3. Şiruri de caractere

3.1. Declararea şi memorarea vectorilor de caractere

Şirurile de caractere sunt de fapt succesiuni de caractere. Am învăţat deja să memorăm succesiuni de caractere cu ajutorul tablourilor.

De exemplu:

char cuvant[15];

este un tablou unidimensional (vector), care poate memora 15 elemente de tip char. Acesta poate fi reprezentat în memorie astfel:

cuvant

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Presupunând că dorim să citim, să memorăm şi să afişăm cuvântul "*programator*", vom scrie următorul program:

```
                                  // CUVANT
#include <iostream.h>
#include <conio.h>
void main()
{char cuvant[15],i;
 cout<<"introduceti cuvantul:";
 for(i=0;i<11;i++)
     {cin>>cuvant[i];}
 cout<<"cuvantul citit:";
 for(i=0;i<11;i++)
    cout<<cuvant[i];
  getch();
}
```

Putem reprezenta memorarea variabilei *cuvânt* astfel:

cuvant

p	r	o	g	r	a	m	a	t	o	r				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

În realitate în fiecare octet al tabloului **se memorează codul ASCII al caracterului respective.**

Octeţii haşuraţi ai tabloului reprezintă zona de memorie neutilizată.

După cum observăm, din exemplul precedent, o astfel de modalitate de lucru este greoaie şi neplăcută.

Limbajul de programare C++, permite prelucrarea şirurilor de caractere într-un mod mult mai facil.

3.2. Iniţializarea şirurilor de caractere

Un şir de caractere poate fi iniţializat la declarare astfel:

char cuvant[ ]=”programator”;

In urma iniţializării, la sfârşitul cuvântului va fi adăugat caracterul **0** (codul ASCII al caracterului nul).

cuvant

p	r	o	g	r	a	m	a	t	o	r	**0**
0	1	2	3	4	5	6	7	8	9	10	11

Compilatorul va calcula numărul de octeţi necesari pentru memorarea cuvântului programator. În acest caz tabloul conţine 12 octeţi, 11 pentru memorarea cuvântului şi 1 pentru memorarea codului caracterului nul.

char cuvant[15]=”programator”;

cuvant

p	r	o	g	r	a	m	a	t	o	r	**0**			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

În acest caz am memorat un vector de 15 octeţi, mai mulţi decât sunt necesari. Ultimii trei octeţi rămân neutilizaţi.

3.3. Citirea şi afişarea şirurilor de caractere

Modalitatea de memorare a şirurilor, prezentată mai sus (memorând la sfârşitul şirului codul ASCII al caracterului nul), ne permite citirea şi afişarea cu mare uşurinţă a şirurilor de caractere. De asemenea Limbaul C deţine numeroase funcţii pentru prelucrarea şirurilor de caractere.

Funcţiile ***cin*** şi ***cout*** suportă şirurile de caractere terminate cu caracterul nul, astfel pot fi citite respectiv afişate şirurile de caractere.

Exemplul de mai sus poate fi scris mult mai simplu astfel:

```
#include <iostream.h>          //CUVANT 1
#include <conio.h>
void main()
{char cuvant[15];
//citeste un sir de caractere
 cout<<"introduceti cuvantul:"; cin>>cuvant;
//afiseaza sirul de caractere de la primul octet pana la întalnirea caracterului nul
cout<<"cuvantul citit:"; cout<<cuvant;
getch();
}
```

În urma introducerii cuvântului “*programator*”, în memorie vom avea următoarea reprezentare:

cuvant

p	r	o	g	r	a	m	a	t	o	r	**0**			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Un vector care conţine un şir de caractere citit sau iniţializat în modul anterior prezentat, poate fi adresat pe componente, în mod normal. De exemplu **cuvant[0]='p'**, **cunvant[1]='r'**, ş.a.m.d.

Din nefericire cu ajutorul funcţiei ***cin*** nu putem citi şiruri de cractere care conţin spaţii sau alte caractere albe.

Funcţia ***cin*** procedează astfel la citirea unui şir de caractere:

- Se sar toate caracterele albe
- Se citeste şirul care începe cu primul caracter care nu este alb
- Citirea se încheie la întâlnirea primului caracter alb.

Exemplu:
Pentru şirul introdus de la tastatură: " *invatam despre siruri de caractere*", programul de mai sus va afişa doar "*invatam*".

