Tehnici de sortare

```
int getNextGap(int gap){
    // Shrink gap by Shrink factor
    gap = (gap*10)/13;
    if (gap < 1)
        return 1;
    return gap;
}
void combSort(int a[], int n){
        int gap=n;
        bool swapped=true;
        while(gap !=1 || swapped == true){
                gap=getNextGap(gap);
        swapped=false;
        for(int i=0; i<n-gap; i++){
                if(a[i]>a[i+gap]){
                    swap(a[i], a[i+gap]);
        swapped=true;
        }}}
```

bubble sort

```
int n, v[100];
    //citire v[] cu n elemente
    bool sortat;
    do{
            sortat = true;
    for(int i = 0 ; i < n - 1 ; i ++)
        if(v[i] > v[i+1]){
            int aux = v[i];
            v[i] = v[i+1];
            v[i+1] = aux;
            sortat = false;
            }
        }
        while(!sortat);
```

Program Fără pointeri - Bubble Sort / Comb Sort

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

```
int *alocare_memorie(int dimensiune)
{
    int *tab = (int*)malloc(dimensiune * sizeof(int));
    if(tab == NULL)
```

```
    {
        printf("Eroare! Nu s-a putut aloca memorie pentru tablou.\n");
        exit(0);
    }
    return tab;
}
void introducere_tastatura(int *tab, int dimensiune)
{
    printf("Introduceti valorile tabloului:\n");
    for(int i = 0; i < dimensiune; i++)
    {
        scanf("%d", &tab[i]);
    }
}
void completare_random(int *tab, int dimensiune)
{
    srand(time(NULL));
    for(int i = 0; i < dimensiune; i++)
    {
        tab[i] = rand() % 100;
    }
}
void bubble_sort(int tab[], int dimensiune)
{
    int i, j, temp;
    for(i = 0; i < dimensiune-1; i++)
    {
        for(j = 0; j < dimensiune-i-1; j++)
        {
            if(tab[j] > tab[j+1])
            {
                temp = tab[j];
                tab[j] = tab[j+1];
                tab[j+1] = temp;
            }
        }
    }
}
void comb_sort(int tab[], int dimensiune)
{
    int gap = dimensiune;
    int i, j, temp;
    int swapped = 1;
    while(gap > 1 || swapped)
    {
        gap = gap * 10 / 13;
        if(gap < 1) gap = 1;
        swapped = 0;
        for(i = 0, j = gap; j < dimensiune; i++, j++)
        {
            if(tab[i] > tab[j])
            {
                temp = tab[i];
                tab[i] = tab[j];
                tab[j] = temp;
                swapped = 1;
            }
        }
    }
```

```
}
void eliberare_memorie(int *tab)
{
    free(tab);
    printf("Memoria a fost eliberata cu succes.\n");
    exit(0);
}
void afisare_meniu(int *tab, int dimensiune)
{
    int optiune;
    printf("\n1. Alocare dinamica a memoriei\n");
    printf("2. Introducere valorilor tabloului de la tastatura\n");
    printf("3. Completare tablou cu valori random\n");
    printf("4. Sortare prin Bubble Sort\n");
    printf("5. Sortare prin Comb Sort\n");
    printf("6. Eliberare memoria si iesire din program\n");
    printf("7. Afisarea tabloului\n");
printf("Introduceti optiunea: ");
scanf("%d", &optiune);
switch(optiune)
{
    case 1:
        printf("Introduceti dimensiunea tabloului: ");
        scanf("%d", &dimensiune);
        tab = alocare_memorie(dimensiune);
        printf("Memorie alocata cu succes pentru un tablou cu %d elemente.\n",
dimensiune);
        break;
    case 2:
        if(tab == NULL)
        {
            printf("Eroare! Tabloul nu a fost alocat.\n");
        }
        else
        {
            introducere_tastatura(tab, dimensiune);
            printf("Valorile au fost introduse cu succes.\n");
        }
        break;
    case 3:
        if(tab == NULL)
        {
            printf("Eroare! Tabloul nu a fost alocat.\n");
        }
        else
        {
            completare_random(tab, dimensiune);
            printf("Valorile au fost completate cu succes.\n");
        }
        break;
    case 4:
        if(tab == NULL)
        {
            printf("Eroare! Tabloul nu a fost alocat.\n");
        }
        else
        {
            bubble_sort(tab, dimensiune);
            printf("Valorile au fost sortate cu succes prin Bubble Sort.\n");
```

```
        }
        break;
    case 5:
        if(tab == NULL)
        {
            printf("Eroare! Tabloul nu a fost alocat.\n");
        }
        else
        {
            comb_sort(tab, dimensiune);
            printf("Valorile au fost sortate cu succes prin Comb Sort.\n");
        }
        break;
    case 6:
        if(tab == NULL)
        {
            printf("Eroare! Tabloul nu a fost alocat.\n");
        }
        else
        {
            eliberare_memorie(tab);
        }
        break;
    case 7:
    if(tab == NULL)
    {
        printf("Eroare! Tabloul nu a fost alocat.\n");
    }
    else
    {
        printf("Valorile din tablou sunt: ");
        for(int i = 0; i < dimensiune; i++)
        {
            printf("%d ", tab[i]);
        }
        printf("\n");
    }
    break;
    default:
        printf("Optiunea selectata nu exista.\n");
        break;
      }
      afisare_meniu(tab, dimensiune);
}
int main() {
      int *tab = NULL;
      int dimensiune = 0;
      afisare_meniu(tab, dimensiune);
      return 0;
}
```

