**ŞIRURI DE CARACTERE**

Un şir de caractere este o succesiune de caractere care se termină cu caracterul **NULL** / **0** / **'\0'**)**.**

O constantă de tip şir de caractere scrie între ghilimele. **Ex.** “Da”, “nu exista”.

**Obs. 'A' ≠ “A”**, deoarece prima constantă este de tip **char** şi ocupă un octet, pe când a doua este de tip **şir de caractere** şi ocupă 2 octeţi (unul pentru caracterul A şi unul pentru caracterul NULL).

* **Declararea şirurilor de caractere**

1. **Ca vector de caractere**

**char nume\_sir[dim\_max]; //** unde **dim\_max** este dimensiunea maximă a şirului şi trebuie să includă şi terminatorul de şir (caracterul **NULL** / **0** / **'\0'**).

**Ex.** char s[11]; // S-a declarat un șir care poate memora maxim 11 caractere, cu indici de la 0 la 10, ultimul caracter memorat fiind caracterul '\0'.

**Obs.**

1. La declararea unui şir de caractere se poate face **iniţializarea** lui:

**Ex.** char s[11]=”calculator”;

sau char s[]=”calculator”; //compilatorul face calculul numarului de octeti necesari

1. Caracterele şirului vor ocupa poziţii consecutive în vector, **începând cu poziţia 0 (caracterele în şir sunt indexate de la 0).**

Pentru exemplul anterior: s[0]=’c’, s[1]=’a’, s[9]=’r’

1. **Ca pointer către tipul char (care va reprezenta adresa de început a şirului)**

**char \*nume\_sir;**

**Ex.** char \*s;

* **Citirea unui șir de caractere de la tastatură**

**char s[100];**

1. **cin >> s;**

Se reţin în şirul s caracterele până la primul spațiu, adică citirea se opreşte la primul spaţiu întâlnit (de exemplu, daca se citeste “ora de informatica”, variabila s va retine numai “ora”). **Deci, citirea cu cin şi operatorul >> se face numai dacă şirul s nu conţine spaţii!**

1. **Pentru a citi un şir care conţine spaţii vom folosi:**

**cin.get(s, dim\_max); // nu citeşte** caracterul new line ('\n') de la apasarea tastei Enter

sau **cin.getline(s, dim\_max); // citeşte** caracterul new line ('\n'), adică sare peste Enter

**Obs.**

1. **Citirea a două şiruri unul după altul**

char s1[31], s2[31];

cin.getline(s1, 31);

cin.getline(s2, 31);

**sau**

char s1[31], s2[31];

cin.get(s1, 31);

**cin.get(); //** citeştecaracterul new line ('\n'), adică sare peste Enter

cin.get(s2,31);

1. **Citirea unui şir de caractere care conţine spaţii după un număr**

char s[31];

int n;

cin>>n;

**cin.get(); //** citeştecaracterul new line ('\n'), adică sare peste Enter

cin.getline(s, 31);

1. **Citirea unui şir din fişier**

**f.getline(s, dim\_max);** // unde, f=numele variabilei de acces la fişier

* **Afişarea unui șir de caractere**

**cout<<s; // pe ecran**

**g<<s; // în fişierul asociat variabilei g**

**Funcţii pentru operaţii cu şiruri de caractere**

Funcţiile pentru operaţii cu şiruri se găsesc în header-ul **<cstring> / <string.h>**

**1)** **strlen(s)** –> returnează lungimea efectivă a şirului **s** (fără a număra terminatorul de şir).

**Ex.** char s[50]=”casa”; // strlen(s) = 4

**Obs. Prelucrarea unui şir de caractere, caracter cu caracter.**

for(int i=0; i<strlen(s); i++) // sau for(int i=0; s[i]; i++)  
 // prelucreaza s[i]

**Aplicaţie.**  Se citeşte de la tastatură o frază cu maxim 100 de caractere, în care cuvintele sunt separate prin câte un singur spaţiu. Să se afişeze numărul de cuvinte din frază. **Ex**. s =„Ana are mere rosii”

#include<iostream>

#include<cstring>

char s[101];

int i, nr; // nr = numărul de spaţii din frază

int main()

{ cin.getline(s,101);

for(i=0; i<strlen(s); i++)

if (s[i] ==' ') nr++;

cout<<nr+1; // numărul de cuvinte = numărul de spaţii + 1

return 0;

}

**Tema (Pbinfo): 12, 1866, 2828, 13, 88, 138, 155, 801**

**2) strcpy(s1, s2);** -> copiaza şirul sursă **s2** în şirul destinaţie **s1**; returnează adresa de început a şirului **s1.**

**Obs.**

* Nu este permisă atribuirea între două şiruri de caractere (declarate ca vector de char) folosind operatorul **=**. **Atribuirea se face folosind funcţia strcpy!**
* Un șir de caractere **s** este indiferent de modul în care îl declarăm **un pointer (o adresă)**. Astfel, valoarea sa **s** este adresa primului element din șir, adică **adresa lui s[0].** De aceea, **s + 1 va reprezenta adresa lui s[1], s + 2 va reprezenta adresa lui s[2]** ş.a.m.d.

