C programavimo kalba

7 paskaita

(Rodyklės, eilutės)

Rodyklių naudojimo pavyzdys

```
#include <stdio.h>
void order(int *, int *);
void main() {
int n1 = 99, n2 = 11, n3 = 22, n4 = 88;
  order (&n1, &n2);
  order (&n3, &n4);

  printf("n1 = %d\n n2 = %d", n1, n2);
  printf("n3 = %d\n n4 = %d", n3, n4);
}

void order (int *numb1, int *numb2)
{ if (*numb1 > *numb2)
     { int temp = *numb1;
       *numb1 = *numb2;
       *numb2 = temp;
      }
 }
```

Rodyklių aritmetika

Operacija	Pavyzdys
Lyginimas	p1 = = p2
Lyginimas ar nelygu	p1 != p2
Mažiau	p1 < p2
Mažiau arba lygu	p1 <= p2
Daugiau	p1 > p2
Daugiau arba lygu	p1 > = p2
Elementų skaičiaus skaičiavimas	p1- p2
Suskaičiuoja nuorodą nutolusią per **n** nuo duotosios	p1 + n; p1 – n

Rodyklių aritmetikos pavyzdys

```
#include <stdio.h>
int Suma(int *, int);
void main()
{ int A[10], i, n;
   printf (“Iveskite n reiksme: “);
   scanf (“%d”, &n);
    for (i=0; i<n; i++) {
        printf ( “\nA[%d]= “, i);
        scanf(“%d”, &A[i]);
    }
 printf ("Suma = %d\n" ,  Suma(A, n) );
}

// Funkcija Suma
int Suma(int *ptr, int k)
{   int S = 0, i = 0;
    while(i < k) { S += *(ptr+i);  i++; }
    return S; }
```

```
#include <stdio.h>

void main()
{    float fa[10], *fp1, *fp2;
      fp1 = fp2 = fa;

      while(fp2 != &fa[10])
      { printf(“Skirtumas: %d\n",
         (int)(fp2-fp1) );
         fp2++;   }
 }
```

Dinaminis atminties išskyrimas

Masyvų elementų skaičius gali priklausyti nuo pradinių duomenų skaičiaus, todėl dažnai naudojami ne statiniai, o dinaminiai masyvai. Tokiems masyvams atmintis dinamiškai išskiriama funkcijomis: **malloc()**, **calloc(),** o atmintis atlaisvinama funkcija **free()**.

Šioms funkcijoms reikalingas **stdlib.h** antraštės failas.

Sintaksė:

```
void *malloc (size_t  size);
```

čia *size* – baitų skaičius, kuris turi būti rezervuotas

```
void *calloc (size_t  num,  size_t  size);      // visi elemtentai lygūs 0
```

čia *num* – elementų skaičius, *size* – elemento dydis baitais.

Jei atmintis rezervuota sėkmingai, grąžinama **rodyklė** į išskirtą atmintį. Jei atmintis nerezervuota, grąžinamas identifikatorius **NULL** ('\0').

Dinaminis atminties išskyrimas

Funkcijos **malloc()** ir **calloc()** grąžina rodyklę į nurodyto tipo atmintį, todėl naudojant šią funkciją nurodomas rodyklės tipas.

```
int* pInt =(int*) malloc(40);   float* pfl= (float*) calloc(4, 4);
```

Atmintis atlaisvinama naudojant funkciją **free().**

Sintaksė:

```
void free (void *block);
```

čia **block* – rodyklė į atminties bloką, kuris turi būti atlaisvintas.

Pavyzdys

```
#include <stdlib.h>
   void main() {
   int *ip, ar[100];
   ip = (int *) malloc( sizeof ar );
   free (ip);
  }
```

Funkcija sizeof()

Norint sužinoti kiek kintamasis užima atminties baitų, naudojama funkcija sizeof()

Sintaksė: sizeof *unarinė išraiška*; sizeof (duomenų tipas);

Kreipinio metu atminties kiekį patogu nurodyti kaip duomenų elementų skaičiaus ir vieno elemento užimamos vietos baitais sandaugą, pavyzdžiui:

n * sizeof(float)

kur *n* – realių skaičių kiekis.