Program Cu pointeri - Bubble Sort / Comb Sort

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

```
int *alocare_memorie(int dimensiune)
{
```

```
    int *tab = (int*)malloc(dimensiune * sizeof(int));
    if(tab == NULL)
    {
        printf("Eroare! Nu s-a putut aloca memorie pentru tablou.\n");
        exit(0);
    }
    return tab;
}
void introducere_tastatura(int *tab, int dimensiune)
{
    printf("Introduceti valorile tabloului:\n");
    for(int i = 0; i < dimensiune; i++)
    {
        scanf("%d", &tab[i]);
    }
}
void completare_random(int *tab, int dimensiune)
{
    srand(time(NULL));
    for(int i = 0; i < dimensiune; i++)
    {
        tab[i] = rand() % 100;
    }
}
void bubble_sort(int *tab, int dimensiune)
{
    int i, j, temp;
    for(i = 0; i < dimensiune-1; i++)
    {
        for(j = 0; j < dimensiune-i-1; j++)
        {
            if(*(tab+j) > *(tab+j+1))
            {
                temp = *(tab+j);
                *(tab+j) = *(tab+j+1);
                *(tab+j+1) = temp;
            }
        }
    }
}
void comb_sort(int *tab, int dimensiune)
{
    int gap = dimensiune;
    int i, j, temp;
    int swapped = 1;
    while(gap > 1 || swapped)
    {
        gap = gap * 10 / 13;
        if(gap < 1) gap = 1;

        swapped = 0;
        for(i = 0, j = gap; j < dimensiune; i++, j++)
        {
            if(*(tab+i) > *(tab+j))
            {
                temp = *(tab+i);
                *(tab+i) = *(tab+j);
                *(tab+j) = temp;
                swapped = 1;
```

```
            }
        }
    }
}
void eliberare_memorie(int *tab)
{
    free(tab);
    printf("Memoria a fost eliberata cu succes.\n");
    exit(0);
}
void afisare_meniu(int *tab, int dimensiune)
{
    int optiune;
    printf("\n1. Alocare dinamica a memoriei\n");
    printf("2. Introducere valorilor tabloului de la tastatura\n");
    printf("3. Completare tablou cu valori random\n");
    printf("4. Sortare prin Bubble Sort\n");
    printf("5. Sortare prin Comb Sort\n");
    printf("6. Eliberare memoria si iesire din program\n");
    printf("7. Afisarea tabloului\n");
printf("Introduceti optiunea: ");
scanf("%d", &optiune);
switch(optiune)
{
    case 1:
        printf("Introduceti dimensiunea tabloului: ");
        scanf("%d", &dimensiune);
        tab = alocare_memorie(dimensiune);
        printf("Memorie alocata cu succes pentru un tablou cu %d elemente.\n",
dimensiune);
        break;
    case 2:
        if(tab == NULL)
        {
            printf("Eroare! Tabloul nu a fost alocat.\n");
        }
        else
        {
            introducere_tastatura(tab, dimensiune);
            printf("Valorile au fost introduse cu succes.\n");
        }
        break;
    case 3:
        if(tab == NULL)
        {
            printf("Eroare! Tabloul nu a fost alocat.\n");
        }
        else
        {
            completare_random(tab, dimensiune);
            printf("Valorile au fost completate cu succes.\n");
        }
        break;
    case 4:
        if(tab == NULL)
        {
            printf("Eroare! Tabloul nu a fost alocat.\n");
        }
        else
```

```
        {
            bubble_sort(tab, dimensiune);
            printf("Valorile au fost sortate cu succes prin Bubble Sort.\n");
        }
        break;
    case 5:
        if(tab == NULL)
        {
            printf("Eroare! Tabloul nu a fost alocat.\n");
        }
        else
        {
            comb_sort(tab, dimensiune);
            printf("Valorile au fost sortate cu succes prin Comb Sort.\n");
        }
        break;
    case 6:
        if(tab == NULL)
        {
            printf("Eroare! Tabloul nu a fost alocat.\n");
        }
        else
        {
            eliberare_memorie(tab);
        }
        break;
    case 7:
    if(tab == NULL)
    {
        printf("Eroare! Tabloul nu a fost alocat.\n");
    }
    else
    {
        printf("Valorile din tablou sunt: ");
        for(int i = 0; i < dimensiune; i++)
        {
            printf("%d ", tab[i]);
        }
        printf("\n");
    }
    break;
    default:
        printf("Optiunea selectata nu exista.\n");
        break;
      }
      afisare_meniu(tab, dimensiune);
}
int main() {
      int *tab = NULL;
      int dimensiune = 0;
      afisare_meniu(tab, dimensiune);
      return 0;
}
```