INFORMATICĂ · CLASA A X-A · ȘIRURI DE CARACTERE · LECȚIA 10

Șiruri de caractere (1): char[], citire, parcurgere, litere și cifre

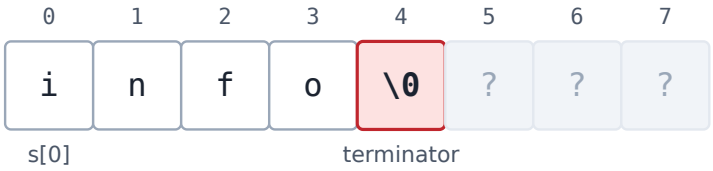
Durată: 60 min Nivel: mate-info Înainte: lecțiile x-02, ix-03

CE VEI ȘTI DUPĂ LECȚIE

- Știi cum se memorează un șir de caractere în C++ (char s[], terminatorul '\0').
- Citești un cuvânt (cin >>) sau o linie întreagă cu spații (cin.getline).
- Parcurgi un șir caracter cu caracter și clasifici caracterele (literă mică/mare, cifră, vocală).
- Transformi literele mici în mari și invers, folosind codurile ASCII.
- Verifici palindromul și numeri aparițiile de litere.

1. Ideea: un text este un vector de caractere

Un cuvânt, o propoziție, o parolă: toate sunt **șiruri de caractere**. Calculatorul le păstrează ca pe un vector în care fiecare căsuță conține un caracter (de fapt, codul lui). La final există un caracter special, `'\0'`, care spune „aici se termină șirul”.



un șir de lungime 4 ocupă 5 căsuțe: literele + '\0'

Șirul în memorie

```
char s[21] = "info";     // poate memora un șir de cel mult 20 de caractere
```

DE CE AȘA?

De ce există `'\0'`? Vectorul `s` are mereu 21 de căsuțe, dar cuvântul poate avea 4 litere. Programul trebuie să știe unde se oprește textul; în loc să țină separat lungimea, C++ pune un semn de sfârșit. De aceea declarăm cu **un caracter în plus** față de lungimea maximă din enunț: ultimul loc este rezervat lui `'\0'`.

DE ȘTIUT

Există și tipul `string` (din `<string>`), mai comod. La Bac, cerințele cu șiruri se rezolvă de obicei cu `char[]` și funcțiile din `<cstring>`, iar baremul le are în vedere pe acestea. Pe pbinfo poți folosi oricare variantă.

2. Citirea unui șir

Instrucțiune	Ce citește	Exemplu (intrare: `ana are mere`)
`cin >> s;`	**un cuvânt** (se oprește la primul spațiu)	`s = "ana"`
`cin.getline(s, 256);`	**o linie întreagă**, cu spații	`s = "ana are mere"`

CAPCANĂ

Dacă înainte de `getline` ai citit un număr cu `cin >> n`, în intrare rămâne caracterul „Enter”, iar `getline` citește o linie goală. Soluție: `cin.get();` între ele (consumă Enter-ul).

3. Parcurgerea

Programul citește o linie și afișează lungimea ei, apoi literele separate prin `-`:

```
#include <iostream>
#include <cstring>
using namespace std;

char s[256];

int main()
{
    cin.getline(s, 256);
    int n = strlen(s);                  // lungimea, fără '\0'; calculată O SINGURĂ DATĂ
    cout << n << '\n';
    for (int i = 0; i < n; i++)         // indicii merg de la 0 la n-1
    {
        cout << s[i];
        if (i < n - 1) cout << '-';
    }
    return 0;
}
```

DE CE AȘA?

De ce `i < n` și nu `i <= n`? Indicii încep de la 0, deci ultimul caracter este `s[n-1]`. `s[n]` este `'\0'`. **De ce `strlen` înaintea buclei?** `strlen` parcurge tot șirul ca să-l măsoare; dacă l-am pune în condiția `i < strlen(s)`, s-ar relua la fiecare pas, deci ar fi $O(n^2)$ în loc de $O(n)$.

Intrare / ieșire

Intrare	Ieșire
`info`	`4` `i-n-f-o`
`a b`	`3` `a- -b`

4. Clasificarea caracterelor: codul ASCII

Fiecare caracter este un număr (codul ASCII). Important: **literele și cifrele sunt scrise în ordine consecutivă**, deci putem face aritmetică cu ele.



