C programavimo kalba

8 paskaita

(Eilučių funkcijos, duomenų failai)

Eilutės pavyzdys

```
#include <stdio.h>
#include <string.h>
  void main() {
  char str1[] = "Sveiki, kaip laikotes?";
  char str2[40];

  for (int i = 0; i < strlen(str1); i++)
      str2[i] = str1[i];
  str2[i] = '\0';

//  arba paprasčiau  strcpy( str2, str1 )

}
```

```
#include <stdio.h>

void main() {
   char Eil[80];
   int i, c;
   c = 'a' - 'A';   // Atstumas tarp
                       raidžių kodų lentelėje
printf( "Įveskite eilutę mažosiomis
            raidėmis ir be tarpų\n";)
scanf ("%s", Eil);

for(i=0; Eil[i] != '\0'; i++)
   if (Eil[i] >= 'a' && Eil[i] <= 'z')
          Eil[i] -= c;

printf("%s", Eil);
 }
```

ASCII lentelė

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`	128	80	Ç	160	A0	á	192	C0	└	224	E0	α
1	01	Start of heading	33	21	!	65	41	A	97	61	a	129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
2	02	Start of text	34	22	"	66	42	B	98	62	b	130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
3	03	End of text	35	23	#	67	43	C	99	63	c	131	83	â	163	A3	ú	195	C3	├	227	E3	π
4	04	End of transmit	36	24	$	68	44	D	100	64	d	132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
5	05	Enquiry	37	25	%	69	45	E	101	65	e	133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
6	06	Acknowledge	38	26	&	70	46	F	102	66	f	134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
7	07	Audible bell	39	27	'	71	47	G	103	67	g	135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
8	08	Backspace	40	28	(	72	48	H	104	68	h	136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i	137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j	138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k	139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l	140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m	141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n	142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o	143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
16	10	Data link escape	48	30	0	80	50	P	112	70	p	144	90	É	176	B0	░	208	D0	╨	240	F0	≡
17	11	Device control 1	49	31	1	81	51	Q	113	71	q	145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
18	12	Device control 2	50	32	2	82	52	R	114	72	r	146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
19	13	Device control 3	51	33	3	83	53	S	115	73	s	147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
20	14	Device control 4	52	34	4	84	54	T	116	74	t	148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u	149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v	150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
23	17	End trans. block	55	37	7	87	57	W	119	77	w	151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
24	18	Cancel	56	38	8	88	58	X	120	78	x	152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
25	19	End of medium	57	39	9	89	59	Y	121	79	y	153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z	154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{	155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|	156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}	157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~	158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□	159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Eilučių lyginimas

int strcmp(const char *str1**, const char** *str2**);**

Palygina dvi eilutes str1 ir str2. Lygina eilutes, pradedant pirmaisiais simboliais iki tol, kol bus surasti nesutampantys simboliai arba bus surasta vienos iš eilučių pabaiga. Skiria didžiąsias ir mažąsias raides.

Jeigu s1 yra	strcmp grąžina reikšmę, kuri yra
mažesnė už eilutę s2	< 0 (tiksliau -1)
lygi eilutei s2	= = 0
didesnė už eilutę s2	> 0 (tiksliau 1)

Pavyzdys:

char str1[]="Jonas joja arkliu";
char str2[]="jonas joja arkliu";

int i = strcmp(str1, str2); // i bus < 0, nes str1 *Jonas* turi "J", o str2 "j"

Eilučių lyginimas

int stricmp(const char *str1**, const char** *str2**);**

Palygina eilutes, kaip ir ankstesnė funkcija, tik mažąsias ir didžiąsias raides laiko vienodomis. Gražinamos analogiškos reikšmės, kaip ir *strcmp()* atveju.

Pavyzdys: str1[ ] = "Moon";
str2[ ] = "MOON";
int j = stricmp(str1, str2); // j = 0

---

int strncmp(const char *str1**, const char** *str2, size_t num**);**

Palygina *num* pirmų simbolių iš abiejų eilučių. Skiria didžiąsias ir mažąsias raides. Kaip ir ankstesnės funkcijos grąžina tas pačias reikšmes.

Pavyzdys: str1[ ] = "Silver moon is shining";
str2[ ] = "Silver MOON is shining in the dark";
int j = strncmp(str1, str2, 12); // j > 0

Eilučių lyginimas

int strnicmp(const char *str1**, const char** *str2, size_t num**);**

Palygina *num* pirmų simbolių iš abiejų eilučių. Ignoruoja didžiąsias ir mažąsias raides. Kaip ir ankstesnės funkcijos grąžina tas pačias reikšmes.

