Probleme rezolvate cu șiuri de caractere

1. Se consideră un text în care unicul separator este spațiul. Știind că între oricare două cuvinte pot exista mai mulți separatori, să se determine numărul de cuvinte din text.

Exemplu: Pentru textul 'Am venit repede ' se va afișa 3.

Soluție

Algoritmul presupune parcurgerea caracter cu caracter a textului şi identificarea numărului de perechi de caractere alăturate care pot reprezintă finalul unui cuvânt (caracter diferit de spațiu urmat de un separator).

```
1  var s:string;
2      i,nr:integer;
3  begin
4   readln(s);
5   s:=s+' '; nr:=0;
6   for i:=1 to length(s)-1 do
7    if(s[i]<>' ')and
8      (s[i+1]=' ')then inc(nr);
9   writeln(nr);
10 end.
11
```

```
#include <stdio.h>
#include <string.h>
char s[256];int nr,i;
void main() {
 gets(s);
 strcat(s," "); nr=0;
 for (i=0;i+1<strlen(s);i++)
 if (s[i]!=' '&&s[i+1]==' ')
    nr++;
 printf("%d\n",nr);
}
```

2. Se citeşte de la tastatură un vers al unei poezii şi o silabă. Să se realizeze un program care determină numărul de apariţii al silabei citite în textul respectiv.
Exemplu : Pentru versul 'Un curcubeu multicolor' şi silaba 'cu' se va afişa 2.

Soluţie
Atât versul citit cât şi silaba vor fi reţinute în variabile de tip şir de caractere *vers* şi *s*. Algoritmul propus se bazează pe căutarea repetată a subşirului *s*, folosindu-ne de funcţia predefinită *pos() / strstr()*.

```
1  var vers,s:string;
2      nr,p,l:integer;
3  begin
4   readln(vers);
5   readln(s);
6   nr:=0;
7   while pos(s,vers)<>0 do begin
8     p:=pos(s,vers);
9     l:=length(s);
10    delete(vers,p,l);
11    inc(nr); {nr:=nr+1}
12  end;
13  writeln(nr);
14 end.
```

```
#include <stdio.h>
#include <string.h>
char vers[256],s[256];
int nr,p,l;
void main() {
 gets(vers); gets(s); nr=0;
 while (strstr(vers,s)!=NULL){
  p=strstr(vers,s)-vers;
  l=strlen(s);
  strcpy(vers+p,vers+p+l);
  nr++;
 }
 printf("%d\n",nr);
}
```

3. O propoziţie se consideră fiind palindrom dacă ignorând diferenţele dintre minuscule şi majuscule şi ignorând separatorii, va fi identică cu propoziţia obţinută prin citirea literelor de la dreapta spre stânga. De exemplu, propoziţia 'Ele fac cafele' este palindrom. Să se realizeze un program care permite citirea propoziţiei de la tastatură şi verifică dacă ca poate fi considerată palindrom. Cuvintele vor fi separate în cadrul propoziţiei prin spaţii (singurul separator prezent).

Soluţie:

Algoritmul propus conţine trei etape:
- ştergerea spaţiilor dintre cuvinte;
- transformarea în majuscule a fiecărei litere;
- cuvântul astfel obţinut se verifică dacă este palindrom.

```
1  var s:string; ok:boolean;
2      i,p:integer;
3  begin
4   readln(s);
5   while pos(' ',s)<>0 do begin
6           p:=pos(' ',s);
7           delete(s,p,1);
8   end;
9   for i:=1 to length(s) do
10        s[i]:=upcase(s[i]);
11  ok:=true;
```

```
#include <string.h>
#include <stdio.h>
char s[256]; int i,p,ok;
void main() {
 gets(s);
 while (strstr(s," ")!=NULL) {
    p=strstr(s," ")-s;
    strcpy(s+p,s+p+1);}
 for (i=0;i<strlen(s);i++)
  if (s[i]>='a'&&s[i]<='z')
   s[i]+='A'-'a';
```

```
12  for i:=1 to length(s)div 2 do
13    if s[i]<>s[length(s)-i+1]
14    then
15        ok:=false;
16  writeln(ok);
17  end.
```

```
ok=1;
for(i=0;i<strlen(s)/2;i++)
 if (s[i]!=s[strlen(s)-i-1])
  ok=0;
printf("%d\n",ok);
}
```

4. Se citeşte de la tastatură un şir de caractere care reprezintă un număr în baza 16. Să se realizeze un program care permite conversia în baza 10 a numărului citit. În situaţia în care şirul conţine caractere nepermise se va afişa mesajul "Imposibil".
Exemplu: Pentru numărul hexazecimal *A1B* se va afişa 2587.

