vectorului a1 si a2 este: %f\n\n", findMed(medArr, &n3)); //Afisam
mediana

        return 0;
}
```

În versiunea dată am adăugat un algoritm pentru aflarea extremelor locale ale vectorului.

Schema Bloc pentru prima versiune:

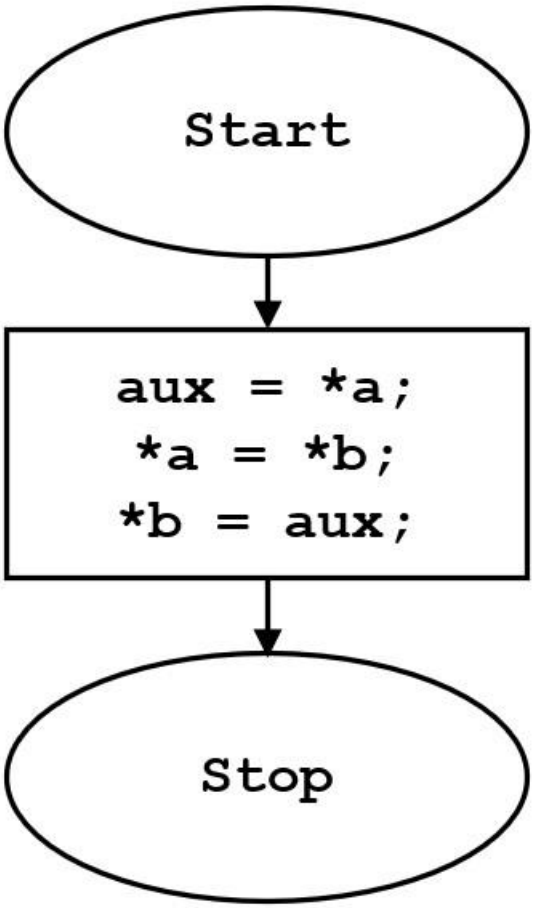

