INFORMATICĂ · CLASA A X-A · ȘIRURI DE CARACTERE · LECȚIA 11

Șiruri de caractere (2): funcțiile din cstring și lucrul cu cuvinte

Durată: 60 min Nivel: mate-info Înainte: lecțiile x-10

CE VEI ȘTI DUPĂ LECȚIE

- Folosești strlen, strcpy, strcat, strcmp, strchr, strstr, strtok.
- Împarți un text în cuvinte (cu strtok sau manual) și prelucrezi fiecare cuvânt.
- Compari șiruri și le ordonezi alfabetic (lexicografic).
- Ștergi / inserezi o secvență într-un șir.
- Rezolvi exercițiul cu șiruri de caractere de la Bac (Subiectul II).

1. Funcțiile din `<cstring>`

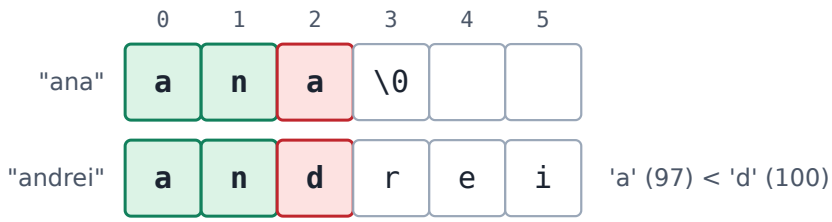
Șirurile nu se pot compara sau copia cu `==` și `=` (aceștia lucrează cu **adrese**). Pentru asta există funcții gata făcute:

Funcție	Ce face	Exemplu
`strlen(s)`	lungimea lui `s`	`strlen("bac")` → 3
`strcpy(d, s)`	copiază `s` în `d`	`strcpy(d, "abc");`
`strcat(d, s)`	lipește `s` la finalul lui `d`	`"ab"` + `"cd"` → `"abcd"`
`strcmp(a, b)`	`<0`, `0`, `>0` după ordinea alfabetică	`strcmp("ana", "andrei") < 0`
`strchr(s, c)`	adresa primei apariții a caracterului `c`, sau `NULL`	`strchr("aeiou", x)`
`strstr(s, t)`	adresa primei apariții a șirului `t` în `s`, sau `NULL`	`strstr(s, "bac")`
`strtok(s, sep)`	taie `s` în „bucăți” separate de caracterele din `sep`	vezi secțiunea 3

ATENȚIE — GREȘELI FRECVENTE

`if (a == b)` pentru două tablouri `char` compară adresele lor (mereu diferite), deci condiția este **falsă** chiar dacă textele sunt identice. Corect: `strcmp(a, b) == 0`. La fel, `a = b` nu copiază: folosește `strcpy(a, b)`.

strcmp: ordinea lexicografică



se compară caracter cu caracter până la prima diferență

strcmp

`strcmp` se oprește la **prima poziție** unde cele două șiruri diferă și compară codurile. De aceea literele mari sunt „mai mici” decât cele mici (`'Z'` = 90 < `'a'` = 97).

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

char a[101], b[101];

int main()
{
    cin >> a >> b;
    int r = strcmp(a, b);
    if (r < 0) cout << '<';
    else if (r == 0) cout << '=';
    else cout << '>';
    return 0;
}
```

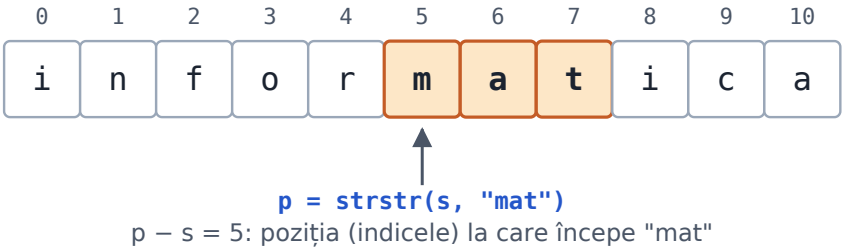
DE CE AȘA?

De ce `r < 0` și nu `r == -1` ? Standardul spune doar că rezultatul este negativ, zero sau pozitiv; valoarea exactă depinde de compilator. Comparația cu 0 funcționează peste tot.

Intrare / ieșire

Intrare	Ieșire
`ana andrei`	`<`
`bac bac`	`=`
`mere ana`	`>`

strchr și strstr: unde apare?



strstr

Ambele întorc o **adresă** (pointer) sau `NULL`. Poziția (indicele) se obține scăzând adresa de început a șirului: `p - s`.

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

char s[256], t[256];

int main()
{
    cin >> s >> t;
    char *p = strstr(s, t);
    if (p == NULL) cout << -1;
    else cout << p - s;            // poziția la care începe t în s
    return 0;
}
```

