

CURS 06– PP

ȘIRURI DE CARACTERE

Șir de caractere = o succesiune de caractere care se termină cu caracterul NUL ('\\0') = un tablou unidimensional cu elemente de tip char

Exemplu:

```
char s[101], t[21]; //trebuie alocat un element în plus
                    //pentru '\\0'
```

```
char s[] = "testare";
char s[] = {'t', 'e', 's', 't', 'a', 'r', 'e', '\\0'};
```

0	1	2	3	4	5	6	7
't'	'e'	's'	't'	'a'	'r'	'e'	'\\0'

1 caracter = 1 octet = codul ASCII al caracterului respectiv

Constante de tip char: 'A', 'b', '@', '?'

Constante de tip șir de caractere: "A", "bac", "@", "@gmail.com"

Observație: 'A' ≠ "A"

'A'	"A"	
65	0	1
1 octet	'A' -> 65	'\\0' -> 0
	2 octeți	

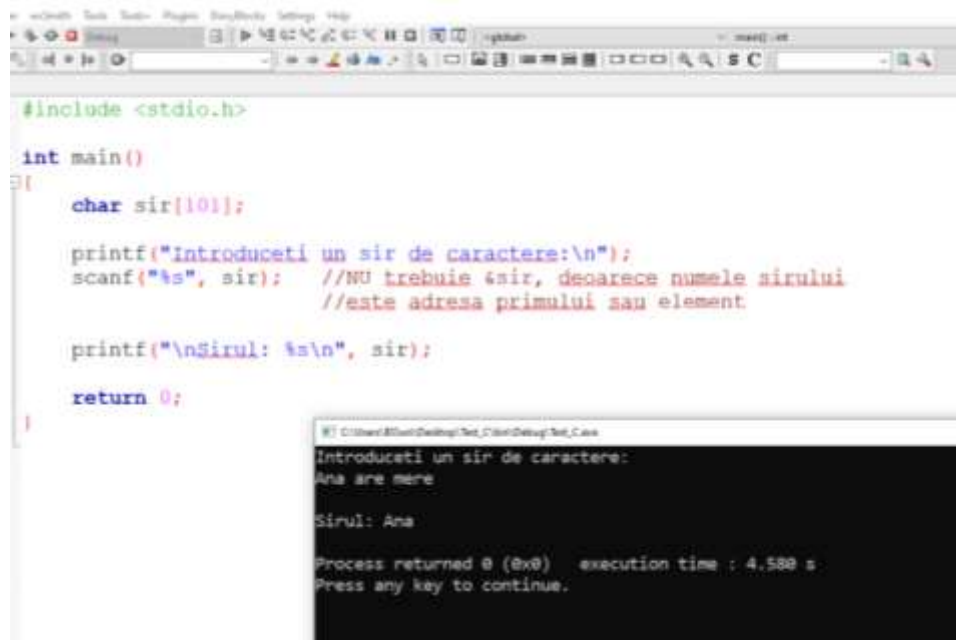
Observație: 'a' > 'A'

Transformarea unui caracter într-un șir:

```
char sir[2];
sir[0] = 'A';
sir[1] = '\\0';
```

Citirea șirurilor de caractere de la tastatură:

- folosind funcția `scanf` cu specificatorul `%s`, dar citirea se va opri la primul separator (de obicei, spațiu)



