**Șiruri de caractere în C++/C. Tot ce trebuie să știi**

Limbajul C permite prelucrarea unor bucăți de text prin intermediul așa numitelor *șiruri de caractere*. Șirurile de caractere sunt defapt niște vectori de elemente char și există o sumedenie de funcții ajutătoare care permit prelucrarea lor cu ușurință, despre care vom vorbi în această lecție.

**Cum se memorează un șir de caractere**

Un șir de caractere se reține ca un vector de elemente char. Spre deosebire de vectorii de numere care puteau începe și de pe poziția 1, șirurile de caractere încep întotdeauna de pe poziția 0. După ultimul caracter se pune un caracter NULL ('\0'), care marchează sfârșitul șirului ([codul ASCII](https://infoas.ro/lectie/90/codul-ascii-tabel-complet) al acestui caracter este 0).

Astfel, de pildă, reprezentarea cuvântului restanță în memorie într-un șir de caractere s se face în felul următor:

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| s[i] | r | e | s | t | a | n | ț | ă | \0 |

Figura 1. Exemplu reperzentare cuvânt

**Cum se declară un șir de caractere**

Un șir de caractere se declară ca un vector de elemente char:

****

Figura 2. Declarare

De obicei, șirurile de caractere se notează cu s, însă orice altă denumire este perfect validă. Avem 101 caractere cu indici între 0 și 100. Cum la finalul oricărui șir de caractere trebuie să se afle un '\0', putem folosi 100 de caractere dintre cele 101 pentru șirul nostru de caractere.

**Valoare inițială pentru șir de caractere**

Unui șir de caractere i se poate atribui o valoare chiar la declarare. Iată câteva moduri de a reține șirul InfoAs:

char a[11] = "inginer"; //primele 7 caractere sunt i,n,g,i,n,e,r iar de acolo toate sunt caractere '\0'

char b[] = "inginer"; //lungimea lui b este 8, pentru a reține doar caracterele i,n,g,i,n,e,r '\0'

char c[11] = {'i', 'n', 'g', 'i', 'n', 'e','r', '\0'}; //similar ca la vectori cu int, putem face așa

char d[] = {'i', 'n', 'g', 'i', 'n', 'e','r', '\0'}; //sau așa

Figura 3. Inițizalizarea unui șir

**Afișarea unui șir de caractere**

Deși sunt vectori în esență, șirurile de caractere se pot afișa direct, ca o variabilă obișnuită:

char s[] = "examen";

printf("%s\n", s);

Figura 4. Afișarea

Afișarea se face automat. Se afișează caracterele începând de pe poziția 0, însă afișarea se oprește când se ajunge la primul caracter '\0'.

**Citirea unui șir de caractere**

Citirea se face, asemănător, ca la o variabilă obișnuită:

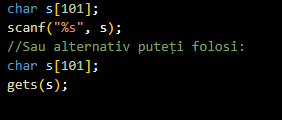


Figura 4. Citire variabilă obișnuită

Citirea se face până la întâlnirea primului spațiu sau până la primul enter. Prin urmare, citirea de acest tip nu va putea citi propoziții.

**Citirea propozițiilor cu șiruri de caractere**

Pentru a citi șiruri care conțin spații (precum propozițiile), se poate folosi funcția cin.getline(), care arată în felul următor:

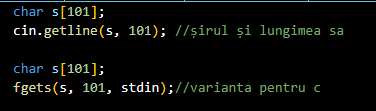


Figura 5. Citire propoziții c/c++

Se așteaptă doi parametri pentru C++

* un șir de caractere în care să se citească;
* lungimea maximă de citire (se recomandă punerea lungimii maxime a șirului, adică numărul din declarare).

Se așteaptă trei parametri pentru C

* un șir de caractere în care să se citească;
* lungimea maximă de citire (se recomandă punerea lungimii maxime a șirului, adică numărul din declarare).
* Stdin – necesar pentru citirea datelor introduse de utilizator

**Atenție!** Funcția cin.getline() citește până la întâlnirea unui caracter enter ('\n', numit și newline sau endline) sau până la finalul citirii. Cu toate acestea, dacă după citire apare un '\n', se va sări peste acesta în continuarea citirii, însă nu se va include în șirul citit.