DE CE AȘA?

De ce `p - s`? `p` este adresa lui `s[5]`, iar `s` este adresa lui `s[0]`; diferența dintre două adrese consecutive din același tablou este numărul de căsuțe dintre ele, deci exact indicele.

CAPCANĂ

Pentru `strchr(s, '\0')` rezultatul **nu** este `NULL`: terminatorul face parte din șir. De aceea, când parcurgi un șir și apelezi `strchr("aeiou", s[i])`, oprește bucla la `s[i] == '\0'`.

Intrare / ieșire

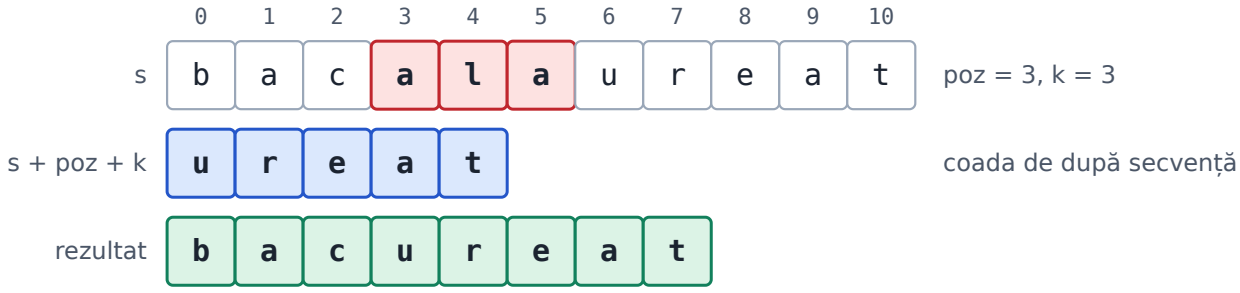
Intrare	Ieșire
`informatica` `mat`	`5`
`informatica` `xyz`	`-1`

2. Șiruri ca „bucăți”: strcpy, strcat și s + poz

- `strcpy(d, s)` copiază și `'\0'`; `d` trebuie să fie destul de mare.

- `strcat(d, s)` caută sfârșitul lui `d` și lipește `s`.
- `s + poz` este „șirul care începe la poziția `poz`”: pentru `s = "bacalaureat"`, `s + 3` este `"alaureat"`.

Ștergerea a k caractere de la poziția poz



Ștergere

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

char s[256], aux[256];

int main()
{
    int poz, k;
    cin.getline(s, 256);
    cin >> poz >> k;
    strcpy(aux, s + poz + k);      // 1) salvăm coada de după secvența ștearsă
    strcpy(s + poz, aux);          // 2) o lipim peste locul secvenței
    cout << s;
    return 0;
}
```

DE CE AȘA?

De ce copiem întâi în `aux`? `strcpy(s + poz, s + poz + k)` copiază din șirul `s` tot în `s`, peste zone care se suprapun, iar standardul nu garantează rezultatul. Cu un șir auxiliar nu avem acest risc.

Inserarea șirului t la poziția poz

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

char s[256], t[256], aux[256];

int main()
{
    int poz;
    cin.getline(s, 256);
    cin >> t >> poz;
    strcpy(aux, s + poz);          // 1) salvăm coada
    strcpy(s + poz, t);            // 2) scriem t peste coadă (șirul se taie aici)
    strcat(s, aux);                // 3) lipim coada înapoi
    cout << s;
    return 0;
}
```