Figura 2.1 – Funcția ”*swap()*”

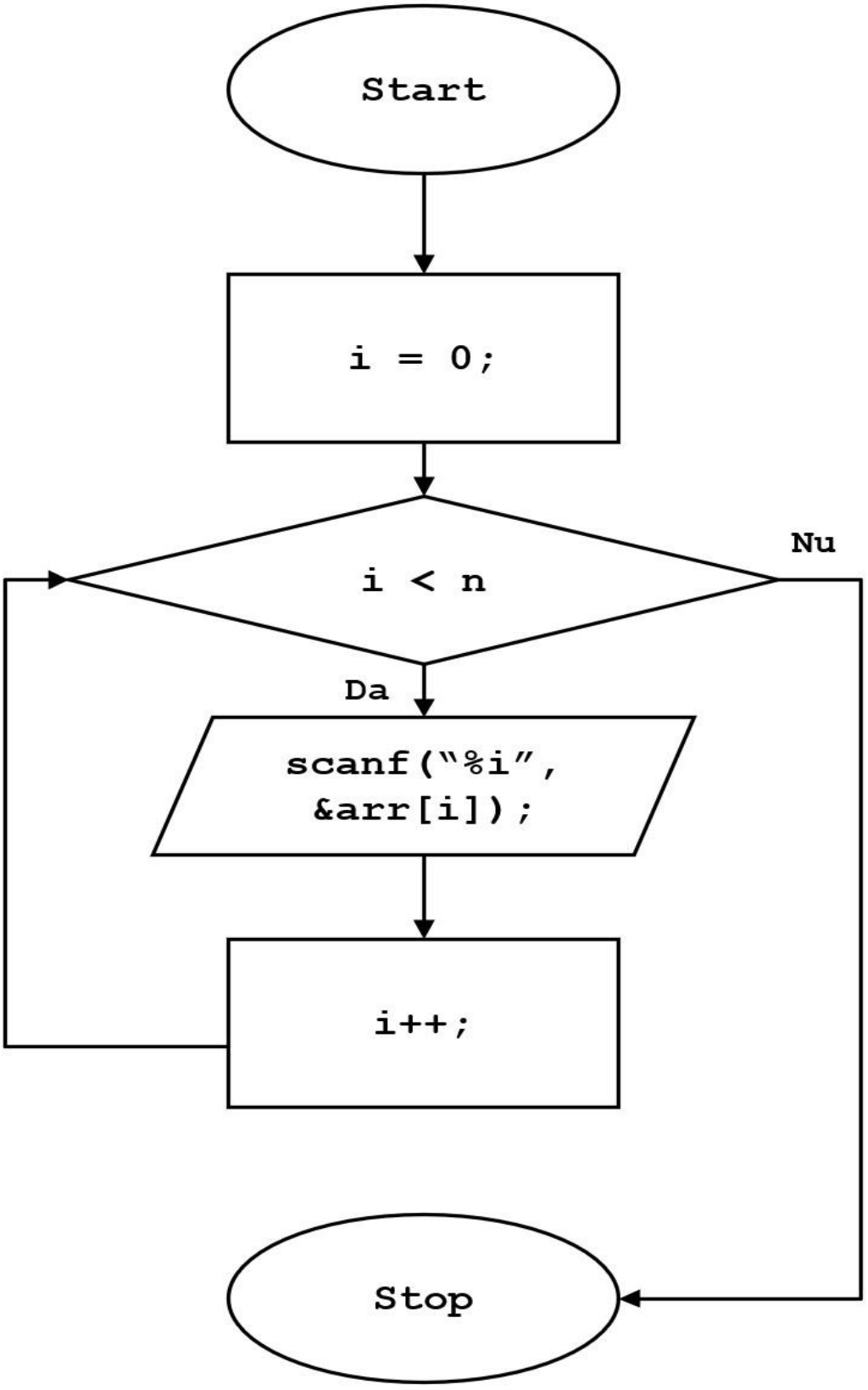

Figura 2.2 – Funcția "*scan()*"

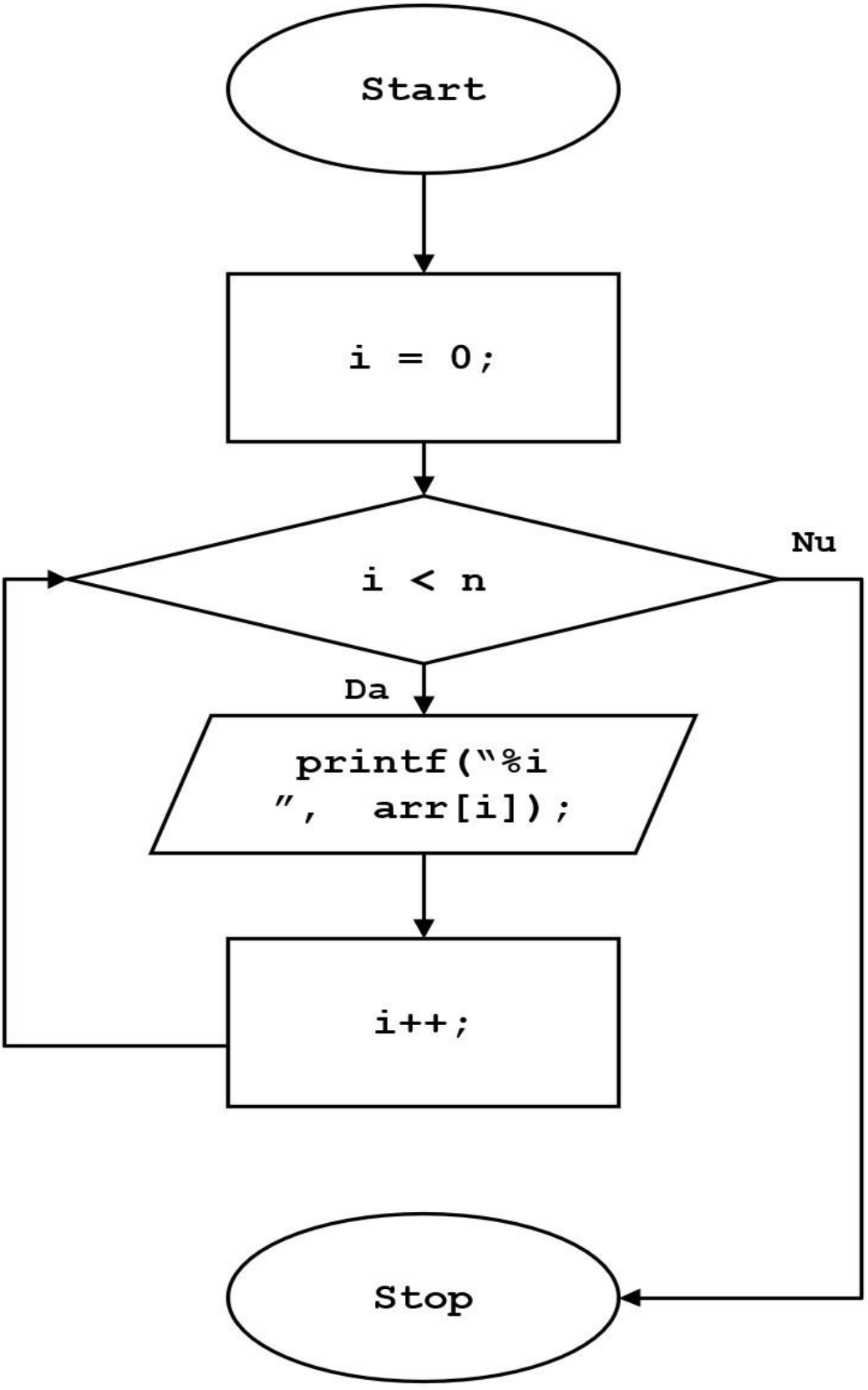

Figura 2.3 – Funcția "*print()*"

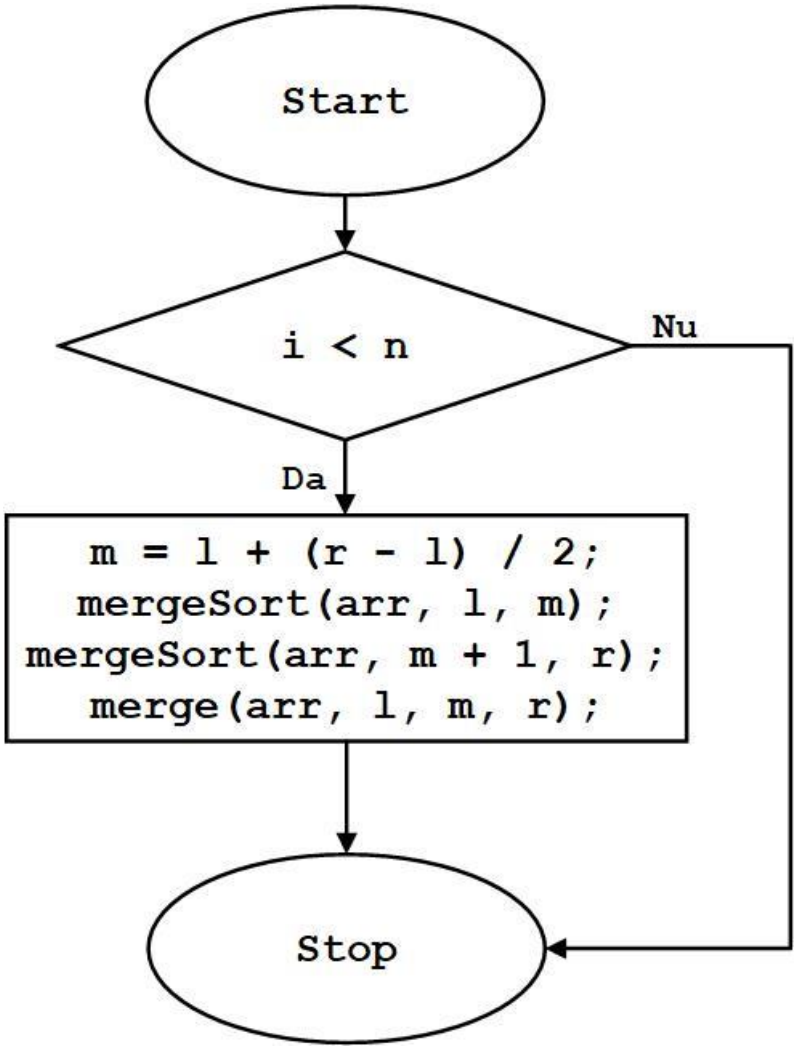

Figura 2.4 – Funcția ”*mergeSort()*”

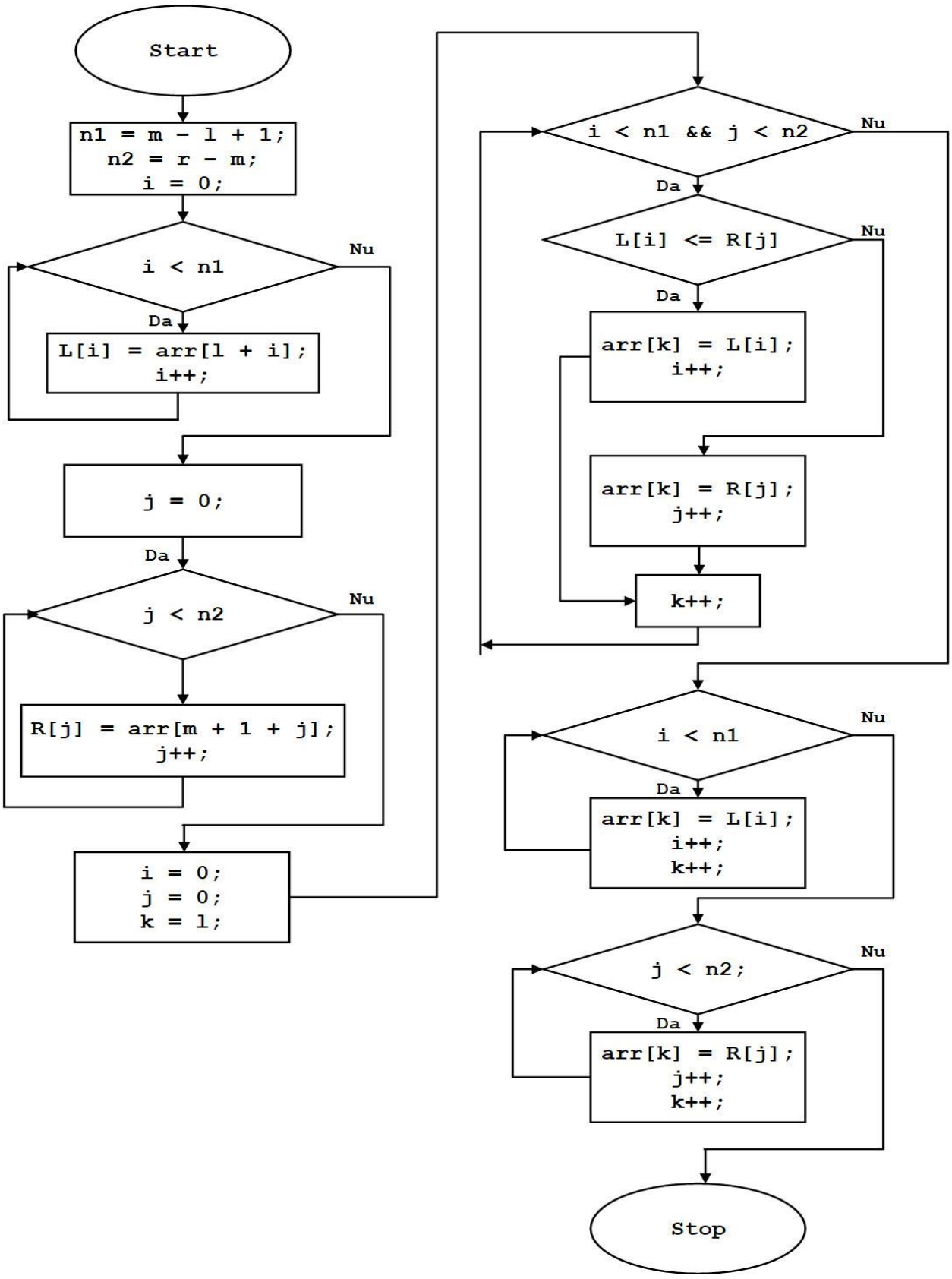

