Laborator 4

```c
/*  Lab 4  Popa Larisa-Ancuta  Prob 1

-implementați metoda bulelor (Bubble-Sort) care folosește un indicator flag și
optimizează ciclul interior
-se cere atât scrierea funcției, cât și partea de program care face citirea și afișarea
șirului inițial și a celui ordonat

*/

#define  _CRT_SECURE_NO_WARNINGS
#include  <stdio.h>

#define DIM 100

void citire(int, int[DIM]); //citeste elementee tabloului
void bubble_sort(int, int[DIM]); //bubble sort cu flag
void afisare(int, int[DIM]); //afiseaza elementele tabloului

int main()
{
	int  n, tab[DIM];

	printf("Introduceti dimensiunea tabloului: ");
	scanf("%d", &n);

	citire(n, tab);

	printf("\nInainte de sortare: \n ");
	afisare(n, tab);

	bubble_sort(n, tab);

	printf("\n\nDupa  sortare: \n ");
	afisare(n, tab);

	printf("\n\n");

	return 0;
}

void citire(int n, int tab[DIM])
{
	int i;

	printf("\nIntroduceti elementele tabloului: \n");
	for (i = 0; i < n; i++)
		scanf("%d", &tab[i]);
}

void bubble_sort(int n, int tab[DIM])
{
	int  i, ok = 0, aux;

	while (ok == 0)
	{
		ok = 1;
```

```
		for (i = 0; i < n - 1; i++)
		{
			if (tab[i] > tab[i + 1])
			{
				aux = tab[i];
				tab[i] = tab[i + 1];
				tab[i + 1] = aux;
				ok = 0;
			}
		}
	}
}

void  afisare(int n,int tab[DIM])
{
	int i;

	for (i = 0; i < n; i++)
		printf("%d  ", tab[i]);
}
```

```
/* Lab 4  Popa Larisa-Ancuta  Prob 2

-modificați programul care exemplifică metoda de sortare rapidă (Quick-Sort) așa încât să
ordoneze șirul inițial
   în ordine descrescătoare

*/
#define  _CRT_SECURE_NO_WARNINGS
#include  <stdio.h>

#define DIM 100

void citire(int, int[DIM]); //citeste elementee tabloului
void quick_sort(int, int, int[DIM]); //sortare rapida in ordine descrescatoare
void afisare(int, int[DIM]); //afiseaza elementele tabloului

int main()
{
	int  n, tab[DIM];

	printf("Introduceti dimensiunea tabloului: ");
	scanf("%d", &n);

	citire(n, tab);

	printf("\nInainte de sortare: \n ");
	afisare(n, tab);

	quick_sort(0, n - 1, tab);

	printf("\n\nDupa  sortare: \n ");
	afisare(n, tab);

	printf("\n\n");
```

```
	return 0;
}

void citire(int n, int tab[DIM])
{
	int i;

	printf("\nIntroduceti elementele tabloului: \n");
	for (i = 0; i < n; i++)
		scanf("%d", &tab[i]);
}

void  quick_sort(int  prim, int  ultim, int tab[DIM])
{
	int  i, j, pivot, temp;

	i = prim;
	j = ultim;
	pivot = tab[ultim];

	do
	{
		while (tab[i]  >  pivot)
			i++;

		while (tab[j]  <  pivot)
			j--;

		if (i  <  j)
		{
			temp = tab[i];
			tab[i] = tab[j];
			tab[j] = temp;
		}

		if (i <= j)
		{
			j--;
			i++;
		}

	} while (i < j);

	if (prim < j)
		quick_sort(prim, j, tab);

	if (i < ultim)
		quick_sort(i, ultim, tab);
}

void  afisare(int n, int tab[DIM])
{
	int i;

	for (i = 0; i < n; i++)
		printf("%d  ", tab[i]);
}
```

```c
/* Lab 4  Popa Larisa-Ancuta  Prob 3

-folosiţi funcţiile de bibliotecă pentru sortări (qsort( )) pentru a aranja un şir de
înregistrări
   cu nume, prenume, cod numeric personal, data angajării după două câmpuri la alegere
   (un exemplu ar fi: crescător după nume şi descrescător după data angajării)

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <cstring>
#include <stdlib.h>

#define DIM 30
#define DIM2 100