**Ex.**

char s[21], t[21] = "copil";

strcpy(s , "pbinfo");

cout << s; // pbinfo

strcpy(s , t);

cout << s; // copil

strcpy(s , t + 2);

cout << s; // pil

strcpy(s + 2 , t);

cout << s; // picopil

**Aplicaţii**

* Stergerea **primului caracter** din şirul s: **strcpy(s, s + 1);**
* Stergerea **primelor k caractere** din şirul s: **strcpy(s, s + k);**
* Stergerea **caracterului de pe poziţia i** din şirul s: **strcpy(s + i, s + i + 1);**
* Funcţia **strncpy(s1, s2, n);** -> copiaza primele **n caractere din** şirul **s2** în şirul **s1.** Nu pune NULL în şirul s1, după cele n caractere copiate.

**Ex 1.**

strncpy(s,”informatica”,4);

S[4]=0; // punem terminatorul de şir!

cout<<s; // info

**Ex 2.**

char s[100]="abcdefghjkl";

strncpy(s, "poveste", 3);

cout << s; // povdefghjkl

**3) strcat(s1, s2);** -> concatenează (adaugă) şirul **s2** la sfârşitul şirului **s1** (inclusiv terminatorul de şir); returnează adresa de început a şirului **s1.**

char s[21]="pan", t[21] = "cake";

strcat(s , t);

cout << s; // pancake

* Funcţia **strncat(s1, s2, n);** -> concatenează (adaugă) primele n caractere din şirul s2 în şirul s1. Nu pune **NULL** în şirul s1, după cele n caractere concatenate.

1. **strcmp(s1, s2) ->** Compară **lexicografic** cele două șiruri de caractere:

* dacă șirul **s1** este mai mic decât **s2** funcția va returna o valoare negativă
* dacă șirul **s1** este mai mare decât **s2** funcția va returna o valoare pozitivă
* dacă cele două șiruri sunt identice funcția va returna valoarea **0**

**Ex.**

char s[21]="abur", t[21]="abecedar";

if(strcmp(s, t) < 0)

cout << "Da"

else

cout << "Nu";

// se va afisa Nu; cuvantul "abur" este lexicografic dupa "abecedar"

* Funcţia **strncmp(s1, s2, n) ->** Compară lexicografic **primele n caractere** din cele două șiruri
* Funcţia **stricmp(s1, s2) ->** Compară lexicografic cele două șiruri fără a face diferenţa dintre literele mici şi mari

1. **strchr(s, c) ->** unde **s** este un şir de caractere şi **c** este un caracter (char), returnează **adresa primei apariţii** a caracterului **c** în şirul **s** sau **NULL** dacă **c** nu se află în şirul **s**.

**Ex 1.**

char s[20]="bacalaureat", \*p;

p = strchr(s, 'a');

cout << p; // acalaureat

**Ex 2.**

char c = 'i';

if(strchr("aeiou" , c)) **// verifică dacă caracterul c este vocală (ca literă mică)**

cout << "DA"

else

cout << "NU";

//se va afisa DA

**Ex 3. Poziţia primei apariţii a unui caracter într-un şir**

char s[20]="cosmos", \*p;

p = strchr(s, 's');

cout << p - s; // 2

* Functia **strrchr(s, c)** –> are acelasi rol cu **strchr**(), cu deosebirea ca returneaza adresa ultimei aparitii a caracterului (cautarea se face de la dreapta spre stanga; r = right)

**Ex 4.**

char s[21]="bacalaureat", \*p;

p = strrchr(s, 'a');

cout << p; // at

1. **strstr(s1,s2) ->** returnează **adresa primei apariţii** a şirului **s2** în şirul **s1** sau **NULL** dacă **s2** nu se află în şirul **s1**.