Figura 2.5 – Funcția ”*merge()*”

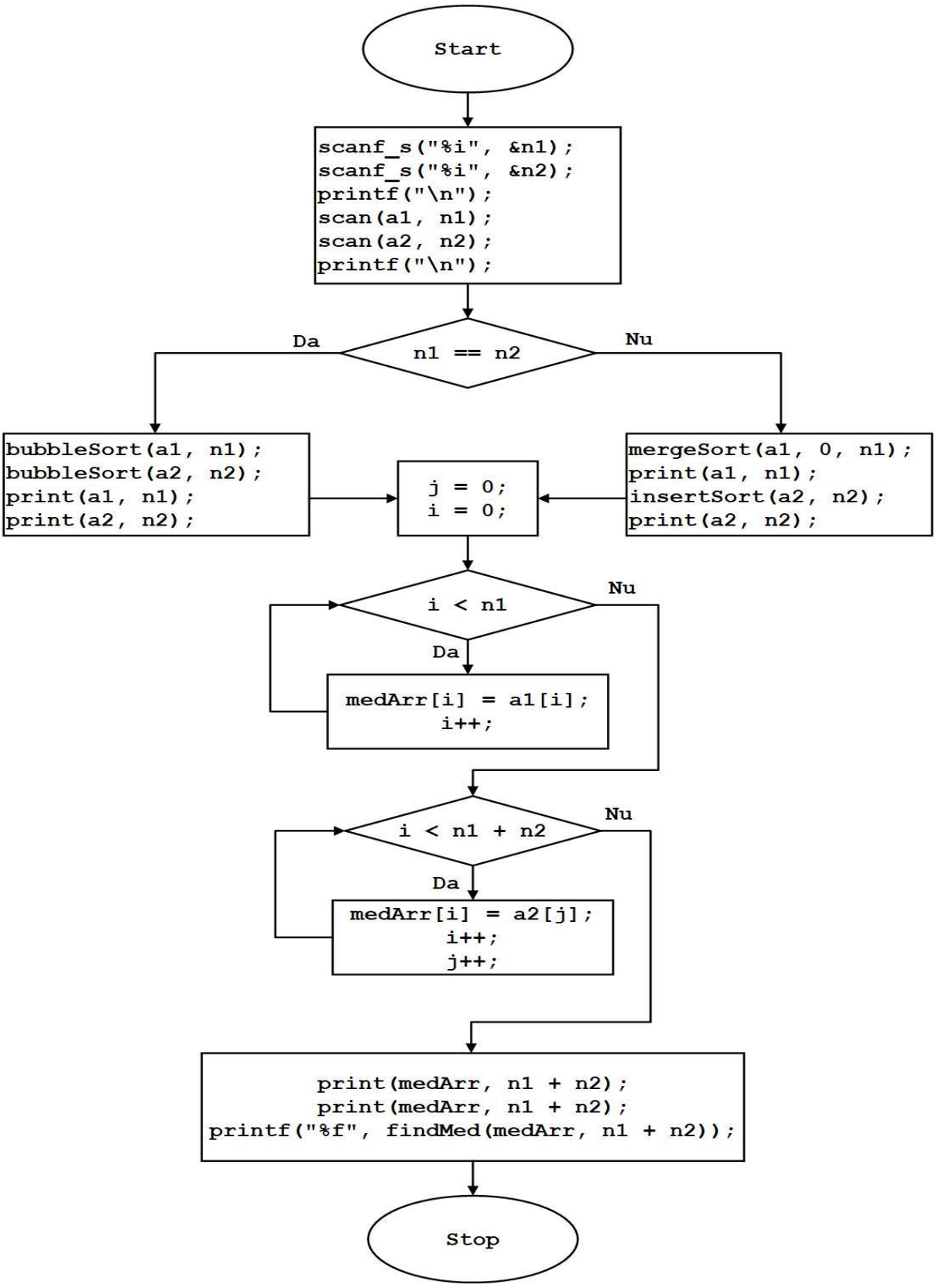

Figura 2.6 – Funcția ”*main()*”

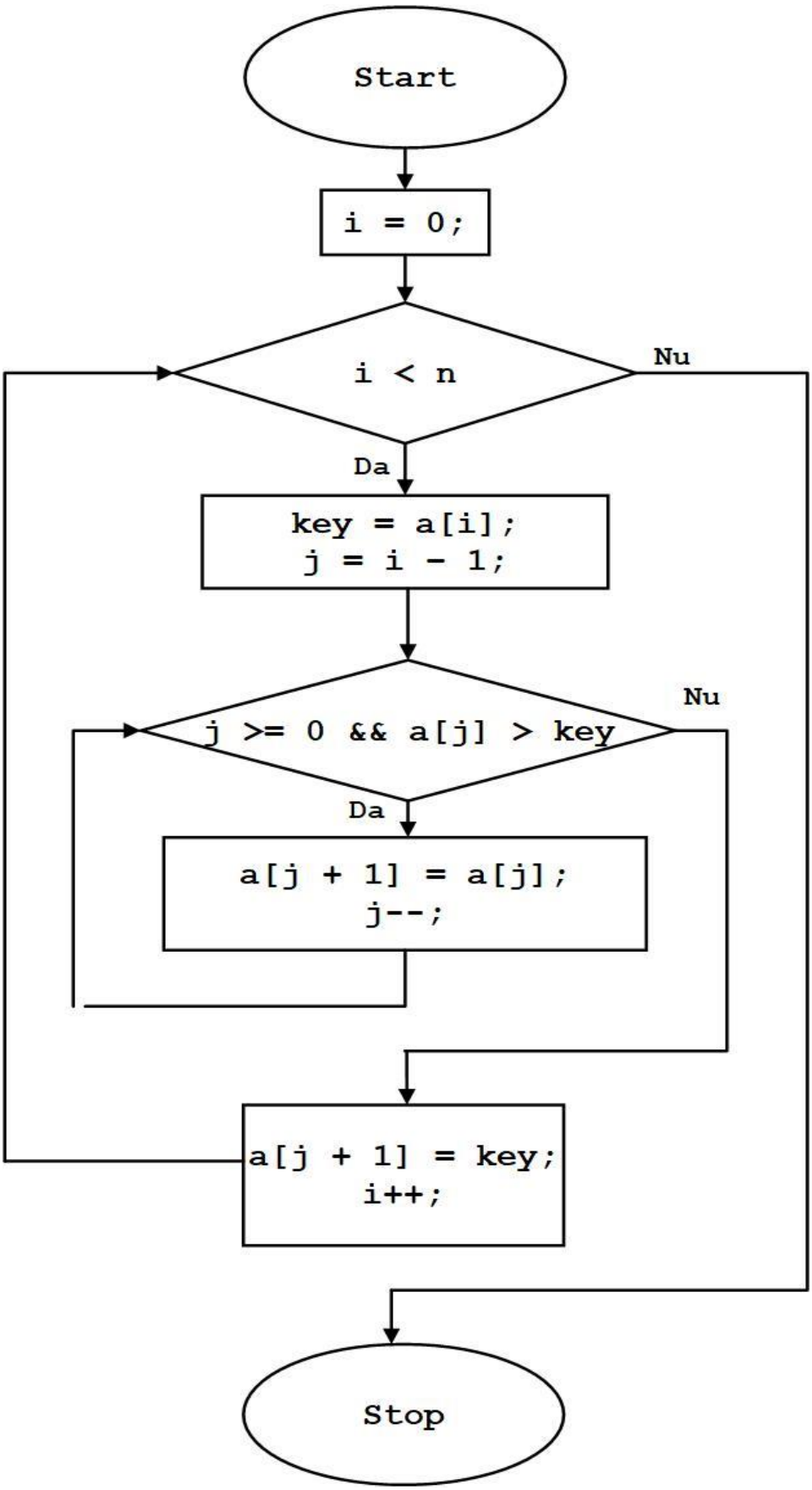

Figura 2.7 – Funcția ”*insertSort()*”

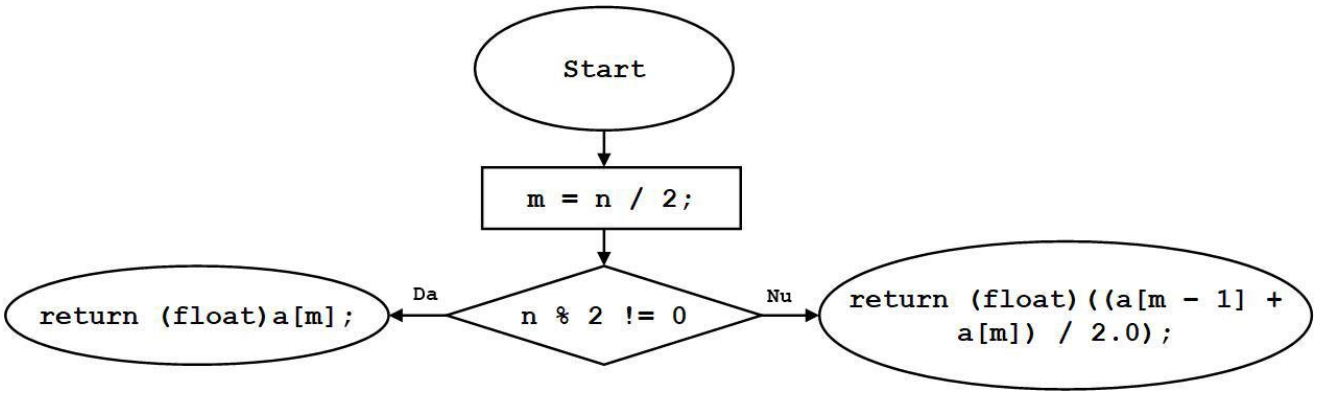

Figura 2.8 – Funcția ”*findMed()*”