The screenshot shows a C program in a code editor and its execution in a terminal. The code defines a character array `sir` of size 101, prompts the user to enter a string, and uses `scanf("%s", sir)` to read the input. The terminal output shows the user entering "Ana are mere" and the program printing "Sirul: Ana".

```
#include <stdio.h>

int main()
{
    char sir[101];

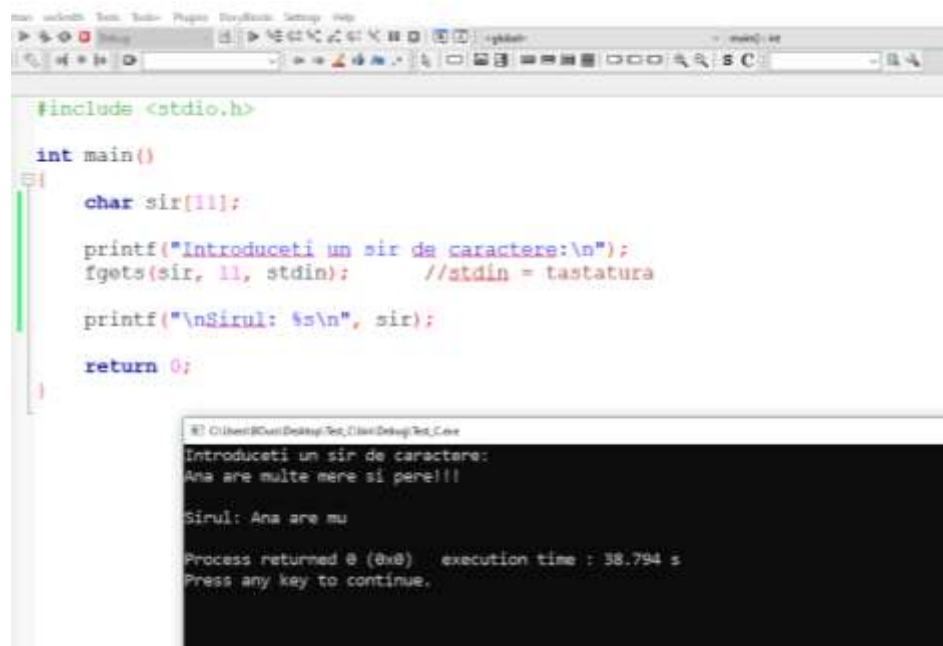
    printf("Introduceti un sir de caractere:\n");
    scanf("%s", sir); //NU trebuie %sir, deoarece numele sirului
                     //este adresa primului sau element

    printf("\nSirul: %s\n", sir);

    return 0;
}
```

Introduceti un sir de caractere:
Ana are mere
Sirul: Ana
Process returned 0 (0x0) execution time : 4.580 s
Press any key to continue.

- folosind funcția `fgets`:
`char* fgets(char *șir, int nr_max_caractere, stdin)`



The screenshot shows a C program in a code editor and its execution in a terminal. The code defines a character array `sir` of size 11, prompts the user to enter a string, and uses `fgets(sir, 11, stdin)` to read the input. The terminal output shows the user entering "Ana are multe mere si pere!!!" and the program printing "Sirul: Ana are mu".

```
#include <stdio.h>

int main()
{
    char sir[11];

    printf("Introduceti un sir de caractere:\n");
    fgets(sir, 11, stdin); //stdin = tastatura

    printf("\nSirul: %s\n", sir);

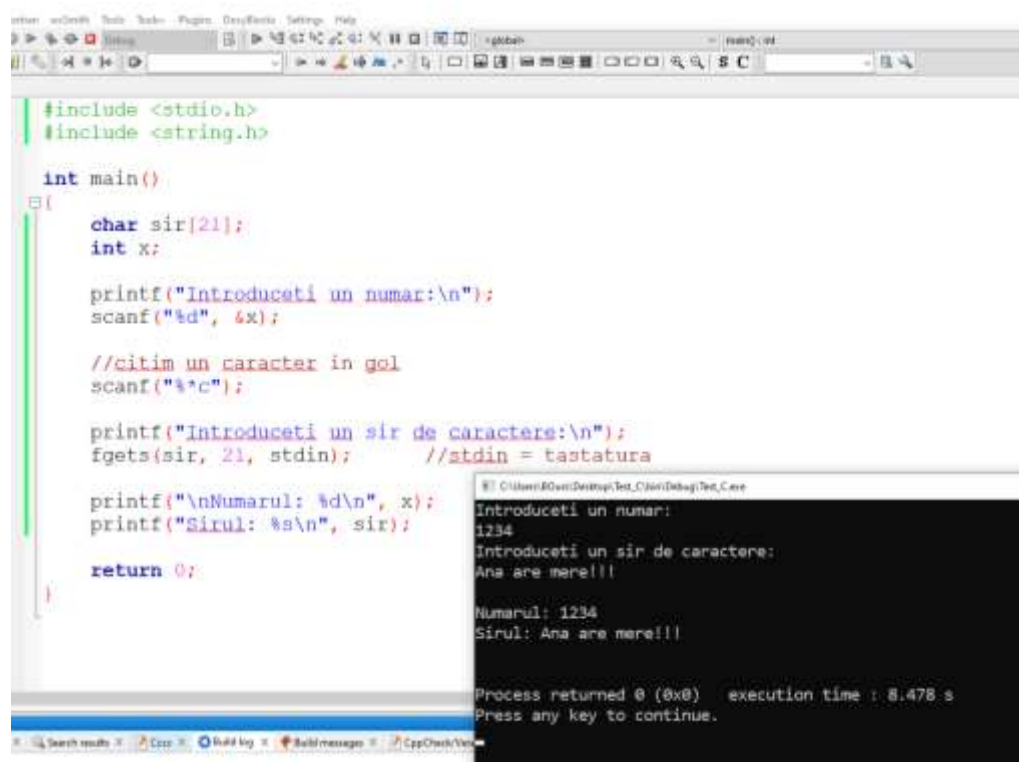
    return 0;
}
```

Introduceti un sir de caractere:
Ana are multe mere si pere!!!
Sirul: Ana are mu
Process returned 0 (0x0) execution time : 38.794 s
Press any key to continue.

