Siruri de caractere

Marius Ghiniei

4 mai 2025

Introducere

Un **șir de caractere** reprezintă o succesiune de caractere **ASCII**, care se încheie cu caracterul `'\0'` (`NULL`).

Pointerul este un tip de variabilă ce ne permite memorarea unei adrese de memorie (de exemplu: `0x7ffdf705aec4`).

1 Caracter vs Sir de caractere

Probabil, una dintre cele mai mari confuzii cand lucram cu sirurile de caractere este aceasta intrebare. Ce este un sir de caractere si ce este un caracter? Am lamurit ca un sir de caractere este o succesiune de caractere, dar un caracter ce este?

Toate caracterele pe care le folosim, au asociat un cod. Acest cod se numeste ASCII. Nu trebuie sa stim toate codurile, doar cateva notiuni.

- Codul asociat lui **a** este **97**.
- Literele mari au un cod mai mic fata de literele mici, de exemplu **A** are codul **65**.
- Diferenta dintre aceeasi litera mica si litera mare **('a' - 'A')** este egala cu **32** si este constanta pentru toate literele alfabetului englez.

Inseamna ca un caracter reprezinta efectiv un cod si un sir de caractere este o succesiune de coduri.
Obs: Un caracter se reprezinta prin ' '(Single quotes / Apostrof), iar un sir de caractere prin " "(Double quotes / Ghilimele).

Sirul "a", este format din **2** caractere. Caracterul a si caracterul `NULL`.

2 Declarare sir de caractere

Pentru a declara un sir de caractere ne vom folosi de tipul de data **char**, care ocupa 1 octet (1 byte).

```
char nume[lungime + 1];
```

La **nume** putem sa dam ce vrem, dar de obicei dam s, t, k etc ... .

La **lungime** este lungimea maxima pe care o poate lua sirul de caractere, adaugam + 1 si pentru caracterul `NULL`.

2.1 Declarare Pointer

Pentru a declara o variabila de tip pointer (mereu folosita la metoda `strtok`) vom proceda astfel:

```
char *p;
```

3 Citirea unui sir de caractere

Putem avea 2 tipuri de citiri standard.

Daca citim un caracter sau un cuvant, cu alte cuvinte citim o propozitie **fara spatii** folosim:

```
char s[101];
cin >> s;
```

Daca avem o propozitie si cu spatii folosim:

```
char s[101];
cin.getline(s,101);
```

4 Afisarea unui sir de caractere

Putem avem multe tipuri de afisare, am sa prezint cele mai uzuale.

Afisarea standard este urmatoarea:

```
char s[101];
cout << s;
```

Obs: Chiar daca consideram variabila **s** ca si un vector, putem face afisarea asa. La vectori de tipul **int** nu o sa functioneze!

Putem afisa caracter cu caracter:

```
char s[101];
for(int i = 0; i < strlen(s); i++)
    cout << s[i];
```

Afisam caracter cu caracter fara a folosi functii predefinite:

```
char s[101];
int i = 0;
while(s[i] != NULL){
    cout << s[i];
    i++;
}
```

Si afisarea care pune mai multe probleme, si uneori confuzii:

```
char s[101];
cout << s + 5;
```

Obs: Este doar un exemplu aceasta afisare, dar ea semnifica ca primele 5 caractere le ignor, adica nu le afisez.

s :	A	n	a	␣	a	r	e	␣	m	e	r	e
i :	0	1	2	3	4	5	6	7	8	9	10	11

In acest caz primele 5 caractere le ignor si se afieaza textul ”re mere”.

Prin urmare, pe caz general instructiunea:

```
cout << s + i;
```

ignora primele **i** caractere.

5 Functii Predefinite

In aceasta sectiune, voi prezenta functii specifice sirurilor de caractere, evident cele mai folosite.

Pentru a putea folosi urmatoarele functii, trebuie sa includem si biblioteca "cstring", in felul urmator:

```
#include <cstring>
```

5.1 strlen

Functia **strlen** $\sim$ **string length** retureaza **numarul** de caractere a unui sir de caractere.

Sintaxa:

```
strlen(char s[])
```

s - trebuie sa fie sir de caractere!

