- **declarate in string.h //cstring**
  - **int strcmp(char *s1,char *s2);** diferenţă lexicografică returneaza

<0, daca s1 < s2 (se pare că -1 chiar dacă diferența de cod ASCII dintre caracterele care nu se potrivesc este <=-1)

= 0, daca s1 = s2

>0, daca s1 > s2 (se pare că 1 chiar dacă diferența de cod ASCII dintre caracterele care nu se potrivesc este >=1)

)

(difereţa de cod ASCII între primele două caractere diferite situate pe aceeaşi poziţie în s1 şi s2)

  - **int strncmp(char *s1,char *s2,int n);**

comparare a doua siruri pe lungimea n

  - **int strncmpi(char *s1,char *s2,int n);**

compară cele două şiruri la fel ca şi funcţia strncmp dar fără să facă diferenţa între literele mari şi mici

  - **char* stricmp(char *s1,char *s2);**

compară cele două şiruri la fel ca şi funcţia strcmp dar fără să facă diferenţa între literele mari şi mici

  - **char* strcpy(char *d,char *s);**

copiaza șirul sursă s în șirul destinatie d; returnează adresa șirului destinație; valoarea anterioară a şirului destinaţie se pierde

ex.

char s[]=”bacalaureat”;

cout<<strcpy(s,”info”);

va afișa șirul *info*

Copieri în cadrul aceluiași șir

Varianta **strcpy(s+i,s+k) unde i<k şterge** dintr-un şir s, prin copiere peste, caracterele aflate începând cu poziţia i şi până la cel din poziţia k-1 inclusiv.

strcpy(s+i,s+i+1) șterge prin copiere peste caracterul din poziția i.

În funcție de compilator aceste copieri în cadrul aceluiași șir pot să nu fie recunoscute, de aceea vom folosi un șir auxiliar pentru transfer

char s[25],t[25];

int i,k;

În loc de strcpy(s+i,s+k) cu i<k vom scrie

strcpy(t,s+k); strcpy(s+i,t);

Observație : nu vom face niciodată copiere în cadrul aceluiași șir spre dreapta deoarece acest lucru generează eroare(deci e greși ~~strcpy(s+4,s+2)~~)

Pentru aceasta vom folosi în șir de transfer auxiliar

char s[25],t[25];

int i,k;

În loc de strcpy(s+4,s+2) vom scrie

strcpy(t,s+2); strcpy(s+4,t);

  - **char* strncpy(char *d,char *s,int n);**

copiaza maxim n caractere de la sursa la destinatie; returneaza adresa sirului destinatie ; nu copiază obligatoriu şi marcajul de sfârşit de şir din şirul s dacă numărul n nu îl ia şi pe acesta în calcul, deci atenţie acesta ar putea fi adăugat prin program d[n]='\0'

  - **int strlen(char *s);**

returnează lungimea șirului fără a număra caracterul care marchează finalul șirului adică ’\0’(sau NULL).

Observație:

Parcurgerea unui șir de caractere de la primul caracter către ultimul

```
char s[25];
cin.get(s);
for(int i=0;i<strlen(s);i++)
        prelucreaza s[i];
```

Sau fără funcția strlen:

```
char s[25];
cin.get(s);
for(int i=0;s[i]!=NULL;i++)
        prelucreaza s[i];
```

Parcurgerea unui șir de caractere de la ultimul caracter la primul

```
char s[25];
cin.get(s);
for(int i=strlen(s)-1;i>=0;i--)
        prelucreaza s[i];
```

- **char* strcat(char *d,char *s);**

concatenează cele două șiruri adăugând la sfârșitul lui d șirul s și returnează adresa de început a șirului șirului rezultat d

ex. char s[]=”info”, d[]=”bac”;

cout<<strcat(d,s); //va afișa șirul *bacinfo*

dar și

```
char s[]=”info”, d[]=”bac”;
strcat(d,s);
cout<<d; // va afișa tot șirul bacinfo deoarece ]n ambele situa’ii se face alipirea
```

- **char *strncat(char *d,char *s,int n);**

lipește maxim n caractere din s in d nu neaparat și \0(NULL), în rest la fel ca la strcat

- **char* strchr(char s,char c);**

returneaza adresa primei aparitii a caracterului c in sirul s(!=NULL) dac[c apare in s, respectiv NULL daca c nu e in s.

exemplu verificare dacă s[i] este vocală literă mică

if(strchr("aeiou",s[i])!=0)) sau if(strchr(”aeiou”,s[i])!=NULL) sau if(strchr(”aeiou”,s[i]))

dacă dorim să verificăm dacă este consoană iar șirul poate conține și alte caractere nu doar litere mici trebuie să verificăm în plus dacă e literă mica

if((s[i]>=’a’ && s[i]<=’z’)&&strchr("aeiou",s[i])==0))
sau if((s[i]>=’a’ && s[i]<=’z’)&&strchr(”aeiou”,s[i])==NULL)

pozitia primei apariții a unui caracter într-un șir: se calculează distanța dintre pointerul furnizat ca rezultat de funcția strchr față de începutul șirului în care se face căutarea

int p=strchr(s,c)-s; va returna poziția primei apariții a caracterului c în șirul s

```
char s[]="macara";
```

cout<<strchr(s,'a')-s; afisează 1 // sau int p= strchr(s,'a')-s; cout<<p;

