

Laborator 12. Rezolvări și explicații

Se citește un număr întreg n după care se citesc n propoziții (fiecare terminată cu tasta Enter). Să se sorteze aceste propoziții (sortare de șiruri de caractere).

Rezolvare: Pentru $n=3$ propoziții, iar propozițiile introduse de la tastatură fiind:

Ana are mere.
Ion are prune.
Alina are alune.

Fiecare propoziție introdusă reprezintă un șir de caractere, iar pentru a stoca în memorie cele 3 propoziții, vom folosi un șir de șiruri de caractere.



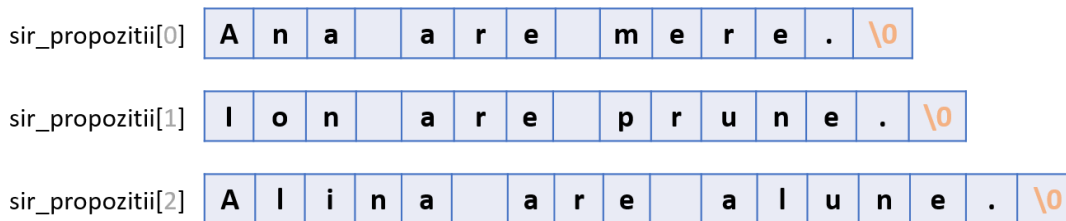
Trebuie să facem o distincție între următoarele definiții de variabile:

```
char propozitie[80]; //un singur șir de caractere (propoziție)
```

și

```
char sir_propozitii[25][80]; //șir ce stochează mai multe șiruri de caractere
```

```
char *sir_propozitii[25]; //șir ce stochează mai multe șiruri de caractere
```



Caracterul null \0 indică sfârșitul șirului de caractere

În continuare, pentru soluționarea problemei, vom furniza două metode de rezolvare: metoda fără alocare dinamică, respectiv metoda cu alocare dinamică a șirului de propoziții.

Metoda 1 – fără alocare dinamică

În implementarea soluției problemei, pentru a stoca șirul de propoziții vom defini o variabilă:

```
char sir_propozitii[25][80];
```

Pentru a citi n propoziții de la intrarea standard, definim o funcție numită *citeste_propozitii()* cu două argumente: șirul de propoziții și numărul n de propoziții. În procesul de citire a fiecărei propoziții (șir de caractere), folosim funcția predefinită *gets()* a librăriei C. Fiecare propoziție citită o stocăm pe poziția i în șirul de propoziții. Prin instrucțiunea *fflush(stdin);* golim buffer-ul de ieșire pentru a citi următorul șir de caractere (adică următoarea propoziție).

```
void citeste_propozitii(char sir_propozitii[25][80], int n)
{
    int i;
    puts("Introduceti propozitiile, apasand tasta enter dupa fiecare:");
    fflush(stdin);
    for (i=0; i<n; i++)
        gets(sir_propozitii[i]);
}
```

Sintaxa funcției *gets()* este:

```
char *gets(char *str);
```

unde parametrul **str** reprezintă pointerul către șirul de caractere în care șirul C e stocat.

Funcția *gets()* citește câte o linie de la intrarea standard (stdin) și stochează această linie în șirul de caractere pointat de **str**.

Pentru a sorta lexicografic propozițiile stocate în șirul de propoziții definim o funcție numită *sorteaza_propozitii()* cu două argumente: șirul de propoziții și dimensiunea n asociată. Parcurgem șirul de propoziții cu ajutorul a doi indecși i și j , comparând propoziția i cu propoziția j din șir. Dacă șirul de caractere asociat propoziției i este mai mare decât șirul de caractere asociat propoziției j atunci interschimbăm șirul de caractere asociat propoziției i cu șirul de caractere asociat propoziției j (înseamnă că, propoziția i este "mai mare" decât propoziția j). În procesul de interschimbare a șirurilor de caractere, folosim un o variabilă auxiliară de tip șir de caractere **char** temp[80]; cu ajutorul căreia vom copia și vom inversa propozițiile în șir.

```
void sorteaza_propozitii(char sir_propozitii[25][80], int n)
{
    int i, j;
    char temp[80];
    for (i=0; i<n-1; i++)
        for (j=i+1; j<n; j++)
            if (strcmp(sir_propozitii[i], sir_propozitii[j])>0)
            {
                strcpy(temp, sir_propozitii[i]);
                strcpy(sir_propozitii[i], sir_propozitii[j]);
                strcpy(sir_propozitii[j], temp);
            }
}
```