struct data_ang
{
	int zi, luna, an;
};

struct angajat
{
	char nume[DIM], prenume[DIM];
	long int CNP;
	data_ang data;
};

void citire(int, struct angajat*); //citire date angajat
void afisare(int, struct angajat*); //afisare date angajat

int ord_ang(const struct angajat *ang1, const struct angajat *ang2); //ordnare crescator
dupa data angajarii si alfabetic dupa nume


int main()
{
	int n, i;
	angajat Ang[DIM2], *ang = &Ang[0];

	printf("\n Introduceti numarul de angajati: ");
	scanf("%d", &n);

	for (i = 0; i < n; i++)
		citire(i, ang);

	qsort((void*)ang, n, sizeof(struct angajat), (int(*)(const void*, const void
*))ord_ang);

	printf("\n\n   Ordonare: crescator data angajarii si alfabetic dup nume\n");
	for (i = 0; i < n; i++)
		afisare(i, ang);

	return 0;
}

void citire(int i, angajat *ang)
```

```
{

	printf("\n\nAngajat %d \n", i + 1);

	printf("\nNume: ");
	scanf("%s", (ang + i)->nume);

	printf("Prenume: ");
	scanf("%s", (ang + i)->prenume);

	printf("CNP: ");
	scanf("%d", &(ang + i)->CNP);

	printf("Data angajarii: \n");
	printf("  Zi: ");
	scanf("%d", &(ang + i)->data.zi);
	printf("  Luna: ");
	scanf("%d", &(ang + i)->data.luna);
	printf("  An: ");
	scanf("%d", &(ang + i)->data.an);
}

void afisare(int i, struct angajat *ang)
{
	printf("\nNume: %s", (ang + i)->nume);
	printf("\nPrenume: %s", (ang + i)->prenume);
	printf("\nCNP: %d", (ang + i)->CNP);
	printf("\nData angajarii: %d/%d/%d\n", (ang + i)->data.zi, (ang + i)->data.luna,
(ang + i)->data.an);
}

int ord_ang(const struct angajat *ang1, const struct angajat *ang2)
{
	if ((ang1->data).an > (ang2->data).an)
		return 1;
	else
		if ((ang1->data).an < (ang2->data.an))
			return -1;
		else {
			if ((ang1->data).luna >(ang2->data).luna)
				return 1;
			else if ((ang1->data).luna < (ang2->data).luna)
				return -1;
			else {
				if ((ang1->data).zi >(ang2->data).zi)
					return 1;
				else if ((ang1->data).zi < (ang2->data).zi)
					return -1;
				else
					if ((strcmp(ang1->nume, ang2->nume) >0))
						return 1;
					else if ((strcmp(ang1->nume, ang2->nume) < 0))
						return -1;

					return 0;
			}
		}
}
```

```
/* Lab 4  Popa Larisa-Ancuta  Prob 4

-scrieți o aplicație C/C++ in care plecând de la două tablouri (unidimensionale) de
numere naturale
-să se obțină un al treilea tablou care să conțină toate elementele tablourilor sursă
fără a se repeta,
   aranjate în ordine crescătoare

*/

#define  _CRT_SECURE_NO_WARNINGS
#include  <stdio.h>
#include  <stdlib.h>

#define DIM 100
#define DIM2 200

void citire(int&, int[DIM]); //citeste elementele tabloului
int  comparare(const  void  *arg1, const  void  *arg2); //comparare crescatoare
void  interclasare(int, int, int[DIM], int[DIM], int&, int[DIM2]); //interclaseaza cele
doua tablouri
void  elim_egale(int&, int[DIM2]); //elimina elementele care se repeta
void  afisare(int, int[DIM2]); //afiseaza elementele tabloului

int main()
{
	int n, m, a[DIM], b[DIM], p, c[DIM2];

	printf("\n   Primul tablou:\n ");
	citire(n, a);
	printf("\n   Al doilea tablou:\n");
	citire(m, b);

	qsort((int*)a, n, sizeof(int), comparare);
	qsort((int*)b, m, sizeof(int), comparare);

	interclasare(n, m, a, b, p, c);

	elim_egale(p, c);

	printf("\nAl treilea tablou:\n ");
