Funcţii care lucrează cu şiruri de caractere

1. Incluse in biblioteca <string>

unsigned int strlen(char *sir);
Efect: returnează numărul de caractere al unui şir de caractere, fără a lua în considerare caracterul nul de la sfârşitul şirului
Exemplu:

```
char a[100]=”mama”;
cout<<“sirul are ”<<strlen(a)<<“ caractere”;  //va afisa 4
```

char *strcpy(char *dest,char *sursa);
Efect: copiază şirul de la adresa sursa la adresa destinaţie. Copierea se termină la întâlnirea caracterului nul. Funcţia returnează adresa şirului destinaţie. Simulează operaţia de atribuire a=b.
Exemplu:

```
char a[100]=“crocodil”,b[100]=“hipopotam”;
strcpy(a,b);
cout<<“sirul a: ”<<a<<endl; //hipopotam
cout<<“sirul b: ”<<b<<endl; //hipopotam
```

char *strncpy(char *dest,char *sursa,unsigned int n);
Efect: copiază primii n octeţi din şirul de la adresa sursă la adresa destinaţie, fără a adăuga caracterul nul. Funcţia returnează adresa şirului destinaţie. Sirul sursă rămâne nemodificat.
Exemplu:

```
char a[100]=“crocodil”,b[100]=“hipopotam”;
strncpy(a,b,4);
cout<<“sirul a: ”<<a<<endl; //hipo
cout<<“sirul b: ”<<b<<endl; //hipopotam
```

char *strcat(char *dest,char *sursa);
Efect: adaugă şirului de la adresa destinaţie, înaintea caracterului nul şirul de la adresa sursă. Şirul de la adresa sursă rămâne nemodificat. Operaţia se numeşte concatenare. La adresa destinaţie vom avea şirul destinaţie urmat de şirul sursă. Şirul destinaţie are lungimea egală cu suma lungimilor şirurilor.
Exemplu:

```
char a[100]=“mama”,b[100]=“merge”;
strcat(a,b);
cout<<“sirul a: ”<<a<<endl; //mamamerge
cout<<“sirul b: ”<<b<<endl; //merge
```

char *strncat(char *dest,char *sursa, unsigned int n);
Efect: adaugă şirului de la adresa destinaţie, înaintea caracterului nul primii n octeţi ai şirul de la adresa sursă. Şirul de la adresa sursă rămâne nemodificat. Funcţia returnează adresa de început a şirului destinaţie.
Exemplu:

```
char a[100]=“mama ”,b[100]=“merge”;
strncat(a,b,3);
cout<<“sirul a: ”<<a<<endl; //mama mer
```

cout<<“sirul b: ”<<b<<endl; //merge

char *strchr(char *sir,int car);
Efect: caută de la stânga la dreapta, caracterul car în şirul de caractere sir. Dacă este găsit, funcţia întoarce adresa subşirului care începe cu prima apariţie a caracterului citit şi se termină cu carcterul nul. Dacă nu este găsit intoarce o expresie de tip char* cu valoarea 0.
Exemplu:

```
char a[100]=“crocodil”;
cout<<strchr(a,’o’); //ocodil
```

char *strrchr(char *sir,int car);
Efect: caută de la dreapta la stânga, caracterul car în şirul de caractere sir. Dacă este găsit, funcţia întoarce adresa subşirului care începe cu ultima apariţie a caracterului citit şi se termină cu carcterul nul. Dacă nu este găsit intoarce o expresie de tip char* cu valoarea 0.
Exemplu:

```
char a[100]=“crocodil”;
cout<<strrchr(a,’o’); //odil
```

char *strstr(char *sir1,char *sir2);
Efect: identifică dacă şirul sir2 este subşir(caractere succesive) al şirului sir1. dacă este găsit, funcţia returnează adresa sa de început în cadrul şirului s1, altfel returnează 0. Căutarea se face de la stânga la dreapta. Dacă sir2 apare de mai multe ori, returnează adresa primei sale apariţii.
Exemplu:

```
char a[100]=“azi ele fac cafele”,b[20]= “ele”;
cout<<strstr(a,b); //ele fac cafele
```

char *strtok(char *sir1,char *sir2);
Efect: separă şirul sir1 în entităţi delimitate de unul sau mai multe carctere din şirul sir2 (acestea având rol de separatori). Apelul funcţiei se face prima dată sub forma strtok(sir1,sir2) - funcţia întoarce adresa primului caracter al primei entităţi - şi a doua oară sub forma strtok(NULL,sir2) şi funcţia întoarce adresa primului caracter al următoarei entităţi şi după el este adăugat caracterul nul. Când şirul iniţial nu mai conţine entităţi, intoarce adresa nulă.
Exemplu:

```
char a[100],sep[]=“, ;.?!”,*p;
cin.get(a,100);
p=strtok(a,sep);
while (p)
{ cout<<p<<endl;
  p=strtok(NULL,sep);
```

int strcmp(char *sir1,char *sir2);
Efect: compară cele două şiruri de caractere. Valoarea returnată este:
<0 dacă sir1<sir2
=0 dacă sir1=sir2
>0 dacă sir1>sir2
Funcţia face distincţie între literele mari si literele mici. Compararea şirurilor se realizează comparând de la stânga la dreapta caracter cu caracter. Un şir este mai mic dacât altul dacă figurează în dicţionar inaintea lui.