Soluţie

Ştim că singurele cifre permise la scrierea unui număr într-o baza b ($2 \le b \le 10$) sunt 0..b-1, corespunzătoare resturilor care se pot obţine la împărţirea cu b. Dacă baza este mai mare ca 10, atunci fiecare din resturile obţinute vor fi codificate în ordine cu literele *A*,*B*,*C*, ş.a.m.d. În baza 16 resturile mai mari ca 9 sunt codificate astfel: '*A*'=10 ; '*B*'=11 ; '*C*'=12 ; '*D*'=13 ; '*E*'=14 ; '*F*'=15.
De exemplu, numărul $2AC_{(16)}$ reprezintă numărul $2*16^2+10*16^1+12*16^0 = 684_{(10)}$;
Pentru a evita calcularea puterii lui 16 din cadrul fiecărui termen putem rescrie expresia de mai sus şi sub forma: $2*16^2 + 10*16^1 + 12*16^0 = ((0*16 + 2)*16 + 10)*16 + 12$.
În cazul în care cifrele sunt exprimate prin litere se va realiza o corespondenţă între codul ASCII al său şi numărul pe care îl reprezintă în baza 16:

Caracter	Codul ASCII	Restul pe care îl indică în baza 16
'A'	65	10 =65-55
'B'	66	11=66-55
'C'	67	12=67-55
'D'	68	13=68-55
'E'	69	14=69-55
'F'	70	15=70-55
		ord(Litera)-55

```
1   var s:string;
2       i,nr,c,er:integer;
3   begin
4    readln(s); nr:=0;
5    for i:=1 to length(s) do
6    begin
7     if s[i]<='9' then
8      val(s[i],c,er)
9     else
10     c:=ord(upcase(s[i]))-55;
11     nr:=nr*16 + c;
12   end;
13   writeln(nr);
14  end.
```

```
#include <string.h>
#include <stdio.h>
char s[256];
int i,nr,c;
void main() {
 gets(s); nr=0;
 for (i=0;i<strlen(s);i++) {
  if (s[i]<='9') c=s[i]-'0';
  else c=s[i]-55;
  nr=nr*16+c;
 }
 printf("%d\n",nr);
}
```

5. Se citeşte de la tastatură un vers al unei poezii. Să se realizeze un program care determină numărul de cuvinte din text. Cuvintele sunt separate între ele prin caracterele: spaţiu (' '), virgula (,), punctul (.) sau punct şi virgula (;).
Exemplu: Pentru propoziţia 'Vremea trece...vremea vine,' se va afişa valoarea 4.

Soluţie

În cadrul algoritmul pe care îl prezentăm vom număra câte cuvinte sunt în vers folosindu-ne de următoarea regulă:
O poziţie în cadrul versului reprezintă începutul unui nou cuvânt dacă tipul caracterului de pe poziţia respectivă este literă iar cel precedent este un separator. Pentru a indentifica mai uşor tipul unui caracter vom folosi o variabilă şir de caractere a cărei valoare o va reprezenta şirul de separatori. ' ,.;' Dacă un caracter al versului nu se regăseşte printre caracterele acestuia atunci el este de tip literă. În C++ se poate folosi functia *strtok()*.

```
1  var s,sep:string;
2      x,y,nr,i:integer;
3  begin
4   readln(s);
5   nr:=0;
6   sep:=' ,.;';
7   s:='.' + s + '.';
8   for i:=2 to length(s) do
9  begin
10     x:=pos(s[i],sep);
11     y:=pos(s[i-1],sep);
12     if (x=0)and(y<>0) then
13       nr:=nr+1;
14   end;
15   writeln(nr);
16 end.
```

```
#include <string.h>
#include <stdio.h>
char s[256],*p;
int nr;
void main()
{
 gets(s);
 nr=0;
 p=strtok(s," ,.;");
 while (p!=NULL) {
  nr++;
  p=strtok(NULL," ,.;");
 }
 printf("%d\n",nr);
}
```

6. Se consideră două cuvinte formate din literele mari şi mici ale alfabetului englez. Verificaţi dacă ele sunt *anagrame*.