Pentru citirea şirurilor de caractere care conţin cuvinte separate prin spaţii sau alte caractere albe vom folosi funcţia ***cin.get().***

Formatul funcţiei:
cin.get(nume_sir, int nr, char='\n')

unde:

- **nume_sir** – reprezintă identificatorul şirului de caractere
- **nr** - reprezintă numărul de caractere care se vor citi împreună cu caracterul terminal nul.
- Al treilea parametru desemnează **caracterul la întâlnirea căruia se opreşte citirea şirului** (valoarea lui implicită este **'\n'**), prin urmare în cazul în care nu dorim ca citirea sa se termine cu un alt cracter diferit de '\n', acest parametru poate să lipsească.

Funcţia ***cin.get*** citeşte caractere pâna când este îndeplinită una din conditiile: **fie s-au citit nr-1 caractere fie a întâlnit caracterul specificat de parametrul al treilea.**

Exemplu:

```
#include <iostream.h>          //SIR
#include <conio.h>
void main()
{char sir[50];
 cout<<"introduceti cuvantul:";
 cin.get(sir,40);          //citeste maximum 40 de caractere sau se termina la <enter>
 cout<<"cuvantul citit:";
 cout<<sir;
 getch();
}
```

În C pot exista mai multe funcţii cu acelaşi nume dar care diferă între ele prin numărul parametrilor, cu acţiuni diferite. (această modalitate de utilizare a funcţiilor poartă numele de *supraîncarcarea funcţiilor*)

Există încă o implementare a funcţiei ***cin.get()*** fară parametri. În acest caz funcţia citeşte un singur caracter alb sau nu.

Observaţie:

În cazul citirii a două şiruri de caractere, în buffer-ul de citire se depune după citirea primului şir, caracterul '\n', datorită faptului că s-a tastat <enter>. Citirea celui de-al doilea şir se opreşte pentru că se preia '\n'.din buffer.

Apelând funcţia *cin.get()* aceasta va prelua caracterul '\n', eliminăm acest inconvenient, iar al doilea şir se va citi în mod normal.

```
#include <iostream.h>          //SIR
#include <conio.h>

void main()
{char sir1[50],sir2[50];
 cout<<"introduceti sir1:";
 cin.get(sir1,40);
 cout<<"cuvantul citit:";
 cout<<sir1<<endl;
 cin.get();                    //citeste caracterul '\n' de la sfarsitul sirului sir1
 cout<<"introduceti sir2:";
 cin.get(sir2,40);
 cout<<"cuvantul citit:";
 cout<<sir2;
 getch();
}
```

3.4. Tipul char*

Tipul de dată ***char**** poartă numele de ***pointer la caracter***.

O variabilă de tip pointer la caracter, este capabiăl să reţină **adresa de memorie a unui caracter.**

Trebuie să spunem în acest moment faptul că **numele unei variabile de tip şir de caractere (sau vector în general), reprezintă adresa de memorie a primului octet al şirului (vectorului).**

Datorită acestui fapt, elementele unui şir de caractere pot fi accesate prin intermediul adreselor lor, astfel:

Presupunem că am declarat şi iniţializat o variabilă de tip şir de caractere:

char sir[]="programator";

să vedem ce va afişa funcţia

cout<<sir;

evident va afişasirul: **programator**, prin urmare am adresat şirul de la primul octet.
dacă scriem **sir+1**, vom adresa şirul de la al doilea octet

cout<<sir+1;

va afişa :**rogramator** deoarece la adresa şirului am adunat 1 şi am obţinut adresa următorului octet, adica octelul al doilea din şir.

cout<<sir+2;

va afişa :**ogramator** şi aşa mai departe, întrucât la adresa şirului am adunat 2 şi am obţinut adresa celui de-al treilea octet din şir.
Expresiile de forma: **sir, sir+1, sir+2… sunt de tip char*, adrese de memorie ale unor elemente de tip caracter.**

