C programavimo kalba

15 paskaita
(Kurso santrauka II)

Simbolių eilutė

Simbolių eilutės – tai simbolių masyvai, kurie užbaigiami *eilutės terminatoriumi* t.y. nuliniu ASCII lentelės simboliu '\0'. Simbolių eilutes sudaro simbolių visuma įrėminta dvigubose kabutėse.

Pavyzdžiui:
char vardas [12];
strcpy(vardas, "Viktorija");
char diena [] = "Treciadienis";

V	I	K	T	O	R	I	J	A	\0		

Darbui su eilutėmis naudojamos funkcijos, kurios saugomos faile **string.h**, Šis failas turi būti prijungtas su direktyva #include , prieš panaudojant eilučių funkcijas.

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 funkcijos

Eilučių kopijavimo funkcijos
char *strcpy(char *str1, const char *str2**);**
char *strncpy(char *str1, const char *str2, num**);**

Eilučių sujungimo funkcijos
char *strcat(char *str1, const char ***str2**);**
char *strncat(char *str1, const char ***str2**,** num**);**

Simbolių paieškos funkcijos
char *strchr(const char ***s, int** c**);**
char *strrchr(const char ***s, int** c**);**
char *strtok(char ***s1, const char ***s2**);**

Eilučių lyginimo funkcijos
int stricmp(const char ***str1, const char ***str2**);**
int strncmp(const char ***str1, const char ***str2**,** size_t num**);**

Pavyzdys

```
#include <stdio.h>
#include <string.h>
void main()
{           char string[50];  char *ptr, *ptt = 0, c='a';
            int i, suma=0;
            puts("Iveskite sakini...");    gets(string);

            if (ptr = strchr (string, c))
            { printf("Simbolio %c vieta eiluteje yra ", c);
               for (i=0; string[i] != '\0'; i++)
                {ptr = strchr ((string+i), c);
                    if ((ptr) && (ptr!=ptt))
                          {printf("%d ", (ptr-string+1) );
                            ptt=ptr;}}
            } else
               printf("Simbolio %c eiluteje nera\n", c);
            for (i=0; string[i] != '\0'; i++)      // kitas kelias
             if (string[i] == c)
            { printf("Simbolio %c vieta eiluteje yra %d \n", c, i+1); suma++; }
            printf ("\nViso eiluteje yra %d simboliai\n", suma);  }}
```

Pavyzdys

Užduotis: parašyti programą:

- nuskaitytų bet kokio ilgio eilutę iš failo;
- surastų kiek žodžių yra eilutėje ir jų skaičių išvestų į ekraną;
- surasti žodžių ilgius.

main()

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char * nuskaitymas(char *);
int  paieska (char *, char *, char *);

int main()
{   char *file = "Duomenys.txt";
    char *string,  *skirtukai = ".,:;!?-+ ";
    string = nuskaitymas (file);
            if (string == NULL)
            { printf("Duomenu eilute nenuskaityta!!!!\n");
              exit (1);
            }

printf("\nEiluteje viso %d zodziai;", paieska(string, skirtukai, file) );
return 0;
}
```

nuskaitymas ()

```
char *nuskaitymas (char *input_FILE)
{ char d;  char *string;
  int number = 0;  FILE *infile;
  infile = fopen(input_FILE, ”r+”);
             if (infile == NULL)
                          return NULL;
d=fgetc(infile);
if(d == '#')                                  //#-simbolis, naudojamas kaip komentaras
 {while (d=fgetc(infile))
       if (d == '\n')
          break; }
else fseek(infile, -1, SEEK_CUR);
  while (d=fgetc(infile))
    if (d != '\n')
        number ++;
    else break;
 fseek(infile, -number-2, SEEK_CUR);
 string = (char *) malloc(number);
 fgets(string, number+1, infile);
             fclose(infile);
             return string;        }
```

paieska()

```
int paieska (char *string, char *skirtukai, char *file)
{int  matrix[80];                                  //masyvas, kuriame saugomi zodziu ilgiai
  int  sum = 1, count = 1;  char *pointer;
  FILE *outfile;
  outfile=fopen(file,”a”);
       pointer = strtok(string, skirtukai);
  fprintf( outfile, "\n------ REZULTATAI------\n“);
  matrix[0]=strlen(pointer);
  fprintf(outfile, “1 zodzio ilgis %d simboliai\n“,matrix[0]);