Pavyzdys: str1[ ] = "Silver moon is shining";
str2[ ] = "Silver MOON is shining in the dark";
int j = strnicmp(str1, str2, 12); // j =0

char *strlwr(char *str**);**

Keičia visas raides mažosiomis.

Pavyzdys: char str[]="Krinta Lapai";
strlwr(str); // dabar str eilutės visos raidės mažosios

char *strupr(char *str**);**

Keičia visas raides didžiosiomis.

Pavyzdys: char str[]="Krinta Lapai";
strupr(str); // dabar str eilutės visos raidės didžiosios

Eilučių funkcijos

char * strrev(char *str**);**

Sukeičia eilutės simbolius vietomis atvirkštine tvarka.

Pavyzdys: str1[ ] = “Silver moon”;
strrev (str1); // dabar str1 eilutė noom revliS

---

char * strstr(const char *str**, const char** *substr**);**

Ieško ar *str* eilutėje yra *substr* eilutė. Jei tokia eilutė rasta, grąžinama rodyklė į pirmąjį *str* simbolį, sutampantį su ieškoma eilute, jei ne - NULL.

Pavyzdys: str1[ ] = “ieškomas elementas”;
str2[ ] = “elementas”;
char *ptr = strstr(str1, str2);
printf(“%s”, ptr); // atspausdinamas elementas

Paieška eilutėje

```
#include <stdio.h>
#include <string.h>
  void main ()
  {   char str[] ="Tai mano eilute";
      char * pch;
      pch = strstr (str,"mano");
      strncpy (pch, "tavo", 4);
      puts (str);
}
```

Ekrane

Tai tavo eilutė

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

Funkcija ieško eilutėje s1 bet kurio simbolio, esančio s2 eilutėje. Sėkmės atveju eilutėje s1 vietoj rasto simbolio įterpia nulinio kodo simbolį NULL. Kartojant kreipinį su NULL vietoje pirmo argumento, grąžinama rodyklė į kitą fragmentą. Kai naujų fragmentų nebėra, grąžinama rodyklė NULL.

Dažnai ši funkcija naudojama žodžių paieškai eilutėje, žodžių atskirimo simbolius nurodant s2 eilutėje.

Pavyzdys

```
#include <string.h>
#include <stdio.h>
  void main() {
  char Eil[30] = "Joju, dairausi ir dainuoju";
  char *p, sep[ ]= ",; !?";
  printf("%s\n", Eil);            // Ekrane visa tiriama eilutė

 // strtok įterpia ribotuvą NULL už surasto žodžio

  p = strtok(Eil, sep);
  if (p)
     printf("%s\n", p);       // Pirmasis žodis

// Tolesni kreipiniai su argumentu NULL grąžina kitų žodžių adresus
while (p)
{   p = strtok(NULL, sep);
      if (p)
         printf("%s\n", p);
}
}
```

Duomenų tipų keitimo funkcijos

Eilučių pertvarkymui į skaičius ir skaičių į eilutes naudojamos bibliotekos **stdlib.h** funkcijos.

int atoi(const char *ptr);

Funkcija paverčia eilutę į kurią rodo *ptr* **int** tipo skaičiumi ir jį grąžina. Jei sutinkamas simbolis, kuris negali būti pakeistas – grąžinamas 0.

int atol(const char *ptr);

Funkcija paverčia eilutę į kurią rodo *ptr* **long int** tipo skaičiumi ir jį grąžina. Jei sutinkamas simbolis, kuris negali būti pakeistas – grąžinamas 0.

```
#include <stdio.h>
#include <stdlib.h>
void main ()
{ char str[] = “200”;
   int i = atoi(str);
   printf (“%d”, i*20);
}
```

Duomenų tipų keitimo funkcijos

double atof(const char *ptr**);**

Funkcija paverčia eilutę į kurią rodo *ptr* **double** tipo skaičiumi ir jį grąžina. Jei sutinkamas simbolis, kuris negali būti pakeistas, funkcija nutraukia darbą.
Eilutėje skaičius turi būti pvz: -1.22e-5 3.5 6.5E+2

---

char itoa(int num**, char** *str**, int** radix**);**
char ltoa(int num**, char** *str**, int** radix**);**

Funkcijos keičia skaičių *num* į eilutę, priklausomai kokia sistema nurodyta kintamajame *radix*. Šios funkcijos nėra apibrėžtos ANSI-C standarte.

