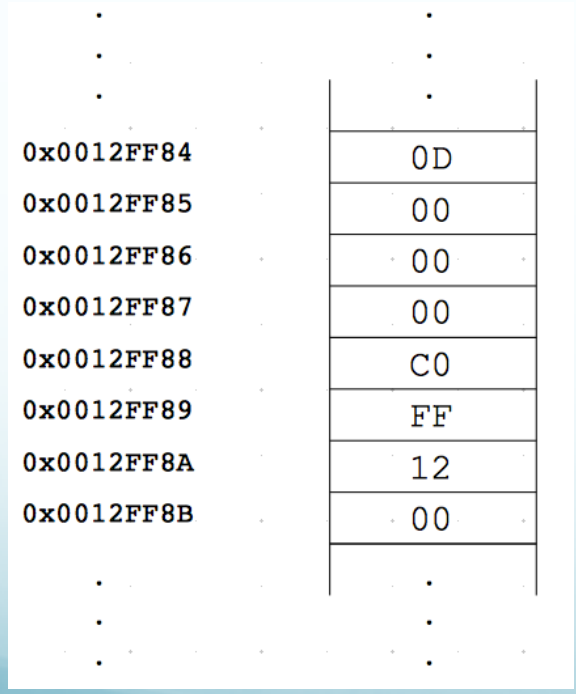

# Programski jezik C - kazalci

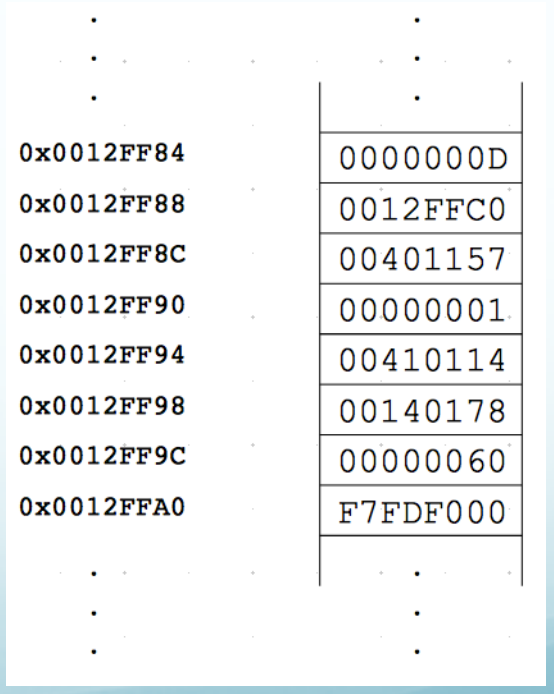
Tomaž Dobravec

# Kazalci

**Kazalec je spremenljivka, ki hrani naslov pomnilniške celice**

```
0012FF7B  00 D8 2A 14 00 78 01 14 00 0D 00 00 00 C0 FF 12 00 57 11
0012FF8E  40 00 01 00 00 00 14 01 41 00 78 01 14 00 60 00 00 00 00
0012FFA1  F0 FD 7F 06 00 00 00 98 FF 12 00 23 51 58 80 E0 FF 12 00
0012FFB4  38 16 40 00 04 70 40 00 00 00 00 00 F0 FF 12 00 37 60 81
0012FFC7  7C 78 01 14 00 60 00 00 00 00 F0 FD 7F 38 B8 54 80 C8 FF
0012FFDA  12 00 E8 F7 12 8A FF FF FF FF 48 9B 83 7C 40 60 81 7C 00
0012FFED  00 00 00 00 00 00 00 00 00 00 00 C0 10 40 00 00 00 00 00
00130000  41 63 74 78 20 00 00 00 01 00 00 00 9C 24 00 00 C4 00 00
```



