



ȘIRURI DE CARACTERE

O variabilă de tip șir de caractere se declară menționând tipul `char`, numele tabloului și dimensiunea maximă (adică numărul de caractere care se pot memora). Se poate efectua și o inițializare cu un conținut oarecare delimitat la început și sfârșit de caracterul `”`. Fiecare componentă a șirului (începând cu cea de indice 0) reține codul ASCII al caracterului pe care îl memorează. Convenția este ca ultimul octet să rețină 0 (codul caracterului NULL). Caracterul NULL este memorat automat. Trebuie rezervate `lungimea_sirului+1` caractere (+1 pentru caracterul NULL).

Exemplu :

```
char vect[11]="calculator";
char vect[]="calculator"; (compilatorul face calculul numarului de octeti necesari și adună 1 pentru
NULL)
char vect[100]="calculator"; (s-au rezervat mai multi octeti decat era necesar)
```

Termenii șir de caractere, respectiv tablou de caractere, sunt identici.

Exemplu:

```
char tc[5] = {'a', 'b', 'c', 'd', 'e'}; // tablou de caractere
char sc[5] = {'a', 'b', 'c', 'd', '\0'}; // sir de caractere cu elementele abcd
Ultima initializare este echivalenta cu:
char sc[5] = "abcd"; //sau char sc[] = "abcd";
char sc1[5] = "abcd";
char s[10];
cout<<sc<<endl; //afiseaza abcd
cout<<tc<<endl; //efect nedeterminat: tabloul de caractere nu contine terminatorul NULL
cout<<s<<endl; // efect nedeterminat: tablou neinitializat
cout<<sc1[0]; // afiseaza primul caracter din sirul sc1
cout<<sc1[2]; // afiseaza al treilea element din sirul sc1
sc1[1]='K'; // elementului din sir de indice 1 i se atribuie valoarea 'K';
```

CITIREA / AFIȘAREA ȘIRURILOR DE CARACTERE

Șirurile de caractere pot fi inițializate încă de la declarare sau citite pe parcursul programului.

- citirea unui șir de caractere se poate face ca citirea oricărui tablou, într-o structură repetitivă, caracter cu caracter (deși nu este recomandată). În acest caz, terminatorul de șir nu este memorat automat, el trebuie pus explicit după ultimul caracter din șir.

Exemplu:

```
char c[20];
for(int i=0;i<=5;i++)
    cin>>c[i];
cout<<c<<endl; //se va afisa sirul format din cele 6 caractere, urmat de caractere „reziduale”,
initializate implicit la compilare, din cauza ca n-a fost pus terminatorul de sir
c[6]=0;
cout<<c<<endl; //a fost pus terminatorul de sir, deci sirul va fi afisat corect
```

- Această formă este recomandată atunci când se dorește citirea unui singur cuvânt. Se poate face menționând doar numele tabloului, folosind sintaxa `cin>>numeTablou;` Caracterul NULL este adăugat automat. Citirea șirului se termină la întâlnirea primului caracter “alb”: spațiu, tab sau Enter (de exemplu, dacă se citește “ora de informatica”, variabila `c` va reține numai “ora”).

Exemplu

```
char c[30];
cin>>c;
```

```
cout<<c;
```

- c. Această formă este recomandată atunci când se dorește citirea unei fraze (mai multe cuvinte separate prin spații). Se pot folosi funcțiile `get()` sau `getline()`. Ambele primesc numele tabloului, numărul de caractere care se doresc a fi citite și opțional un caracter până la întâlnirea căruia se va realiza citirea (implicit este marcatorul de sfârșit de linie).

Exemplu

```
char a[30],x;int nr;  
cin.get(a,nr,x);
```

Funcțiile citesc un șir de caractere până cand au fost citite `nr-1` caractere sau până se întâlnește caracterul din variabila `x`. Al treilea parametru poate lipsi, caz în care el este implicit caracterul `'\n'` (new line). Sunt citite și caracterele albe, caracterul `NULL` este inserat automat iar caracterul transmis ca ultim parametru nu este inserat în șir.

Exemplu

```
char a[30];  
cin.get(a,5,'s'); //daca se citeste sirul "maimuta, variabila a va retine "maim"  
cin.get(a,15,'s'); //daca se citeste sirul "maimuta, variabila a va retine "maimuta"  
cin.get(a,15,'t'); //daca se citeste sirul "maimuta, variabila a va retine "maimu"  
cin.get(a,4,'t'); //daca se citeste sirul "maimuta, variabila a va retine "mai"  
cin.get(a,10); //daca se citeste sirul "maimuta, variabila a va retine "maimuta"
```

Funcția **`cin.get()`** fara parametri are rolul de a citi un caracter (alb sau nu).