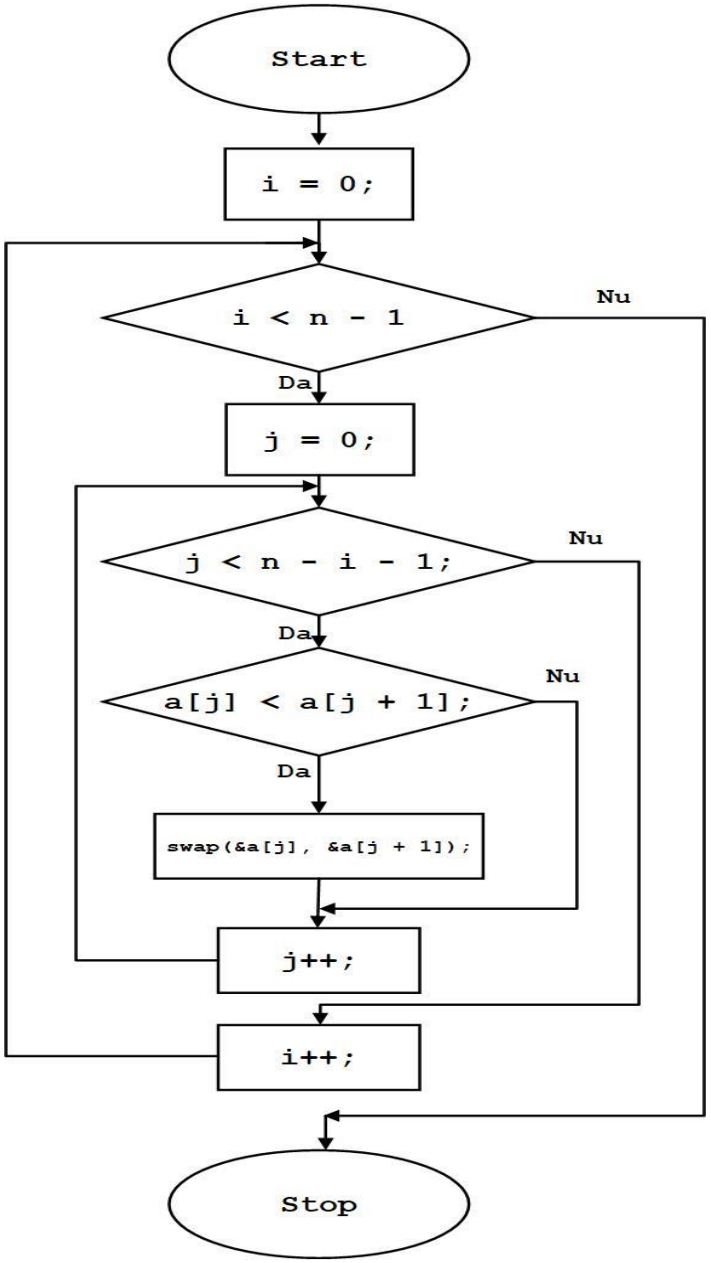

Figura 2.9 – Funcția ”*bubbleSort()*”

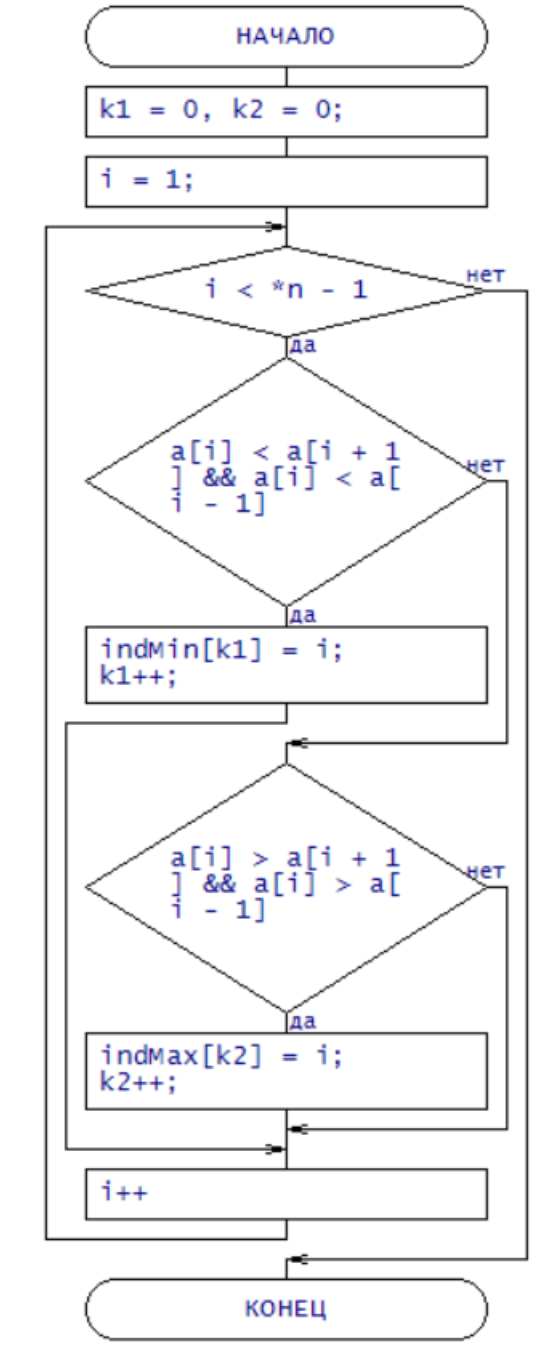

Figura 2.10 – Funcția ”*localExtrem()*”

Output prima versiune:

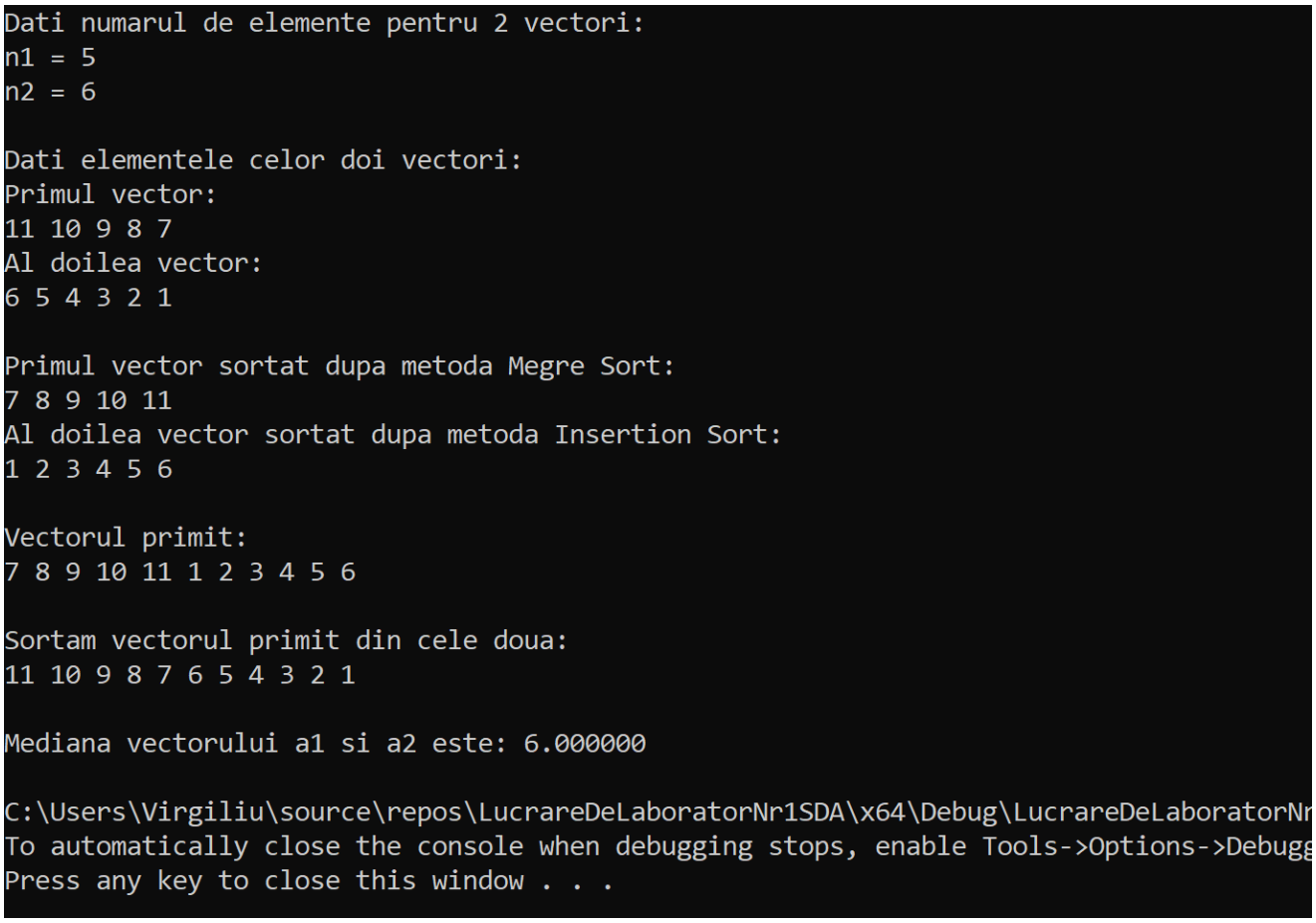

Figura 3.1 – Fereastra de output când n1 != n2