în timp ce **cout<<strchr(“macara”,’a’)** va afișa **acara** //deoarece afișează de la adresa furnizată ca rezultat și până la sfârșitul șirului

- **char* strrchr(char s,char c);**

returnează adresa ultimei apariții a caracterului c în șirul s, respectiv NULL dacă c nu e în s, în rest analog ca la strchr

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

returnează adresa de început a primei apariții a lui s2 în s1, dacă șirul s2 apare în șirul s1, respectiv NULL dacă s2 nu apare în s1.
pozitia primei aparitii a unui șir în alt sir: analog strchr, calculăm diferența dintre pointerul furnizat de apelul funcției și pointerul de început a lui s1: int poz=strstr(s1,s2)-s1;
char s[]="macara";
cout<<strstr(s,"ar")-s; afisează 3 // sau int poz= strstr(s,"ar")-s; cout<<poz;
în timp ce cout<<strstr("macara","ar") va afișa **ara**

- **char* strset (char *s,char c);**

şirul s este parcurs începând de la primul caracter, până la sfârşitul lui, fiecare caracter fiind înlocuit cu caracterul c, mai puţin caracterul NULL, funcția returnând pointerul de început(adresa de început) a șirului s
exemplu:
char s[]="info";
cout<<strset(s,'*'); va afișa ****

- **char* strnset(char *s,char c,int n)**

la fel ca şi funcţia strset doar că sunt parcurse primele n caractere

- **char* strlwr(char* s);**

transformă toate literele mari din şirul s în litere mici, restul rămân nemodificate și returnează adresa de început a șirului s
exemplu: cout<<strlwr("BAC09"); //va afișa bac09

- **char* strupr(char* s);**

transformă toate literele micid in şirul s în litere mari, restul rămân nemodificate și returnează adresa de început a șirului s
exemplu: cout<<strupr("bac09"); //va afișa BAC09

- **int strspn(char* s1,char* s2);**

furnizează ca rezultat numărul de caractere consecutive din şirul s1 începând cu primul caracter care se găsesc printre caracterele din s2

- **int strcspn(char* s1,char* s2);**

furnizează ca rezultat numărul de caractere consecutive din şirul s1 începând cu primul caracter care nu se găsesc printre caracterele din s2

- **char* strpbrk(char* s1,char*s2);**

furnizează ca rezultat un pointer către primul caracter din şirul s1 care se găseşte în şirul s2

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

şirul s2 este un şir de caractere ce pot f folosite ca separatori, iar şirul s1 este format din mai multe entităţi separate prin unul dintre separatorii din s2. Funcţia înlocuieşte separatorii prin caracterul NULL şi furnizează ca rezultat un pointer către primul caracter al primei entităţi. Pentru a găsi următoarea entitate vor apela strtok(NULL,s2)
Cel mai adesea se foloseşte pentru a extrage cuvintele dintr-un text atunci când se cunosc caracterele care joacă rol de separator de cuvinte. Şirul iniţial se distruge prin adăugarea de valori NULL aşa că e bine să prelucrăm o copie a acestuia

```
char s[101],aux[101],cuv[26],sep[]=",.!? ";
char *p;
cin.get(s,100);
strcpy(aux,s); //daca nu dorim sa distrugem sirul s, altfel lucram direct cu s dar se distruge
```

```
p=strtok(aux,sep);
while(p)
 {strcpy(cuv,p);
  //prelucreaza cuv
p=strtok(NULL,sep);
}
```

Dacă nu dorim să distrugem șirul initial iar cuvintele sunt separate doar prin spații:

```
int i=0;
while(i<=strlen(s))
        { while(i<=strlen(s) &&s[i]==' ')
                 i++;
          j=i+1;
          while(j<=strlen(s) &&s[i]!=' ')
                j++;
          // se  analizează cuvântul de lungime j-i format din s[i],…s[j-1]
          i=j;
         }
```

Dacă nu sunt separate prin spații vom scrie strchr(sep,s[i])!=NULL în loc de s[i]==' ' și în loc de strchr(sep,s[i])==NULL în loc de s[i]!=' '

- **declarate în ctype.h**

    - **char tolower(char c)**

returnează codul ASCII a caracterului c transformat din literă mare în literă mică, ex.char c='A'; dacă c nu e literă returnează codul ASCII corespunzător al lui c

```
char c='A';
cout<<c<<'  '<<tolower(c);
```

va afișa A 97

iar

```
char c='A';
char b= tolower(c);
    cout<<c<<' '<<b;
```

va afișa: A a

iar cout<<tolower('3'); va afișa 51 adică codul ASCII al lui 3 deoarece acesta nu este literă

s[i]=s[i]+32; sau s[i]=s[i]+('a'-'A') sau s[i]=tolower(s[i]) va trasnforma pe s[i] din literă mare în mică