**Ex.**

char s[21]="parasolara", \*p;

p = strstr(s, 'ara');

cout << p; // arasolara

1. Functia **strlwr(s)** –> converteşte toate literele mari din şirul **s** în litere mici. Restul caracterelor rămân neschimbate**.**
2. Functia **strupr(s)** –> converteşte toate literele mici din şirul **s** în litere mari. Restul caracterelor rămân neschimbate**.**
3. Funcţia **strrev(s)** –> returnează adresa şirului invesat/oglindit. Şirul **s** iese modificat!

**Ex.**

char s[21]="masa";

cout << strrev(s); // asam

cout << s; // asam

1. **strtok(s1,s2)** –> extrage pe rând din şirul de caractere **s1** câte un subșir/cuvânt, subşirurile/cuvintele fiind delimitate de caractere din șirul **s2**.

Funcţia **strtok** acţionează în felul următor:

* Primul apel trebuie să fie de forma **strtok(s1, s2)**, apel care returnează adresa de început a primului subşir/cuvânt. Separatorul de după cuvânt este înlocuit prin caracterul NULL.
* Următoarele apeluri sunt de forma **strtok(NULL, s2)** De fiecare data, funcţia întoarce adresa de început a următorului subşir/cuvânt, înlocuind separatorul de după el cu NULL.
* Cand şirul **s1** nu mai conţine subşiruri/cuvinte, funcţia returnează NULL.

**Obs.** Șirul **s1** din care se face extragerea se modifică în urma apelurilor. Dacă este nevoie de el mai târziu trebuie să-i facem o copie.

**Aplicaţii**

**Se dă o frază în care cuvintele sunt separate prin spaţiu, virgulă, punct sau o combinaţie a lor. Să se afişeze cuvintele din frază, fiecare pe căte o linie.**

**Ex.** Dacă se dă şirul “Ana are mere, pere.” se va afişa:

Ana

are

mere

pere

char s[100], \*p, sep[]=" .,";

cin.getline(s, 100);

p = strtok(s, sep); // p = adresa de început a primului cuvânt

while(p)

{

cout << p << endl; // se afişează cuvântul p

p = strtok(NULL , sep); // se extrage următorul cuvânt

}

**Se dă o frază în care cuvintele sunt separate prin câte un spaţiu. Să se afişeze cuvintele din frază, în ordinea inversă apariţiei lor, fiecare pe căte o linie. Fraza are maxim 1000 de caractere şi fiecare cuvant are maxim 20 de caractere.**

**Ex.** Dacă se dă şirul “Ana are mere si pere.” se va afişa:

pere

si

mere

are

Ana

char s[1001], \*p, v[1000][21]; // **v = vector de cuvinte/siruri**

int n=0, i;

cin.getline(s, 1001);

p = strtok(s, “ ”); // p = adresa de început a primului cuvânt

while(p)

{

strcpy(v[++n], p); // se adaugă cuvântul p în vector

p = strtok(NULL , “ ”); // se extrage următorul cuvânt

}

for (i = n; i >= 1; i--)

cout << v[i] << endl;

**11) Funcţii de conversie**

Funcţiile pentru aceste operaţii cu şiruri se găsesc în header-ul **<cstdlib> / <stdlib.h>**

**a) Conversii - sir de caractere -> numar**

* Functia **atoi(s)** –> converteste sirul **s** la tipul **int**. Daca aceasta conversie esueaza (se intalneste un caracter nenumeric), valoarea intoarsa este 0.
* Functia **atol(s)** – converteste sirul **s** la tipul **long**. Daca aceasta conversie esueaza (se intalneste un caracter nenumeric), valoarea intoarsa este 0.
* Functia **atof(s)** –> converteste sirul **s** la tipul **double**. Daca aceasta conversie esueaza (se intalneste un caracter nenumeric), valoarea intoarsa este 0.
* Functia **atold(s)** –> converteste sirul **s** la tipul **long double**. Daca aceasta conversie esueaza, valoarea intoarsa este 0.