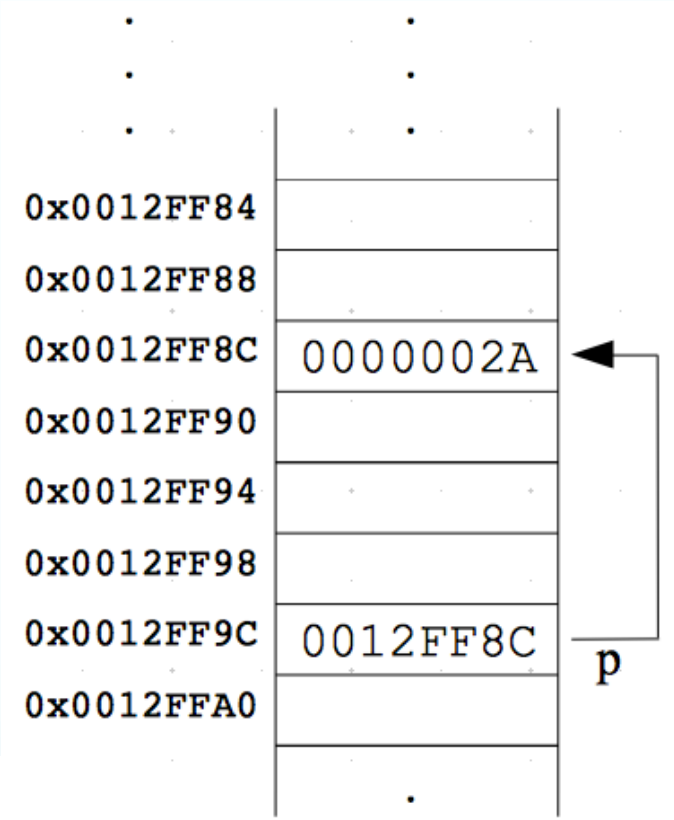




Shematski prikaz vsebine pomnilniških lokacij (trije načini)

# Primer uporabe kazalcev (absolutno naslavljanje)

Če bi želeli v pomnilniško celico številka 0x0012FF8C vpisati število 42 (0x0000002A), bi v programskem jeziku C postopali takole:

- deklarirali bi spremenljivko p kot kazalec na tip int,
- v p bi vpisali 0x0012FF8C (kazalec p bo s tem ''pokazal" na celico s tem naslovom),
- v celico, na katero kaže p, bi vpisali 42.




```
int *p;           // naj bo p kazalec na int
p = 0x0012FF8C;   // p naj pokaze na 0x0012FF8C
*p = 42;          // kamor kaže p, vpiši 42
```

# Primer uporabe kazalcev (relativno naslavljanje)

Spremeniti želimo vsebino spremenljivke `x` brez neposrednega prirejanja `x=...`

```
int x;     // spremenljivka tipa int
int *p;    // kazalec na int
p  = &x;   // p pokaze na celico, ki pripada x
*p = 15;   // kamor kaze p (torej v x) vpisemo 15
```

# Operatorja * in &

**Operator * (dereferenčni operator)**

- deklaracija kazalca .... `int *p;`

  (`p` je kazalec na spominske lokacije tipa `int`)

- vrednost na naslovu .... `*p = 15;`

  (na naslov `p` se zapiše vrednost 15)

**Operator &**

- naslov .... `p = &x;`

  (`p` dobi vrednost naslova `x;` `p` "pokaže na" `x`)

# Naslov je število

Tip kazalec (na primer `int *`) je celoštevilski tip, saj hrani naslove (torej cela števila).

Primer (za razumevanje, ne za uporabo!)

```
int x;    int y = 7;
x = (int) &y;
*((int *)x) = 5;

printf("%d\n", y);
```

# Kazalci in tabele

Ime tabele je sinonim za naslov njenega prvega elementa

```
char a[10];
```

`a` pomeni isto kot `&a[0]`

```
Test: printf("%p, %p \n", a, &a[0]);
```

# Kazalci in tabele

Če imamo tabelo, lahko namesto nje uporabimo kazalec

```
int a[10];
int *pa;

// isto kot pa=a
pa = &a[0]; // pa pokaze na prvi element tabele a

*pa      = 15; // isto kot a[0] = 15;
*(pa+5) = 5;  // isto kot a[5] = 5;

pa = pa + 3;  // pa pomaknemo za tri polja naprej
*pa      = 7;  // isto kot a[3] = 7;
```