    - **char toupper(char c)**

returnează codul ASCII al caracterului c transformat din literă mică în literă mare sau codul ASCII al lui c dacă acesta nu este literă

```
char c='A';
cout<<c<<'  '<<toupper(c);
```

va afişa a 65

iar

```
char c='A';
char b= toupper(c);
cout<<c<<' '<<b;
```

va afişa: A a

iar cout<<toupper('3'); *va afișa 51* adică codul ASCII al lui 3 deoarece acesta nu este literă

s[i]=s[i]-32; sau s[i]=s[i]-('a'-'A') sau s[i]=toupper(s[i]) va trasnforma pe s[i] din literă mică în mare

Pentru a transforma un caracter cifră în cifra numerică corespunzătoare vom scrie c=s[i]-48; sau c=s[i]-'0'; (de ex. '7'-'0' va avea valoarea numerică 7)

Funcţii de conversie

- **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.
- **long double _atold(sir);** – converteste un sir catre tipul long double. Daca aceasta conversie esueaza, valoarea intoarsa este 0.
- **int atoi(sir);** – converteste un sir catre tipul int. Daca aceasta conversie esueaza (se intalneste un caracter nenumeric), valoarea intoarsa este 0.
- **long atol(sir);** – converteste un sir catre tipul long. Daca aceasta conversie esueaza (se intalneste un caracter nenumeric), valoarea intoarsa este 0.
- **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 -.
- **ltoa(long valoare,sir,int baza**); – converteste o valoare de tip long int in sir, care este memorat in variabila sir.
- **ultoa(unsigned long valoare,sir,int baza);** – converteste o valoare de tip unsigned long in sir, care este memorat in variabila sir.

```
#include<iostream.h>
#include<string.h>
#include<stdlib.h>
void main()
{char sir[100];  int n;
cout<<"dati sirul ";cin.get(sir,100);
n=atoi(sir);
cout<<"numarul dupa conversie "<<n<<"\n";
itoa(n,sir,10);
cout<<n<<" in baza 10 "<<sir<<"\n";
itoa(n,sir,8);
cout<<n<<" in baza 8 "<<sir<<"\n";
itoa(n,sir,16);
cout<<n<<" in baza 16 "<<sir<<"\n";
itoa(n,sir,2);
cout<<n<<" in baza 2 "<<sir<<"\n";
}
```

Vectorul de cuvinte:
Se declară prin char v[nr_cuv][lung max cuvant];
Exemplu: 1. Citirea și parcurgerea pentru prelucare a vectorului de cuvinte

```
char v[100][25]; //100 de cuvinte maxim, fiecare putând memora maxim 24 de litere
int n; //nr de cuvinte din vector
cin>>n;
for(int i=0;i<n;i++)
    cin.get(v[i]); //fiecare v[i] reprezintă un cuvânt
//afișare
for(int i=0;i<n;i++)
    cout<<v[i]<<' ';
//afișarea cuvintelor care încep cu litera 'a'
for(int i=0;i<n;i++)
  if(v[i][0]=='a')
    cout<<v[i]<<' ';
```

```
// afișarea cuvintelor care încep și se termină cu aceeași literă
for(int i=0;i<n;i++)
  if(v[i][0]==v[i][strlen(v[i]-1])  //deci prima literă este v[i][0], iar ultima v[i][strlen(v[i]-1)
    cout<<v[i]<<' ';
//ordonarea lexicografică a cuvintelor
for(int i=0;i<n-1;i++)
    for(int j=i+1;j<n;j++)
       if(strcmp(v[i],v[j])>0)
          {char aux[25];
           strcpy(aux,v[i]);
           strcpy(v[i],v[j]);
           strcpy(v[j],aux);
          }
for(int i=0;i<n;i++)
    cout<<v[i]<<' ';

//ordonarea cuvintelor descrescător după lungime
for(int i=0;i<n-1;i++)
    for(int j=i+1;j<n;j++)
       if(strlen(v[i])<strlen(v[j]))
          {char aux[25];
           strcpy(aux,v[i]);
           strcpy(v[i],v[j]);
           strcpy(v[j],aux);
          }
for(int i=0;i<n;i++)
    cout<<v[i]<<' ';
```

Observație: dacă dorim să extragem cuvintele dintr-un text în care cuvintele sunt separate prin diferiți separatori

```
char s[101],aux[101],sep[]=",.!?";
char *p, v[100][25];
int n=0;
cin.get(s,100);
strcpy(aux,s);//daca nu dorim sa distrugem sirul s
p=strtok(aux,sep);
while(p)
 {strcpy(v[n],p);
  n++;
  p=strtok(NULL,sep);
}
for(int i=0;i<n;i++)
    prelucreaza v[i];
```

Dacă nu dorim să distrugem șirul initial iar cuvintele sunt separate doar prin spații:

```
int i=0;
int n=0ș
while(i<=strlen(s))
        { while(i<=strlen(s)&&s[i]==' ')
                i++;
          j=i+1;
          while(j<=strlen(s)&&s[i]!=' ')
```

```
            j++;
        n++;
        for(int k=i;k<j;k++)
            v[n][[k-i]=s[i];
        v[n][j-i]='\0';
        i=j;
    }
```

Prelucrare vector