**Ex.**

int x;

double z;

x=atoi(“30”); // x=30;

z=atof(“-3.14”); // z=-3.14

x=atoi(“-3.14”); //x=-3

**a) Conversii - numar -> sir de caractere**

* Functia **itoa(n, s, b);** – converteste numărul **n** de tip **int*,*** aflat în baza de numeraţie **b**,în şirul de caractere **s.**
* Functia **ltoa(n, s, b);** – converteste numărul **n** de tip **long**,aflat în baza de numeraţie **b**,în şirul de caractere **s.**
* Functia **ultoa(n, s, b);** – converteste numărul **n** de tip **unsigned** **long**, aflat în baza de numeraţie **b**,în şirul de caractere **s.**

**Ex.**

char s[100];

itoa(-104, s, 10); cout<<s; // s=”-104”

itoa(13, s, 2); cout<<s; // s=”1101”

itoa(13, s, 16); cout<<s; // s=”d”

**Link-uri utile:**

* [*https://www.pbinfo.ro/articole/19/siruri-de-caractere-in-c*](https://www.pbinfo.ro/articole/19/siruri-de-caractere-in-c)
* <http://www.worldit.info/uncategorized/siruri-de-caractere-si-lucrul-cu-acestea-in-cc-partea-i/>
* <https://invata.info/2017/03/31/functii-pentru-operatii-cu-siruri-de-caractere-c/>

**Probleme**

1. Sa se verifice daca un cuvant este palindrom.

2. Sa se transforme un sir din litere mici in litere mari.

3. Sa se transforme un sir din litere mari in litere mici.

4. Sa se determine frecventa de aparitie a unui caracter intr-un text.

5. Sa se genereze toate prefixele / sufixele unui cuvant.

6. Se citeste un text dintr-un fisier si un caracter c. Sa se determine de cate ori se gaseste caracterul in text (nu se face distinctie intre literele mari si literele mici).

7. Se citeste un text de la tastatura astfel incat cuvintele sa fie separate printr-un singur spatiu si imediat dupa ultimul cuvant se scrie punct. Textul va fi scris pe un singur rand.

a) Sa se determine cate cuvinte contine textul. De ex : "Ana are mere." Contine 3 cuvinte.

b) Sa se determine daca textul are cuvinte distincte (se ignora diferenta de cheie). c) Sa se determine daca textul contine cifre.

8. Sa se determine de cate ori se gaseste un cuvant intr-un text.

9. Codificati un text astfel incat litera a sa devina c, b sa devina e s.a.m.d.

10. Sa se sorteze alfabetic un sir de cuvinte (eventual, fara a distinge literele mici de cele mari).

11. Codificarea pasareasca a unui cuvant (dupa fiecare vocala, se pune litera p urmata de acea vocala). Ex : informatica: ipinfopormapatipicapa.

12. Se citesc n cuvinte. Sa se afiseseze grupurile de cuvinte care rimeaza (au ultimele 2 caractere identice).

13. Sa se desparta un text in cuvinte si sa se afiseze cuvintele separate. Sa se afiseze cuvantul de lungime maxima.

14. Sa se verifice daca doua cuvinte sunt sau nu anagrame.

15. Sa se numere aparitiile unui cuvant intr-un text.

16. Se citeste un numar real de la tastatura. Sa se trunchieze astfel incat cifrele ramase sa formeze o secventa monoton crescatoare.

Ex. 34.59483 →34.59 ; 24.1276 → 24

17. Se citeste un sir de caractere alfanumerice. Consideram ca literele sunt separatorii numerelor. Afisati datele de tip numeric preluate in ordine din sirul citit. Numerele vor fi scrise cate unul pe o linie.

Ex.

in.txt out.txt

a23sc345ss5e

23

345

5

18. In directorul curent se afla fisierul cuvinte.txt care contine mai multe linii de text formate din cuvinte separate de cate un spatiu. Sa se afiseze cuvintele care au cel putin 3 consoane sau 3 vocale consecutive.

19. Se citeste un sir de caractere. Sa se afiseze sirul oglindit din care lipsesc vocalele. 20. Se da un text de maxim 30 de caractere. Sa se listeze toate cuvintele de doua caractere din acest text.

21. Se introduc de la tastatura cuvinte pana la citirea cuvantului stop. Afisati pe ecran cuvintele mai mari in sens lexicografic decat primul citit.

22. Se dau doua texte. Sa se stabileasca o vocala comuna celor doua texte, care apare de cele mai putine ori.

23. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu. Sa se determine cate cuvinte contine textul.

24. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu. Se va genera un nou text care va contine cuvintele ordonate alfabetic

25. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu. Sa se scrie intr un alt fisier, pe linii separate, fiecare cuvant care apare in text urmat de un numar care va reprezenta de cate ori apare cuvantul in text. Sa se determine cuvantul care apare de cele mai multe ori.

26. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu. Intr-un alt fisier se va scrie pe linii separate fiecare cuvant si liniile pe care apare.

27. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu sau mai multe. Se va genera un nou fisier care va contine textul initial avand spatiile de prisos eliminate (intre cuvinte va ramane numai cate un spatiu).