Exemplu:

```
#include <iostream.h>          //ADRESE
#include <fstream.h>
void main()
{char sir[15],*p;                          //p-pointer la o variabila de tip caracter
 ofstream  f("sir.out");
 cout<<"introduceti sir:";
 cin>>sir;
 f<<"sir:"<<sir<<endl;
 f<<"sir+1:"<<(sir+1)<<endl;
 f<<"sir+2:"<<(sir+2)<<endl;
 f<<"sir+3:"<<(sir+3)<<endl;

 f<<"(sir)[0]:"<<(sir)[0]<<endl;
 f<<"(sir+1)[0]:"<<(sir+1)[0]<<endl;
 f<<"(sir+1)[1]:"<<(sir+1)[1]<<endl;
 f<<"(sir+1)[2]:"<<(sir+1)[2]<<endl;
 f<<"(sir+2)[0]:"<<(sir+2)[0]<<endl;
 f<<"(sir+2)[1]:"<<(sir+2)[1]<<endl;;

 p=sir+3;        //se depune adresa celui de al patrulea element in pointerul p (sir+3)
 f<<"sir+3="<<p<<endl;
 p++;
 f<<"sir+3="<<p<<endl;
 f.close();
}
```

Introducând şirul ***abcdefgijk*,** în exemplul de mai sus, în fişierul sir.out se va afişa:
sir:abcdefghijk
sir+1:bcdefghijk
sir+2:cdefghijk
sir+3:defghijk
(sir)[0]:a
(sir+1)[0]:b
(sir+1)[1]:c
(sir+1)[2]:d
(sir+2)[0]:c
(sir+2)[1]:d
sir+3=defghijk
sir+3=efghijk

sir

a	b	c	d	e	f	g	h	i	j	k	**0**			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

3.5. Funcţii care operează cu şiruri de caractere

Şirurile de caractere fiind în realitate tablouri unidimensionale nu sunt permise atribuiri de forma sir1=sir2 şi comparări între sir1 şi sir2. Pentru efectuarea acestora şi a altor operaţii sunt implementate funcţii care lucrează cu şiruri de caractere.

Pentru ca aceste funcţii să poată fi folosite, trebuie inclus heder-ul ***string.h***.

3.5.1. Determinarea lungimii unui şir de caractere:

size_t strlen(const char *s);

unde:
size_t – tip întreg fără semn (al funcţiei)
const char *s **-** argumentul este de tip char* şi identifică un şir de caractere.

Exemplu: Se citeşte un şir de caractere, să se afişeze lungimea şirului citit.

```
#include <conio.h>          //LUNGIME
#include <iostream.h>
#include <string.h>

void main()
{char sir[50];
 cout<<"introduceti sir:";
 cin>>sir;
 cout<<strlen(sir);   //determina si afiseaza lungimea sirului
 getch();
}
```

3.5.2. Copierea unui şir într-un alt şir.

char *strcpy(char *dest, const char *sursa);

unde:
char *dest - argumentul identifică şirul destinaţie
const char *sursa - argumentul identifică sirul sursă
Descriere:

Copiază şirul ***sursa*** în şirul ***dest***, copierea se termină după ce a fost copiat caracteru nul.

Valoare returnată:

Funcţia **strcpy** returnează adresa şirului ***dest*** (destinaţie)

Exemplu: Se citeşte un şir de caractere, şă se copieze într-un alt şir.

```
#include <iostream.h>       //COPIERE
#include <iostream.h>
#include <string.h>
void main()
{char sir1[50],sir2[50];
 cout<<"introduceti sir1:"; cin>>sir1;
 strcpy(sir2,sir1); cout<<sir2;   //copiaza sir1 in sir2
}
```

3.5.3. Concatenarea a două şiruri

a.) Funcţia strcat

char *strcat(char *dest, const char *sursa);

Descriere:

Funcţia **strcat** adaugă la sfârşitul şirului ***dest***, o copie a sirului ***sursa***. Lungimea sirului rezultat va fi *strlen(dest) + strlen(sursa).*

Valoare returnată:

Funcţia **strcat** returnează adresa şirului ***dest***.

Exemplu: Se citesc două şiruri de caractere. Să se concateneze la primul şir, la al doilea şir citit.

```
#include <iostream.h>          //CONCATENARE 1
#include <string.h>
#include <conio.h>
void main()
{char sir1[50],sir2[50];
 cout<<"introduceti sir1:"; cin>>sir1;
 cout<<"introduceti sir2:"; cin>>sir2;
 strcat(sir1,sir2);              //concateneaza sir2 la sir1
 cout<<sir1;
getch();
}
```

b.) Funcţia strncat

char *strncat(char *dest, const char *sursa, size_t maxlen);

Descriere

Copiază din şirul ***sursa*** cel mult ***maxlen*** caractere la sfărşitul şirului ***dest***. Şirul dest va avea lungimea strlen(dest) + maxlen.