Atenție, funcția `fgets` păstrează la sfârșitul șirului caracterul `'\n'` corespunzător tastei Enter dacă nu s-a atins lungimea maximă a șirului, deci trebuie să îl eliminăm noi!!!

```
if(sir[strlen(sir)-1] == '\n')
    sir[strlen(sir)-1] = '\0';
```

Atenție, în cazul în care citim un număr de la tastatură, caracterul `'\n'` corespunzător tastei Enter va rămâne în buffer-ul tastaturii, deci, dacă vom citi imediat un șir de caractere, acesta va fi format doar din acel caracter `'\n'` !!! Pentru a evita această problemă, vom efectua o citire în gol a unui caracter, după ce vom citi numărul!



```
#include <stdio.h>
#include <string.h>

int main()
{
    char sir[21];
    int x;

    printf("Introduceti un numar:\n");
    scanf("%d", &x);

    //citim un caracter in gol
    scanf("%c");

    printf("Introduceti un sir de caractere:\n");
    fgets(sir, 21, stdin); //stdin = tastatura

    printf("\nNumarul: %d\n", x);
    printf("Sirul: %s\n", sir);

    return 0;
}
```

Output from console:

```
Introduceti un numar:
1234
Introduceti un sir de caractere:
Ana are mere!!!

Numarul: 1234
Sirul: Ana are mere!!!

Process returned 0 (0x0)   execution time : 8.478 s
Press any key to continue.
```

Aceeași problemă apare și dacă vrem să citim de la tastatură două caractere despărțite printr-un spațiu!!!

```
char a, b;
```

```
printf("Introduceti doua caractere:\n");
//citim in gol separatorul dintre cele doua caractere
scanf("%c%c", &a, &b); //scanf("%c %c", &a, &b);
printf("\nCaracterele: %c %c\n", a, b);
```

Observație:

Un șir de caractere este, de fapt, o succesiune de caractere care începe la o adresă (un pointer) de tip `char*` și se termină cu caracterul NUL (`'\0'`)!!!

Exemplu:

`char s[10];`

S	0	1	2	3	4	5	6	7	8	9	indici (poziții)
	s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]	accesare directă
											accesare indirectă
	*s	*(s+1)	*(s+2)	*(s+3)	*(s+4)	*(s+5)	*(s+6)	*(s+7)	*(s+8)	*(s+9)	
	s	s+1	s+2	s+3	s+4	s+5	s+6	s+7	s+8	s+9	adrese

Exemplu:

Să se afișeze caracterele, sufixele și prefixele dintr-un șir de caractere.

```
#include <stdio.h>
#include <string.h>

int main()
{
    char sir[] = "exemplu", aux;
    int i;

    printf("Caracterele sirului:\n");
    for(i = 0; i < strlen(sir); i++)
        printf("%c\n", sir[i]);

    printf("\nCaracterele sirului:\n");
    for(i = 0; i < strlen(sir); i++)
        printf("%c\n", *(sir+i));

    printf("\nSufixe sirului:\n");
    for(i = 0; i < strlen(sir); i++)
        printf("%s\n", sir+i);

    printf("\nSufixe sirului:\n");
    for(i = strlen(sir)-1; i >= 0; i--)
        printf("%s\n", sir+i);
}
```

```

printf("\nPrefixele sirului:\n");
for(i = 1; i <= strlen(sir); i++)
{
    aux = sir[i];
    sir[i] = '\0';
    printf("%s\n", sir);
    sir[i] = aux;
}

return 0;
}

```

Biblioteca string.h

Biblioteca `string.h` conține funcții predefinite pentru manipularea șirurilor de caractere.

Funcții diverse:

- **`unsigned int strlen(char *sir)`** – furnizează numărul de caractere din șirul dat ca parametru (lungimea sa), fără `'\0'`. De exemplu, `strlen("test") = 4`.

Funcții pentru comparare lexicografică:

- **`unsigned int strcmp(char *sir_1, char *sir_2)`** – compară lexicografic (`<`) conținuturile celor două șiruri și returnează un număr întreg, astfel:
 - un număr strict negativ dacă `sir_1 < sir_2`
 - un număr strict pozitiv dacă `sir_1 > sir_2`
 - numărul 0 dacă `sir_1 == sir_2`

Exemple:

- `strcmp("exemplu", "exemplu") = 0`
- `strcmp("exemplu", "Exemplu") > 0`
- `strcmp("exemplu", "exemplu") > 0`
- `strcmp("examinator", "exemplu") < 0`

Atenție, o literă mică este strict mai mare decât o literă mare!!!