Două şiruri de caractere sunt anagrame, dacă unul dintre ele este format din caracterele celuilalt, eventual într-o altă ordine. *Exemplu:* 'are', 'era'.

Soluţie:

Algoritmul propus căută succesiv prima literă a primului cuvânt citit (x), în cel de al doilea (y). În cazul în care este găsită ea va fi ştearsă din ambele cuvinte. Procedeul continuă până când fie litera nu este regăsită, fie când lungimea lui x este 0, caz în care cuvintele sunt anagrame.

O altă metodă ar consta în ordonarea caracterelor ambelor cuvinte şi compararea acestora la final.

```
var x,y:string;
begin
readln(x);
readln(y);
if length(x)=length(y) then
 begin
 while pos(x[1],y)<>0 do
  begin
     delete(y,pos(x[1],y),1);
     delete(x,1,1);
  end;
  if (x='')and(y='')then
       write('Da')
  else
       write('Nu')
 end
else write('Nu')
end.
```

```
#include <iostream.h>
#include <string.h>
char x[21],y[21],*p;
void main() {
  cin>>x>>y;
  if (strlen(x)==strlen(y)) {
   while (strchr(y,x[0])!=NULL
          && x[0]!=0){
    p=strchr(y,x[0]);
    strcpy(p,p+1);
    strcpy(x,x+1);
   }
   if (x[0]==0 && y[0]==0)
       cout<<"Da";
   else cout<<"Nu";
  }
  else cout<<"Nu";
}
```

7. De la tastatură se citeşte un text codificat după regula următoare: în faţa fiecărui caracter este scris numărul de apariţii consecutive ale acestuia. Realizaţi un program care decodifică textul. Numărul de apariţii consecutive ale unui caracter este strict mai mic decât 10.

Exemplu: Pentru codificarea '1c1o1p3i' se va afişa textul 'copiii'

Soluţie

În funcţie de tipul fiecărui caracter (literă/cifră) se decide dacă trebuie realizată afişarea caracterului curent sau actualizarea cifrei care va reprezenta numărul de scrieri ale caracterului următor.

```
var s:string;
    i,j,er,cifra:integer;
begin
 readln(s);
 for i:=1 to length(s) do
 begin
  if s[i] in ['0'..'9'] then
    val(s[i],cifra,er)
  else
    for j:=1 to cifra do
      write(s[i]);
 end;
end.
```

```
#include <stdio.h>
#include <string.h>
char s[256];
int i,j,cifra;
void main() {
 gets(s);
 for (i=0;i<strlen(s);i++)
  if (s[i]>='0'&&s[i]<='9')
   cifra=s[i]-'0';
  else
   for (j=0;j<cifra;j++)
  printf("%c",s[i]);
}
```

8. Se citeşte de la tastatură un şir de caractere. Identificaţi în cadrul acestuia o secvenţă de lungime maximă care poate fi convertită către o variabilă de tip întreg.

Exemplu: Pentru şirul „25AB32042Xs23" se va afişa 32042.

