

CURS 08 – PP

ȘIRURI DE CARACTERE

Biblioteca string.h

Funcții pentru căutare:

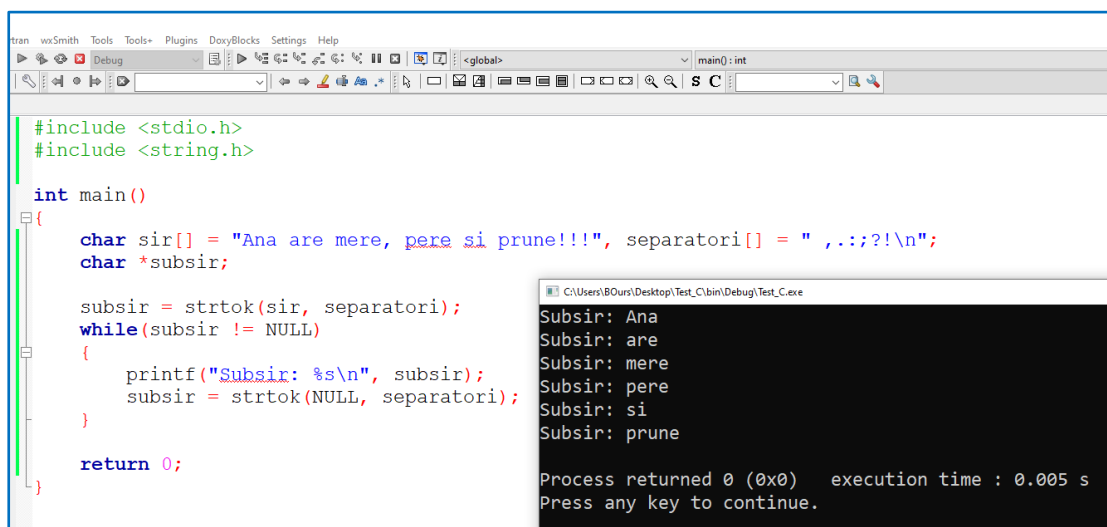
- **char* strtok(char *sir, char *separatori)** – returnează, pe rând, subșirurile șirului dat care sunt delimitate de separatorii indicați.

Modalitatea de utilizare:

```
char sir[1001], separatori[] = " ,.:;?!";
char *subsir;

//inițializare șir (de exemplu, prin citire de tastatură)
subsir = strtok(sir, separatori);
while(subsir != NULL)
{
    //prelucrarea subșirului curent
    subsir = strtok(NULL, separatori);
}
```

Exemple:



The screenshot shows a C program in a code editor and its execution output in a console window. The program uses `strtok` to split the string "Ana are mere, pere si prune!!!" into tokens separated by commas. The output shows each token on a new line, labeled "Subsir:". The console window title is "C:\Users\BOurs\Desktop\Test_C\bin\Debug\Test_C.exe".

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

int main()
{
    char sir[] = "Ana are mere, pere si prune!!!", separatori[] = " ,.:;?!\\n";
    char *subsir;

    subsir = strtok(sir, separatori);
    while(subsir != NULL)
    {
        printf("Subsir: %s\\n", subsir);
        subsir = strtok(NULL, separatori);
    }

    return 0;
}
```

Subsir: Ana
Subsir: are
Subsir: mere
Subsir: pere
Subsir: si
Subsir: prune

Process returned 0 (0x0) execution time : 0.005 s
Press any key to continue.

```

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

int main()
{
    char sir[] = "Ana are mere, pere si prune!!!", separatori[] = "rea";
    char *subsir;

    subsir = strtok(sir, separatori);
    while(subsir != NULL)
    {
        printf("Subsir: %s\n", subsir);
        subsir = strtok(NULL, separatori);
    }

    return 0;
}

```

Subsir: An
Subsir: m
Subsir: , p
Subsir: si p
Subsir: un
Subsir: !!!
Process returned 0 (0x0) execution time : 0.021 s
Press any key to continue.

Modalitatea de funcționare:

```

char sir[] = "...Ana are mere, pere si prune!!!",
    separatori[] = " ,.:;?!";
char *subsir;