	afisare(p, c);

	printf("\n\n");

	return 0;
}


void citire(int &n, int tab[DIM])
{
	int i;

	printf("Introduceti dimensiunea tabloului: ");
	scanf("%d", &n);

	printf("Introduceti elementele tabloului: \n ");
	for (i = 0; i < n; i++)
```

```
		scanf("%d", &tab[i]);
}

int  comparare(const  void  *arg1, const  void  *arg2)
{
	if (*(int  *)arg1<*(int  *)arg2)
		return  -1;
	if (*(int  *)arg1 == *(int  *)arg2)
		return  0;
	if (*(int  *)arg1>*(int  *)arg2)
		return  1;
}

void  interclasare(int n, int m, int a[DIM], int b[DIM], int &p, int c[DIM2])
{
	int  i = 0, j = 0, k;

	p = 0;

	while (i < n  &&  j < m)
	{
		if (a[i] < b[j])
			c[p++] = a[i++];
		else
			c[p++] = b[j++];
	}

	if (i < n)
	{
		for (k = i; k < n; k++)
			c[p++] = a[k];
	}

	if (j < m)
	{
		for (k = j; k < m; k++)
			c[p++] = b[k];
	}
}

void elim_egale(int &p, int c[DIM2])
{
	int i, j, k;

	for (i = 0;i < p - 1; i++)
	{
		for (j = i + 1;j < p; j++)
		{
			if (c[i] == c[j])
			{
				for (k = j; k < p - 1; k++)
					c[k] = c[k + 1];
				p--;
				j--;
			}
		}
	}
}
```

```
void  afisare(int n,int tab[DIM2])
{
	int i;

	for (i = 0; i < n; i++)
		printf("%d  ", tab[i]);
}
```

```
/* Lab 4  Popa Larisa-Ancuta  Prob 6

-cititi de la tastatura m elemente de tip intreg intr-un tablou unidimensional si o
valoare intreaga n<m
-impartiti tabloul citit in doua sub-tablori astfel:
   a) primul subtablou va contine primele n elemente din tabloul initial
   b) al doilea subtablou va contine restul elementelor din tablul initial.
-sa se realizeze urmatoarele operatii:
   1. sa se ordoneze crescator cele doua subtablouri
   2. sa se sorteze tabloul initial, prin interclasarea celor doua subtablori ordonate.
(merge-sort)

*/

#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>

#define DIM 100

void citire(int, int[DIM]); //citeste elementee tabloului
void afisare(int, int[DIM]); //afiseaza elementele taboului
void ordonare(int, int[DIM]);
void interclasare(int, int, int[DIM], int[DIM], int, int[DIM]);

int main()
{
	int  m, tab1[DIM], n, tab2[DIM], tab3[DIM], i, n2 = 0, n3 = 0, tab[DIM], m2 = 0;

	printf("Introduceti dimensiunea tabloului: ");
	scanf("%d", &m);

	citire(m, tab1);

	printf("\nIntroduceti o valoare intreaga: ");
	scanf("%d", &n);

	if (n >= m)
		printf("\n   Valoarea introdusa este prea mare!\n\n");
	else
	{
		for (i = 0;i < n;i++)
			tab2[n2++] = tab1[i];

		for (i = n;i < m;i++)
			tab3[n3++] = tab1[i];
```

```
			ordonare(n2, tab2);
			printf("\nSubtabloul 1: \n");
			afisare(n2, tab2);

			ordonare(n3, tab3);
			printf("\nSubtabloul 2: \n");
			afisare(n3, tab3);

			printf("\n\nSirul sortat prin interclasarea a celor 2 subsiruri: \n");
			interclasare(n2, n3, tab2, tab3, m2, tab);
			afisare(m, tab);
		}

		return 0;
}

void citire(int m, int tab[DIM])
{
		int i;

		printf("\nIntroduceti elementele tabloului: \n");
		for (i = 0; i < m; i++)
			scanf("%d", &tab[i]);
}

void afisare(int n, int tab[DIM])
{
		int i;

		for (i = 0;i<n;i++)
			printf("%d  ", tab[i]);
}

void ordonare(int n, int tab[DIM])
{
		int i, j;

		for (i = 0;i < n;i++)
		{
			for (j = i + 1;j < n - 1;j++)
			{
				if (tab[i] > tab[j])
				{
					int aux;
					aux = tab[i];
					tab[i] = tab[j];