Soluție

Algoritmul propus determină, după o singură parcurgere a caracterelor din șir, care este secvența de lungime maximă ce poate fi convertită către o dată de tip numeric. Se va reține în final, poziția de început (*poz*) a secvenței în cadrul șirului și lungimea acesteia(*max*).

```
1  var s,c:string;
2  i,lung,max,poz,er,x:integer;
3  begin
4   readln(s); lung:=0;
5   s:=s+' ';
6   for i:=1 to length(s) do
7   begin
8     val(s[i],x,er);
9     if (er=0) then inc(lung)
10    else begin
11      if max<lung then begin
12        max:=lung;
13        poz:=i-lung;
14      end;
15      lung:=0;
16    end;
17  end;
18  for i:=poz to poz+max-1 do
19     write(s[i]);
20 end.
```

```
#include <stdio.h>
#include <string.h>
char s[256];
int i,lung,max,poz;
void main() {
 gets(s); strcat(s," ");
 lung=0;
 for (i=0;i<strlen(s);i++)
   if (s[i]>='0'&&s[i]<='9')
     lung++;
   else {
    if (max<lung) {
      max=lung;
      poz=i-lung;
    }
    lung=0;
   }
 for (i=poz;i<poz+max;i++)
    printf("%c",s[i]);
}
```

9. Se dorește ca operația *Find-Replace* să fie executată asupra unui text care nu conține mai mult de 250 de caractere. Această operație constă în înlocuirea tuturor aparițiilor unui subșir *s1* cu un alt subșir *s2*. În cazul de față cele două subșiruri se consideră a fi diferite și de lungimi egale. Creați un program care simulează această operație.

Exemplu: Considerând textul „care caracatita", dacă subșirul „ca" se va înlocui cu „ta" atunci textul afișat va fi „tare taratatita".

Soluție

Programul sursă realizat pentru varianta Pascal folosește apeluri la subprogramele predefinite pentru tipul șir de caractere. Sursa C/C++ construiește un nou șir de caractere *S* în care subșirul *s1* a fost înlocuit cu *s2*.

```
1  var
2      s,s1,s2:string;
3      c:integer;
4
5  begin
6   readln(s);
7   readln(s1);
8   readln(s2);
9   c:=pos(s1,s);
```

```
#include <stdio.h>
#include <string.h>
char S[256],s2[256];
int u,i,p;
char s[56],s1[56];
void main() {
gets(s);
gets(s1);
gets(s2);
```

```
10  while c<>0 do
11  begin
12    delete(s,c,length(s1));
13    insert(s2,s,c);
14    c:=pos(s1,s);
15  end;
16  writeln(s);
17 end.
18
19
20
21
22
```

```
do { char* pt=strstr(s,s1);
 if (!pt){
  for(i=u;i<strlen(s);i++)
  strncat(S,s+i,1); break;
 } p=pt-s;
 for(i=u;i<p;i++)S[i]=s[i];
 for(i=p;i<p+strlen(s1);i++)
  s[i]='!';
 for(i=p;i<p+strlen(s2);i++)
  strncat(S,s2+i-p,1);
 u=p+strlen(s1);} while (1);
 puts(S);
}
```

10. Se consideră un şir de *n* cuvinte. Identificaţi mulţimea cu număr maxim de cuvinte care sunt anagrame între ele două câte două. Se va afişa cardinalul mulţimii şi un element al acesteia, considerat reprezentantul ei.
Exemplu: pentru *n=6* şi cuvintele 'arc', 'rac', 'voi', 'car', 'armata', ' tamara' se va afişa mulţimea: arc 3.

Soluţie
Algoritmul verifică pentru oricare două cuvinte dacă sunt anagrame. Se memorează elementul pentru care s-a determinat numărul maxim de anagrame.

```
1  var a:array[1..100]of
2  string[20];
3      n,i,nr,j,max:integer;
4      cuv,x,y:string;
5  begin
6   readln(n); max:=0;
7   for i:=1 to n do
8              readln(a[i]);
9   for i:=1 to n-1 do begin
10   nr:=1;
11   for j:=i+1 to n do begin
12    y:=a[j]; x:=a[i];
13    if length(x)=length(y) then
14    begin
15     while pos(x[1],y)<>0 do
16      begin
17       delete(y,pos(x[1],y),1);
18       delete(x,1,1);
19      end;
20     if (x='')and(y='')then
21                          inc(nr)
22    end;
23    if max<nr then begin
24      max:=nr; cuv:=a[i];
25    end;
26   end;
27  end;
28  writeln(cuv,' ',max);
29 end.
```

```
#include <iostream.h>
#include <string.h>
char a[101][21],x[21],y[21];
char *p ,cuv[21];
int nr,n,i,j,k,max,poz;
void main() {
 cin>>n; max=0;
 for (i=0;i<n;i++) cin>>a[i];
 for (i=0;i<n-1;i++){
  for (nr=1, j=i+1;j<n;j++){
   strcpy(x,a[i]);
   strcpy(y,a[j]);
   if (strlen(x)==strlen(y)) {
    while(strchr(y,x[0])!=NULL
          && x[0]!=0){
     p=strchr(y,x[0]);
     strcpy(p,p+1);
     strcpy(x,x+1);
    }
    if (x[0]==0 && y[0]==0)
        nr++;
   }
   if (max<nr){
    max=nr;strcpy(cuv,a[i]); }
  }
 }
 cout<<cuv<<" "<<max;
}
```