De exemplu:

- Când $i=0$ și $j=1$ atunci `sir_propozitii[0]="Ana are mere."` iar
`sir_propozitii[1]="Ion are prune."`

Rezultatul comparației `strcmp(sir_propozitii[0], sir_propozitii[1])` este un număr negativ deci primul șir de caractere este mai mic decât al doilea fiindcă caracterul 'A' din propoziția "Ana are mere." este mai mic decât caracterul 'I' din propoziția "Ion are prune." (*)

Observație: În procesul de comparare lexicografică a propozițiilor (șirurilor de caractere), se compară codul ASCII asociat caracterelor din propoziții.

(*) Caracterul 'A' are codul ASCII 65, iar caracterul 'I' are codul ASCII 73.

Deci, condiția din `if` nu e îndeplinită → Nu au loc interschimbări la acest pas.

- Când $i=0$ și $j=2$ atunci `sir_propozitii[0]="Ana are mere."` iar
`sir_propozitii[2]="Alina are alune."`

Rezultatul comparației `strcmp(sir_propozitii[0], sir_propozitii[2])` este un număr pozitiv deci primul șir de caractere este mai mare decât al doilea fiindcă caracterul 'n' din propoziția "Ana are mere." este mai mare decât caracterul 'l' din propoziția "Alina are prune." (codul ASCII asociat caracterului 'n' este 110 care este mai mare decât codul ASCII asociat caracterului 'l' și anume 105)

Deci, condiția din `if` e îndeplinită → copiem temporar în variabila `temp` propoziția 0, pentru a interschimba propoziția 0 cu propoziția 2 în șir

În urma rearanjării celor două propoziții (0 și 2), șirul de propoziții va stoca în memorie:

`sir_propozitii[0]="Alina are alune."`
`sir_propozitii[1]="Ion are prune."`
`sir_propozitii[2]="Ana are mere."`

s-a interschimbato propoziția 0 cu
 propoziția 2, restul rămânând nemodificat

- Când $i=1$ și $j=2$ atunci `sir_propozitii[1]="Ion are prune."` iar
`sir_propozitii[2]="Ana are mere."`

Rezultatul comparației `strcmp(sir_propozitii[1], sir_propozitii[2])` este un număr pozitiv deci primul șir de caractere este mai mare decât al doilea fiindcă caracterul 'I' din propoziția "Ion are prune." este mai mare decât caracterul 'A' din propoziția "Ana are mere."

Deci, condiția din `if` e îndeplinită → copiem temporar în variabila `temp` propoziția 1, pentru a interschimba propoziția 1 cu propoziția 2 în șir

În urma rearanjării celor două propoziții (1 și 2), șirul de propoziții va fi sortat și va stoca în memorie următoarele șiruri de caractere:

```

sir_propozitii[0]="Alina are alune."
sir_propozitii[1]="Ana are mere."
sir_propozitii[2]="Ion are prune."

```

s-a interschimbat propoziția 1 cu propoziția 2, restul fiind deja sortat

Pentru a tipări la ieșire șirul sortat, definim o funcție *afiseaza_propozitii()* a cărei definiție cuprinde următoarele instrucțiuni:

```

void afiseaza_propozitii(char sir_propozitii[25][80], int n)
{
    int i;
    for(i=0; i<n; i++)
        puts(sir_propozitii[i]);
}