Exemplu:
```
char a[20]="adriana",b[20]= "ana", c[20]= "Ana";
cout<<strcmp(a,b); //<0 deoarece 'a'='a' si 'd'<'n' => "adriana"< "ana"
cout<<strcmp(a,c); //>0 deoarece 'a'>'A'
cout<<strcmp(b,c); //>0 deoarece 'a'>'A'
```

int stricmp(char *sir1,char *sir2);
Efect: are acelaşi efect ca şi strcmp dar nu face diferenţă între literele mari şi literele mici.
Exemplu:
```
char b[20]= "ana", c[20]= "Ana";
cout<<stricmp(b,c); //==0
```

int strncmp(char *sir1,char *sir2,int n);
Efect: are acelaşi efect ca şi strcmp dar compara doar primele n caractere din cele doua siruri
Exemplu:
```
char b[20]= "adriana", c[20]= "adina";
cout<<strncmp(b,c,2); //==0
```

int strncmpi(char *sir1,char *sir2,int n);
Efect: are acelaşi efect ca şi strncmp dar nu face diferenţă între literele mari şi literele mici.
Exemplu:
```
char b[20]= "adriana", c[20]= "ADina";
cout<<strncmpi(b,c,2); //==0
```

char *strupr(char *s)
Efect: transformă un şir de caractere din litere mici în litere mari. Restul caracterelor rămân nemodificate.
Exemplu:
```
char a[100]="1 crocodil";
cout<<strupr(a); //1 CROCODIL
```

char *strlwr(char *s)
Efect: transformă un şir de caractere din litere mari în litere mici. Restul caracterelor rămân nemodificate.
Exemplu:
```
char a[100]="1 CROCODIL";
cout<<strupr(a); //1 crocodil
```

2. Incluse in biblioteca <stdlib.h>

int atoi(char *s)
Efect: transformă un şir de carcatere într-un întreg (int).
Exemplu:

```
int n;
char *s=”1234.56”;
n=atoi(s);
cout<<n; // va afisa 1234
```

long atol(char *s)
Efect: transformă un şir de carcatere într-un întreg (long).

double atof(char *s)
Efect: transformă un şir de carcatere într-un număr real.
Exemplu:

```
float n;
char *s=”-4521234.56”;
n=atof(s);
cout<<n; // va afisa  -4521234.56
```

char *itoa(int val, char *sir, int baza)
Efect: transformă un numar întreg (int) într-un şir de caractere. Baza reprezintă baza in care este scris noul număr.
Exemplu:

```
int n=12345;
char s[20];
itoa(n,s,10);
cout<<s // va afisa  sirul “12345”
```

char *ltoa(long val, char *sir, int baza)
Efect: transformă un numar întreg (long) într-un şir de caractere.

char *ultoa(unsigned long val, char *sir, int baza)
Efect: transformă un numar întreg (unsigned long) într-un şir de caractere.

3. Funcţii care lucrează cu caractere

Sunt incluse in biblioteca <ctype.h>. Testează dacă un cracter primit ca parametru îndeplineşte o condiţie. Returnează 0 dacă acel caracter nu indeplineşte condiţia şi valoare diferită de 0 dacă o îndeplineşte.

int isalnum(int c);
Efect:testează dacă un caracter este literă sau cifră
Exemplu:
```
char s='y';
cout<<isalnum(s); // va afisa o valore diferita de 0
```

int isalpha(int c);
Efect: testează dacă un caracter este literă

int isdigit(int c);
Efect: testează dacă un caracter este cifră
Exemplu:
```
char s='y';
cout<<isdigit(s); // va afisa 0
```

int islower(int c);
Efect: testează dacă un caracter este literă mică

int isupper(int c);
Efect: testează dacă un caracter este literă mare

int isspace(int c);
Efect: testează dacă un caracter este spaţiu

int isxdigit(int c);
Efect: testează dacă un caracter este cifră în baza 16
Exemplu:
```
char s='d';
cout<<isxdigit(s); // va afisa o valore diferita de 0, deoarece d este o cifra in baza 16
```

int toupper(int c);
Efect: transformă ***un caracter*** care este litera mică în literă mare
Exemplu:
```
char s='y';
cout<<toupper(s); // va afisa 'Y'
```

int tolower(int c);
Efect: transformă ***un caracter*** care este litera mare în literă mică
Exemplu:
```
char s='Y';
cout<<tolower(s); // va afisa 'y'
```