Iată un exemplu de citire cu cin.getline()/ fgets()

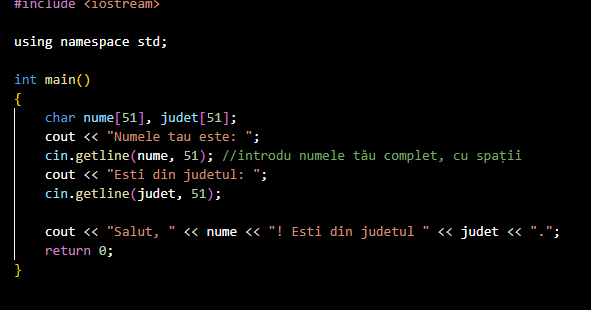


Figura 6. Program citire/afișare în C++

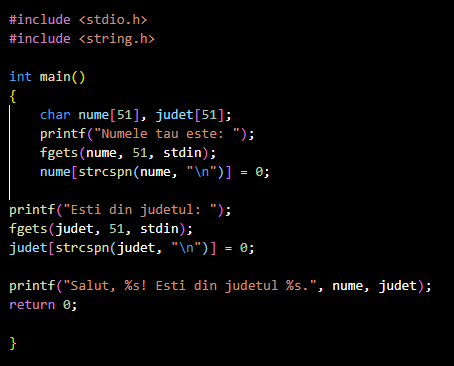


Figura 7. Program citire/afișare în C++

**Parcurgerea unui șir de caractere. Extragerea unui caracter**

Șirurile de caractere sunt vectori de elemente char. Din acest motiv, putem să extragem caracterele pe rând exact ca la un vector de numere:

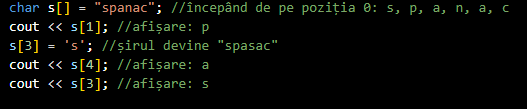


Figura 8. Extragere element

Parcurgerea unui șir de caractere se poate realiza în două moduri:

* dacă știm lungimea șirului (mai jos se prezintă o astfel de metodă), putem parcurge caracterele de la 0 la lungime - 1;
* dacă nu știm lungimea, putem parcurge folosind o structură repetitivă for, cu condiția de oprire s[i] != '\0' (adică nu ne aflăm la finalul șirului).

Iată parcurgerea unui șir de caractere și afișarea caracterelor cu spații între ele:

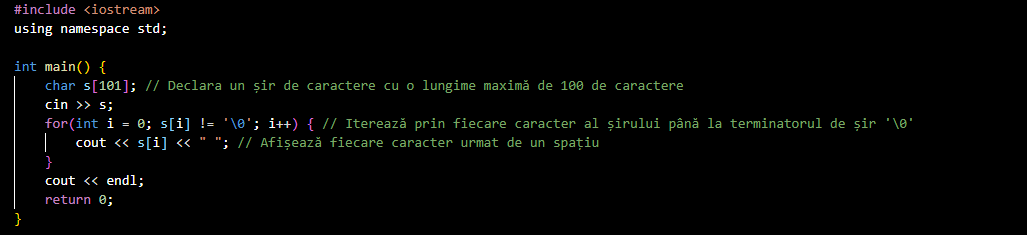


Figura 9 . Afișare cu spații

**Funcția isalpha — cum funcționează**

Funcția isalpha așteaptă un caracter c și returnează:

* 1, dacă c este literă (mică sau mare) din alfabetul englez;
* 0, în caz contrar.

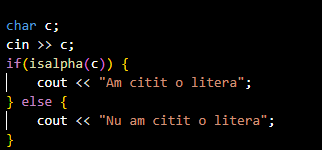


Figura 10. Funcția isalpha

**Funcția isdigit — cum funcționează**

Funcția isdigit așteaptă un caracter c și returnează:

* 1, dacă c este cifră (de la 0 la 9);
* 0, în caz contrar.

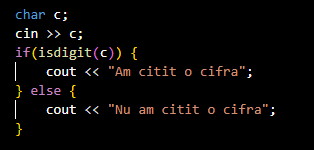


Figura 11. Funcția isdigit

**Funcția isalnum — cum funcționează**

Funcția isalnum așteaptă un caracter c și returnează:

* 1, dacă c este literă a alfabetului englez (mică sau mare) sau cifră (de la 0 la 9);
* 0, în caz contrar.

Practic, funcția isalnum(c) este echivalentă cu expresia is isalpha(c) || isdigit(c).