while (pointer)
{ pointer = strtok(NULL, skirtukai);
  if (pointer)
  {             matrix[count]=strlen(pointer);
                fprintf(outfile, “%d zodzio ilgis %d simboliai\n“, count+1, matrix[count]);
                count++;
  }
} fclose(outfile);
return count;
}
```

Struktūros

Visi kintamieji, iki šiol kuriais naudojotės, priklausė baziniams C/C++ duomenų tipams:

- Sveiko tipo (int) kintamieji ir konstantos;
- Realaus tipo (float) kintamieji ir konstantos;
- Dvigubo tikslumo (double) kintamieji ir konstantos;
- Simbolinio tipo (char) kintamieji ir konstantos;

Šie duomenys galėjo sudaryti masyvus, tačiau masyvo elementais gali būti tik to paties bazinio tipo duomenys.

Struktūra – tai vienodo arba skirtingo tipo kintamųjų rinkinys.

Struktūra priklauso išvestinių (vartotojo) duomenų tipui.

Struktūra realiame pasaulyje

Sąrašai

- Įmonės darbuotojai
  - Vardas, pavardė, adresas, gim.metai, išsilavinimas, paso Nr.
- Telefono numerių
  - Vardas, pavardė, adresas, tel.Nr.

Kartotekos

- Knygų, prekių, CD ...

Struktūros

Struktūros sintaksė:

struct [*struktūrinio tipo vardas*] **{** [*laukų aprašų sąrašas*] **}**
[*kintamųjų sąrašas*];

Laukų aprašų sąrašų elementai atskiriami **kabliataškiais**, o kintamųjų sąrašo elementai – **kableliais**. Jei apraše nėra kintamųjų sąrašo, jis apibrėžia tik naujų struktūrinių duomenų tipą, tačiau jo realizacijoms atmintyje vietos neskiria.

Atskiro struktūros tipo apibrėžimo ir realizavimo pavyzdys:

```
struct telefonas { char vardas[30]; unsigned long int tel; } asmeninis;
```

Struktūros pavyzdys su funkcija

```
#include <iostream.h>
#include <string.h>
#include <stdlib.h>

struct filmai_t {
  char title [50];
  int year;
} mano, tavo;

void printmovie (struct filmai_t );

void main ()
{  char buffer [50];
   strcpy (mano.title, "2001 A Space Odyssey");
   mano.year = 1968;
   printf( "Ivesk pavadinima: ");
   fgets (tavo.title, 50, stdin);
   printf( "Ivesk metus: ");
   scanf("%d", &tavo.year);
```

```
prinf("Mano filmas:\n ");
   printmovie (mano);
printf("Tavo filmas:\n ");
   printmovie (tavo);
return 0;
}

void printmovie (filmai_t movie)
{
  puts(movie.title);
  printf( " ( %d )\n", movie.year);
}
```

Struktūrų masyvas

Dažniausiai naudojamos ne pavienės struktūros, o jų masyvai, kurie sudaro duomenų bazių pagrindą. Struktūrų masyvai apibrėžiami analogiškai, kaip ir įprastiniai masyvai.

```
struct CD
{    char name[20];
     float price;
     int number;
}
```

Struktūrų masyvas

struct CD disc[10];

```
strcpy(disc[0].name, "Aliukai");
strcpy(disc[1].name, "Broliukai");
.............
for (int i = 0; i < 10; i++)
        { cout << "Pavadinimas"<<disc[i].name<<end;
  ........
 }
```

Rodyklės ir struktūros

Kaip ir bet kuriam duomenų tipui, taip ir struktūrai gali būti sukurta rodyklė. Taip pat struktūra gali būti rodyklės tipu.

Rodyklės deklaravimas

struct Struktūros_pavadinimas * rodyklės_pavadinimas;

Pavyzdžiui:

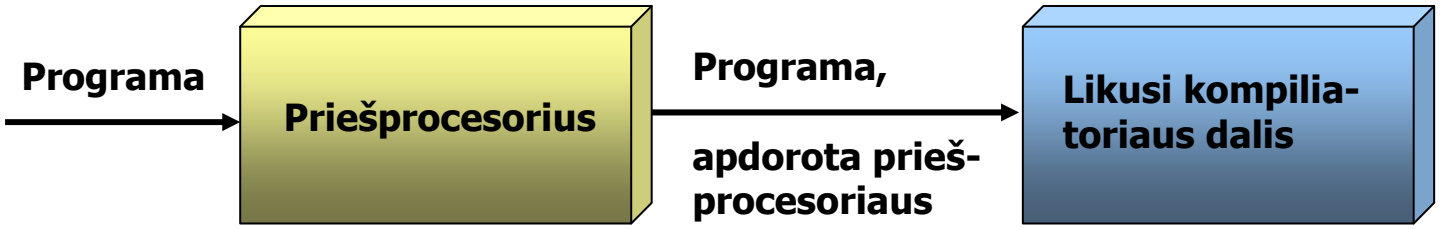
```
struct movies_t
{   char title [50];
    int year; };
 struct movies_t  amovie, *pmovie;