Funkcija sizeof() grąžina nurodyto duomenų tipo užimamą atminties kiekį baitais.

Dinaminis masyvas

```
#include <stdio.h>
#include <stdlib.h>
int main()
{ int N, *ptr;
  printf( "Iveskite masyvo elementu skaiciu: ");
  scanf ("%d", &N);
  ptr = (int*) malloc(N*sizeof(int) );      // išskiriama atmintis masyvui
   if ( ptr == NULL)
    {  printf ("Truksta atminties \n");
     return 1;}

  for (int i = 0; i < N; i++)
     *(ptr+i) = i;                          //arba tas pats būtų   ptr[i]=i;

  for ( i = 0; i < N; i++)
     {printf( "%d elementas = %d\n", i, *(ptr+i) );
       }
  free (ptr);
  return 0; }
```

Dinaminis masyvas

```
#include <stdio.h>
#include <stdlib.h>
int *Duomenys(int);     // Duomenų įvedimas
void Sp (int *, int);   // Išvedimas ekrane

// Pagrindine programa
void main() {
   int n, *A;
   printf( "Iveskite elementu skaiciu: ");
   scanf ("%d",&n);

A = Duomenys(n);
if (A != NULL) {
   printf(" Ivestas masyvas:\n");
   Sp(A, n);
}
free(A);
}
```

```
   int *Duomenys(int n) {
   int *x, i;
   x = (int *) malloc(n * sizeof(int));
   if (x == NULL)
   printf("Truksta atminties\n");
   else {
   for(i=0; i<n; i++) {
   printf("Iveskite %d-a elementa", i+1);
   scanf("%d", (x+i)); }
   }
   return x;}
```

```
void Sp(int *x, int n) {
 int i = 0;
 while(i < n) {
    printf("%d-elementas: %d\n", i+1, *(x+i));
    i++; }
}
```

Dvimatis dinaminis masyvas

```
#include <stdio.h>
#include <stdlib.h>
void main()
{ int n, m, i, j, *ptr;
  printf( "Iveskite masyvo eiluciu ir stulpeliu skaiciu:");
  scanf ("%d %d",&n, &m);
  if ( (ptr = (int*) malloc(n*m*sizeof(int) ) == NULL);
       printf ("Truksta atminties \n");
  else {
     for ( i = 0; i < n*m; i++)
       *(ptr+i) = i;

     for ( i = 0; i < n; i++)
      {  for (j = 0; j<m; j++)
          { printf( " %d  ", *(ptr+i*m+j));
                      }
         printf ("\n");
        }}
}
```

Dvimatis dinaminis masyvas

```
#include <stdio.h>
#include <stdlib.h>
void main()
{ int i, eilute, stulpelis;
  printf("Iveskite matricos eiluciu ir stulpeliu skaiciu:");
  scanf ("%d %d", &eilute, &stulpelis);
        int **array1 = (int **)malloc(eilute * sizeof(int *));
        for(i = 0; i < eilute; i++)
                array1[i] = (int *)malloc(stulpelis * sizeof(int));
  for ( i = 0; i < eilute; i++)
     for (int j = 0; j < stulpelis; j++)
        array1[i][j] = rand();

  for ( i = 0; i < eilute; i++)
     {for ( int j = 0; j < stulpelis; j++)
          printf(" %d  ", array1[i][j]);
      printf("\n");
     } }
```

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 t.y.