					tab[j] = aux;
				}
			}
		}
}

void interclasare(int n2, int n3, int tab2[DIM], int tab3[DIM], int m2,int tab[DIM])
{
		int i = 0, j = 0;

		while (i <= n2 - 1 && j <= n3 - 1)
		{
```

```
		if (tab2[i]<tab3[j])
			tab[m2++] = tab2[i++];
		else
			tab[m2++] = tab3[j++];
	}

	while (i <= n2 - 1)
		tab[m2++] = tab2[i++];

	while (j <= n3 - 1)
		tab[m2++] = tab3[j++];
}
```

```
/*  Lab 4  Popa Larisa-Ancuta  Prob 7

-să se scrie un program care permite sortarea unui stoc de calculatoare
-acestea să se reprezinte în program ca o structură formată din caracteristicile
calculatoarelor (nume (sir caractere),
  tip de procesor (sir caractere), frecventa de tact (long int), dimensiunea memoriei RAM
(int), preț (float)
-sortarea se va face, la alegerea utilizatorului, după: pret, memorie, tact sau tip de
procesor
-folosiți, de preferință, funcția de bibliotecă pentru sortări qsort( ) sau o altă metodă
la alegere
-sortați apoi considerând un câmp sir de caractere si unul numeric
-afișați rezultatele

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <stdlib.h>

#define DIM 50
#define DIM2 100

struct calculator
{
	char nume[DIM], procesor[DIM];
	long int frecventa;
	int RAM;
	float pret;
};

void citire(int, struct calculator *); //citeste specificatiile calculatorului
void afisare(int, struct calculator *); //afiseaza specificatiile calculatorului

int ord_pret(const struct calculator *c1, const struct calculator *c2); //ordonare
crescatoare dupa pret
int ord_RAM(const struct calculator *c1, const struct calculator *c2); //ordonare
crescatoare dupa dimensiunea memoriei RAM
int ord_procesor(const struct calculator *c1, const struct calculator *c2); //ordonare
alfabetica dupa tipul procesorului
```

```
int main()
{
	int n, i, optiune;
	calculator C[DIM2], *c = &C[0];

	printf("\nIntroduceti numarul de calculatoare: ");
	scanf("%d", &n);

	for (i = 0; i < n; i++)
		citire(i, c);

	printf("\n\nOrdonare\n 1.Pret cresctor\n 2.Dimensiune memorie crescator\n 3.Tip
procesor alfabetic\n  Alegeti optiunea dorita: ");
	scanf("%d", &optiune);

	do
	{
		if (optiune == 1)
		{
			qsort((calculator *)c, n, sizeof(struct calculator), (int(*)(const
void*, const void*))ord_pret);

			for (i = 0; i < n; i++)
				afisare(i, c);

			printf("\n\n");
		}
		else if (optiune == 2)
		{
			qsort((calculator *)c, n, sizeof(struct calculator), (int(*)(const
void*, const void*))ord_RAM);

			for (i = 0; i < n; i++)
				afisare(i, c);

			printf("\n\n");
		}
		else if (optiune == 3)
		{
			qsort((calculator *)c, n, sizeof(struct calculator), (int(*)(const
void*, const void*))ord_procesor);

			for (i = 0; i < n; i++)
				afisare(i, c);

			printf("\n\n");
		}
		else printf("\nNu exista optiunea introdusa!\n\n");

		printf("\nOrdonare\n 1.Pret cresctor\n 2.Dimensiune memorie crescator\n
3.Tip procesor alfabetic\n  Alegeti optiunea dorita(0-iesire): ");
		scanf("%d", &optiune);

	} while (optiune != 0);

	return 0;
}
```

```
void citire(int i, struct calculator *c)
{
	printf("\nCalculatorul %d \n", i + 1);

	printf("Nume: ");
	scanf("%s", (c + i)->nume);

	printf("Tip procesor: ");
	scanf("%s", (c + i)->procesor);

	printf("Frecventa(intreg): ");
	scanf("%d", &(c + i)->frecventa);

