Procedūrinio programavimo pagrindai

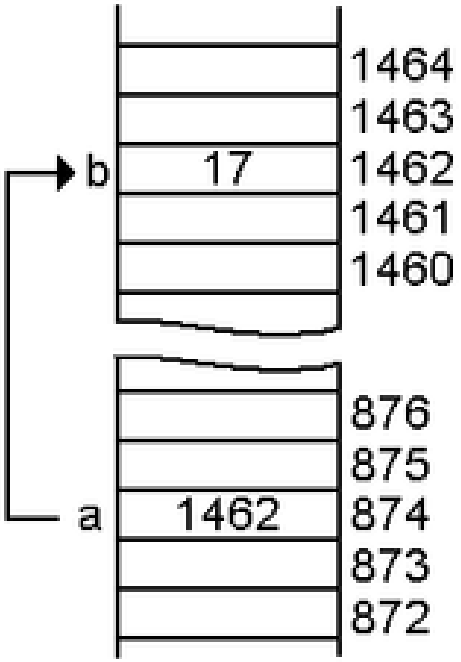
Rodyklės

lekt. Irmantas Radavičius
irmantas.radavicius@mif.vu.lt
Informatikos institutas, MIF, VU

Turinys

Rodyklės

Rodyklė (angl. pointer) – tai kintamasis, skirtas adresams saugoti.



Rodyklių taikymai:

- masyvai
- parametrų perdavimas
- ...

Rodyklės

Rodyklės apibrėžimas:

- tipas * vardas;

Rodyklių tipai:

- tipizuotos

  int * ptrToInt;
- netipizuotos

  void * ptrToSmth;

Rodyklių tipai

```
int * i_ptr; char * c_ptr;  void * v_ptr;          /* rodyklės */

v_ptr = c_ptr;  i_ptr = v_ptr;                     /* gerai */
i_ptr = c_ptr; c_ptr = i_ptr;                      /* perspėjimas, klaida */

c_ptr = (char *)i_ptr; i_ptr = (int *)c_ptr;       /* gerai */

c_ptr = (char *)100;                               /* gerai! */
```

Operatoriai

- adreso operatorius &
- išrodyklinimo operatorius *

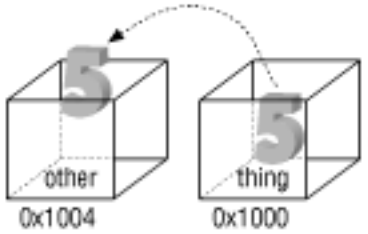
C Code	Description
`thing`	Simple thing (variable)
`&thing`	Pointer to variable `thing`
`thing_ptr`	Pointer to an integer
`*thing_ptr`	Integer

Operatoriai

A `thing_ptr = &thing;`

Assigns **`thing`**'s address to **`thing_ptr`**.

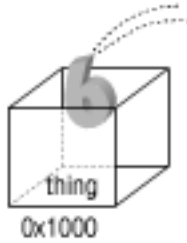
B `other = *thing_ptr;`



0x1000
thing_ptr

Assigns to **`other`** the value at the address **`thing_ptr`** carries.

C `*thing_ptr = 6;`



0x1000
thing_ptr

Assigns to a value to what **`thing_ptr`** points to.

Rodyklės į rodykles

Rodyklės tipas yra išvestinis!

```
int x;
int *p = &x;
int **q = &p;
…
```

Rodyklių inicializacija

- rodyklė “į niekur”

```
#include <stdlib.h>
int *p1 = NULL;                    /* p1 = 0; bet p1 = (int *) 1; */
```

- rodyklė į kintamąjį

```
int x = 0, *p2 = &x;
```

- rodyklė masyve

```
int a[5], *p3 = &a[3];
```

- dinaminio atminties skirstymo funkcijos

```
int *p4  = (int *) malloc (sizeof(int));
```

Rodyklių aritmetika

- pridėti/atimti sveiką skaičių

  p = p + 1; p--; p+=5;

- rasti skirtumą tarp rodyklių (tame pačiame bloke)

  p – q; (p+1) – (p-1);