Utilizare:

```
int n = strlen(s)
```

Salvam in variabila n, numarul de caractere din sirul s.

5.2 strcpy

Functia **strcpy** $\sim$ **string copy** copiaza un sir in alt sir

Sintaxa:

```
strcpy(char dest[], char sursa[])
```

dest, sursa - trebuie sa fie siruri de caractere!
Se copiaza sirul sursa, in sirul dest.

Utilizare:

```
strcpy(s, "Invatam siruri de caractere");
```

dest este s, deci in s se salveaza.
sursa = "Invatam siruri de caractere", deci asta se copiaza.

Obs: Putem folosi copiera de siruri pentru a elimina caractere. De exemplu daca avem:

```
char s[101];
cin.getline(s,101);
for(int i = 0; i < strlen(s); i++)
    if(strchr("aeiou", s[i]) != 0)
        strcpy(s + i, s + i + 1); //elimina caracterul de pe pozitia i
cout << s;
```

Codul de mai sus, identifica pozitiile unde am vocale, si elimina vocalele de pe pozitiile respective.

Problema pot avea si urmatoarea varianta:

```
char s[101], t[101];
cin.getline(s,101);
for(int i = 0; i < strlen(s); i++)
    if(strchr("aeiou", s[i]) != 0){
        strcpy(t, s + i + 1);
        strcpy(s + i, t);
    }
cout << s;
```

Aici folosesc un sir temporar, dar se observa ca este aceasi copiere, sa copiaza sirul **s + i + 1** in sirul **s + i**.

Fiindca **s + i** igonora primele **i** caractere, la copiere aceste caractere raman neafectate, adica raman in **s** si face copierea de la pozita **i + 1**.

5.3 strncpy

Functia **strncpy** $\sim$ **string n copy** copiaza primele n caractere din al doilea sir in primul.

Sintaxa:

```
strncpy(char dest[], char sursa[], int n)
```

dest, sursa - trebuie sa fie siruri de caractere!

Utilizare:

```
strncpy(s, "Invatam siruri de caractere", 5);
```

dest este s, deci in s se salveaza.
sursa = "Invatam siruri de caractere", deci din asta copiaza primele 5 caractere adica **Invat**.

5.4 strcmp

Functia **strcmp** $\sim$ **string compare** compara 2 siruri de caractere.

Sintaxa:

```
strcmp(char s1[], char s2[])
```

s1, s2 - trebuie sa fie siruri de caractere!
Functia este case - sensitive, adica **'a' != 'A'**

Utilizare:

```
if(strcmp(s1, s2) == 0)
    cout << "Sirurile sunt identice.";
else if(strcmp(s1, s2) > 0)
    cout << "Primul sir este mai mare, lexicografic";
else if(strcmp(s1, s2) < 0)
    cout << "Al doilea sir este mai mare, lexicografic";
```

Se parcurg sirurile pana cand se gaseste o diferenta la un caracter sau se termina sirurile. Se returneaza dupa caz. Lexicografic inseamna ordine alfabetica, deci daca primul sir este mai mare, inseamna ca al doilea sir, alfabetic, este inaintea primul sir.

```
char s1[8] = "ana are";
char s2[8] = "ama are";
if(strcmp(s1,s2) > 0)
    cout << "Primul sir este dupa al doilea";
```

5.5 strcat

Functia **strcat** $\sim$ **string concatenation** lipeste, adica concateneaza, un sir de alt sir.

Sintaxa:

```
strcat(char s1[], char s2[])
```

s1, s2 - trebuie sa fie siruri de caractere!

Utilizare:

```
char s[101];
strcpy(s, "Invat");
strcat(s, " siruri de caractere");
cout << s;
```

Se copiaza in sirul **s** sirul "Invat", de care se lipeste sirul " siruri de caractere". In final se afiseaza sirul s, adica **Invat siruri de caractere**.