Valoare returnată

Funcţia **strncat** returnează adresa şirului ***dest***.

Exemplu: Se citesc două şiruri de caractere, să se concateneze primele 3 caractere din şirul al doilea la primul şir citit.

```
#include <iostream.h>          //CONCATENARE 2
#include <string.h>
#include <conio.h>
void main()
{char sir1[50],sir2[50];
 cout<<"introduceti sir1:"; cin>>sir1;
 cout<<"introduceti sir2:"; cin>>sir2;
 strncat(sir1,sir2,3);           //concateneaza 3 caractere din sir2 la sir1
 cout<<sir1;
 getch();
}
```

3.5.4. Căutarea într-un şir

a.) Funcţia strchr

char *strchr(const char *s, int c);

Descriere

Caută prima apariţie a caracterului c în şirul s. Căutarea se face de la stânga la dreapta. Caracterul nul este considerat ca făcând parte din şir.

Valoare returnată

Funcţia **strchr** returnează adresa primei apariţii a caracterului c în şir sau o expresie de tip char* cu valoarea 0 (null).

Exemplu: Se citeşte un şir de caractere şi un caracter. Căutăm caracterul citit în şir, în cazul în care îl găsim afişăm poziţia în şir unde se găseşte acest caracter, în cazul în care caracterul nu se găseşte în şir vom afişa un mesaj.

```
#include <iostream.h>          //CAUTARE CARACTER
#include <string.h>
#include <conio.h>
void main()
{char sir[50],*p,c;
 cout<<"introduceti sir:"; cin>>sir;
 cout<<"introduceti caracterul cautat:"; cin>>c;
 p=strchr(sir,c);                        //cauta caracterul c in sir
 if(p)
   cout<<c<<" se gaseste pe pozitia "<<(p-sir)<<" in sir"; //afiseaza pozitia din sir
 else
   cout<<c<<" nu se gaseste in sir";
 getch();
}
```

Observaţie:

Am determinat poziţia caracterului găsit în şir scăzând din adresa acestuia, adresa de început a şirului **(p-sir)**. Astfel am obţinut numărul de elemente cuprinse între cele două adrese.

b.) Funcţia strrchr

char *strrchr(const char *s, int c);

Are acelaşi rol ca şi funcţia ***strchr,*** dar căutarea caracterului c se face de la dreapta la stânga.

c.) Funcţia strstr

char *strstr(const char *s1, const char *s2);

Descriere

Caută prima apariţie a subşirului s2 în cadrul şirului s1. Căutarea se face de la stânga la dreapta.

Valoarea returnată

Funcţia returnează adresa elementului din s1, unde începe subşirul s2. Dacă s2 nu apare în şirul s1, returnează o expresie de tip char* cu valoarea 0 (null).

Exemplu: Se citesc două şiruri de caractere şir1 şi şir2. Dacă şir2 este subşir al lui şir1 se va afişa poziţia în şir1 unde se găseşte. În caz contrar se afişează un mesaj.

```
#include <iostream.h>          //CAUTARE SIR
#include <string.h>
#include <conio.h>

void main()
{char sir1[50],sir2[50],*p;
 cout<<"introduceti sir1:"; cin>>sir1;
 cout<<"introduceti sir2:"; cin>>sir2;
 p=strstr(sir1,sir2);
 if(p)
   cout<<sir2<<" se gaseste pe pozitia "<<(p-sir1)<<" in sir";
 else
   cout<<sir2<<" nu se gaseste in sir1";
 getch();
}
```

3.5.5. Compararea a două şiruri

int strcmp(const char *s1, const char *s2);

Descriere

Funcţia ***strcmp*** compară şirul s1 cu şirul s2. Compararea începe cu primul caracter din fiecare şir şi continuă secvenţial, până când cele două caractere comparate sunt diferite sau până când unul dintre şiruri se termină.

Se compară codurile ASCII ale caracterelor, astfel, un caracter este mai mic decât altul, dacă are codul ASCII mai mic. Se consideră mai mic şirul care conţine caracterul mai mic.