Funcția **`cin.get(char c)`** are rolul de a citi un caracter (alb sau nu) pe care îl încarcă în variabila `c`.

Observatie: În cazul utilizării repetate a funcției `cin.get(a,nr,x)`, după fiecare folosire trebuie citit caracterul de la sfârșitul fiecărui șir, adică `'\n'` (în caz contrar, acest caracter va fi încărcat la începutul următorului șir, deci citirea celui de-al doilea șir se termină înainte de a începe, adică va fi șirul vid). Această citire a caracterului `'\n'` se realizează folosind `cin.get()` fără parametri. Funcția `getline()` nu are acest dezavantaj, deci este recomandată pentru citiri repetate de fraze.

Exemplu

```
char a[30],b[30];  
cin.get(a,15);  
cin.get(b,10);
```

Daca se incearca citirea sirurilor „sarbatoare” si „vacanta”, se observa ca `a="sarbatoare"`, `b=""` (nici nu apucam sa citim sirul `b`). Varianta corecta este:

```
cin.get(a,15);  
cin.get();  
cin.get(b,10);
```

sau:

```
cin.getline(a,15);  
cin.getline(b,10);
```

Afisarea unui sir de caractere se face folosind `cout`, cu condiția să existe terminatorul `NULL`. Dacă acesta nu există, efectul este nedeterminat.

```
cout<<a;
```

Se poate afișa și caracter cu caracter, ca în cazul tablourilor, dar această variantă nu este recomandată.

FUNCȚII PENTRU OPERAȚII CU ȘIRURI DE CARACTERE

Funcțiile pentru operații cu șiruri de caractere se găsesc în header-ul `<cstring>`.

➤ **Functia strlen**

int **strlen**(nume_sir); – returnează *lungimea efectivă* a unui șir (fără a număra terminatorul de șir).

Exemplu:

```
char a[50]="ora de informatica"; // strlen(a) = 18
```

➤ **Functia strcpy**

strcpy(sir_destinatie, sir_sursa); – copiaza sirul sir_sursa in sir_destinatie (se simuleaza atribuirea a=b).

ATENȚIE!! Nu este permisă atribuirea între două șiruri de caractere folosind operatorul =. Atribuirea se face folosind functia strcpy.

Exemplu:

```
char a[50]="primul sir",b[40]="al doilea sir";  
a=b; //eroare  
strcpy(a,b); // a = "al doilea sir"; b="al doilea sir";
```

➤ **Functia strcat**

strcat(dest, sursa); – adauga sirului dest sirul sursa. Sirul sursa ramane nemodificat. Operatia se numeste *concatenare* si nu este comutativa.

Exemplu:

```
char a[]="vine ",b[]="vacanta?";  
strcat(a,b); // a = "vine vacanta?";
```

➤ **Functia strncat**

strncat(dest, sursa, nr); – adauga la dest primele nr caractere din sirul sursa. Sirul sursa ramane nemodificat.

Exemplu:

```
char a[]="vine ",b[]="vacanta?";  
strncat(a,b,6); // a = "vine vacant";
```

➤ **Functia strchr**

strchr(sir, c); – are rolul de a cauta caracterul c în șirul sir. Căutarea se face de la stânga la dreapta, iar funcția întoarce adresa subșirului care începe cu prima apariție a caracterului c. Dacă nu este găsit caracterul, funcția returnează 0. Diferența dintre adresa șirului inițial și cea a subșirului returnat reprezintă chiar poziția caracterului căutat în șirul dat.

Exemplu:

```
char a[]="acesta este un sir",b='t',c='x',d;  
cout<<strchr(a,b); // se tipareste "ta este un sir";  
cout<<strchr(a,c); // nu se tipareste nimic (se tipareste 0 daca se face o conversie la int a  
lui strchr(a,c) ;  
d=strchr(a,b);  
cout<<"Caracterul apare prima data la pozitia "<<d-a;
```

➤ **Functia strrchr**

strrchr(sir, c); – are același rol cu strchr, cu deosebirea ca returnează adresa ultimei apariții a caracterului (cautarea se face de la dreapta spre stanga; r = right)

➤ **Functia strcmp**

int strcmp(sir1, sir2); – are rolul de a compara lexicografic două șiruri de caractere. Valoarea returnată este -1 (daca sir1<sir2), 0 (daca sir1==sir2) sau 1 (daca sir1>sir2). Functia strcmp face distincție între literele mari și cele mici ale alfabetului.