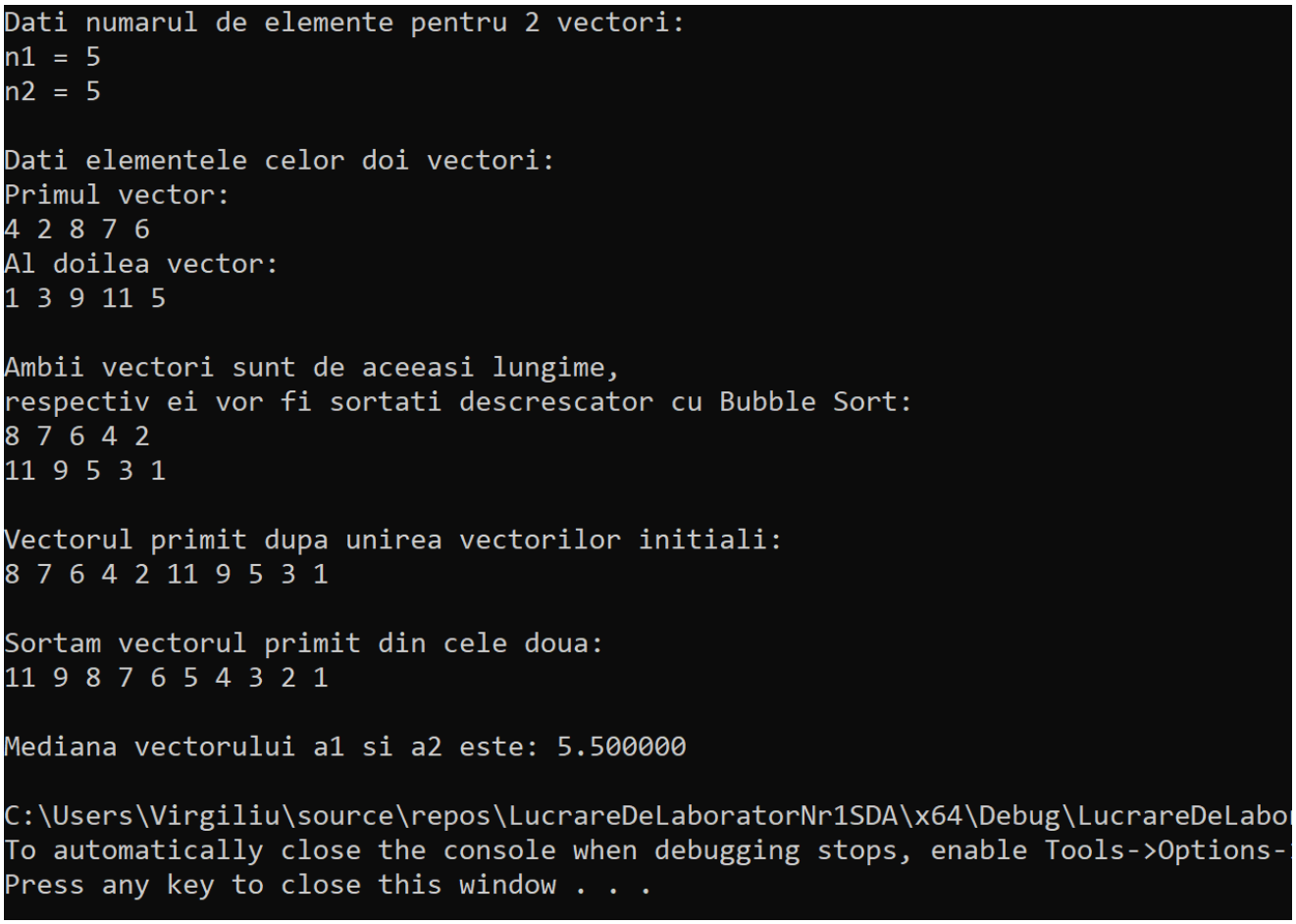

Figura 3.2 – Fereastra de output când n1 == n2

Output versiunea modificata:

```
Dati numarul de elemente pentru 2 vectori:
n1 = 5
n2 = 5

Dati elementele celor doi vectori:
Primul vector:
5 7 8 9 4

Al doilea vector:
2 3 4 7 5

Vectorul primit dupa unirea vectorilor initiali:
5 7 8 9 4 2 3 4 7 5

Indexii extremelor locale minime:
   5
Indexii extremelor locale maxime:
   3 8

Ambii vectori sunt de aceeasi lungime,
respectiv ei vor fi sortati descrescator cu Bubble Sort:
9 8 7 5 4
7 5 4 3 2

Sortam vectorul primit din cele doua cu Bubble Sort:
9 8 7 7 5 5 4 4 3 2

Mediana vectorului a1 si a2 este: 5.000000


C:\Users\Virgiliu\source\repos\LucrareDeLaboratorNr1SDA\x64\Debug\
To automatically close the console when debugging stops, enable To
Press any key to close this window . . .