= 32; s[i] − 'a' = poziția în alfabet; s[i] − '0' = valoa

ASCII

Verificare	Condiție	Cu `<cctype>`
literă mică	`s[i] >= 'a' && s[i] <= 'z'`	`islower(s[i])`
literă mare	`s[i] >= 'A' && s[i] <= 'Z'`	`isupper(s[i])`
cifră	`s[i] >= '0' && s[i] <= '9'`	`isdigit(s[i])`
vocală	`strchr("aeiouAEIOU", s[i]) != NULL`	n/a

DE CE AȘA?

De ce funcționează `s[i] >= 'a' && s[i] <= 'z'` ? Comparația se face între coduri (`'a'` = 97, `'z'` = 122). Un caracter este literă mică dacă valoarea lui se află între cele două coduri. Nu trebuie să scrii 97 și 122: scrie `'a'` și `'z'`, ca să se vadă ce înseamnă.

```cpp
#include <iostream>
#include <cstring>
using namespace std;

char s[256];

int main()
{
    cin.getline(s, 256);
    int mari = 0, mici = 0, cifre = 0;
    for (int i = 0; s[i] != '\0'; i++)     // parcurgere fără strlen
    {
        if (s[i] >= 'A' && s[i] <= 'Z') mari++;
        else if (s[i] >= 'a' && s[i] <= 'z') mici++;
        else if (s[i] >= '0' && s[i] <= '9') cifre++;
    }
    cout << mari << ' ' << mici << ' ' << cifre;
    return 0;
}
```

Urmărire pentru `Bac 2026!` (s-au omis pașii pe care nu se schimbă nimic):

i	s[i]	clasificare	mari	mici	cifre
0	`B`	literă mare	1	0	0
1	`a`	literă mică	1	1	0
2	`c`	literă mică	1	2	0
3	`␣`	niciuna	1	2	0
4–7	`2 0 2 6`	cifre	1	2	4
8	`!`	niciuna	1	2	4

Intrare / ieșire

Intrare	Ieșire
`Bac 2026!`	`1 2 4`
`INFO`	`4 0 0`

5. Transformări cu coduri ASCII

```cpp
if (s[i] >= 'a' && s[i] <= 'z') s[i] = s[i] - 'a' + 'A';   // mică → mare   (sau s[i] -= 32)
if (s[i] >= 'A' && s[i] <= 'Z') s[i] = s[i] - 'A' + 'a';   // mare → mică   (sau s[i] += 32)
int cifra = s[i] - '0';                                     // '7' → 7
```

DE CE AȘA?

De ce `s[i] - 'a' + 'A'` ? `s[i] - 'a'` dă poziția literei în alfabet (0 pentru `a`, 25 pentru `z`); adunând `'A'` obținem litera mare de pe aceeași poziție. Este același lucru cu a scădea 32 (diferența dintre `'a'` și `'A'`), dar se înțelege mai bine ce face.

```
#include <iostream>
using namespace std;

char s[256];

int main()
{
    cin.getline(s, 256);
    for (int i = 0; s[i]; i++)                    // s[i] != 0 este același lucru cu
s[i] != '\0'
        if (s[i] >= 'a' && s[i] <= 'z')
            s[i] = s[i] - 'a' + 'A';
    cout << s;
    return 0;
}
```

Cifrul lui Cezar

Cezar muta fiecare literă cu `k` poziții în alfabet; la sfârșit se revine la `a`.



% 26 face deplasarea circulară: după 'z' urmează din nou 'a'

Cezar

```
#include <iostream>
using namespace std;

char s[256];

int main()
{
    int k;
    cin >> k;
    cin.get();                        // consumă Enter-ul rămas după k
    cin.getline(s, 256);
    for (int i = 0; s[i]; i++)
    {
        if (s[i] >= 'a' && s[i] <= 'z') s[i] = 'a' + (s[i] - 'a' + k) % 26;
        else if (s[i] >= 'A' && s[i] <= 'Z') s[i] = 'A' + (s[i] - 'A' + k) % 26;
    }
    cout << s;
    return 0;
}
```

DE CE AȘA?

De ce `% 26` ? Alfabetul are 26 de litere. `(poziție + k) % 26` rămâne mereu între 0 și 25, deci după `z` revine la `a`. Fără `%`, `'y' + 3` ar ieși din alfabet. Literele mari și cele mici sunt tratate separat ca să rămână mari, respectiv mici; celelalte caractere nu se schimbă.