```

Main-ul complet este:

```

int main()
{
    int n;
    char sir_propozitii[25][80];
    printf("Introduceti numarul de propozitii n=");
    scanf("%d", &n);
    citeste_propozitii(sir_propozitii, n);
    printf("\nPropozitiile introduse de la tastatura sunt:\n");
    afiseaza_propozitii(sir_propozitii, n); // pentru verificare

    sorteaza_propozitii(sir_propozitii, n);
    printf("\nPropozitiile sortate sunt:\n");
    afiseaza_propozitii(sir_propozitii, n);
    return 0;
}

```

Metoda 2 – cu alocare dinamică

În implementarea soluției problemei, pentru a stoca șirul de propoziții vom defini o variabilă:

```
char *sir_propozitii[25];
```

Funcțiile de citire, sortare și de afișare a propozițiilor vor avea următoarea sintaxă:

```

void citeste_propozitii(char *sir_propozitii[25], int n);
void sorteaza_propozitii(char *sir_propozitii[25], int n);
void afiseaza_propozitii(char *sir_propozitii[25], int n);

```

În cazul acestei metode de rezolvare, modificările survin în funcția de citire a propozițiilor. În cadrul acestei funcții, citim cu *gets()* fiecare propoziție într-o variabilă temporară, după care stocăm propoziția citită, în șir pe poziția *i*, unde în prealabil s-a alocat dinamic memoria.

```

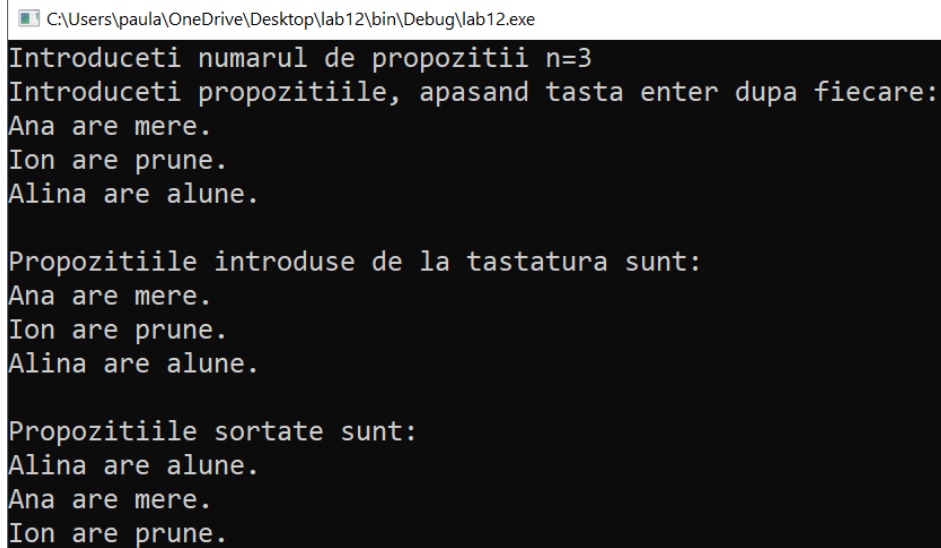
void citeste_propozitii(char *sir_propozitii[25], int n)
{
    int i;

```

```
char temp[80];
puts("Introduceti propozitiile, apasand tasta enter dupa fiecare:");
fflush(stdin);
for(i=0;i<n;i++)
{
    gets(temp);
    sir_propozitii[i]=malloc(strlen(temp)+1);
    strcpy(sir_propozitii[i],temp);
}
```

Funcțiile *sorteaza_propozitii()* și *afiseaza_propozitii()* vor deține exact același grupaj de instrucțiuni ca în cazul primei metode de rezolvare. De asemenea, main-ul este identic.

În ambele metode de rezolvare, output-ul este:



```
C:\Users\paula\OneDrive\Desktop\lab12\bin\Debug\lab12.exe
Introduceti numarul de propozitii n=3
Introduceti propozitiile, apasand tasta enter dupa fiecare:
Ana are mere.
Ion are prune.
Alina are alune.

Propozitiile introduse de la tastatura sunt:
Ana are mere.
Ion are prune.
Alina are alune.

Propozitiile sortate sunt:
Alina are alune.
Ana are mere.
Ion are prune.
```