	printf("Memoria RAM(intreg): ");
	scanf("%d", &(c + i)->RAM);

	printf("Pret(real): ");
	scanf("%f", &(c + i)->pret);
}

void afisare(int i, struct calculator *c)
{
	printf("\nNume: %s", (c + i)->nume);
	printf("\nTip de procesor: %s", (c + i)->procesor);
	printf("\nFrecventa: %d", (c + i)->frecventa);
	printf("\nMemorie RAM: %d", (c + i)->RAM);
	printf("\nPret: %0.2f\n", (c + i)->pret);
}

int ord_pret(const struct calculator *c1, const struct calculator *c2)
{
	if ((c1->pret) > (c2->pret))
		return 1;
	else
	{
		if ((c1->pret) < (c2->pret))
			return -1;
	}

	return 0;
}

int ord_RAM(const struct calculator *c1, const struct calculator *c2)
{
	if ((c1->RAM) > (c2->RAM))
		return 1;
	else
	{
		if ((c1->RAM) < (c2->RAM))
			return -1;
	}

	return 0;
}

int ord_procesor(const struct calculator *c1, const struct calculator *c2)
{
	if ((strcmp(c1->procesor, c2->procesor) < 0))
```

```
			return -1;
		else
		{
			if ((strcmp(c1->procesor, c2->procesor) > 0))
				return 1;
		}

		return 0;
}
```

```
/*  Lab 4  Popa Larisa-Ancuta  Prob 8

-preluați din două fișiere doua tablouri unidimensionale ce conțin valori reale
-generați un al treilea tablou care să conțină toate valorile din cele doua tablouri
  și toate valorile obținute prin medierea valorilor de pe aceeași poziție din cele doua
tablouri inițiale
-dacă numărul de elemente ale tablourilor diferă, media se va face considerând valoarea
0.0 pentru elementele lipsă
-ordonați al treilea tablou
-numărați câte valori neunice sunt în șir

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

#define DIM 100
#define DIM2 300

void ordonare(int, float[DIM2]); //ordoneaza crescator elementele tabloului
void val_neunice(int, float[DIM2]); //afiseaza numarul de valori neunice

int main()
{
		char fisier1[] = "Fisier1.txt", fisier2[] = "Fisier2.txt";
		FILE *fis1, *fis2;

		int n = 0, m = 0, i, p = 0, j;
		float tab1[DIM], tab2[DIM], v1, v2, tab3[DIM2];

		if (!(fis1 = fopen(fisier1, "r")))
			printf("\nEroare la deschiderea fisierului!\n\n");
		else
		{
			while (!feof(fis1))
			{

				fscanf(fis1, "%f", &v1);
				tab1[n++]=v1;
			}
		}

		if (!(fis2 = fopen(fisier2, "r")))
			printf("\nEroare la deschiderea fisierului!\n\n");
		else
		{
```

```
        while (!(feof(fis2)))
        {
                fscanf(fis2, "%f", &v2);
                tab2[m++] = v2;
        }
}

for (i = 0;i<n;i++)
{
        tab3[p] = tab1[i];
        p++;
}

for (i = 0;i<m;i++)
{
        tab3[p] = tab2[i];
        p++;
}

if (n == m)
{
        for (i = 0;i < n;i++)
        {
                tab3[p] = (tab1[i] + tab2[i]) / 2;
                p++;
        }
}
else
{
        i = 0;
        j = 0;

        while (i < n && j < m)
        {
                tab3[p]= (tab1[i] + tab2[i]) / 2;
                p++;
                i++;
                j++;
        }
        while (i < n)
        {
                tab3[p] = tab1[i] / 2;
                p++;
                i++;
        }
        while (j < m)
        {
                tab3[p] = tab2[j] / 2;
                p++;
                j++;
        }
}

ordonare(p, tab3);

val_neunice(p, tab3);

fclose(fis1);
```

```
	fclose(fis2);

	return 0;
}

void ordonare(int p,float tab3[DIM2])
