*SIRURI DE CARACTERE*

O constanta de tip sir de caractere de declara intre doua caractere “. In memoria interna, o constanta de acest tip este retinuta sub forma unui vector de caractere. Fiecare componenta a sirului (incepand cu cea de indice 0) retine codul ASCII al caracterului pe care il memoreaza. Conventia este ca ultimul octet sa retina 0 (codul caracterului nul). Caracterul nul este memorat automat. Trebuie rezervate *lungimea\_sirului+*1 caractere char (+1 pentru caracterul nul). Poziționarea caracterului null la finalul unui șir construit de utilizator trebuie să se facă manual print-o operație de atribuire .... = ‘\0’

Limbajul C/C++ permite initializarea unui tablou de caractere printr-o constanta sir, care include automat caracterul null.

***Exemplu :***

char vect[11]=”calculator”;

char vect[]=”calculator”; (compilatorul face calculul numarului de octeti necesari)

char vect[100]=”calculator”; (s-au rezervat mai multi octeti decat era necesar)

 Sirurile de caractere sunt de fapt *tablouri de caractere*, care au ca ultim element un terminator de sir, caracterul null.

***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; //afiseaza *abcd*

cout<<tc; //eroare: tabloul de caractere nu contine terminatorul de sir, deci nu poate fi afisat ca sir

cout<<s; // eroare: 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 / AFISAREA SIRURILOR DE CARACTERE**

Sirurile de caractere pot fi initializate inca de la declarare sau citite pe parcursul programului.

a. Citirea unui sir de caractere se poate face ca citirea oricarui tablou, intr-o instructiune for, caracter cu caracter (desi nu este recomandata). In acest caz, terminatorul de sir nu este memorat automat, el trebuie pus explicit dupa ultimul caracter din sir.

 ***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

 b. Se poate face pur si simplu, folosind ***cin>>***. Caracterul nul este adaugat automat. Dezavantajul este ca in acest fel nu se pot citi siruri care contin mai multe cuvinte separate prin spatii. Citirea sirului se sfarseste la intalnirea primului caracter blank (de ex, daca se citeste “ora de informatica”, variabila c va retine numai “ora”).

***Exemplu***

char c[30];

cin>>c;

cout<<c;

* 1. Se poate folosi o functie speciala pentru citirea sirurilor de caractere, inclusa in biblioteca string.h (varianta recomandata).

***Exemplu***

char a[30],x;int nr;

cin.get(a,nr,x);

Functia ***cin.get*** citeste un sir de caractere sau pana cand au fost citite nr-1 caractere, sau daca s-a intalnit caracterul x. Al treilea parametru poate lipsi, caz in care el este implicit caracterul ’\n’ (new line). Sunt citite si caracterele albe, caracterul nul este inserat automat iar caracterul transmis ca ultim parametru nu este inserat in sir.

***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”

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

Functia ***cin.get(char c)*** are rolul de a citi un caracter (alb sau nu) pe care il incarca in variabila c.

*Observatie*: In cazul utilizarii repetate a functiei cin.get(a,nr,x), dupa fiecare folosire trebuie citit caracterul de la sfarsitul fiecarui sir , adica ’\n’ (in caz contrar, acest caracter va fi incarcat la inceputul urmatorului sir, a carui citire se termina la caracterul Enter, deci citirea celui de-al doilea sir se termina inainte de a incepe, iar al doilea sir va fi sirul vid). Aceasta citire a caracterului ’\n’ se realizeaza folosind cin.get() fara parametri.

***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);

Afisarea unui sir de caractere se face folosind ***cout***.

cout<<a;

Se poate afisa si caracter cu caracter, ca in cazul tablourilor, dar aceasta varianta nu este recomandata.

**FUNCTII PENTRU OPERATII CU SIRURI DE CARACTERE**

 Functiile pentru operatii cu siruri se gasesc in header-ul **<string.h>**.