Pavyzdys:
```
int number = 567;
char str[10];
itoa (number, str,10);   // verčia dešimtainį skaičių į eilutę str
printf ("%s",str);
```

Pavyzdys

```
/* atof pavyzdys: sin kalkuliatorius */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main ()
{  double n,m;
   double pi=3.1415926535;
   char Input [256];
   printf ( “Ivesk laipsnius: " );
      gets ( Input );          // nuskaitoma eilutė
  n = atof ( Input );
  m = sin (n*pi/180);
     printf ( "sin  %f laipsniu yra %f\n" , n, m );
  return 0;}
```

Ekrane:
Ivesk laisnius: 45
sin 45.000000 laipsniu yra 0.707101

Duomenų tipų keitimo funkcijos

char *gcvt(double val, int ndec, char *buf**);**

Funkcija keičia trupmeninį skaičių val į eilutę, kuriai priskirta rodyklė *buf. Eilutėje gali būti ne daugiau skaičių nei nurodyta kintamajame ndec.

Pavyzdys:

```
#include <stdio.h>
#include <stdlib.h>
  void main ()
  { char str[10];   double num;
    int sig = 4;
    num = 3.243;
    gcvt(num, sig, str);
    printf("%s\n", str);       //    ekrane  3.243

    num = -876.5432;
    gcvt(num, sig, str);
    printf("%s\n", str);       //    ekrane  -876.5

    num = 0.134e4;
    gcvt(num, sig, str);
    printf("%s\n", str);  }     //    ekrane  1340
```

Duomenų failai

Apdorojant didesnės apimties duomenų srautus, patogu duomenis saugoti failuose. Failo kintamieji aprašomi tipo FILE rodykle:

FILE *F;

Failai paruošiami darbui funkcija, kurios prototipas toks:

```
#include <stdio.h>
   FILE *fopen(const char *FailoVardas, const char *M);
```

čia *FailoVardas* nurodomas kaip simbolių eilutė: t.y. tekstas tarp dvigubų kabučių arba konstantos vardu, arba kintamuoju, turinčiu tą reikšmę.
Failo paruošimo darbui būvis nurodomas antruoju argumentu *M* (simbolių eilutė: konstanta, jos vardas arba kintamasis).
*Jei failo nepavyko atidaryti, grąžinamas **NULL**.*

Galimos *M reikšmės:

r	tik skaitymui;
w	tik rašymui; jeigu failas egzistavo, tai naikinamas ir sukuriamas naujai;
a	papildymui gale; jeigu failo nebuvo, tai jis sukuriamas;

Duomenų failai

Būvio *M reikšmę papildžius raide **t** (rt, wt, at), failas bus paruošiamas darbui kaip tekstinis.

Būvio *M reikšmę papildžius raide **b** (rb, wb, ab), failas bus paruošiamas darbui kaip binarinis.

Jeigu nebus panaudota nei **t**, nei **b** raidės, tai failas bus atidaromas darbui priklausomai nuo globalinio kintamojo – **fmode** reikšmės (žr. fcntl.h).

Kiekvienas būvio variantas gali būti papildytas ženklu **+** (plius) (pvz.: r+, w+, wt+). Tai reiškia, kad failas paruošiamas atnaujinimui t.y. leidžiama skaityti ir rašyti.

Pavyzdys:

```
FILE *F;
F = fopen("duomenys.dat", "r");
    if (F == NULL)
        printf("Failas neatidarytas");
```

I/O funkcijos

int **fprintf** (**FILE** *F, const ***char** temp [,<sąrašas>]);

Funkcija skirta informacijos išvedimui į failą su rodykle *F*, naudojant šabloną *temp*, rašomi sąrašo elementai. Šablonas ir argumentų sąrašas sudaromi taip, kaip ir funkcijai printf().

int **fscanf** (**FILE** *F, **const *char** temp [,<sąrašas>]);

Funkcija skirta skaitymui duomenų iš failo pagal šabloną *temp*. Funkcijos reikšmė – sėkmingai perskaitytų elementų skaičius. Šablonas ir argumentų sąrašas sudaromi taip, kaip ir funkcijai scanf().

int **fclose**(FILE *F);

Funkcija, uždaranti failą.

int **feof**(FILE *rodykle);

Funkcija, tikrinanti ar nepasiekta failo pabaiga. Jei failo pabaiga pasiekta, grąžinama nenulinė funkcijos reikšmė.