{
	int i, j;

	for (i = 0; i < p - 1; i++)
	{
		for (j = i + 1; j < p; j++)
			if (tab3[i] > tab3[j])
			{
				float aux = tab3[i];
				tab3[i] = tab3[j];
				tab3[j] = aux;
			}
	}

	printf("\nTabloul 3 ordonat: \n");
	for (i = 0;i < p;i++)
		printf("%0.2f ", tab3[i]);
	printf("\n\n");
}

void val_neunice(int p, float tab3[DIM2])
{
	int nr = 0, i = 0, poz, j;

	while (i < p)
	{
		poz = 1;

		for (j = i + 1; j < p;j++)
		{
			if (tab3[i] == tab3[j])
				poz++;
		}

		if (poz != 1)
			nr++;

		i += poz;
	}

	printf("Sunt %d valori neunice in sir.\n\n", nr);
}
```

```
/* Lab 4  Popa Larisa-Ancuta  Prob 9

-generati in mod aleatoriu un tablou de maxim 200 valori intregi, valori nu mai mari de
100
-determinati si afisati valoarea minima, mediana si maxima generata, sortand elementele
printr-o metoda la alegere
```

```
-determinati valoarea medie si comparati aceasta valoare cu cea mediana (afisati
diferenta)
-verificati daca valoarea medie este in tabloul initial generat

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define DIM 200
#define MAX 100

void ord_cresc(int[DIM]);
int comp(int *, int *);
void afisare(int[DIM]);

int main()
{
	srand(time(NULL));

	int tab[DIM], i, sum = 0;
	float med;

	for (i = 0; i < DIM; i++)
	{
		tab[i] = rand() % MAX;
		sum += tab[i];
	}

	ord_cresc(tab);

	printf("\nValoarea minima generata: %d", tab[0]);
	printf("\nValoarea mediana generata: %d", tab[99]);
	printf("\nValoarea maxima generata: %d", tab[199]);

	med = (float)sum / DIM;
	printf("\nValoarea medie: %0.2f", med);

	printf("\n\nDiferenta dintre val. mediana si val. medie este: %0.2f", tab[99] -
med);

	int *caut = (int *)bsearch(&med, tab, DIM, sizeof(int), (int(*) (const void *,
const void *)) comp);

	if (caut == 0)
		printf("\n\nValoarea medie nu se afla in tabloul initial generat.\n");
	else
		printf("\n\nValoarea medie se afla in tabloul initial generat.\n");


	printf("\nTabloul sortat:\n ");
	afisare(tab);

	return 0;
}
```

```
void ord_cresc(int tab[DIM])
{
	int i, j;
	for (i = 0; i < DIM - 1; i++)
	{
		for (j = i + 1; j < DIM; j++)
		{
			if (tab[i] > tab[j])
			{
				int aux = tab[i];
				tab[i] = tab[j];
				tab[j] = aux;
			}
		}
	}
}

int comp(int *nr1, int *nr2)
{
	return(*nr1 - *nr2);
}

void afisare(int tab[DIM])
{
	int i;

	for (i = 0; i < DIM; i++)
		printf("%d ", tab[i]);
}
```

```
/* Lab4  Popa Larisa-Ancuta  Prob 10

-generati printr-un mecanism aleatoriu un tablou de maxim 200 de valori reale
  (prin doua tablouri de aceiași dimensiune, primul fiind partea intreaga (nu mai mare de
100)
  al doilea partea fractionara (limitata la 20 ca intreg ce devine .20 fractionar)
-tabloul real fiind obtinut prin combinarea partii intregi și fractionare
-afișati tablourile generate, cel real obtinut
-sortati folosind functia qsort( ) tabloul real si afisati rezultatul obtinut

*/
#define  _CRT_SECURE_NO_WARNINGS
#include  <stdio.h>
#include  <stdlib.h>
#include  <time.h>

#define DIM 200
#define MAX 100
#define MIN 20

void  afisare_int(int, int[DIM]);
void afisare_real(int, float[DIM]);
int  comp(const  void  *arg1, const  void  *arg2);

int main()
{
```

```
	int n, tab1[DIM], tab2[DIM], i;
	float tab3[DIM];

