

CURS 07 – PP

ȘIRURI DE CARACTERE

Biblioteca `string.h`

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

Funcții diverse:

- `unsigned int strlen(char *sir)`

Funcții pentru compararea lexicografică:

- `unsigned int strcmp(char *sir_1, char *sir_2)`
- `unsigned int strncmp(char *sir_1, char *sir_2, int n)`

Funcții pentru copiere:

- `char* strcpy(char *destinatie, char *sursa)`
- `char* strncpy(char *destinatie, char *sursa, int n)`

Funcții pentru concatenare:

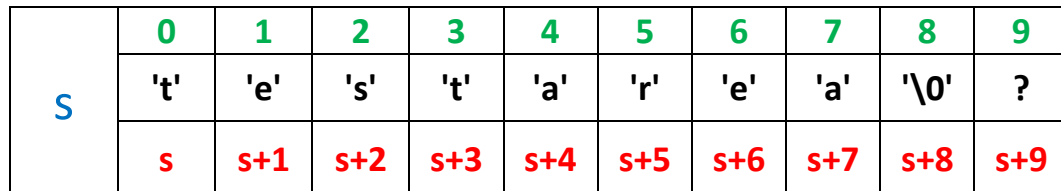
- `char* strcat(char *destinatie, char *sursa)`
- `char* strncat(char *destinatie, char *sursa, int n)`

Funcții pentru căutare:

- `char* strchr(char *sir, char caracter)` – returnează adresa primei apariții, de la stânga spre dreapta, a caracterului indicat sau pointerul NULL dacă respectivul caracter nu apare în șir.
- `char* strrchr(char *sir, char caracter)` – returnează adresa ultimei apariții, de la stânga spre dreapta, a caracterului indicat sau pointerul NULL dacă respectivul caracter nu apare în șir.

Exemplu:

```
char s[10] = "testarea";  
char *p = strchr(s, 'e'); //Poziția: p-s = (s+1)-s = 1
```



	0	1	2	3	4	5	6	7	8	9
s	't'	'e'	's'	't'	'a'	'r'	'e'	'a'	'\0'	'?'
	s	s+1	s+2	s+3	s+4	s+5	s+6	s+7	s+8	s+9

```
char *q = strrchr(s, 'e'); //Poziția: q-s = (s+6)-s = 6
```

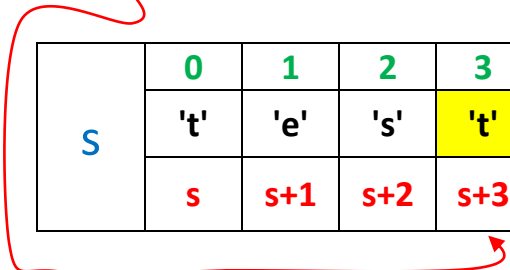
```
#include <stdio.h>  
#include <string.h>
```

```
int main()  
{  
    char s[51];  
    char c;  
  
    printf("Sirul: ");  
    fgets(s, 51, stdin);  
  
    if(s[strlen(s)-1] == '\n')  
        s[strlen(s)-1] = '\0';  
  
    printf("Caracterul cautat: ");  
    scanf("%c", &c);  
  
    char *p = strchr(s, c);  
  
    if(p == NULL)  
    {  
        printf("\nCaracterul '%c' nu apare in sirul \"%s\"!\n", c, s);  
        return 0;  
    }  
  
    char *q = strrchr(s, c);  
  
    printf("\n Prima pozitie pe care apare caracterul '%c'"  
          " in sirul \"%s\": %d\n", c, s, p-s);  
    printf("Ultima pozitie pe care apare caracterul '%c'"  
          " in sirul \"%s\": %d\n", c, s, q-s);  
  
    return 0;  
}
```

- **char* strstr(char *sir_1, char *sir_2)** – returnează adresa primei apariții în șirul sir_1, de la stânga spre dreapta, a șirului sir_2 ca subșir sau pointerul NULL dacă sir_2 nu este subșir în sir_1.

Exemplu:

```
char s[10] = "testarea", t[21] = "tare";
char *p = strstr(s, t); //Poziția: p-s = (s+3)-s = 3
```



	0	1	2	3	4	5	6	7	8	9
s	't'	'e'	's'	't'	'a'	'r'	'e'	'a'	'\0'	'?'
	s	s+1	s+2	s+3	s+4	s+5	s+6	s+7	s+8	s+9

Conversia unui caracter într-un șir formă dintr-un singur caracter:

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

Exemplu:

Să se afișeze toate pozițiile pe care apare un șir s într-un șir t sau un mesaj corespunzător dacă șirul s nu apare în șirul t. **Atenție la variantele precizate în comentarii!!!**

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

int main()
{
    char s[21], t[21];
    char *p;

    printf("Sirul s: ");
    fgets(s, 21, stdin);

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

    printf("Sirul t: ");
    fgets(t, 21, stdin);
```

```

if(t[strlen(t)-1] == '\n')
    t[strlen(t)-1] = '\0';

p = strstr(s, t);

if(p == NULL)
    printf("\nSirul \"%s\" nu apare in sirul \"%s\"!\n", t, s);
else
{
    //Problema are doua variante:
    //a. subsiruri disjuncte (non-overlapping)
    //b. subsiruri suprapuse (overlapping)