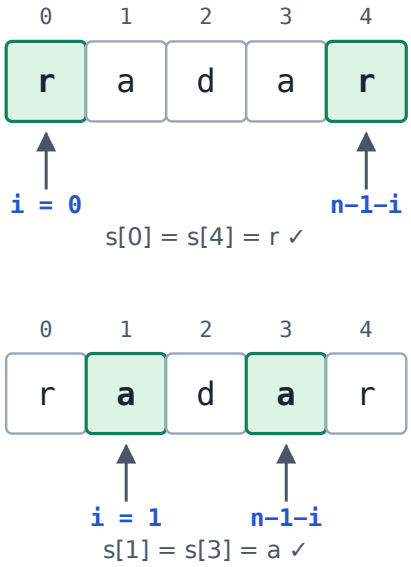
Urmărire (`k = 3`, șirul `xyz`):

s[i]	s[i] – 'a'	+ k	% 26	rezultat
x	23	26	0	a
y	24	27	1	b
z	25	28	2	c

6. Tipare uzuale

Palindrom

Un șir este palindrom dacă se citește la fel în ambele sensuri: comparăm capetele care se apropie de mijloc.



Palindrom

```cpp
#include <iostream>
#include <cstring>
using namespace std;

char s[256];

int main()
{
    cin >> s;
    int n = strlen(s);
    bool pal = true;
    for (int i = 0; i < n / 2; i++)          // doar jumătatea: fiecare pereche o dată
        if (s[i] != s[n - 1 - i]) pal = false;
    cout << (pal ? "DA" : "NU");
    return 0;
}
```

DE CE AȘA?

De ce `n / 2` pași? Perechea `(i, n−1−i)` se verifică o singură dată; dacă mergeam până la `n`, am compara aceleași perechi încă o dată în sens invers. Elementul din mijloc (la `n` impar) se compară cu el însuși, deci nu trebuie verificat.

Frecvența literelor

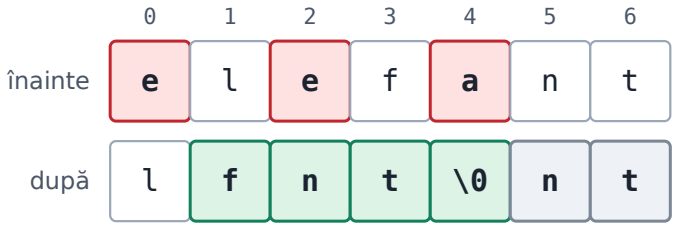
```cpp
#include <iostream>
using namespace std;

char s[256];
int fr[26];

int main()
{
    cin >> s;
    for (int i = 0; s[i]; i++)
        if (s[i] >= 'a' && s[i] <= 'z')
            fr[s[i] - 'a']++;                // litera -> poziția ei în alfabet
    for (int c = 0; c < 26; c++)
        if (fr[c] > 0)
            cout << char('a' + c) << ' ' << fr[c] << '\n';
    return 0;
}
```

Același tipar ca „vectorul de frecvență” de la lecția X.6, doar că indicele se obține din literă cu `s[i] - 'a'`.

Ștergerea unor caractere (doi indici)



s[k] = '\0' taie șirul; ce rămâne după el (gri) nu mai contează

Compactare

```cpp
#include <iostream>
#include <cstring>
using namespace std;

char s[256];

int main()
{
    cin >> s;
    int k = 0;                              // unde scriem
    for (int i = 0; s[i]; i++)              // de unde citim
        if (!strchr("aeiouAEIOU", s[i]))    // păstrăm doar ce NU e vocală
            s[k++] = s[i];
    s[k] = '\0';                            // NU uita terminatorul!
    cout << s;
    return 0;
}
```

Urmărire (`elefant`):

i	s[i]	vocală?	k după pas	începutul șirului (s[0..k−1])
0	`e`	da	0	
1	`l`	nu	1	`l`
2	`e`	da	1	`l`
3	`f`	nu	2	`lf`
4	`a`	da	2	`lf`
5	`n`	nu	3	`lfn`
6	`t`	nu	4	`lfnt`

DE CE AȘA?

De ce `s[k] = '\0'` ? Prin copiere am scris doar în primele `k` poziții; restul șirului vechi a rămas în memorie. Fără terminator, `cout << s` ar afișa și resturile (`lfntant` în exemplul de mai sus).