       Functia **strlen**
int **strlen**(nume\_sir); – returneaza lungimea efectiva a unui sir (fara a numara terminatorul de sir).
*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).

**ATENTIE**!! **Nu** este permisa atribuirea intre doua siruri 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 dest primele nr caractere din sirul sursa. Sirul sursa ramane nemodificat.

***Exemplu:***
char \*a=”vine ”,\*b=”vacanta?”;
strncat(a,b,4); 🡪 a = ”vine vaca”;

        Functia **strchr**
strchr(sir,c); – are rolul de a cauta caracterul c in sirul sir. Cautarea se face de la stanga la dreapta, iar functia intoarce adresa subsirului care incepe cu prima aparitie a caracterului c. Daca nu este gasit caracterul, functia returneaza 0. Diferenta dintre adresa sirului initial si cea a subsirului returnat reprezinta chiar pozitia caracterului cautat in sirul 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;

 Ex: Sa se afiseze toate pozitiile unui caracter intr-un sir

#include <iostream.h>

#include <string.h>

void main()

{char a[100],\*p,c;

cin.get(a,100);

cin>>c;

p=strchr(a,c);

while (p)

{cout<<"Pozitia "<<p-a<<endl;

 p++;

 p=strchr(p,c);}}

        Functia **strrchr**
strrchr(sir,c); – are acelasi rol cu strchr, cu deosebirea ca returneaza adresa ultimei aparitii a caracterului (cautarea se face de la dreapta spre stanga; r = right)

       Functia **strcmp**
int strcmp(sir1,sir2); – are rolul de a compara doua siruri de caractere. Valoarea returnata este <0 (daca sir1<sir2), =0 (daca sir1=sir2) si >0 (daca sir1>sir2). Functia strcmp *face distinctie* intre literele mari si cele mici ale alfabetului.
Obs: Functia **strcmp** returneaza diferenta dintre codurile ASCII ale primelor caractere care nu coincid

        Functia **stricmp**
int stricmp(sir1,sir2); – are acelasi rol cu strcmp, cu deosebirea ca *nu face distinctie* intre literele mari si cele mici ale alfabetului (i = ignore).

        Functia **strstr**
strstr(sir1,sir2); – are rolul de a identifica daca sirul sir2 este subsir al sirului sir1. Daca este, functia returneaza adresa de inceput a subsirului sir2 in sirul sir1, altfel returneaza adresa 0. In cazul in care sir2 apare de mai multe ori in sir1, se returneaza adresa de inceput a primei aparitii. Cautarea se face de la stanga la dreapta

       Functia **strtok**
strtok(sir1,sir2); – are rolul de a separa sirul sir1 in mai multe siruri (cuvinte) separate intre ele prin unul sau mai multe caractere cu rol de separator. Sirul sir2 este alcatuit din unul sau mai multe caractere cu rol de separator.
Functia **strtok** actioneaza in felul urmator:

o        Primul apel trebuie sa fie de forma strtok(sir1,sir2); Functia intoarce adresa primului caracter al primei entitati. Dupa prima entitate, separatorul este inlocuit automat prin caracterul nul.

o        Urmatoarele apeluri sunt de forma strtok(NULL,sir2); De fiecare data, functia intoarce adresa de inceput a urmatoarei entitati, adaugand automat dupa ea caracterul nul.

o        Cand sirul nu mai contine entitati, functia returneaza adresa nula.

*Exemplu*:

//Sa se separe cuvintele dintr-un text.

#include <iostream.h>

#include <conio.h>

#include <string.h>

void main()

{char text[100],cuv[10][10],\*p,\*r,separator[]=",. !?";int i=0,nr=0;

clrscr();

cout<<"Dati sirul:";cin.get(text,100);

strcpy(p,text);

p=strtok(p,separator);

while (p)

 {strcpy(cuv[++nr],p);

 p=strtok(NULL,separator);}

cout<<"Sunt "<<nr<<" cuvinte:"<<endl;

for (i=1;i<=nr;i++) cout<<cuv[i]<<endl;

getch();}

       Functia **strspn** cu forma generala
int strspn(sir1,sir2); – are rolul de a returna numarul de caractere ale sirului sir1 (caractere consecutive care incep obligatoriu cu primul caracter) care se gasesc in sirul sir2.