- palyginti rodykles (tame pačiame bloke)

  p != q; p < q

Draudžiama

- aritmetika netipizuotoms rodyklėms
- suma ir kiti aritmetiniai veiksmai

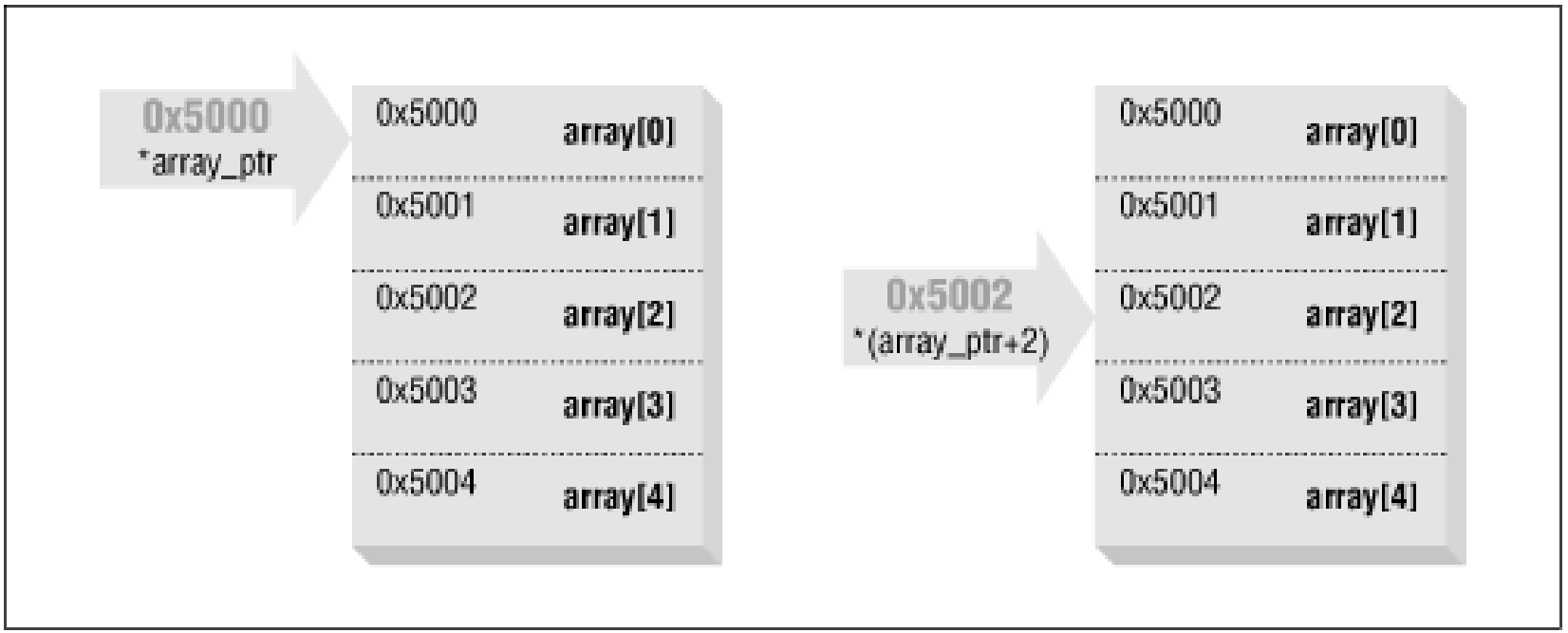
Operatorių prioritetai

++*p;	*++p;	*p++;
(*p++);	*(p++);	(*p)++;
(*p + 1);	(*p) + 1;	*(p + 1);

OPERATORS	ASSOCIATIVITY
`()  []  ->  .`	left to right
`!  ~  ++  --  +  -  *  &  (type)  sizeof`	right to left

Vėl masyvai

Ryšys tarp masyvo ir rodyklių: a[i] atitinka *(a+i), &a[i] atitinka (a+i)

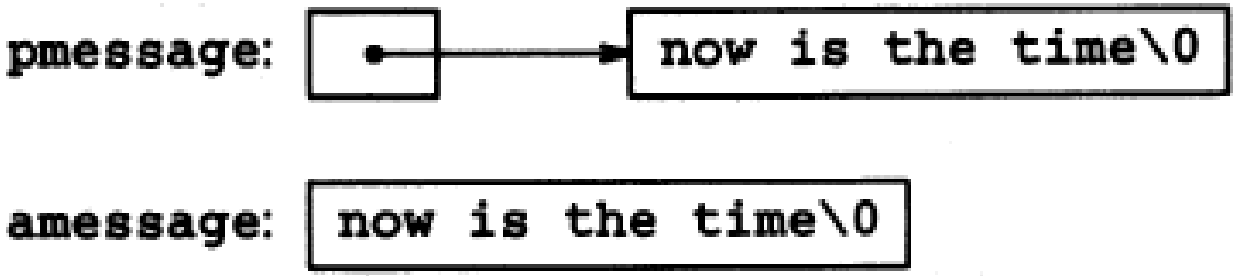


Skirtumas: masyvo vardas nėra kintamasis!