Intrare / ieșire

Intrare	Ieșire
`elefant`	`lfnt`
`aeiou`	(nimic, șir vid)

7. Greșeli frecvente

Greșeala	Efect	Corect
`char s[10]` pentru un cuvânt de 10 litere	depășire, `'\0'` nu mai încape	dimensiune ≥ lungime + 1
`for (i = 0; i <= strlen(s); i++)`	ajunge și la `'\0'`	`i < strlen(s)`
`strlen(s)` în condiția buclei	$O(n^2)$	calculează `n` înainte
`getline` după `cin >> n` fără `cin.get()`	linie goală	`cin.get();`
`s == "info"`	compară adrese, nu conținut	`strcmp(s, "info") == 0` (lecția următoare)
uiți `s[k] = '\0'` după ștergere	rămân resturi	pune terminatorul
`'a' + 32` crezând că e literă mare	ai litera mică ori alt caracter	`s[i] - 'a' + 'A'`

8. La BAC

> **LA BAC**
>
> La Bac, Subiectul I are aproape mereu o secvență pe un șir („ce se afișează?”), iar Subiectul II sau III cere o prelucrare: numărarea unor litere, transformarea unor caractere, ștergerea sau inserarea. Simulează pe hârtie, scriind sub șir indicii 0, 1, 2, … Ține minte: `'A' = 65`, `'a' = 97`, `'0' = 48` și că expresia `s[i] - 'a'` dă poziția în alfabet.

9. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

1. Ce afișează?

```
char s[] = "bacalaureat";
for (int i = 0; i < strlen(s); i += 2) cout << s[i];
```

2. `s` memorează un cuvânt cu litere mici. Scrie o secvență care transformă în literă mare fiecare vocală.

3. Ce valoare are expresia `s[3] - 'a'` pentru `s = "informatica"`?

4. Ce afișează `cout << 'E' + 1;`, dar `cout << char('E' + 1);`? Explică diferența.

5. Ce dimensiune minimă trebuie să aibă `s` ca să memoreze `bacalaureat`?

6. Pentru `s = "anaa"`, ce află programul de palindrom? Scrie perechile comparate.

7. Cu cifrul lui Cezar și `k = 5`, în ce se transformă `v`? Dar `x`?

8. Din `bacalaureat` se șterg vocalele ca în secțiunea 6. Ce afișează programul dacă **lipsește** linia `s[k] = '\0';`?

9. Scrie condiția care este adevărată dacă `s[i]` este o cifră **impară**.

10. Mai departe (pentru cine vrea mai mult)

`string` și `getline`. În C++ modern: `string s; getline(cin, s);`, cu `s.size()`, `s += "abc"`, `s.substr(i, k)`, `s.find("mat")`. Mult mai comod, dar la Bac trebuie să știi ambele variante.

Analiza frecvenței. Cifrul lui Cezar se sparge ușor: în limba română cea mai frecventă literă este `e`; calculezi `fr[]` pentru textul criptat, găsești litera cea mai frecventă și deduci `k`.

Anagramele. Două cuvinte sunt anagrame dacă au aceleași frecvențe de litere: calculezi `fr1[]` și `fr2[]` și le compari, în O(n).

Hash de șiruri. Pentru a compara rapid mii de șiruri, fiecare șir poate fi transformat într-un număr (hash) cu o formulă de tip `h = h * 31 + s[i]`. Așa funcționează tabelele de dispersie (`unordered_map<string, int>`).

Rezumat

- `char s[N]`, indici 0…n−1, terminator `'\0'`; declară cu un loc în plus.
- `cin >> s` citește un cuvânt; `cin.getline(s, N)` o linie.
- ASCII: `'0' = 48`, `'A' = 65`, `'a' = 97`; literele și cifrele sunt consecutive. De aici: `s[i] - 'a'`, `s[i] - '0'`, diferența 32.
- Clasificare cu comparații de coduri sau `<cctype>`; Cezar cu `% 26`.
- Palindrom: `n / 2` perechi. Frecvență: `fr[s[i] - 'a']++`.
- După ștergeri pune `'\0'`.

Probleme de lucru (pbinfo)

Începe cu acestea, în ordine: sunt alese să urmeze pașii din lecție.