Mai jos este prezentată o funcție care validează un nume de utilizator, care poate conține doar litere și cifre:

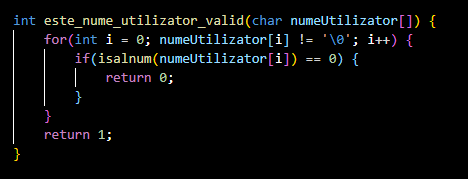


Figura 12. Funcția isalnum

**Funcția islower — cum funcționează**

Funcția islower așteaptă un caracter c și returnează:

* 1, dacă c este o literă mică;
* 0 în caz contrar (nu este literă sau este literă mare).

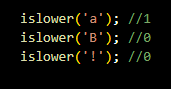


Figura 13. Funcția islower

**Funcția isupper — cum funcționează**

Funcția isupper așteaptă un caracter c și returnează:

* 1, dacă c este o literă mare;
* 0 în caz contrar (nu este literă sau este literă mică).

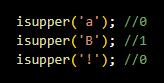


Figura 14. Funcția isupper

**Funcția tolower — cum funcționează**

Funcția tolower așteaptă un caracter c și returnează:

* c, dacă c este o literă mică sau nu este literă;
* c ca literă mică, dacă este o majusculă.

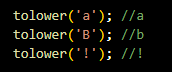


Figura 15. Funcția tolower

**Funcția toupper — cum funcționează**

Funcția toupper așteaptă un caracter c și returnează:

* c, dacă c este o literă mare sau nu este literă;
* c ca literă mare, dacă este o literă mică.

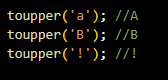


Figura 16. Funcția toupper

*Este necesară biblioteca ctype.h pentru aceste funcții!!!*

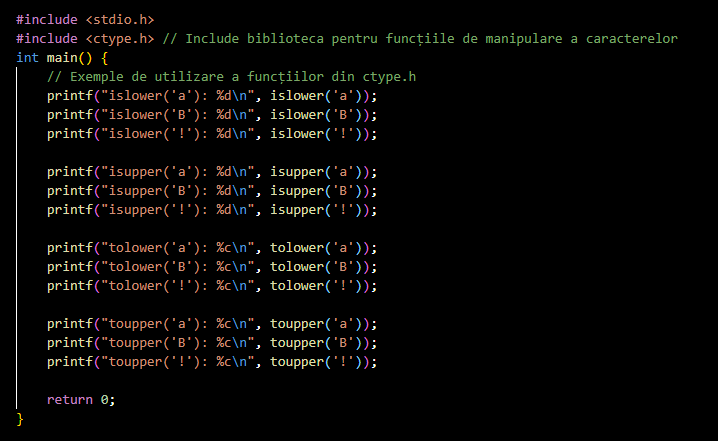


Figura 17. Exemplu utilizare funcții

**Funcția strlen — cum funcționează**

Funcția strlen așteaptă un șir de caractere s și returnează lungimea acestuia. Caracterul '\0' este exclus din numărare.

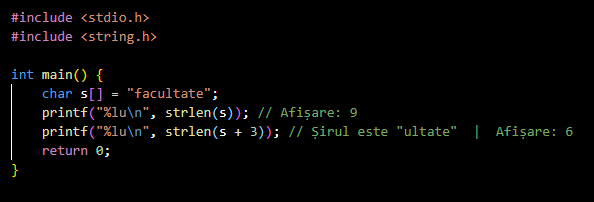


Figura 18.Funcția strlen

**Funcția strcpy — cum funcționează**

Funcția strcpy așteaptă două șiruri de caractere dest și src, după care copiază conținutul lui src în dest (ștergând ce era înainte în dest).

**Atenție**! Funcția strcpy se folosește uneori în ștergerea sau inserarea caracterelor, unde dest și src au părți care se suprapun. Se recomandă folosirea unui șir auxiliar în astfel de situații, pentru că dacă dest și src se suprapun, comportamentul este deseori impredictibil.

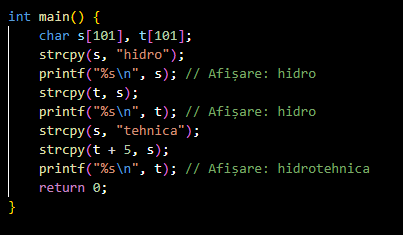


Figura 19. Funcția strcpy

**Eliminarea unui caracter dintr-un șir cu strcpy**

Pentru a șterge caracterul de pe poziția poz dintr-un șir s, ne folosim de un șir auxiliar t:

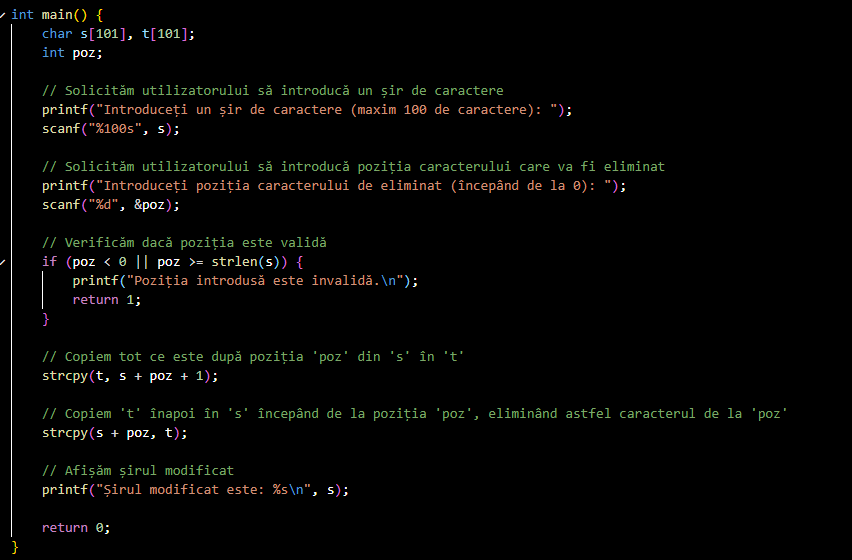


Figura 20. Rezolvare cu strcpy

**Inserarea unui caracter într-un șir cu strcpy**

Pentru a insera un caracter c pe poziția poz a unui șir s, ne folosim de un șir auxilar t:

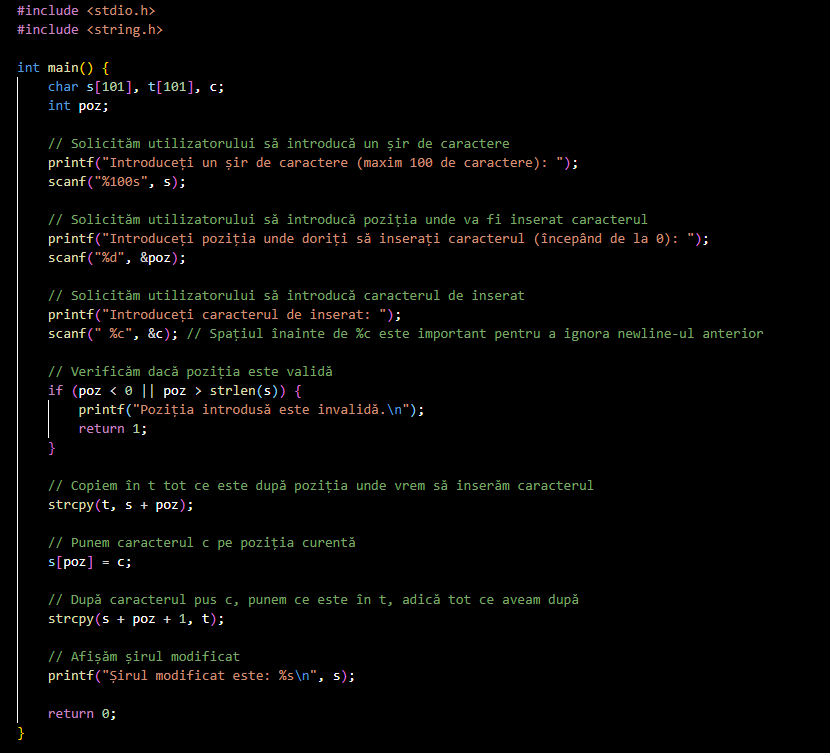


Figura 21. Exemplu inserare

**Funcția strncpy — cum funcționează**

Funcția strncpy așteaptă două șiruri de caractere dest și src, împreună cu un număr cnt și copiază în dest primele cnt caractere din src (sau șirul întreg src dacă cnt este mai mare decât lungimea acestuia). După cele cnt caractere, nu se plasează caracterul '\0' și trebuie plasat manual pentru a evita erorile.

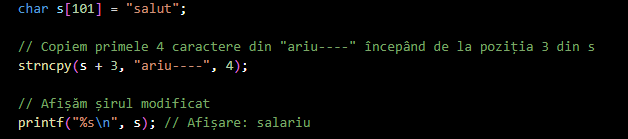


Figura 22. Exemplu utilizare strncpy

**Atenție!** Șirurile dest și src nu trebuie să se suprapună, similar ca la funcția strcpy.

**Funcția strcat — cum funcționează**

Funcția strcat așteaptă două șiruri de caractere dest și src și concatenează (lipește) la finalul șirului dest, șirul src.