Dacă cele două şiruri pe percursul comparării au toate caracterele egale, dar unul dintre ele se termină, se consideră mai mic şirul care se terminat (are mai puţina caractere).

Valoarea returnată

- Returnează o valoare **< 0**, dacă **s1<s2**
- Returnează o valoare **=0**, dacă **s1=s2**
- Returnează o valoare **>0**, dacă **s1>s2**

Exemplu: Se citesc două cuvinte. Să se afişeze cuvintele în ordine lexicorafică. Se va afişa un mesaj în cazul în care cele două cuvinte sunt egale.

```
#include <iostream.h>          //COMPARARE SIRURI
#include <string.h>
#include <conio.h>
void main()
{char sir1[50],sir2[50];
 int k;
 cout<<"introduceti sir1:"; cin>>sir1;
 cout<<"introduceti sir2:"; cin>>sir2;
 k=strcmp(sir1,sir2);                    //compară cele două siruri
 if(k<0)
   cout<<sir1<<" "<<sir2;
 else
   if(k>0)
      cout<<sir2<<" "<<sir1;
   else
      cout<<"sirurile sunt egale";
 getch();
}
```

3.5.6. Separarea entităţilor

char *strtok(char *s1, const char *s2);

Descriere

Caută în s1, un şir care este separat de delimitatori definiţi în sirul s2. Să numim şirul cu această proprietate ***entitate***.

Prima apelare a funcţiei strtok caută de la începutul şirului şi returnează adresa primului caracter al entităţii determinate, apoi inserează caracterul 0 (null) la sfârşitul acestei entităţi

Urmatoarea căutare trebuie să înceapă de la caracterul nul inserat.

Valoarea returntă

Funcţia returnează adresa entităţite determinate în s1 sau o expresie de tip char* cu valoarea 0 (null) dacă nu găseşte nici o entitate.

Exemplu: Se citeşte un text care contine cuvinte separate prin spaţii, virgule sau punct. Să se afişeze cuvintele căte unul pe o linie şi să se numere cuvintele din text.

```
#include <iostream.h>    //NUMAR CUVINTE
#include <conio.h>
#include <string.h>
char text[101],*p,separator[]=" ,.";
int k=0;
void main()
{
 clrscr();
 cout<<"Introduceti textul:";
 cin.get(text,100);
 cout<<"\n";
 p=strtok(text,separator);                    //determina prima entitate
 while(p)
   {k++;
    cout<<p<<"\n";
    p=strtok(NULL,separator);                //determină următoarele entităţi
   }
 cout<<"\nNumarul de cuvinte:"<<k;
 getch();
}
```

3.5.7. Funcţii de conversie

a.) Conversia unui întreg în sir de caractere

char *itoa(int valoare, char *s, int baza);

valoare –valoarea de convertit
s -variabila sir care va conţine şirul obţinut
baza -specifica baza utilizată în conversie (trebuie să fie între 2 şi 36 inclusiv). In cazul în care valoarea este negativa şi baza este 10, primul caracter al şirului va fi (-).

Descriere

Pentru utilizarea funcţiei trebuie inclus heder-ul ***stdlib.h***.

Converteşte un întreg într-un şir de caractere. Şirul obţinut prin conversie va conţine caracterul 0 pe ultima poziţie.

Lungimea şirului returnat poate fi cel mult 17 bytes.

Valoarea returnată

Funcţia returnează adresa şirului obţinut.

Exemplu:Să se convertească un număr întreg într-un şir de caractere, folosind baza 2.

```
#include <iostream.h>    //ITOA
#include <conio.h>
#include <string.h>
#include <stdlib.h>

void main()
{int k,baza;
 char s[17];
 cout<<"numar:";cin>>k;
 cout<<"baza:";cin>>baza;
 itoa(k,s,baza);                 //converteste numarul in sir
 cout<<"sirul:"<<s;
 getch();
}
```

Alte funcţii de conversie similare:

- Funcţia **ltoa** converteşte o valoare de tip *long int* într-un şir de caractere.

char *ltoa(int valoare, char *s, int baza);

- Funcţia **ultoa** converteşte o valoare de tip *unsigned longt* într-un şir de caractere.

char *ltoa(int valoare, char *s, int baza);

b.) Conversia unui şir de caractere într-un întreg

long strtol(const char *s, char **adresa, int baza);

s -sirul se convertit
adresa -adresa unei variabile de tip char*, rwţine adresa caracterului din şir care nu poate fi convertit.
baza -baza în care este considerat numărul sub formă de şir. Poate avea valorile 8, 10 sau 16.