#include <string.h>

Pavyzdys

```
#include<stdio.h>
#include<string.h>
main ( )
{ float weight, volume; int size, letters; char name[40];
   printf("Labas! Koks Jūsų vardas? \n");
      scanf( "%s", name);
      printf("%s, kokia Jūsų masė?\n", name); scanf("%f", &weight);
size = sizeof( name);              //masyvo name[] dydis baitais
letters = strlen(name);            //eilutės simbolių skaičius
volume = weight/1200;

printf("Nuostabu, %s, Jūsų tūris %2.2f kūbiniai metrai.\n", name, volume);
printf("Be to, Jūsų vardas sudarytas iš %d raidžių, \n", letters);
printf("ir jo patalpinimas kompiuterio atmintyje užima %d baitų.\n", size);

fflush(stdin);
getchar();      // laukiama kol bus įvestas bet koks simbolis
return 0;}
```

Eilučių kopijavimas

char *strcpy(char *str1, const char *str2**);**

Eilutę str2 kopijuoja į str1. Kopijavimas atliekamas iki str2 eilutės terminatoriaus. Grąžina rodyklę į str1. Eilutės str2 ilgis <= str1 ilgį.

Pavyzdžiai:

```
char str[20];
strcpy(str,"Mano batai buvo du");
```

```
char str1[]="Mano batai";
char str2[20];
char *ptr= str1;
ptr +=5;
strcpy(str2,  ptr);
```

char *strncpy(char *str1, const char *str2, num**);**

Eilutę str2 kopijuoja į str1. Kopijuoja *num* simbolių iš str2 eilutės. Grąžina rodyklę į eilutę str1.

Pavyzdys:

```
char long[]="abcde";
char shot[3];
strncpy(shot, long,2);
```

Eilučių sujungimas

char *strcat(char *str1, const char ***str2);**

Eilutės str2 kopiją prijungia prie eilutės str1 galo. Grąžina rodyklę į sujungtas eilutes t.y str1. Grąžinamos eilutės ilgis: strlen(str1) + strlen(str2).

Pavyzdys: char str[80];
strcpy(str, "Sveiki atvyke");
strcat(str, "i VGTU");

char *strncat(char *str1, const char ***str2, num);**

Eilutės str2 *num* simbolių prijungia prie eilutės str1 galo. Grąžina rodyklę į sujungtas eilutes t.y. str1. Grąžinamos eilutės ilgis: strlen(str1) + num.

Pavyzdys: char str1[] = "Sveikas atvykes";
char str2[]= "Petrai Petraiti";
strncat (str1, str2, 6);
printf ("%s", str1);

Simbolių paieška

char *strchr(const char *s, int c);

Eilutėje **s** ieško simbolio **c**. Paieška pradedama nuo eilutės pradžios ir baigiama suradus pirmąjį simbolį, lygų nurodytam. Nulinis simbolis laikomas eilutės dalimi, todėl jo paieška taip pat galima.
Pavyzdžiui, **strchr(strs,'\ 0')** grąžina rodyklę į nulinį simbolį.

Funkcija grąžina **rodyklę** į surastą simbolį eilutėje, arba NULL neradus.

```
#include <string.h>
#include <stdio.h>
  void main(void) {
    char string[20] = "Nagrinejama eilute";
    char *ptr, c = 'r';
    ptr = strchr(string, c);
    if (ptr != NULL)
        while (ptr != NULL)
         {   printf("Simbolis %c rastas: %d-as \n", c, (ptr – string+1) );
             ptr = strchr(ptr+1, c); }
     else printf ("Simbolio r eiluteje nera\n"); }
```

Simbolių paieška

char *strrchr(const char *s, int c);

Eilutėje **s** ieško simbolio **c** pradedant nuo s eilutės galo. Jei simbolis nerastas, grąžinamas NULL, jei rastas – rodyklė į rastą simbolį.

Pavyzdys:
```
char str[ ] = "abc abc";
char *ptr;
ptr = strrchr(str, 'a');    // suranda pirmąją a iš galo
```

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

Tikrina kiekvieną eilutės str1 ir str2 simbolį. Funkcija grąžina sutapusių simbolių skaičių. Tikrinimas baigiamas, kai sutinkamas **pirmas** nesutampantis simbolis.

Pavyzdys:
```
char str1[ ] = "Mano batai buvo du";
char str2[ ] = "Mano batai buvo trys";
int index = 0;
index = strspn (str1, str2);     // 16
printf("Sutapo %d simboliu \n", index);
```