- [ ] **#11 vocale** — ușoară
  https://www.pbinfo.ro/probleme/11/vocale
- [ ] **#84 interschimbarelitere** — ușoară
  https://www.pbinfo.ro/probleme/84/interschimbarelitere
- [ ] **#88 palindrom** — ușoară
  https://www.pbinfo.ro/probleme/88/palindrom
- [ ] **#94 eliminarevocale** — ușoară
  https://www.pbinfo.ro/probleme/94/eliminarevocale

Temă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] #13 **prefixe** — ușoară
  https://www.pbinfo.ro/probleme/13/prefixe
- [ ] #85 **inserareasterisc** — ușoară
  https://www.pbinfo.ro/probleme/85/inserareasterisc
- [ ] #138 **frecventa** — ușoară
  https://www.pbinfo.ro/probleme/138/frecventa
- [ ] #155 **cifmax** — ușoară
  https://www.pbinfo.ro/probleme/155/cifmax
- [ ] #886 **secvcons** — ușoară
  https://www.pbinfo.ro/probleme/886/secvcons
- [ ] #890 **nrvocale** — ușoară
  https://www.pbinfo.ro/probleme/890/nrvocale
- [ ] #903 **cezar** — ușoară
  Cifrul lui Cezar: deplasare circulară în alfabet.
  https://www.pbinfo.ro/probleme/903/cezar
- [ ] #2715 **rot13** — ușoară
  https://www.pbinfo.ro/probleme/2715/rot13

Grile din subiectele de BAC (pbinfo)

Exerciții de tip grilă din variantele de Bacalaureat, rezolvate online pe pbinfo, cu verificare automată a răspunsului.

- [ ] #514 **Exercițiul #514**
  valoarea expresiei strlen("bine")
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=514
- [ ] #66 **Exercițiul #66**
  secvență pe un șir s: ce se afișează
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=66
- [ ] #179 **Exercițiul #179**
  valoarea unei expresii pentru șirul "examen"
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=179
- [ ] #128 **Exercițiul #128**
  expresie care are valoarea 1 dacă și numai dacă șirul s are o anumită proprietate
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=128

- [ ] #333 **Exercițiul #333**
  palindrom: care secvență verifică corect proprietatea
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=333

- [ ] #1418 **Exercițiul #1418**
  ce se afișează după prelucrarea unui șir dat (char t[] = ...)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1418

Citește și exersează în plus

- Codul ASCII https://www.pbinfo.ro/articole/56/codul-ascii
- Șiruri de caractere în C++ https://www.pbinfo.ro/articole/19/siruri-de-caractere-in-cpp
- Tabelul ASCII — toate codurile; găsești 'A' = 65, 'a' = 97, '0' = 48 https://www.asciitable.com/
- cppreference — <cctype> (isalpha, isdigit, toupper...) — funcțiile de clasificare a caracterelor (engleză) https://en.cppreference.com/w/cpp/header/cctype
- Cifrul lui Cezar — istoria și spargerea lui prin analiza frecvenței literelor
  https://ro.wikipedia.org/wiki/Cifrul_lui_Cezar
- Codeforces — probleme cu șiruri (tag: strings) — pentru cine vrea să se antreneze cu probleme internaționale https://codeforces.com/problemset?tags=strings

Răspunsuri

Exersează pe hârtie

1. `bcluet` (pozițiile 0, 2, 4, 6, 8, 10: b, c, l, u, e, t).

2.

```
for (int i = 0; s[i]; i++)
    if (strchr("aeiou", s[i])) s[i] = s[i] - 'a' + 'A';
```

3. `s[3]` este `'o'`, deci `'o' - 'a' = 14`.

4. `cout << 'E' + 1` afișează **70**: adunarea transformă `'E'` (69) în număr întreg, iar rezultatul este de tip `int`. `cout << char('E' + 1)` afișează **F**, pentru că numărul 70 este convertit înapoi în caracter.

5. `bacalaureat` are 11 litere, deci minim **12** (11 + `'\0'`).

6. `n = 4`, deci `n / 2 = 2` perechi: `(s[0], s[3]) = (a, a)` ✓, `(s[1], s[2]) = (n, a)` ×. Programul afișează `NU`.

7. `v` → `a` ((21 + 5) % 26 = 0). `x` → `c` ((23 + 5) % 26 = 2).

8. Rămân în primele 5 poziții `bclrt`, iar restul șirului vechi nu e tăiat: se afișează `bclrtaureat`.

9. `s[i] >= '0' && s[i] <= '9' && (s[i] - '0') % 2 == 1`.