*Exemplu*:
strspn(“AB2def”,”1B3AQW”); 🡪 returneaza 2, pentru ca primele 2 caractere ‘A’ si ‘B’

din sir1 se gasesc in sir2.

strspn(“FAB2def”,”16A32BF”); 🡪 returneaza 0, deoarece caracterul ‘F’ cu care incepe sir1 nu se gaseste in sir2.

       Functia **strcspn** cu forma generala
int strspn(sir1,sir2); – are rolul de a returna numarul de caractere ale sirului sir1 (caractere consecutive care incep obligatoriu cu primul caracter) care *nu* se gasesc in sirul sir2.

*Exemplu*:
strspn(“AB2def”,”123”); 🡪 returneaza 2, pentru ca primele 2 caractere din sir1 nu se gasesc in sir2.

//Se citeste un sir de caractere care nu contine caractere albe. Sa se decida daca sirul este alcatuit exclusiv din caractere numerice.

#include <iostream.h>

#include <conio.h>

#include <string.h>

void main()

{char text[100],cifre[]="0123456789";

clrscr();

cout<<"Dati sirul:";cin.get(text,100);

if (strcspn(cifre,text)==strlen(text))

cout<<"exclusiv numeric";

else cout<<”nenumeric”;

getch();}

       Functia **strlwr** cu forma generala
strlwr(sir); – are rolul de a converti toate literele mari din sir in litere mici. Restul caracterelor raman neschimbate.

        Functia **strupr** cu forma generala
strupr(sir); – are rolul de a converti toate literele mici din sir in litere mari. Restul caracterelor raman neschimbate

       Functia **strbrk** cu forma generala
strpbrk(sir1,sir2); – actioneaza in felul urmator:

o        Cauta primul caracter al sirului sir1 in sir2. Daca este gasit, returneaza adresa sa din cadrul sirului sir1 si executia se termina. Altfel, se trece la pasul urmator.

o        Cauta al doilea caracter al sirului sir1 in sir2. Daca este gasit, returneaza adresa sa din cadrul sirului sir1 si executia se termina. Altfel, se trece la pasul urmator.

o        …

o        Daca nici un caracter al sirului sir1 nu apartine sirului sir2, functia returneaza adresa nula.

       Functia **atof** cu forma generala
double atof(sir); – converteste un sir catre tipul double. Daca aceasta conversie esueaza (se intalneste un caracter nenumeric), valoarea intoarsa este 0. Aceasta functie (ca si cele similare) necesita includerea librariei stdlib.h.

       Functia \_**atold** cu forma generala
long double \_atold(sir); – converteste un sir catre tipul long double. Daca aceasta conversie esueaza, valoarea intoarsa este 0.

       Functia **atoi** cu forma generala
int atoi(sir); – converteste un sir catre tipul int. Daca aceasta conversie esueaza (se intalneste un caracter nenumeric), valoarea intoarsa este 0.

       Functia **atol** cu forma generala
long atol(sir); – converteste un sir catre tipul long. Daca aceasta conversie esueaza (se intalneste un caracter nenumeric), valoarea intoarsa este 0.

        Functia **itoa** cu forma generala
itoa(int valoare,sir,int baza); – converteste o valoare de tip int in sir, care este memorat in variabila sir. Baza retine baza de numeratie catre care sa se faca conversia. In cazul bazei 10, sirul retine si eventualul semn -.

       Functia **ltoa** cu forma generala ltoa(long valoare,sir,int baza); – converteste o valoare de tip long int in sir, care este memorat in variabila sir.

        Functia **ultoa** cu forma generala
ultoa(unsigned long valoare,sir,int baza); – converteste o valoare de tip unsigned long in sir, care este memorat in variabila sir.