- **unsigned int strncmp(char *sir_1, char *sir_2, int n)** – compară lexicografic doar primele n caractere din cele două șiruri:

Exemple:

- `strncmp("exemplu", "exemplu", 6) > 0`
- `strncmp("exemplu", "exemplu", 4) = 0`
- `strncmp("examinator", "exemplu", 5) < 0`
- `strncmp("examinator", "exemplu", 2) = 0`

Funcții pentru copierea șirurilor:

- **char* strcpy(char *destinație, char *sursa)** – copiază șirul sursă în șirul destinație (inclusiv '\0'). Funcția returnează șirul destinație. **Atenție, șirul destinație TREBUIE să aibă alocată o zonă de memorie suficientă pentru a memora șirul sursă, inclusiv caracterul '\0' !!!**

Exemplu:

Alocarea dinamică a unui șir înainte de a se copia alt șir în el:

```
#include <stdio.h>
#include <string.h>

int main()
{
    char *s, t[21] = "exemplu";

    //s = (char*)malloc((strlen(t) + 1)*sizeof(char));

    //intotdeauna sizeof(char) = 1, deci putem scrie mai simplu
    s = (char*)malloc(strlen(t) + 1);
    strcpy(s, t);

    printf("Sirul s: %s\n", s);
    printf("Sirul t: %s\n", s);

    free(s);

    return 0;
}
```

Exemplu:

Utilizarea valorii returnate de funcția strcpy într-o copiere multiplă:

```
char s[51], t[51], v[51];
strcpy(s, strcpy(t, strcpy(v, "exemplu")));
```

- **char* strncpy(char *destinație, char *sursa, int n)** – copiază primele *n* caractere din șirul sursă în șirul destinație, **fără a adăuga '\0' la sfârșit!!!** Dacă valoarea lui *n* este strict mai mică decât strlen(sursa), atunci va trebui să adăugăm explicit '\0' la sfârșitul șirului destinație, altfel caracterul '\0' va fi copiat automat!

```
int main()
{
    char s[21], t[21] = "exemplu";

    strncpy(s, t, 3);
    //nu este corect s[strlen(s)] = '\0', deoarece
    //șirul s nu este încă închis (nu are '\0' la sfârșit)!!!

    //închidem șirul s folosind al treilea parametru din strncpy
    s[3] = '\0';

    printf("Șirul s: %s\n", s);
    return 0;
}
```

Exemplu:

Să se afișeze toate prefixele unui șir:

```
#include <stdio.h>
#include <string.h>

int main()
{
    char sir[21], aux[21];
    int i;

    printf("Șirul:\n");
    fgets(sir, 21, stdin);

    //eliminăm eventualul caracter '\n' de la sfârșitul șirului
    if(sir[strlen(sir)-1] == '\n')
        sir[strlen(sir)-1] = '\0';

    printf("\nPrefixele șirului:\n");
    for(i = 1; i <= strlen(sir); i++)
    {
        strncpy(aux, sir, i);
        aux[i] = '\0';
        printf("%s\n", aux);
    }
}
```

```

    return 0;
}

```

Funcții pentru concatenare:

- **char* strcat(char *destinație, char *sursa)** – concatenează șirul sursă la sfârșitul șirului destinație. Atenție, șirul destinație TREBUIE să aibă alocată o zonă de memorie suficientă pentru a memora șirul sursă și șirul destinație, inclusiv caracterul '\0'!!! De asemenea, sursa și destinația ar trebui să nu se suprapună (vezi exemplul cu strcat(s,s))!

Exemple:

```
char s[21] = "test", t[21] = "simplu";
```

- `strcat(s, t) = "testsimplu"`
- `strcat(s, "greu") = "testgreu"`
- `strcat(t, s) = "simplutest"`

Concatenare multiplă:

```

#include <stdio.h>
#include <string.h>

int main()
{
    char s[21] = "test", t[21] = "simplu";

    strcat(strcat(s, " "), t);

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

    return 0;
}

```

- **char* strncat(char *destinație, char *sursa, int n)** – concatenează primele *n* caractere din șirul sursă în șirul destinație, adăugând și '\0' la sfârșitul șirului destinație!!!

Exemple:

```
char s[21] = "test", t[21] = "simplu";
```

- `strncat(s, t, 3) = "testsim"`
- `strncat(s, t, 10) = "testsimplu"`