Obs: Daca vreau sa lipesc doar un caracter, adica:

```
char s[101];
strcpy(s, "Invat");
strcat(s, ' ');
cout << s;
```

Este eroare de compilare pentru ca ' ' este un caracter si nu un sir!
Pentru a concatena corect un spatiu de un cuvant de exemplu, facem in felul urmator:

```
char s[101];
strcpy(s, "Invat");
strcat(s, " ");
cout << s;
```

5.6 strchr

Functia **strchr** $\sim$ **string character/search** cauta un caracter intr-un sir de caractere.

Sintaxa:

```
strchr(char s[], char c)
```

s - trebuie sa fie sir de caractere si c - trebuie sa fie caracter!

Utilizare:

```
if(strchr("aeiou", c) == 0)
    cout << c << " este consoana";
else if(strchr("aeiou", c) != 0)
    cout << c << " este vocala";
```

Poate nu are foarte mult sens acum, dar pe viitor o sa aiba.

Functia strchrreturneaza un pointer, adica o adresa de memorie.

Daca returneaza 0, de fapt returneaza **pointer-ul nul**, adica nu gaseste caracterul respectiv, cu alte cuvinte daca nu gaseste caracterul c in sirul "aeiou", inseamnca ca c este consoana.

Daca strchrreturneaza ceva diferit de 0, returneaza locatia din memorie unde se gaseste prima aparitie a caracterului c, cu alte cuvinte gaseste caracterul c in sirul "aeiou", inseamna ca c este vocala.

Putem face diferite combinatii, depinde de enunt. Mai sus am presupus ca c - ul poate fi doar litera mica.

Sa consideram urmatoarea problema, afisati numarul de vocale din sirul s citit de la tastatura.

```
char s[101];
int voc = 0;
cin.getline(s,101);
for(int i = 0; i < strlen(s); i++)
    if(strchr("aeiouAEIOU", s[i]) != 0)
        voc++;
cout << "Numarul de vocale este: " << voc;
```

5.7 strtok

Functia **strtok** $\sim$ **string token** separa o propozitie in token - uri.
O tinem minte pentru simplitatea ei de a putea toca propozitia in cuvinte.

Sintaxa:

```
char *p = strtok(char s[], char separatori[]); // primul apel
p = strtok(NULL, char separatori[]); // al doilea apel
```

s - este un sir de caractere
separatori - este o insiruire a toti separatorii pentru cuvinte de exemplu:

```
char separatori[] = " .,/?!;:[](){}";
```

Punem in **sirul** separatori, toti separatorii posibili pentru a delimita 2 cuvinte, pentru probleme unde separatorii pot fi multiplii.

Obs: Foarte important, `strtok` se salveza neaparat intr - un **pointer**! De fapt, strtok returneaza adresa de memorie, unde se intalneste primul separator.

Utilizare:

```
char separatori[] = " .,/?!;:[](){}";
char s[101];
cin.getline(s,101);
char *p = strtok(s, separatori);
while(p != NULL){
    // aici prelucram cuvantul p care este tot un sir de caractere
    cout << p << endl;
    p = strtok(NULL, separatori); // trec la urmatorul cuvant
}
```

Se salveaza primul cuvant in p, si cat timp mai am cuvinte in sirul s, determin toate cuvintele.

Dupa caz pot parcurge sirul p cu un *for* si determin valorile din probleme.

6 Alte idei importante

Daca vrem sa verificam daca un caracter este litera mare folosim:

```
if(c >= 'A' && c <= 'Z')
```

Daca vrem sa verificam daca un caracter este litera mica folosim:

```
if(c >= 'a' && c <= 'z')
```

Daca vrem sa verificam daca un caracter este cifra folosim:

```
if(c >= '0' && c <= '9')
```

Daca vrem sa verificam daca un caracter este spatiu folosim:

```
if(c == ' ')
```

Pentru a transforma o litera mare in litera mica (tinem cont de codurile ASCII):

```
int k = 'a' - 'A'; // diferenta dintre litera mica si mare este 32
if(c >= 'A' && c <= 'Z')
    c = c - k;
```

Pentru a transforma o litera mica in litera mare (tinem cont de codurile ASCII):

```
int k = 'a' - 'A'; // diferenta dintre litera mica si mare este 32
if(c >= 'a' && c <= 'z')
    c = c + k;
```