    //Exemplu: s = "aaaaa", t = "aa"
    //Varianta a: pozitiile 0, 2
    //Varianta b: pozitiile 0, 1, 2, 3

    printf("\nSirul \"%s\" apare in sirul \"%s\"
                                                    pe pozitiile:\n", t, s);
    while(p != NULL)
    {
        printf("%d ", p-s);
        //cautam, din nou, sirul t incepand cu
        //urmatoarea adresa fata de cea la care l-am gasit
        //ultima oara, adica p+strlen(t)

        //Varianta a:
        p = strstr(p+strlen(t), t);
        //Varianta b:
        //p = strstr(p+1, t);
    }
}

printf("\n");

return 0;
}

```

- **int strspn(char *sir_1, char *sir_2)** – returnează lungimea maximă a prefixului şirului sir_1 format doar din caractere din sir_2.

Exemple:

- strspn("bacalaureat", "carbon") = 4
- strspn("carbon", "raspuns") = 0
- strspn("casa", "mocasin") = 4

Observație:

Prin instrucțiunea `if(strspn(sir_1, sir_2) == strlen(sir_1))...` se poate verifica dacă toate caracterele din `sir_1` au o anumită proprietate!

Exemple:

- `if(strspn(cuvant, "aeiouAEIOU") == strlen(cuvant))...` se poate utiliza pentru a verifica dacă respectivul cuvânt este format doar din vocale
- `if(strspn(cuvant, "ABCDEFGHIJKLMNOPQRSTUVWXYZ") == strlen(cuvant))...` se poate utiliza pentru a verifica dacă respectivul cuvânt este format doar din litere mari
- `if(strspn(cuvant, "0123456789") == strlen(cuvant))...` se poate utiliza pentru a verifica dacă respectivul cuvânt este format doar din cifre
- funcție care verifică dacă un șir de caractere conține un număr întreg:

```
int testareNrIntreg(char *sir)
{
    char *p = sir;

    //sirul trebuie sa aiba un prefix format
    //din cel mult un caracter '+' sau '-'
    if(sir[0] == '-' || sir[0] == '+')
        p = sir+1;

    //restul caracterelor trebuie sa fie cifre
    if(strspn(p, "0123456789") == strlen(p))
        return 1;

    return 0;
}
```

- funcție care verifică dacă un șir de caractere conține un număr "real" (cu zecimale):

```
int testareNrReal(char *sir)
{
    //sirul "aux" va contine sirul initial fara eventualul semn
    char *aux = sir;
    char *pozp;

    //sirul trebuie sa aiba un prefix format
    //din cel mult un caracter '+' sau '-'
    if(sir[0] == '-' || sir[0] == '+')
        aux = sir+1;
```

```

//cautam caracterul '.' in sirul aux
pozp = strchr(aux, '.');

//daca in sirul "aux" nu apare niciun '.',
//atunci tot sirul "aux" trebuie sa fie format doar din cifre
if(pozp == NULL)
    if(strspn(aux, "0123456789") == strlen(aux))
        return 1;
    else
        return 0;
//daca in sirul "aux" apare un '.' la adresa "pozp",
//atunci subsirul pana in '.' si subsirul dupa '.'
//trebuie sa fie formate doar din cifre
else
{
    //verific subsirul aflat dupa caracterul '.'
    if(strspn(pozp+1, "0123456789") != strlen(pozp+1))
        return 0;
    //verific subsirul aflat pana in caracterul '.'
    *pozp = '\0'; //aux[pozp-aux] = '\0';
    if(strspn(aux, "0123456789") != strlen(aux))
    {
        *pozp = '.';
        return 0;
    }
    *pozp = '.';
    return 1;
}
}

```

- **if(strspn(sir_1, sir_2) == strlen(sir_1))... NU se poate utiliza pentru a verifica dacă şirurile respective sunt anagrame!!!**

```

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

int main()
{
    char sir_1[] = "aaab", sir_2[] = "abbb";

    if(strlen(sir_1) == strlen(sir_2) && strspn(sir_1, sir_2) == strlen(sir_1) &&
        strspn(sir_2, sir_1) == strlen(sir_2))
        printf("DA");
    else
        printf("NU");

    return 0;
}

```

C:\Users\BOurs\Desktop\Test_C\bin\Debug\Test_C.exe

DA
Process returned 0 (0x0) execution time : 0.043 s
Press any key to continue.

- `int strcspn(char sir_1[], char sir_2[])` – returnează lungimea maximă a prefixului șirului `sir_1` format doar din caractere care **NU** apar în șirul `sir_2`.

Caracterul 'c' nu apare în șirul "raspuns",
dar caracterul 'a' apare!

Exemple:

- `strcspn("carbon", "raspuns") = 1`
- `strcspn("bacalaureat", "carbon") = 0`
- `strcspn("casa", "brut") = 4`