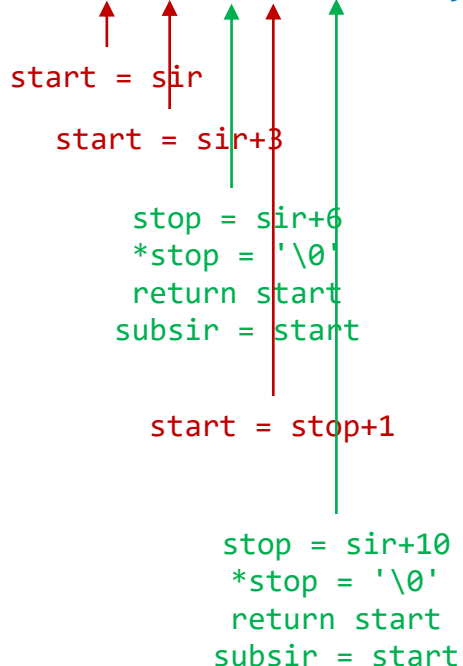
```

```

subsir = strtok(sir, separatori);
while(subsir != NULL)
{
    printf("Subsir: %s\n", subsir);
    subsir = strtok(NULL, separatori);
}

```

sir = "...Ana\0are\0mere, pere si prune!!!\0"



```

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

int main()
{
    char sir[] = "...Ana are mere, pere si prune!!!",
    separatori[] = " ,.;;?!";
    char *subsir;
    int i, lsir;

    lsir = strlen(sir);

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

    subsir = strtok(sir, separatori);
    while(subsir != NULL)
    {
        printf("Subsir: %s\n", subsir);
        subsir = strtok(NULL, separatori);
    }

    //se va afisa sirul "...Ana"
    printf("\nSirul dupa prelucrare: %s\n", sir);

    for(i = 0; i < lsir; i++)
        if(sir[i] == '\0')
            sir[i] = '?';

    //se va afisa sirul "...Ana?are?mere? pere?si?prune?!!"
    printf("\nSirul reconstituit: %s\n", sir);

    return 0;
}

```

Observații:

- Funcția strtok modifică șirul inițial, respectiv, după utilizarea funcției, șirul inițial va fi restrâns la primul subșir găsit și eventualii separatori dinaintea sa.
- NU se pot utiliza apeluri imbricate ale funcției strtok, deoarece variabilele start și stop folosite în implementarea sa sunt statice! Astfel, de obicei, funcția va prelucra doar primul subșir găsit (vezi exemplul de mai jos).

```

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

int main()
{
    char sir[] = "Popa Ion,101,9.50;
                Ionescu Anca Elena,102,9.80;Mihai Radu,101,8.95";
    char separatori_studenti[] = ";";
    char separatori_informatii[] = ",";
    char *student, *informatii;

    student = strtok(sir, separatori_studenti);
    while(student != NULL)
    {
        printf("Student: %s\n", student);
        informatii = strtok(student, separatori_informatii);
        while(informatii != NULL)
        {
            printf("Informatie: %s\n", informatii);
            informatii = strtok(NULL, separatori_informatii);
        }
        printf("\n");

        student = strtok(NULL, separatori_studenti);
    }

    return 0;
}

```

```

C:\Users\BOurs\Desktop\Test_C\bin\Debug\Test_C.exe
Student: Popa Ion,101,9.50
Informatie: Popa Ion
Informatie: 101
Informatie: 9.50

Process returned 0 (0x0)   execution time : 0.008 s
Press any key to continue.

```

Funcții pentru clasificarea caracterelor

Funcțiile pentru clasificarea caracterelor se găsesc în biblioteca `ctype.h`.
Toate funcțiile pentru clasificarea caracterelor au anteturi de forma următoare:

`int` `istip(char c)`

Toate funcțiile furnizează o valoare nenulă în cazul în care caracterul `c` este de tipul indicat sau 0 în caz contrar.

Cele mai utilizate funcții pentru clasificarea caracterelor sunt următoarele:

- **isalpha** – caracterul este literă mică sau mare;
- **islower** – caracterul este literă mică;
- **isupper** – caracterul este literă mare;
- **isblank** – caracterul este spațiu sau tabulator;
- **isdigit** – caracterul este o cifră în bază 10;
- **ispunct** – caracterul este semn de punctuație (în afara semnelor de punctuație uzuale, mai sunt luate în considerare și parantezele, acoladele, precum și alte simboluri grafice speciale).

În afara funcțiilor de clasificare, biblioteca `ctype.h` mai conține și două funcții de conversie pentru caractere:

- **char tolower(char c)** – transformă o literă mare în litera mică echivalentă, orice alt caracter rămânând neschimbat;
- **char toupper(char c)** – transformă o literă mică în litera mare echivalentă, orice alt caracter rămânând neschimbat.

Atenție, toate funcțiile din biblioteca `ctype.h` pot fi utilizate doar pentru caractere, ci nu pentru șiruri de caractere! De exemplu, dacă dorim să verificăm dacă un șir de caractere este format doar din litere mici, atunci trebuie să verificăm, pe rând, fiecare caracter:

```
int isLowerString(char sir[])
{
    int i;

    for(i = 0; i < strlen(sir); i++)
        if(islower(sir[i]) == 0)
            return 0;
    return 1;
}
```

Într-un mod asemănător trebuie să procedăm pentru a transforma toate literele dintr-un șir în litere mari:

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

```

int main()
{
    char sir[] = "Ana are mere si pere!";
    int i;
    printf("\nSirul initial: %s\n", sir);

    for(i = 0; i < strlen(sir); i++)
        sir[i] = toupper(sir[i]);

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

    return 0;
}

```

Programul de mai sus rescris fără funcția toupper:

```

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

int main()
{
    char sir[] = "Ana are mere si pere!";
    int i;

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

    for(i = 0; i < strlen(sir); i++)
        if(sir[i] >= 'a' && sir[i] <= 'z')
            sir[i] = sir[i] - ('a' - 'A');

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

    return 0;
}

```

Funcții pentru conversii

Funcțiile pentru conversii sunt utilizate pentru a transforma un șir de caractere într-o valoare numerică și sunt definite în biblioteca `stdlib.h` (<http://www.cplusplus.com/reference/cstdlib/>). Dezavantajul acestor funcții de conversie constă în faptul că fiecare este specifică unui anumit tip de date. De exemplu, funcția `strtol` trebuie folosită pentru a realiza conversia unui șir într-un număr de tip `long int`, funcția `strtoll` trebuie folosită pentru conversia într-un număr de tip `unsigned long int` etc. Mai mult, nu există definite funcții de conversie dintr-o valoare numerică într-un șir de caractere.

O soluție mai simplă pentru realizarea conversiilor dintre șiruri de caractere și valori numerice o reprezintă utilizarea funcțiilor `sscanf` și `sprintf` din biblioteca `stdio.h`. Cele două funcții sunt foarte asemănătoare cu funcțiile `scanf` și `printf`, singura diferență constând în faptul că ele utilizează un șir de caractere pentru citire/scriere în locul tastaturii/monitorului (șirul respectiv este primul parametru al ambelor funcții). De exemplu, după rularea secvenței de cod de mai jos, se vor afișa pe ecran valorile 1234 78.500000 -900:

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

int main()
{
    char sir[] = "1234 78.5 -900";
    int x, y;
    float f;

    //char *subsir;
    //
    //subsir = strtok(sir, " ");
    //x = atoi(subsir);
    //
    //subsir = strtok(NULL, " ");
    //f = atof(subsir);
    //
    //subsir = strtok(NULL, " ");
    //y = atoi(subsir);
    //
    //printf("x = %d\nf = %f\nny = %d\n", x, f, y);

    sscanf(sir, "%d %f %d", &x, &f, &y);
    printf("x = %d\nf = %f\nny = %d\n", x, f, y);

    return 0;
}
```

În mod asemănător se poate utiliza funcția `sprintf` pentru a realiza conversia unor valori numerice într-un șir de caractere:

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

int main()
{
    char sir[] = "1234 78.5 -900";
    int x, y;
    float f;
```

```

    sscanf(sir, "%d %f %d", &x, &f, &y);
    printf("x = %d\nf = %f\ny = %d\n", x, f, y);

    sprintf(sir, "%.2f", x + y + f);
    printf("\nSuma = %s\n", sir);

    return 0;
}

```

TABLOURI DE ȘIRURI DE CARACTERE

În multe probleme, avem nevoie să memorăm mai multe șiruri de caractere, de obicei, într-un tablou unidimensional. De exemplu, într-un program trebuie să determinăm cuvintele distincte dintr-un text sau să sortăm niște cuvinte după un anumit criteriu.

Deoarece în limbajul C o linie a unei matrice este un tablou unidimensional, rezultă că o linie a unei matrice având elemente de tip caracter (`char`) este chiar un șir de caractere. De exemplu, matricea `șiruri` declarată prin `char siruri[50][21]` poate fi utilizată pentru a memora cel mult 50 de șiruri de caractere, fiecare șir fiind format din maxim 20 de caractere, iar prin `siruri[i]` vom putea accesa/prelucra individual fiecare șir.

De exemplu, prin secvența de cod de mai jos, fiecare linie a matricei `cuvinte` va conține câte un cuvânt din șirul `text`:

```

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

int main()
{
    char text[] = "Ana are mere, pere si prune!!!";
    char cuvinte[50][21], *p;
    int nrc, i;

    //nrc = numarul curent de cuvinte
    nrc = 0;

    //p = cuvantul curent
    p = strtok(text, " ,.:;!?\n");
    while(p != NULL)

```



```

{
    //copiem cuvantul curent pe prima linie disponibila
    //din matricea siruri, iar apoi crestem numarul de cuvinte
    strcpy(cuvinte[nrc++], p);
    p = strtok(NULL, " ,.?:!?\n");
}

printf("Cuvintele din propozitia data:\n");
for(i = 0; i < nrc; i++)
    printf("%s\n", cuvinte[i]);

return 0;
}

```

O altă variantă pentru memorarea mai multor şiruri de caractere o constituie utilizarea unui tablou de pointeri la şiruri, astfel încât fiecare şir să poată fi alocat dinamic:

```

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

int main()
{
    char text[] = "Ana are mere, pere si prune!!!";
    char **cuvinte = NULL, *p;
    int nrc, i;

    //nrc = numarul curent de cuvinte
    nrc = 0;

    //p = cuvantul curent
    p = strtok(text, " ,.?:!?\n");
    while(p != NULL)
    {
        //aloc o linie noua in "tabloul bidimensional" cuvinte,
        //utilizand functia realloc
        cuvinte = (char**)realloc(cuvinte, (nrc+1) * sizeof(char*));

        //aloc memorie pentru cuvantul curent pe linia curenta
        cuvinte[nrc] = (char*)malloc(strlen(p) + 1);

        //copiem cuvantul curent pe linia curenta
        strcpy(cuvinte[nrc], p);

        //cresc numarul curent de cuvinte
        nrc++;

        p = strtok(NULL, " ,.?:!?\n");
    }
}

```

```

}

printf("Cuvintele din propozitia data:\n");
for(i = 0; i < nrc; i++)
    printf("%s\n", cuvinte[i]);

//eliberez zonele de memorie alocate cuvintelor
for(i = 0; i < nrc; i++)
    free(cuvinte[i]);

//eliberez zonele de memorie alocata tabloului bidimensional
free(cuvinte);

return 0;
}

```

Împărțirea unui șir în subșiruri folosind mai multe criterii:

```

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

int main()
{
    char sir[] = "Popa Ion,101,9.50;
                Ionescu Anca Elena,102,9.80;Mihai Radu,101,8.95";
    char matrice_studenti[100][51];
    int nrs, i;

    char separatori_studenti[] = ";";
    char separatori_informatii[] = ",";
    char *student, *informatii;

    nrs = 0;
    student = strtok(sir, separatori_studenti);
    while(student != NULL)
    {
        strcpy(matrice_studenti[nrs++], student);
        student = strtok(NULL, separatori_studenti);
    }

    printf("Studentii:\n");
    for(i = 0; i < nrs; i++)
        printf("%s\n", matrice_studenti[i]);
}

```

```

printf("\nInformatiile despre fiecare student:\n");
for(i = 0; i < nrs; i++)
{
    printf("\tStudent %d: ", i+1);
    informatii = strtok(matrice_studenti[i],
                        separatori_informatii);

    while(informatii != NULL)
    {
        printf("%s / ", informatii);
        informatii = strtok(NULL, separatori_informatii);
    }
    printf("\b\b \n");
}

return 0;
}

```