Descriere

Pentru utilizarea funcţiei trebuie inclus heder-ul ***stdlib.h***.
Funcţia ***strtol*** converteşte şir de caractere într-o valoare de tip ***long***.

Valoarea returnată

Funcţia returnează adresa şirului obţinut.

Exemplu:Să se convertească un şir de caractere conţinând un număr în baza 10 în număr. În cazul în care nu poate fi convertit să se afişeze de unde nu poate fi

convertit.

```
#include <iostream.h>    //STRTOL
#include <conio.h>
#include <string.h>
#include <stdlib.h>

void main()
{int baza;
 long k;
 char s[17],*p=NULL;
 cout<<"sir:";cin>>s;
 cout<<"baza(8,10,16):";cin>>baza;
 k=strtol(s,&p,baza);                    //face conversia
 cout<<"valoarea numerica:"<<k<<endl;
 cout<<"nu s-a putut converti de la:"<<p;     //afiseaza de unde nu poate fi convertit
 getch();
}
```

Alte funcţii de conversie similare:

- Funcţia ***strtod*** converteşte şir de caractere într-o valoare de tip ***double***.

double strtod(const char *s, char **adresa, int baza);

- Funcţia ***strtold*** converteşte şir de caractere într-o valoare de tip ***long double***.

long double strtold(const char *s, char **adresa, int baza);

- Funcţia ***strtoul*** converteşte şir de caractere într-o valoare de tip ***unsigned long***.

unsigned long strtol(const char *s, char **adresa, int baza);

3.6. Probleme rezolvate

1. Se citesc de la tastatură două şiruri s1 şi s2. Să se elimine toate apariţiile lui s2 în s1. Şirul s2 conţine doar un cuvânt.

```
#include <conio.h>          //STERGE
#include <iostream.h>
#include <string.h>

void main()
{char s1[100],s2[30],*p;
 int l;
 cout<<"introduceti textul:";cin.get(s1,100);
 cout<<"introduceti cuvantul:";cin>>s2;
 l=strlen(s2);                    //determina lungimea sirului s2
 p=strstr(s1,s2);                 //cauta s2 in s1
 while(p)
   {strcpy(p,p+l);                //sterge sirul s2, copiind sirul de pa pozitia p+l pe pozitia p
    p=strstr(s1,s2);              //cauta o noua aparitie a lui s2 in s1
   }
 cout<<s1;
 getch();
}
```

2. Se citeşte un şir s şi două cuvinte c1 şi c2. Să se înlocuiască toate apariţiile lui c1 în s cu c2.

```
#include <conio.h>          //INLOCUIRE
#include <iostream.h>
#include <string.h>
void main()
{char s[100],ss[100]="",c1[30],c2[30],*p;     //ss sir vid
 int l;
 cout<<"introduceti textul:";cin.get(s,100);
 cout<<"introduceti primul cuvantul:";cin>>c1;
 cout<<"introduceti al doilea cuvantul:";cin>>c2;
 l=strlen(c1);              //determina lungimea lui c1
 p=strstr(s,c1);            //determina prima aparitie a lui c1 in s
 while(p)
   {strncat(ss,s,p-s);      //concateneaza p-s caractere la sirul ss
    strcat(ss,c2);          //concateneaza c2 la ss
    strcat(ss,p+l);         //concateneaza sirul de dupa aparitia lui c1 la ss
    strcpy(s,ss);           //copiaza ss in s
    strcpy(ss,"");          //anuleaza sirul ss
    p=strstr(s,c1);         //cauta urmatoarea aparitie a lui c1 in s
   }
 cout<<s; getch();
}
```

3.Se citeşte de la tastatură un text care conţine cuvinte separate prin spaţiu, virgulă sau punct. Să se afişeze cuvintele textului în ordine alfabetică.