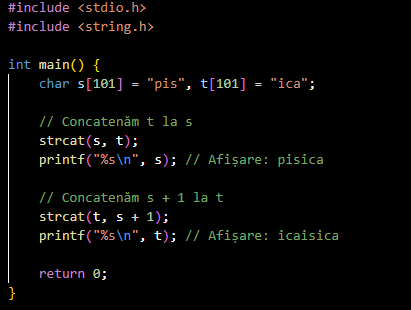


Figura 23. Utilizare strcat

**Atenție**! Ca și la funcția strcpy, la funcția strcat, șirurile dest și src nu se pot suprapune.

**Funcția strchr — cum funcționează**

Funcția strchr așteaptă un șir de caractere s și un caracter c și returnează:

* NULL (adică 0), dacă caracterul c nu se găsește deloc în s;
* adresa primei apariții a lui c în s, dacă există.

***Ce face programul acesta?***

if(strchr("aeiou", c) != NULL) {

    cout << "V";

} else {

    cout << "C";

}

Figura 24. Exemplu strchr

* Determinare pozitie

#include <stdio.h>

#include <string.h>

int main() {

    char alfabet[] = "abcdefghijklmnopqrstuvwxyz", c;

    // Solicităm utilizatorului să introducă o literă mică

    printf("Introduceți o literă mică din alfabetul englez: ");

    scanf(" %c", &c);

    // Calculăm poziția literei în alfabet

    int pozitie = (strchr(alfabet, c) - alfabet + 1);

    // Afișăm rezultatul

    printf("Litera %c se află pe poziția %d din alfabet.\n", c, pozitie);

    return 0;

}

Figura 24. Exemplu strchr - rezolvare

**Funcția strstr — cum funcționează**

Funcția strstr așteaptă doi parametri șiruri de caractere s și t și returnează:

* NULL (adică 0), dacă șirul t nu se află deloc în șirul s;
* adresa prime apariții a lui t în s, dacă există.

int main() {

    char s[] = "abcabc", t[] = "bc";

    // Afișăm direct rezultatul funcției strstr

    printf("%s\n", strstr(s, t)); // Afișare: bcabc

    return 0;

}

Figura 25. Exemplu strstr

**Funcția strcmp — cum funcționează**

Funcția strcmp așteaptă două șiruri de caractere s și t și returnează:

* Un număr negativ (nu neapărat -1), dacă s este lexicografic mai mic decât t;
* 0, dacă cele două șiruri sunt identice;
* Un număr pozitiv (nu neapărat 1), dacă s este lexicografic mai mare decât t.

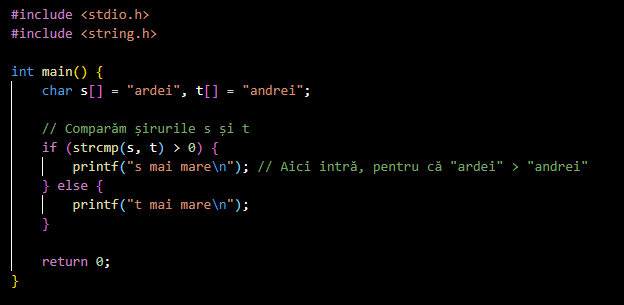


Figura 26. Utilizare strcmp

**Funcția strtok — cum funcționează**

Funcția strtok este folosită pentru a extrage dintr-un șir de caractere mai multe subșiruri (de pildă, cuvinte), separate prin mai multe caractere diferite. Cel mai simplu mod de a vedea cum funcționează funcția este de a urmări cel mai general exemplu:

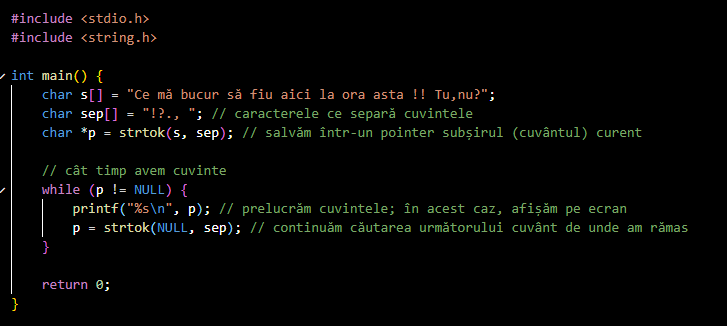


Figura 27. Utilizare strtok

Mai exact, **strtok** se așteaptă prima oară la șirul nostru și la caracterele de separare, după care, de la următoarele apelări, trebuie să primească NULL pentru a continua de unde a rămas. Atunci când funcția strtok returnează NULL, înseamnă că am ajuns la punctul de oprire.