 pmovie = &amovie;      // adreso priskyrimas
```

Struktūros elementai, kai naudojama rodyklė, pasiekiami naudojant ne ".", o "**->**".

Pavyzdžiui: pmovie->title; pmovie->year;

Priešprocesorius

Priešprocesorius - tai integruota C/C++ kompiliatoriaus dalis, kuri atlieka paruošiamuosius veiksmus, prieš pradedant kompiliuoti programą.



Priešprocesorius valdomas direktyvomis, t.y. komandomis, kurios nurodo, kokius veiksmai turi būti atlikti prieš pradedamos programos kompiliavimą.

Direktyvos – tai tam tikras raktinis žodis, kuris pradedamas **#** simboliu.

Direktyvos apdorojamos pirminiame programos kompiliavimo etape, o jų naudojimas apibrėžtas C standarte.

Direktyvos

Direktyvų sąrašas:

#define	#elif	#else	#endif
#error	#if	#ifdef	#ifndef
#include	#line	#pragma	#undef

Direktyva #include skirta nuskaityti ir sukompiliuoti nurodytą failą.

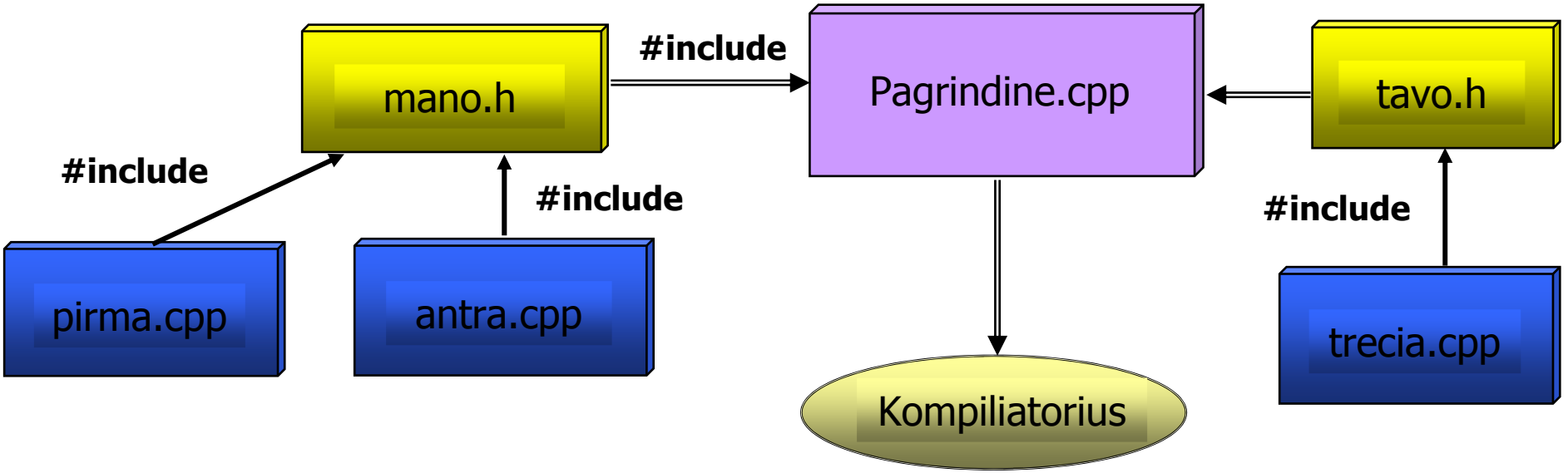
Sintaksė:

```
#include <header_name>      \\ paieška vykdoma kataloge  ....\include
#include "header_name"      \\ paieška vykdoma .\   po to  ....\include
#include macro              \\ makro komandos prijungimas
```

Programa keliuose failuose

Galimos failų prijungimo modifikacijos (3 skyriaus):

- Sukuriamas antraštės failas (*pvz. header.h*) , jame prijungiami visi failai, o
  pagrindinis failas prisijungia tik *header.h* failą.



Pavyzdys

```
// pagrindine.cpp

#include <stdio.h>
#include <stdlib.h>
#include "mano.h"

#define VAR 10

void main()
{ int k, arr[VAR];
   k = ivedimas();
   srand (k);
   for(int j =0; j < VAR; j++)
   {  arr[j] = rand();
      printf("arr[%d] = %d\n", j, arr[i] );
   }
 pabaiga();
}
```

```
// Failas mano.h

#include "pirmas.cpp"
int ivedimas()
{  int i;
    puts("Ivesk skaiciu");
    scanf("%d",&i);
    return i;
 }
```

```
// Failas pirmas.cpp

void pabaiga()
{ printf ("Programa baige
            darba sekmingai\n");
  }
```

Laikas ir data

time.h – failas, kuris turi būti prijungtas naudojant datos ir laiko funkcijas.

Laiko skaičiavimui skaičiavimui naudojami tokie duomenų tipai:

- clock_t; time_t; tm

Laiko duomenų tipai clock_t, time_t skirti atvaizduoti sisteminį laiką ir datą, kaip sveiką skaičių, todėl šie duomenų tipai yra long int.

tm – tai struktūra, kurioje gali būti saugoma pilna informacija apie laiką ir datą.

time.h faile apibrėžta makrokomanda CLOCKS_PER_SEC, kuri parodo sisteminio laikrodžio tiksėjimų skaičių per sekundę.

Ji naudojama su funkcija clock(void), kai reikia sužinoti laiką sekundėmis.

Sėkmės, laikant pirmąją sesiją!!!!