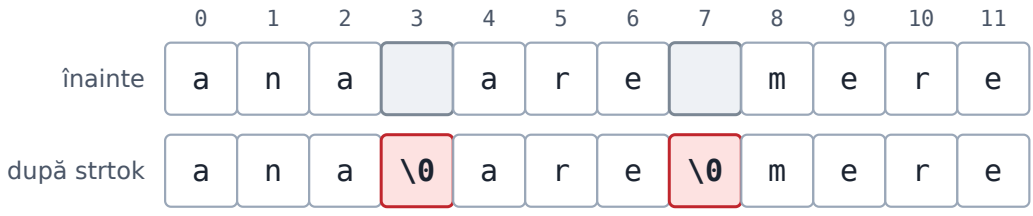
Urmărire (`s = bacalaureat` , `t = XY` , `poz = 3`):

pas	aux	s
(început)		`bacalaureat`
1	`alaureat`	`bacalaureat`
2	`alaureat`	`bacXY`
3	`alaureat`	`bacXYalaureat`

Intrare / ieșire

Intrare	Ieșire
`bacalaureat` `3 3` (ștergere)	`bacureat`
`bacalaureat` `XY 3` (inserare)	`bacXYalaureat`

3. Împărțirea în cuvinte cu strtok



fiecare cuvânt devine un șir de sine stătător, terminat cu '\0'

strtok

`strtok` caută primul cuvânt, pune `'\0'` după el și întoarce adresa lui. La apelurile următoare, cu `NULL`, continuă de unde a rămas.

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

char s[256];

int main()
{
    cin.getline(s, 256);
    char *p = strtok(s, " ,.");          // primul cuvânt; separatori: spațiu, virgulă, punct
    while (p != NULL)
    {
        cout << p << ' ' << strlen(p) << '\n';
        p = strtok(NULL, " ,.");         // NULL = continuă în același șir
    }
    return 0;
}
```

Urmărire pentru `ana are mere`:

apel	p arată spre	strlen(p)	rămâne de prelucrat
`strtok(s, " ,.")`	`ana`	3	`are mere`
`strtok(NULL, …)`	`are`	3	`mere`
`strtok(NULL, …)`	`mere`	4	(nimic)
`strtok(NULL, …)`	`NULL`		sfârșit

DE CE AȘA?

De ce primul apel are `s`, iar următoarele `NULL`? `strtok` ține minte singur unde a rămas în șirul precedent. Când îi dai `NULL` îi spui „continuă”. Dacă dai din nou `s`, ar relua de la început.

CAPCANĂ

`strtok` **modifică** șirul original (pune `'\0'` în locul separatorilor). Dacă mai ai nevoie de text, fă mai întâi o copie cu `strcpy`.

Cel mai lung cuvânt

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

char s[256], maxim[256];

int main()
{
    cin.getline(s, 256);
    char *p = strtok(s, " ");
    while (p)
    {
        if (strlen(p) > strlen(maxim))      // strict: la egalitate rămâne primul
            strcpy(maxim, p);              // copiem conținutul, nu adresa
        p = strtok(NULL, " ");
    }
    cout << maxim;
    return 0;
}
```

DE CE AȘA?

De ce `strcpy(maxim, p)` și nu `maxim = p`? După `strtok`, `p` arată în interiorul lui `s`. Dacă am păstra doar adresa, conținutul s-ar schimba când `strtok` modifică șirul. Copia ne garantează valoarea.

Numărarea cuvintelor, fără strtok

Un cuvânt **începe** la poziția `i` dacă `s[i]` nu e spațiu și (`i == 0` sau `s[i-1]` e spațiu):