11. Se consideră un fişier text *in.txt* ce conţine numere întregi dispuse pe mai multe linii. Orice caracter ce nu reprezintă un caracter numeric este considerat separator. Scrieţi un program care creează un fişier *out.txt* ce conţine pe fiecare linie media aritmetică a numerelor situate pe aceeaşi linie în fişierul *in.txt*. Media aritmetică va fi scrisă cu două zecimale. Pe fiecare linie a fişierului de intrare se află maximum 200 de caractere.

Exemplu: in.txt

```
2..3a 403bx
2..2 A,..5
1.92
```

out.txt

```
136.00
3.00
46.50
```

Soluţie:

În problema de faţă, separatorii nu sunt reprezentaţi prin *spaţii albe*. Această situaţie impune folosirea la citire a unei variabile de tip şir de caractere.

Pentru determinarea mediei se va parcurge şirul citit caracter cu caracter, relizându-se conversia secvenţelor de caractere numerice către date de tip numeric(întreg).

```
1  var f,g:text; x,c:string;
2      i,v,s,n,er:integer;
3  begin
4   assign(f,'in.txt'); reset(f);
5   assign(g,'out.txt');rewrite(g);
6   while not eof(f) do begin
7    s:=0; n:=0; c:='';
8    readln(f,x);
9    x:=x+'.';
10   for i:=1 to length(x) do
11    if x[i] in ['0'..'9'] then
12     c:=c+x[i]
13    else
14     if c<>'' then begin
15      val(c,v,er); c:='';
16      s:=s+v;
17      inc(n);
18    end;
19    writeln(g,s/n:0:2);
20  end;
21  close(f); close(g);
22 end.
```

```
#include <stdio.h>
#include <string.h>
FILE *f,*g; char x[256];
int i,n;
float c,s;
void main() {
 f=fopen("in.txt","r");
 g=fopen("out.txt", "w");
 while (!feof(f)) {
  s=0; n=0; c=0;
  if (!fgets(x,256,f)) break;
  strcat(x,".");
  for (i=0;i<strlen(x);i++)
   if (x[i]>='0'&&x[i]<='9')
    c=c*10+x[i]-'0';
   else if (c) {
    s+=c; n++; c=0;
   }
  fprintf(g,"%.2f\n",s/n);
 }
 fclose(f); fclose(g);
}
```

12. Se consideră un cuvânt din maximum 50 de caractere format numai din literele alfabetului englezesc. Se cere inserarea minusculei minime din punct de vedere lexicografic (aparţinând cuvântului), între oricare două litere identice aflate pe poziţii alăturate. La inserare nu se va face distincţie între majuscule şi minuscule. Dacă nu există minuscule în cuvânt atunci acesta nu va suferi nici o modificare.

Exemplu:

Pentru cuvântul "inNorat" minuscula minimă este '*a*' şi se va afişa „inaNorat". Pentru şirul de caractere „boOlutton" minuscula minimă este '*b*' şi se va afişa „bobOlutbton".