Vom separa cuvintele textului folosind funcţia strtok şi le vom memora rând pe rând într-un tablou. Vom ordona tabloul.

```
#include<iostream.h>                    //ORDONARE CUVINTE
#include<conio.h>
#include<string.h>
char s[100],*p,a[20][20],separator[]=" ,.";//a-tablou de siruri,retine cuvintele
int k=0,i;
void separare();
void afisare();
void ordonare();
void main()
{ cout<<"Dati sirul:"<<endl;cin.get();cin.get(s,100,'\n');
  separare();
  cout<<"cuvintele:"<<endl;   afisare();
  ordonare();
  cout<<"cuvintele ordonate:"<<endl;  afisare();
  getch();
}
void separare()          //separa cuvintele
{  p=strtok(s,separator);
  while (p)
  { strcpy(a[k],p);
    k++;
    p=strtok(NULL,separator);
  }
 }
void afisare()         //afiseaza cuvintele
{ int i;
  for(i=0;i<k;i++)
   cout<<a[i]<<endl;
}
void ordonare()         //ordoneaza cuvintele
{int sw,i;
 char aux[20];
 do {sw=1;
      for(i=0;i<k-1;i++)
      if (strcmp(a[i],a[i+1])>0)
           {       strcpy(aux,a[i]);
                   strcpy(a[i],a[i+1]);
                   strcpy(a[i+1],aux);
                   sw=0;
           }
    }
 while(!sw);
}
```

3.7. Evaluare

TESTUL 1

1. Descrieţi modul de funcţionare al comenzii strcmp.
2. Dacă sir1=”calculator” iar sir2=”peformant”.
   Spuneţi ce va conţine sir1 dupa aplicarea funcţiei:
   Strncat(sir1,sir2,6)
3. Se citeste un text de la tastatură. Să se afiseze histograma vocalelor din text.
4. Se citesc de la tastatură n numere naturale <=10. Să se convertească fiecare număr într-un sir de caractere care reprezintă numărul în baza b (2<=b<=10).

TESTUL 2

1. Descrieţi modul de funcţionare al comenzilor: strchr, strstr.
2. Spuneţi care este efectul programului de mai jos:

```
 void main()
{ char sir1[ ]="calculator", sir2[ ]="performant";
  strcpy(sir1+5," ");
  strcat(sir1,sir2);
  cout<<sir1;
  getch();
}
```

3. Se citesc de la tastatură numele colegilor de clasă. Să se afiseze în ordine alfabetică numele citite.

TESTUL 3

1. Descrieţi modul de funcţionare al comenzilor:

```
char *ltoa(int valoare, char *s, int baza);
long strtol(const char *s, char **adresa, int baza);
```

2. Spuneţi care este efectul programului de mai jos:

```
void main()
{
 char sir1[]="calculator", sir2[]="performant";
  strcpy(sir1+3,sir2+6);
  cout<<sir1;
  getch();
}
```

3. Se citeşte un text. Să se afişeze cuvintele din text care contin cel puţin 3 vocale.

3.8. Probleme propuse

1. Scieţi o funcţie care inserează pe poziţia p a unui şir un alt sir dat.

2. Scrieţi o funcţie care şterge dintr-un şir dat n caractere, începând cu poziţia p.

3. Fiind dat un cuvânt, să se afişeze toate sufixele sale.

4. Fiind dat un cuvânt, să se afişeze toate prefixele sale.

5. Se citeste un text de la tastatură, să se afiseze cuvintele în ordinea crescătoare a lungimii lor.

6. Se citeşte un text de la tastatură, care conţine cuvinte separate prin spaţii. Să se elimine spaţiile din text.

7. Se citesc cuvinte până la introducerea cuvantului END. Afişaţi cuvintele în ordinea inversă citirii lor.

8. Număraţi ăpariţiile unui cuvânt într-un text.

9. Număraţi apariţiile fiecărei litere într-un text, fără a face distincţie între literele mari şi mici.

10. Se citeste de la tastatură un text care conţine mai multe propoziţii. Cuvintele textului sunt separate prin : spatiu, virgulă sau punct. Să se numere câte propoziţii conţine textul, iar fiecare propoziţie, câte cuvinte are.

11. se citeşte un cuvânt de la tastatură. Să se ghicească cuvântul prin încercări repetate. Fiecare tentativă va fi urmată de afisarea literelor ce se corespund.

12. Căutaţi un cuvânt într-un text şi afişaţi numărul de apariţii ale cuvântului. Dacă acest cuvânt este inclus în alte cuvinte din text, nuu se va număra.