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

char s[256];

int main()
{
    cin.getline(s, 256);
    int cuvinte = 0;
    for (int i = 0; s[i]; i++)
        if (s[i] != ' ' && (i == 0 || s[i - 1] == ' '))
            cuvinte++;
    cout << cuvinte;
    return 0;
}
```

DE CE AȘA?

De ce numărăm „începuturile", nu spațiile? Numărul de spații nu este numărul de cuvinte dacă între cuvinte sunt mai multe spații sau dacă textul începe/se termină cu spațiu. Fiecare cuvânt are un singur început, oricâte spații l-ar înconjura.

4. Ordonarea alfabetică a cuvintelor

Memorăm cuvintele într-o matrice de caractere (fiecare linie = un cuvânt) și sortăm cu `strcmp`:

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

char s[256], cuv[101][31], aux[31];

int main()
{
    cin.getline(s, 256);
    int n = 0;
    for (char *p = strtok(s, " "); p; p = strtok(NULL, " "))
        strcpy(cuv[n++], p);

    for (int i = 0; i < n - 1; i++)
        for (int j = i + 1; j < n; j++)
            if (strcmp(cuv[i], cuv[j]) > 0)           // cuv[i] vine după cuv[j]:
interschimbăm
            {
                strcpy(aux, cuv[i]);
                strcpy(cuv[i], cuv[j]);
                strcpy(cuv[j], aux);
            }

    for (int i = 0; i < n; i++) cout << cuv[i] << ' ';
    return 0;
}
```

DE CE AȘA?

De ce o matrice `cuv[101][31]`? Fiecare linie a matricei este un șir (`char[31]`), deci `cuv[i]` poate fi dat direct lui `strcmp` și `strcpy`. 31 = 30 de litere + `'\0'`. Interschimbarea se face cu trei `strcpy`, la fel ca la numere, dar cu șirul `aux`.

5. Exercițiu tip Bac (Subiectul II)

„Un text are cel mult 100 de caractere, iar cuvintele sale sunt formate din litere mici și separate prin câte un spațiu. Scrieți un program care citește textul și afișează, pe câte o linie, cuvintele cu cel puțin două vocale. Dacă nu există, se afișează `nu exista`."

Pașii: `getline` → `strtok` → numeri vocalele din fiecare cuvânt → afișezi dacă ≥ 2 → un indicator `gasit` pentru mesajul final.

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

char s[101];

int main()
{
    cin.getline(s, 101);
    bool gasit = false;
    for (char *p = strtok(s, " "); p; p = strtok(NULL, " "))
    {
        int voc = 0;
        for (int i = 0; p[i]; i++)              // p[i] != 0: nu apelăm strchr pe '\0'
            if (strchr("aeiou", p[i])) voc++;
        if (voc >= 2)
        {
            cout << p << '\n';
            gasit = true;
        }
    }
    if (!gasit) cout << "nu exista";
    return 0;
}
```

LA BAC

Baremul punctează separat: declararea variabilelor, citirea, identificarea cuvintelor, verificarea proprietății, afișarea și tratarea cazului „nu există”. Chiar dacă nu iese perfect, scrie toate etapele.

Intrare / ieșire

Intrare	Ieșire

`ana are mere si pere`	`ana`
	`are`
	`mere`
	`pere`

`pt si`	`nu exista`

6. Greșeli frecvente

Greșeala	Efect	Corect
`if (a == b)` pe șiruri	mereu fals	`strcmp(a, b) == 0`
`a = b`	nu copiază (eroare de compilare)	`strcpy(a, b)`
`maxim = p` după `strtok`	erori de tip	`strcpy(maxim, p)`
`strcpy` într-un șir prea mic	depășire de memorie	dimensiune suficientă

Greșeala	Efect	Corect
`strtok` direct pe textul de care mai ai nevoie	textul e tăiat	copie înainte
`strchr(..., s[i])` fără a opri la `'\0'`	numără și terminatorul	`for (i = 0; s[i]; i++)`
`strcmp(...) == -1`	nu merge pe toate compilatoarele	`< 0`

7. La BAC

LA BAC

Funcțiile `strlen`, `strcpy`, `strcat`, `strcmp`, `strchr`, `strstr` apar la Subiectul I („ce se afișează?"). La Subiectul II cu text, `strtok` cu `getline` rezolvă „cuvintele unui text". Atenție la indicii: `strstr(s, t) - s` este poziția, numerotată de la 0.

8. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

1. Cât este `strlen("info bac")`?

2. Ce semn are `strcmp("ana", "anca")`? Explică.

3. Pentru `char s[] = "informatica";`, ce afișează `cout << strstr(s, "mat");`? Dar `cout << strstr(s, "mat") - s;`?

4. Ce afișează `cout << strchr(s, 'a') - s;` pentru același `s`?

5. Ce afișează?