Soluție

Operația de inserare se efectuează simultan cu operația de parcurgere a șirului de caractere. La inserarea unui nou caracter între două litere identice, indicele curent se incrementează cu două poziții.

```
var
  s:string; m:char; i:byte;
begin
 readln(s);
 m:=chr(127);
 for i:=1 to length(s)do
  if (s[i]<m)
     and(s[i]<>upcase(s[i]))
  then m:=s[i];
 i:=1;
 if upcase(m)<>m then begin
  while i<length(s)do
   if upcase(s[i])=
      upcase(s[i+1])
   then begin
    insert(m,s,i+1);
    inc(i,2);
   end
   else
    inc(i);
   end;
   writeln(s);
end.
```

```
#include <string.h>
#include <stdio.h>
char s[256],m,c1,c2; int i;
void main() {
 gets(s); m=127;
 for(i=0;i<strlen(s);i++){
  c1=s[i]>='A'&&s[i]<='Z'?
     s[i]+'a'-'A':s[i];
  if (m>c1) m=c1;
 }
 for (i=0;i+1<strlen(s);i++){
  c1=s[i]>='A'&&s[i]<='Z'?
     s[i]+'a'-'Z':s[i];
  c2=s[i+1]>='A'&&s[i+1]<='Z'?
     s[i+1]+'a'-'A':s[i+1];
  if (c1==c2) {
   memmove(s+i+1,s+i,
           strlen(s)-i);
   s[++i]=m;
  }
 }
 puts(s);
}
```

13. Fișierul text „*P.txt*" conține pe mai multe linii un algoritm reprezentat în pseudocod. Știm că singurele cuvinte cheie ce se regăsesc în fișier sunt: *intreg*, *daca*, *atunci*, *altfel*, *citeste*, *scrie*, *stop*. Să se determine numărul de variabile folosite în algoritm și identificatorii acestora. Toate caracterele care intervin în fișier au coduri ASCII asociate. Liniile din fișier se termină cu caracterul punct și virgulă(;).

Exemplu: Pentru secvența „daca a>b atunci xx=c altfel d=a+b stop;" se va afișa:

5

a b xx c d

Soluție

Tabloul unidimensional *v* va memora toate variabilele identificate. Fișierul va fi parcurs caracter cu caracter, formându-se de fiecare dată o secvență ce poate reprezenta un cuvânt cheie sau identificatorul unei variabile. Dacă acesta nu reprezintă un cuvânt cheie, atunci este salvat în vectorul *v*, numai în cazul în care nu a fost deja memorat.

```
type
sir=array[1..7] of string[10];
const a : sir =('intreg',
'citeste','scrie', 'daca',
'atunci','altfel','stop');
var f:text; i,n,m:integer;
```

```
#include <string.h>
#include <stdio.h>
const char a[8][11]={
 "intreg","citeste","scrie",
 "daca","atunci","altfel",
 "stop"};
```

```
7  s:string; x:char; ok:boolean;
8  v:array[1..50]of string[10];
9  begin
10 assign(f,'p.txt'); reset(f);
11 m:=0;
12 while not eof(f) do begin
13  s:='';
14  while not eoln(f) do begin
15   read(f,x);
16   if x in ['a'..'z'] then
17        s:=s+x
18   else if s<>'' then begin
19    ok:=true;
20    for i:=1 to 7 do
21     if a[i]=s then ok:=false;
22    for i:=1 to m do
23      if s=v[i] then ok:=false;
24    if ok then begin
25         inc(m); v[m]:=s;
26    end; s:='';
27  end; end; readln(f); end;
28 writeln(m);
29 for i:=1 to m do
30      write(v[i],' ');
31 end.
```

```
int i,j,m,ok; FILE *f;
char s[256],x[256],v[51][11];
void main() {
 f=fopen("p.txt","r");
 for (m=0;!feof(f);) {
  fgets(x,256,f);
  strcat(x," ");
  for(i=0;i<strlen(x);i++)
   if(x[i]>='a'&&x[i]<='z')
    strncat(s,x+i,1);
   else if (strlen(s)) {
    ok=1;
    for(j=0;j<7;j++)
     if(!strcmp(a[j],s))ok=0;
    for(j=0;j<m;j++)
     if(!strcmp(v[j],s))ok=0;
    if (ok) strcpy(v[m++],s);
    memset(s,0,sizeof(s));
   }
 }
 printf("%d\n",m);
 for(i=0;i<m;i++)
  printf("%s ",v[i]);
}
```

14. Se consideră un şir de *n* cuvinte. Se doreşte afişarea lor pe verticală, respectând următoarele cerinţe:

- pe fiecare coloană se află în ordine câte un cuvânt, de la primul până la ultimul;
- pe ultima linie se află primele litere ale fiecărui cuvânt;
- pe prima linie se află ultimele litere ale celor mai lungi cuvinte.