	srand(time(NULL));

	printf("\nIntroduceti dimensiunea tabloului generat: ");
	scanf("%d", &n);

	for (i = 0;i < n;i++)
	{
		tab1[i] = rand() % MAX;
		tab2[i] = rand() % MIN;

		tab3[i] = (float)tab1[i] + (float)100 / tab2[i];
	}

	printf("\nTabloul 1: \n");
	afisare_int(n, tab1);

	printf("\n\nTabloul 2: \n");
	afisare_int(n, tab2);

	printf("\n\nTabloul obtinut: \n");
	afisare_real(n, tab3);

	qsort((float*)tab3, n, sizeof(float), comp);
	printf("\n\nTabloul obtinut sortat: \n");
	afisare_real(n, tab3);

	printf("\n\n");
	return 0;
}

void  afisare_int(int n, int tab[DIM])
{
	int i;

	for (i = 0; i < n; i++)
		printf("%d  ", tab[i]);
}

void afisare_real(int n , float tab[DIM])
{
	int i;

	for (i = 0; i < n; i++)
		printf("%0.2f  ", tab[i]);
}

int  comp(const  void  *arg1, const  void  *arg2)
{
	if (*(float  *)arg1<*(float  *)arg2)
		return  -1;
	if (*(float  *)arg1 == *(float  *)arg2)
		return  0;
	if (*(float  *)arg1>*(float  *)arg2)
		return  1;
}
```

```c
/* Lab 4  Popa Larisa-Ancuta  Prob 11

-alocati in mod dinamic un tablou de n numere intregi, care vor fi citite si afisate
-cititi o valoare cheie de la intrarea standard
-folosind functia _lfind( ) cautati si afisati toate pozitiile in care aceasta cheie se
gaseste in tabloul citit
-tratati cazul in care sunt valori multiple, sau valoarea nu e in tablou
-folosind functia qsort( ) sortati apoi acest tablou, pe care il afisati
-cautati folosind functia bsearch() aceiasi valoare in tabloul sortat si afisati pozitia
ei

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <search.h>

void citire(int, int *);
void afisare(int, int *);

int comparare1(int*, int*);
int comparare2(const int *, const int *);


int main()
{
	int n, *tab = NULL, cheie;

	printf("\nIntroduceti numarul de elemente ale tabloului: ");
	scanf("%d", &n);

	if (!(tab = (int*)malloc(n * sizeof(int))))
		printf("\nAlocare nereusita!");
	else
	{
		citire(n, tab);

		printf("\nIntroduceti cheia: ");
		scanf("%d", &cheie);

		int *rez1 = (int *)_lfind(&cheie, tab, (size_t*)&n, sizeof(int),
(int(*)(const void*, const void*))comparare1);

		if (rez1 == 0)
			printf("\n(lfind)Numarul %d nu se gaseste in tablou!\n", cheie);
		else
			printf("\n(lfind)Numarul %d se gaseste in tablou pe pozitia : %d
\n", cheie, rez1 - tab + 1);

		printf("\nTabloul ordonat: \n");
		qsort((void*)tab, n, sizeof(int), (int(*)(const void*, const void
*))comparare2);
		afisare(n, tab);

		int *rez2 = (int*)bsearch(&cheie, tab, n, sizeof(int*), (int(*)(const
void*, const void*))comparare1);
```

```
		if (rez2 == 0)
			printf("(bsearch)Numarul %d nu se gaseste in tablou!\n\n", cheie);
		else
			printf("(bsearch)Numarul %d se gaseste in tablou pe pozitia : %d
\n\n", cheie, rez2 - tab + 1);

		if (tab)
			free(tab);
	}

	return 0;
}

void citire(int n, int *tab)
{
	int i;

	printf("\nIntroduceti elementele tabloului: \n");
	for (i = 0;i < n;i++)
		scanf("%d", (tab + i));
}

void afisare(int n, int *tab)
{
	int i;

	for (i = 0;i < n;i++)
		printf("%d  ", *(tab + i));

	printf("\n\n");
}

int comparare1(int *nr1, int *nr2)
{
	return (*nr1 - *nr2);
}

int comparare2(const int *nr1, const int *nr2)
{
	if (*nr1 > *nr2)
		return 1;
	else
		return -1;

	return 0;
}
```