```
char s[] = "a,b;;c", *p = strtok(s, ",;");
while (p) { cout << p << '\n'; p = strtok(NULL, ",;"); }
```

6. Care este valoarea lui `s` după `strcpy(s, s + 3);` dacă înainte `s = "bacalaureat"`? Ce operație a făcut?

7. Inserează `XY` la poziția 2 în `info`, urmărind pașii din tabel.

8. Ordonează lexicografic cuvintele `ar`, `Ana`, `mere`, `Zoe` cu `strcmp`.

9. De ce `if (a == b)` este greșit pentru două tablouri `char a[10], b[10]` cu același conținut?

10. Ce afișează?

```
char s[20] = "info";
strcat(s, "matica");
cout << strlen(s) << ' ';
strcpy(s + 2, "X");
cout << s;
```

9. Mai departe (pentru cine vrea mai mult)

Căutarea rapidă într-un text. `strstr` caută în O(n·m) în cel mai rău caz. Algoritmul **KMP** (Knuth–Morris–Pratt) folosește „funcția prefix" și o face în O(n + m). Merită studiat după clasa a X-a.

Distanța dintre cuvinte. Distanța **Levenshtein** (câte inserări, ștergeri sau înlocuiri trebuie pentru a transforma un cuvânt în altul) stă la baza corectoarelor ortografice și se calculează cu programare dinamică pe o matrice (lecția X.9 ți-a arătat cum se umple o matrice din vecini).

`stringstream`. Alternativă modernă la `strtok`: `stringstream ss(linie); string cuvant; while (ss >> cuvant) { ... }`. Nu modifică textul și lucrează cu `string`.

Text în fișiere. Combinat cu lecția X.1: `getline(fin, linie)` pe un fișier cu mii de rânduri, apoi `strtok` pe fiecare.

Rezumat

- `strlen`, `strcpy`, `strcat`, `strcmp`, `strchr`, `strstr`, `strtok`.
- Compari cu `strcmp(a, b) == 0` și copiezi cu `strcpy`; nu cu `==` și `=`.
- Poziția unei apariții: `p - s`. Sufixul de la poziția `poz`: `s + poz`.
- Cuvinte: `p = strtok(s, sep); while (p) { …; p = strtok(NULL, sep); }`.
- Sortare de cuvinte: matrice de caractere + `strcmp` + trei `strcpy`.

Probleme de lucru (pbinfo)

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

- [ ] #136 **numararecuvinte** — ușoară
  https://www.pbinfo.ro/probleme/136/numararecuvinte
- [ ] #973 **cuvinte1** — ușoară
  https://www.pbinfo.ro/probleme/973/cuvinte1
- [ ] #12 **initiale** — medie
  https://www.pbinfo.ro/probleme/12/initiale
- [ ] #199 **eliminaresubsir** — medie
  strstr + ștergere repetată.
  https://www.pbinfo.ro/probleme/199/eliminaresubsir

Temă (pbinfo)

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

- [ ] #137 **afisarelitere** — ușoară
  https://www.pbinfo.ro/probleme/137/afisarelitere
- [ ] #156 **eliminare1** — ușoară
  https://www.pbinfo.ro/probleme/156/eliminare1

- [ ] #609 **dublarevocale** — ușoară
  https://www.pbinfo.ro/probleme/609/dublarevocale
- [ ] #891 **pasareste** — ușoară
  https://www.pbinfo.ro/probleme/891/pasareste
- [ ] #975 **cuvinte2** — ușoară
  https://www.pbinfo.ro/probleme/975/cuvinte2
- [ ] #2828 **acronim** — ușoară
  Subiect de Bac.
  https://www.pbinfo.ro/probleme/2828/acronim
- [ ] #3112 **bac-cuvinte** — ușoară
  Subiect de Bac.
  https://www.pbinfo.ro/probleme/3112/bac-cuvinte
- [ ] #188 **inlocuirecuvant** — medie
  https://www.pbinfo.ro/probleme/188/inlocuirecuvant

Pentru cei care vor mai mult

Probleme mai grele sau de concurs, pentru cine vrea să meargă dincolo de BAC.

- [ ] #101 **sortarecuvinte** — medie
  https://www.pbinfo.ro/probleme/101/sortarecuvinte