28. Se citesc de la tastatura elementele unei matrici de caractere (nr. linii=nr. coloane), A(NXN), N<=10. a) Sa se afiseze matricea A;

b) Sa se formeze si sa se afiseze cuvantul format din caracterele pe pe diagonala principala a matricii A; c) Sa se calculeze si sa se afiseze numarul de litere mari, litere mici si cifre din matrice;

d) Sa se afiseze cuvantul format din caracterele de pe diagonala secundara;

e) Sa se afiseze procentul literelor mari, al literelor mici si al cifrelor de pe cele 2 diagonale;

f) Sa se afiseze caracterele comune aflate pe liniile p si q (p, q < N, p si q citite de la tastatura);

g) Sa se afiseze in ordine alfabetica, crescatoare, literele mari aflate pe coloanele impare.

29. Simulati comanda REPLACE astfel incat intr-un text veti inlocui un caracter x citit de la tastatura cu un alt caracter y citit de la tastatura. Se ignora sau nu diferenta de cheie dupa optiunea utilizatorului. 30. Simulati comanda REPLACE astfel incat intr-un text veti inlocui un sir x citit de la tastatura cu un alt caracter sir y citit de la tastatura. Se ignora sau nu diferenta de cheie dupa optiunea utilizatorului.

31. Se citeste de la tastatura un cuvant. Sa se stabileasca daca el contine doua litere alaturate identice, afisandu-se un mesaj corespunzator.

32. Dintr-un fisier se citesc numele a n persoane. Sa se modifice continutul fisierului astfel incat toate numele sa fie scrise astfel: prima litera mare si restul litere mici.

33. Se citesc n siruri. Pentru fiecare sir se va determina suma codurilor ASCII.

34. Intr-un fisier sunt scrise cuvinte pe linii separate. Sa se afiseze cuvintele care contin majuscule. 35. Intr-un fisier sunt scrise pe randuri diferite numele a n copii. Sa se modifice continutul fisierului astfel incat sa contina numele ordonate crescator.

36. Sa se afiseze vocalele unui cuvant.

37. Sa se afiseze cuvintele care incep si se termina cu consoana, (vocala) etc.

38. Sa se desparta un text in cuvinte si sa se afiseze cuvintele separate. Sa se afiseze cuvantul de lungime maxima.

39. Intr-un text exista un cuvant. Codificati/decodificati cuvantul dupa un algoritm generat de voi.

40. Aceeasi problema pentru un text.

41. Se dau doua texte. Sa se stabileasca o vocala comuna celor doua texte, care apare de cele mai putine ori. 42. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu. Sa se determine cate cuvinte contine textul.

43. Se citesc n cuvinte. Sa se afiseze perechile de 2 cuvinte care rimeaza.

44. Aceeasi problema, numai ca se vor afisa grupurilede cuvinte care rimeaza

45. Se citesc cuvinte pana la citirea cuvantului "stop". Sa se afiseze cate dintre cuvintele citite sunt egale cu primul cuvant citit.

46. Se citeste un text. Textul contine cuvinte separate printr-un spatiu. Se va genera un nou text care va contine cuvintele ordonate alfabetic

47. Se citeste un text. Textul contine cuvinte separate printr-un spatiu. Sa se scrie, pe linii separate, fiecare cuvant care apare in text urmat de un numar care va reprezenta de cate ori apare cuvantul in text. Sa se determine cuvantul care apare de cele mai multe ori.

48. Se citeste un text. Textul contine cuvinte separate printr-un spatiu. Sa se scrie, pe linii separate, fiecare cuvant si liniile pe care apare.

49. Se citeste un text care contine cuvinte separate printr-un spatiu sau mai multe. Se va genera un nou text care va contine textul initial avand spatiile de prisos eliminate (intre cuvinte va ramane numai cate un spatiu).

50. Simulati scrierea unei parole intr-un fisier. La tastarea parolei pe ecran in locul fiecarui caracter se va scrie caracterul '\*'. Eventual realizati si incriptarea parolei inainte de a fi scrisa intr-un fisier.

51. Fie un sir de forma: cifra-litera, cifra litera …etc. Ex 2a4b5c. Sa se genereze un astfel de sir: aabbbbccccc. 52. Fie un sir format din replicarea de un numar de ori a unui subsir redundant. Sa se determine cea mai scurta secventa redundanta. Se va afisa subsirul redundant si numarul sau de aparitii. Ex pt: aabcaabcaabcaabc se va fisa: aabc si nu aabcaabc. Numarul de aparitii este 4 si nu 2.