Pastaba: 2[a] leistina?!!

Vėl eilutės

```
char amessage[] = "now is the time";    /* an array */
char *pmessage = "now is the time";     /* a pointer */
```



Pastaba:

pabandžius modifikuoti konstantinę (!) eilutę, rezultatas neapibrėžtas.

Dinaminiai vs statiniai masyvai

```
int x1[10];          /* C89 – masyvo dydis yra konstanta */
int x2[n = 10];      /* C99 - kintamo dydžio masyvai */
```

Statiniai masyvai:

- ✓ paprastesni
- – dydis turi būti žinomas kompiliavimo metu (C89)
- – negalima keisti jau esamo masyvo dydžio (C99)

Dinaminiai masyvai!

- ! reikia mokėti naudoti

Dinaminio atminties skirstymo funkcijos

Biblioteka <stdlib.h>

Atminties skirstymas:

```
void * malloc(unsigned size);
void * calloc(unsigned num , unsigned size);
```

Atminties perskirstymas:

```
void * realloc(void * ptr , unsigned size);
```

Atminties atlaisvinimas:

```
void free(void * ptr);
```

Dinaminio atminties skirstymo funkcijos

Funkcija malloc: `void * malloc(unsigned size);`

- gauna pageidaujamo bloko dydį (baitais)
- grąžina rodyklę į bloko pradžią (arba NULL)

Funkcija calloc: `void * calloc(unsigned num , unsigned size);`

- gauna pageidaujamą (dinaminio masyvo) elementų skaičių ir vieno elemento dydį
- grąžina rodyklę į bloko pradžią (arba NULL)
- visus bloko bitus inicializuoja nuliais

Dinaminio atminties skirstymo funkcijos

Funkcija realloc: `void * realloc(void * ptr , unsigned size);`

- gauna nuorodą į jau rezervuoto bloko pradžią ir pageidaujamo bloko (naują) dydį baitais
- grąžina rodyklę į bloko pradžią (nepasikeitusią, pasikeitusią arba NULL)

Funkcija free: `void free(void * ptr);`

- gauna nuorodą į rezervuoto bloko pradžią
- atlaisvina bloką (rodyklė išlieka nepakitusi!)

Parametrų perdavimas

Perdavimas vyksta per reikšmę (angl. by value).

Perduodamos kintamųjų kopijos.

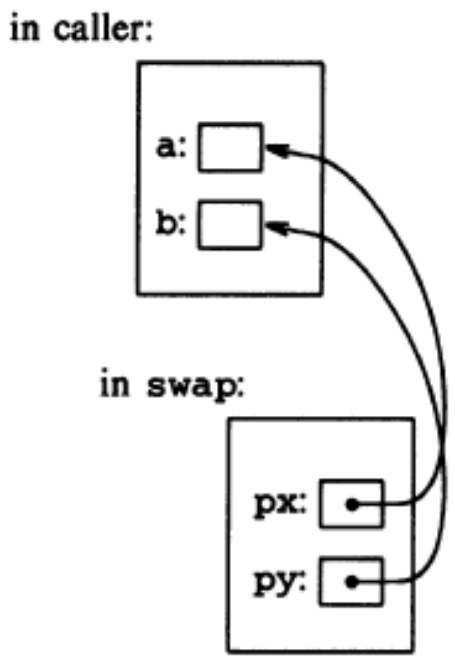
Parametrai-kintamieji yra modeliuojami parametrais-reikšmėmis.

Norint pakeisti perduodamų kintamųjų turinį, naudojamos rodyklės.

Parametrų perdavimas

```
void swap_blogas(int x,int y) {
  int temp;
  temp=x;
  x=y;
  y=temp;
}
int main() {
  int a=1, b=2;
  swap_blogas(a,b);
  /* a=1,b=2 */
...
}
```

```
void swap(int* x,int* y) {
  int temp;
  temp=*x;
  *x=*y;
  *y=temp;
}
int main() {
  int a=1, b=2;
  swap(&a,&b);
  /* a=2,b=1 */
...
}
```



Masyvų perdavimas!

Komandinės eilutės parametrai

Funkcija main – programos įeities taškas.

```
int main(void) { . . . }

int main(int argc, char *argv[ ]) { . . . })
```

argc – komandinės eilutės parametrų skaičius

argv – parametrų reikšmės (eilučių masyvas)

free(students)