



- ```
char s[]="Sir de caractere";
```

|                            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Index                      | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Cod ASCII<br>(hexazecimal) | 53 | 69 | 72 | 20 | 64 | 65 | 20 | 63 | 61 | 72 | 61 | 63 | 74 | 65 | 72 | 65 | 00 |



- ```
printf ("%s\n", s);
```

```
puts (s) ;
```



- 4



- 5



- 6



- Câteva prototipuri din biblioteca **ctype.h**
 - În limbajul C: **false** este 0, **true** este orice diferit de 0

Prototip	Descriere
int isdigit(int c)	Returnează true dacă c este cifră și false altfel
int isalpha(int c)	Returnează true dacă c este literă și false altfel
int islower(int c)	Returnează true dacă c este literă mică și false altfel
int isupper(int c)	Returnează true dacă c este literă mare și false altfel
int tolower(int c)	Dacă c este literă mare, tolower returnează c ca și literă mică. Altfel, tolower returnează argumentul nemodificat
int toupper(int c)	Dacă c este literă mică, toupper returnează c ca și literă mare. Altfel, toupper returnează argumentul nemodificat
int isspace(int c)	Returnează true dacă c este un caracter <i>white-space</i> — <i>newline</i> (<code>"\n"</code>), <i>space</i> (<code>" "</code>), <i>form feed</i> (<code>"\f"</code>), <i>carriage return</i> (<code>"\r"</code>), <i>horizontal tab</i> (<code>"\t"</code>), <i>vertical tab</i> (<code>"\v"</code>) — și false altfel



Rezultate afișate:

3 Pere



- | Prototip | Descriere |
|--|---|
| size_t strlen(const char *s) | Returnează numărul de caractere conținut de <i>string</i> -ul s aflate înaintea caracterului terminal '\0' |
| char *strdup(const char *s) | Copiază <i>string</i> -ul s într-un nou string alocat dinamic. Dacă alocarea dinamică eșuează, funcția returnează pointer la NULL , altfel pointer la primul caracter din noul <i>string</i> duplicat generat |
| char *strcpy(char *s1, const char *s2) | Copiază <i>string</i> -ul s2 în șirul de caractere s1 . Valoarea lui s1 este returnată |
| char *strncpy(char *s1,
 const char *s2,
 size_t n) | Copiază cel mult n caractere din <i>string</i> -ul s2 în șirul de caractere s1 . Valoarea lui s1 este returnată |



- Câteva prototipuri din biblioteca **string.h**

Prototip	Descriere
char *strcat(char *s1, const char *s2)	Concatenează <i>string</i> -ul s2 la <i>string</i> -ul s1 . Primul caracter din s2 suprascrie caracterul terminal '\0' din s1 . Valoarea lui s1 este returnată
char *strncat(char *s1, const char *s2, size_t n)	Concatenează cel mult n caractere din <i>string</i> -ul s2 la <i>string</i> -ul s1 . Primul caracter din s2 suprascrie caracterul terminal '\0' din s1 . Valoarea lui s1 este returnată
int strcmp(const char *s1, const char *s2)	Compară conținutul <i>string</i> -urilor s1 și s2 la nivelul caracterelor componente. Returnează un număr negativ dacă s1 este mai mic decât s2 (ordinea este dată de codurile ASCII), zero dacă sunt identice și un număr pozitiv dacă s1 este mai mare decât s2
int strncmp(const char *s1, const char *s2, size_t n)	Compară conținutul primelor n caractere din <i>string</i> -urile s1 și s2 . Funcția se comportă asemenea funcției strcmp



- Câteva prototipuri din biblioteca **string.h**

Prototip	Descriere
int stricmp(const char *s1, const char *s2)	Compară conținutul <i>string</i> -urilor s1 și s2 la nivelul caracterelor componente, ignorând tipul literelor (mici/mari) . Returnează un număr negativ dacă s1 este mai mic decât s2 (ordinea este dată de codurile ASCII), zero dacă sunt identice și un număr pozitiv dacă s1 este mai mare decât s2
int strnicmp(const char *s1, const char *s2, size_t n)	Compară conținutul primelor n caractere din <i>string</i> -urile s1 și s2 ignorând tipul literelor (mici/mari) . Funcția se comportă asemenea funcției stricmp
char *strchr(const char *s, int c)	Găsește prima apariție a caracterului c în <i>string</i> -ul s . Returnează pointer către acel caracter în <i>string</i> dacă a fost găsit și NULL în caz contrar.