Exemplu: Pentru şirul de 4 cuvinte: 'eu', 'masa', 'noi', 'vine' se va afişa:

```
  a   e
  s i n
u a o i
e m n v
```

Soluţie

Se foloseşte ca auxiliar un vector *h* în care este memorată lungimea fiecărui cuvânt. Numărul de linii pe care se va face afişarea este egală cu elementul maxim din *h*. Pentru fiecare cuvânt, în cadrul unei linii, se poate afişa fie caracterul spaţiu, dacă lungimea acestuia este mai mică decât valoarea variabilei *mx*(care contorizează poziţia literei ce urmează a fi afişată), fie litera de pe poziţia *mx*.

```
1  type
2  sir=array[1..99] of string;
3  var a:sir;
4  l,i,n,mx:integer;
```

```
#include <string.h>
#include <stdio.h>
int n,i,max,h[100];
char a[100][256];
```

```
5  h:array[1..99]of byte;
6  begin
7   readln(n);mx:=0;
8   for i:=1 to n do begin
9    readln(a[i]);
10   h[i]:=length(a[i]);
11   if h[i]>mx then mx:=h[i]
12  end;
13  for l:=1 to mx do begin
14   for i:=1 to n do
15    if h[i]=mx then begin
16     write(a[i][h[i]]);
17     dec(h[i])
18    end
19    else write(' ');
20   dec(mx);
21   writeln;
22  end;
23 end.
```

```
void main() {
 scanf("%d",&n); max=0;
 for(i=0;i<n;i++){
  scanf("%s",a[i]);
  h[i]=strlen(a[i])-1;
  if (max<h[i]) max=h[i];
 }
 for(;max>=0;max--){
  for (i=0;i<n;i++)
   if (h[i]==max){
    printf("%c",a[i][h[i]]);
    h[i]--;
   }
   else printf(" ");
  printf("\n");
 }
}
```

15. Se citeşte de la tastatură un şir de maximum 255 litere mici ale alfabetului englez. Să se realizeze un program care identifică cea mai lungă secvenţă de caractere care se repetă de minimum 2 ori în cadrul şirului.
Exemplu: Pentru şirul de caractere:" masectrsecsacrrrr" se va afişa „sec".

Soluţie
Algoritmul parcurge fiecare poziţie din şir şi identifică lungimea maximă a secvenţei care poate începe cu caracterul respectiv. Variabila *x* reţine această secvenţă.

```
1  var l,max,i:integer;
2  s,r,x,y:string;
3  begin
4  readln(s); max:=0;
5  for i:=1 to length(s)-1 do
6  begin
7   l:=0; x:='';
8   repeat
9    inc(l); y:=s;
10   x:=x+s[i+l];
11   delete(y,i,length(x));
12  until (pos(x,y)=0)or
13        (l+i>length(s));
14  if length(x)-1>max then begin
15   max:=length(x)-1;
16   r:=copy(x,1,max);
17  end;
18 end;
19 writeln(r);
20 end.
21
```

```
#include <string.h>
#include <stdio.h>
int l,max,i; char
s[256],r[256],x[256],y[256];
void main() {
 gets(s); max=0;
 for(i=0;i+1<strlen(s);i++){
  l=0; memset(x,0,sizeof(x));
  do {
   strncat(x,s+i+1,1);l++;
   strcpy(y,s);
   strcpy(y+i,y+i+strlen(x));
  } while (i+1<strlen(s) &&
    strstr(y,x));
  if (max<strlen(x)-1){
   max=strlen(x)-1;
   strncpy(r,x,max);
  }
 }
 puts(r);
}
```