- [ ] #96 **palindrommaxim** — medie
  https://www.pbinfo.ro/probleme/96/palindrommaxim

Grile din subiectele de BAC (pbinfo)

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

- [ ] #83 **Exercițiul #83**
  ce se afișează după o secvență cu funcții din <cstring> (char s[50], x[50])
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=83
- [ ] #152 **Exercițiul #152**
  valoarea expresiei strlen(s) pentru un șir dat
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=152
- [ ] #208 **Exercițiul #208**
  șirul "bacalaureat": ce se afișează după o instrucțiune cu funcții de șiruri
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=208

- [ ] #370 **Exercițiul #370**
  șirul CARACATITA: valoarea unei expresii
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=370

- [ ] #374 **Exercițiul #374**
  șirul ABAC: care dintre expresii are o anumită valoare
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=374

- [ ] #970 **Exercițiul #970**
  char s[] = "info"; ce afișează o expresie cu strchr
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=970

- [ ] #1186 **Exercițiul #1186**
  ce se afișează în urma unei secvențe cu strcpy / strcat
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1186

- [ ] #1548 **Exercițiul #1548**
  variabilele s, x, y: ce expresie dă rezultatul cerut
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1548

Citește și exersează în plus

- Șiruri de caractere în C++ https://www.pbinfo.ro/articole/19/siruri-de-caractere-in-cpp
- cppreference — <cstring> — toate funcțiile pentru șiruri C, cu exemple (engleză)
  https://en.cppreference.com/w/cpp/header/cstring
- cppreference — strtok — comportamentul exact al lui strtok și avertismentele
  https://en.cppreference.com/w/cpp/string/byte/strtok
- CP-Algorithms — Funcția prefix (KMP) — pentru cine vrea să înțeleagă cum se caută rapid un cuvânt într-un text https://cp-algorithms.com/string/prefix-function.html
- Distanța Levenshtein — cum măsori cât de „asemănătoare" sunt două cuvinte (corectoare ortografice) https://ro.wikipedia.org/wiki/Distanța_Levenshtein

Răspunsuri

Exersează pe hârtie

1. `8` (4 + 1 + 3; spațiul se numără).

2. Negativ. Primele două caractere sunt egale (`a`, `n`); apoi `'a'` (97) < `'c'` (99), deci `"ana"` este înaintea lui `"anca"`.

3. Primul afișează `matica` (șirul de la prima apariție), al doilea `5` (`m` este pe poziția 5: i0 n1 f2 o3 r4 m5).

4. `6` (primul `a` din `informatica` e la poziția 6).

5. Pe trei rânduri: `a`, `b`, `c`. Separatorii consecutivi (`;;`) nu produc cuvinte vide.

6. `s = "alaureat"`: s-au șters primele 3 caractere (s-a copiat sufixul `s + 3` peste începutul lui `s`). Atenție: zonele se suprapun, iar standardul nu garantează rezultatul; varianta sigură folosește un șir auxiliar.

7. `aux = s + 2 = "fo"`; `s = "in"` apoi `"inXY"`; `strcat` → **`inXYfo`**.

8. Literele mari au coduri mai mici: `Ana`, `Zoe`, `ar`, `mere`.

9. `a` și `b` sunt adrese de început ale unor tablouri diferite. Comparația `a == b` compară adresele (mereu diferite), nu conținutul. Se folosește `strcmp(a, b) == 0`.

10. `10 inX`: după `strcat`, `s = "infomatica"` (10 caractere); `strcpy(s + 2, "X")` scrie `X` și `'\0'` începând cu poziția 2, deci `s = "inX"`.