Rezultate afișate:

Emil



Funcții pentru manipularea *string*-urilor – cont. exemplu

```
strncpy(u+4,t+1,3);
puts(u);
u[7]='\0';
puts(u);
strcat(u,s+2);
puts(u);
strncat(u,s,3);
puts(u);
printf("%d %d\n",
        strcmp(s,"Ali"),
        strncmp(s,"Ali",3));
printf("%d %d\n",
        stricmp(s,"ALONA"),
        strnicmp(s,"ALI",3));
char *fs=strchr("Liliana",'a');
puts(fs);
```

Rezultate afișate:

Alinmil.■ :A¥t=E¥t→@

Alinmil

Alinmilina

AlinmilinaAli

1 0

-1 0

ana



Funcție pentru delimitarea unor sub-șiruri dintr-un șir de caractere

- Funcția **strtok**

```
char *strtok(char *sir, const char *delimitatori)
```

- Delimitează string-ul **sir** în sub-*string*-uri componente delimitate de unul sau mai multi **delimitatori**
- *String*-ul inițial se trimite doar la primul apel al funcției, obținându-se primul sub-*string*
- La următoarele apeluri, pentru obținerea celorlalte sub-*string*-uri, se trimite ca și prim argument **NULL**
- *String*-ul inițial este distrus în urma acestor apeluri!
- Exemplu:

```
char sir[100];
gets(sir); //"., Ana ;. are,,,mere;si ; pere..."
char *subsir=strtok(sir,"., ;");
while (subsir!=NULL) {
    puts(subsir);
    subsir=strtok(NULL,"., ;");
}
```



- Câteva prototipuri din biblioteca **string.h**

Prototip	Descriere
void *memcpy(void *s1, const void *s2, size_t n)	Copiază simplu n octeți începând de la adresa s2 la adresa s1 . Se poate ca în timpul copierii sursa să fie suprascrisă. Returnează pointer la începutul zonei de memorie s1
void *memmove(void *s1, const void *s2, size_t n)	Copiază n octeți începând de la adresa s2 la adresa s1 prin intermediul unei zone de memorie temporară. Se evită astfel posibilitatea ca în timpul copierii sursa să fie suprascrisă. Returnează pointer la începutul zonei de memorie s1
int memcmp(const void *s1, const void *s2, size_t n)	Compară primii n octeți corespondenți începând de la adresele s1 și s2 . Returnează 0 dacă octeții sunt identici, ceva mai mic decât 0 dacă s1 < s2 , ceva mai mare ca 0 dacă s1 > s2



- Câteva prototipuri din biblioteca **string.h**

Prototip	Descriere
void *memchr(const void *s, int c, size_t n)	Determină prima apariție a octetului c (convertit la tip unsigned char) în primii n octeți care încep de la adresa s . Dacă c este găsit se returnează pointerul la c în s , altfel se returnează NULL
void *memset(void *s, int c, size_t n)	Copiază valoarea c (convertită la tipul unsigned char) în primii n octeți începând de la adresa s . Returnează pointer la începutul zonei de memorie s



Funcții pentru manipularea blocurilor de memorie – exemplu

```
int a[]={25,-36,91,7415};  
int *b=(int*)malloc(sizeof(a));  
memcpy(b,a,sizeof(a));  
for (int i=0;i<sizeof(a)/sizeof(int);i++)  
    printf("%d ",b[i]);  
printf("\n");  
char c[50]="Ana are mere";  
char d[50]="Ana are mere";  
memcpy(c+8,c,12);  
puts(c);  
memmove(d+8,d,12);  
puts(d);
```

Rezultate afișate
(compilare pe Release):
25 -36 91 7415
Ana are Ana are Ana
Ana are Ana are mere



Rezultate afișate (compilare pe Release):

-1

25 0 0 7415