Exemplu:

```
cout<<strcmp("abcde", "abcwww"); // se afisează -1
```

➤ Funcția **stricmp**

`int stricmp(sir1, sir2);` – are același rol cu `strcmp`, cu deosebirea ca nu face distincție între literele mari și cele mici ale alfabetului (i = ignore).

➤ Funcția **strstr**

`strstr(sir, subSir);` – are rolul de a identifica dacă șirul `subSir` este subșir al șirului `sir`. Dacă este, funcția returnează adresa de început a subșirului în cadrul șirului altfel returnează NULL. În cazul în care subșirul apare de mai multe ori în șir se returnează adresa de început a primei apariții. Căutarea se face de la stânga la dreapta.

➤ Funcția **strtok**

`strtok(sir, sep);` – are rolul de a separa șirul `sir` în mai multe entități (de regulă cuvinte) separate între ele prin unul sau mai multe caractere cu rol de separator. Șirul `sep` este alcătuit din acești separatori.

Funcția **strtok** acționează în felul următor:

- Primul apel trebuie să fie de forma `strtok(sir, sep);` Funcția întoarce adresa primului caracter al primei entități. După prima entitate, separatorul este înlocuit automat prin caracterul NULL.
- Următoarele apeluri sunt de forma `strtok(NULL, sep);` De fiecare dată, funcția întoarce adresa de început a următoarei entități, adăugând automat după ea caracterul NULL.
- Când șirul nu mai conține entități, funcția returnează adresa nulă.

Exemplu:

```
//Sa se separe cuvintele dintr-un text.
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char text[100], cuv[10][10], *p, sep[]=" . !?";
    int nr=0, i;
    cout<<"Dati fraza: "; cin.getline(text, 100);
    p=strtok(text, sep);
    while(p!=NULL)
    {
        strcpy(cuv[++nr], p);
        p=strtok(NULL, sep);
    }
    cout<<"Sunt "<<nr<<" cuvinte:"<<endl;
    for (i=1; i<=nr; i++)
        cout<<"Cuvantul "<<i<<": "<<cuv[i]<<endl;
}
```

➤ Funcția **strlwr** cu forma generală

`strlwr(sir);` – are rolul de a converti toate literele mari din `sir` în litere mici. Restul caracterelor rămân neschimbate.

➤ Funcția **strupr** cu forma generală

`strupr(sir);` – are rolul de a converti toate literele mici din `sir` în litere mari. Restul caracterelor rămân neschimbate

➤ Funcția **atof** cu forma generală

`double atof(sir);` – convertește un șir care conține cifre către tipul `double`. Dacă aceasta conversie eșuează (se întâlnește un caracter nenumeric), valoarea întoarsă este 0.

- Funcția **atoi** cu forma generală
`int atoi(sir);` – convertește un șir care conține cifre către tipul `int`. Dacă aceasta converție eșuează (se întâlnește un caracter nenumeric), valoarea întoarsă este 0.
- Funcția **itoa** cu forma generală
`itoa(valoare, sir, baza);` – convertește o valoare de tip `int` în șir, care este memorat în variabila `sir`. Parametrul `baza` reprezintă baza de numeratie în care se va realiza conversia. În cazul bazei 10, șirul reține și eventualul semn -.
- Funcția **isalpha(c)** returnează o valoare nenulă dacă parametrul `c` reține o literă și 0 în caz contrar.
- Funcția **isupper(c)** returnează o valoare nenulă dacă parametrul `c` reține o majusculă și 0 în caz contrar.
- Funcția **islower(c)** returnează o valoare nenulă dacă parametrul `c` reține o minusculă și 0 în caz contrar.
- Funcția **isdigit(c)** returnează o valoare nenulă dacă parametrul `c` reține o cifră și 0 în caz contrar.
- Funcția **toupper(c)** returnează litera din parametrul `c` transformată în majusculă, sau același parametru dacă acesta nu memorează o literă.
- Funcția **tolower(c)** returnează litera din parametrul `c` transformată în minusculă, sau același parametru dacă acesta nu memorează o literă.