**Atenție**! Șirul inițial s se modifică în urma apelării funcției strtok. Prin urmare, pentru a păstra șirul original, trebuie făcută o copie.

**Pointeri în C++. Tipul char \* (char steluță)**

Să declarăm un șir de caractere:

*char s[] = " testare ";*

Am discutat că s este un tablou unidimensional — un șir de caractere, reprezentat printr-un vector de char. Însă, în esență, s este defapt un pointer, adică atunci când îl apelăm pe s în programul nostru, tot ce facem este să mergem pe poziția unde este salvată informația din s, adică " testare " în cazul nostru. Mai precis de atât, s este un pointer către primul caracter din șirul nostru, adică către caracterul 't'.

Tipul char \* este o metodă de a reține un pointer, însă fără să avem un loc definit unde să salvăm informația (spre deosebire de s, unde am făcut loc pentru a reține "testare").

Un pointer se declară ca mai jos și i se poate atribui o anumită adresă de memorie (spre exemplu, locația de unde începe s), lucru ce îi dă posibilitatea să funcționeze ca șirul de caractere căruia i s-a dat adresa:

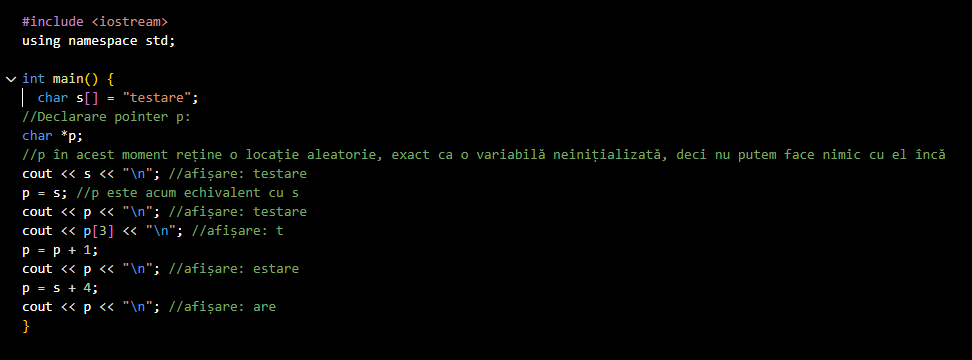


Figura 28. Utilizare pointer

1. Se dă un șir de caractere format din litere mici, litere mari și spații. Să se transforme toate literele cuvântului în litere mari, după care să se afișeze șirul modificat.
2. Se dă un cuvânt format din litere mici ale alfabetului englez, împreună cu două poziții x și y din cuvânt. Să se interschimbe literele aflate pe pozițiile x și y.
3. Iulian, mare iubitor de animale, tocmai a descoperit noul lui animal preferat: **zibelina**.



Fiind uimit de frumusețea ei deosebită, Iulian numește toate cuvintele care încep cu litera z, au lungime pară și sunt formate din consoane și vocale așezate alternativ, cuvinte *frumoase*.

1. Cifrul Cezar (Caesar cipher) este o metodă simplă și foarte populară de criptare a unui text. Cifrul se folosește de o valoare p, iar toate literele textului se înlocuiesc cu litera din alfabet aflată la o distanță de p față de litera curentă.

Spre exemplu, pentru textul aBc și p = 3, litera a se înlocuiește cu d (deoarece d se află la p = 3 poziții față de a), litera B se înlocuiește cu E și litera c se înlocuiește cu f, textul criptat devenind dEf.

* Cerință

Se dă un șir de caractere ce reprezintă o propoziție, împreună cu valoarea p. Cuvintele propoziției, formate din litere mici și mari ale alfabetului englez, sunt despărțite prin câte un spațiu. Să se cripteze textul dat folosind cifrul Cezar.

* Date de intrare

Programul citește de la tastatură, de pe prima linie, șirul format din litere mici și mari ale alfabetului englez și spații, iar de pe a doua linie, numărul natural p.