```

Figura 3.3 − Output v. modificată

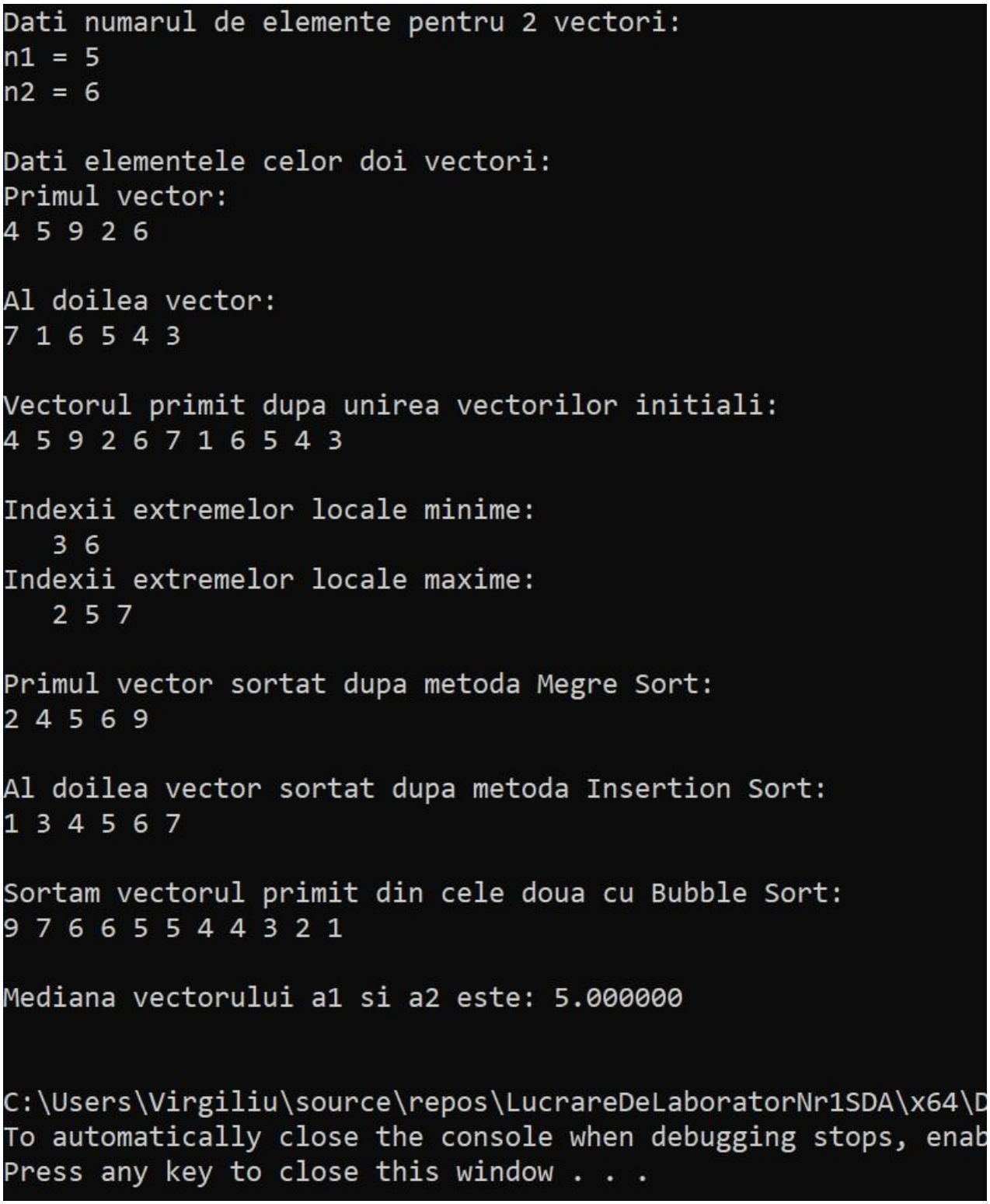

Figura 3.4 - Output v. modificată

Concluzie:

În timpul executării lucrării de laborator am aplicat cunostințele obținute la cursuri și seminare și anume am folosit diferite metode de sortare pentru a sorta un vector, începând cu cele ușoare cum sunt: Bubble Sort, Insertion Sort și terminând cu unul ceva mai complicat cum este Merge Sort. Am aflat ce este mediana unui vector și cum se calculează, am elaborat un algoritm care găsește extremele locale dintr-un vector. În general aș spune că acesta e un început destul de bun, având în vedere că prima lucrare are 27 pagini.