# Kazalci in tabele

- Kazalci in tabele so tako močno povezani, da lahko izmenično uporabljam enega ali drugega

```c
// strlen s tabelo
int strlen(char s[]) {
  int n=0;
  while (s[n] != '\0') {
    n++;
  }
  return n;
}
```

```c
// strlen s kazalcem
int strlen(char *s) {
  int n=0;
  while (*s != '\0') {
     n++; s++;
  }
  return n;
}
```

- Še več: zgornji funkciji bi delali pravilno, tudi če bi zamenjal samo njuni glavi!

# Kazalci in tabele

- Če imamo tabelo, lahko namesto nje uporabimo kazalec:

```
int a[10];
int *pa;
pa = a;
// ... od tu naprej namesto z a delamo s pa
// ... namesto a[i] pisemo *(pa+i)
```

- Kaj pa obratno? Imamo kazalec in bi ga radi uporabili kot tabelo?

# Kazalci in tabele

- Imamo torej kazalec

```
int *tp;
```

- Kaj manjka, da bi lahko pisali

```
tp[10] = 15;
```

?

# Kazalci in tabele

- Odgovor: rezervirati moramo prostor v pomnilniku (za celotno tabelo) in s kazalcem pokazati na začetek tega prostora:

```
int *tp;

// rezerviramo prostor za tabelo velikosti 100
// (potrebujem torej 100krat toliko prostora,
// kot ga zasede en podatek tipa int)
tp = (int *) malloc (100 * sizeof(int));

tp[10]=15;
```

!

# Dinamično delo s pomnilnikom

```
int *p;
// rezerviramo prostor na kopici
p = (int *) malloc(100 * sizeof(int));

// ... uporabljamo tabelo p
p[10]= 15
// sprostimo prostor na kopici
free(p);
```

- Pomembno: če rezerviranega prostora (`malloc`) ne potrebuješ več, ga moraš OBVEZNO sprostiti (`free`)!

# Dinamično delo s pomnilnikom

`malloc()` ... rezervira pomnilnik in vrne kazalec

`calloc()` ... isto kot `malloc()` + vrednosti v rezerviranem delu pomnilnika postavi na 0

`realloc()` ... spremeni velikost rezerviranega prostora

`free()` ... sprosti rezerviran del pomnilnika

Primer: program, ki v zanki povečuje velikost rezerviranega pomnilnika.

# Linearizacija dvodimenzionalne tabele

- Dvodimenzionalna tabela (npr. `int a[3][3]`) števila zapiše v pomnilnik v row-major vrstnem redu
- Do elementov lahko dostopamo tako, da sami izračunamo položaj v enodimenzionalni tabeli:

```
int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
int *b = (int *)a;  // enodim. tabela

int i=2, j=1;
printf("%d\n", a[i][j]);
printf("%d\n", b[i*3+j]);
```

# Kazalci in strukture

```
struct kompleksno w;
w.re = 5;
w.im = 1;
```

```
struct kompleksno *z;

z = (kompleksno*) malloc(sizeof(kompleksno));

(*z).re = 3;
(*z).im = 2;
```


Primer: tabela podatkov cplx in kazalec na tabelo.

# Prenos parametrov v funkcije

- Kaj izpiše spodnji program?

```
bla(int x) {
  x=10;
}

main() {
  a=5;
  bla(a);
  izpisi(a);
}
```

# Prenos parametrov v funkcije

- V programskem jeziku C se VSI parametri v funkcije prenašajo po vrednosti!
- Vprašanje: Kako bi napisali funkcijo, ki zamenja vrednost svojih parametrov?
- Odgovor: V funkcijo bi prenesli vrednosti referenc parametrov (kazalce)!

# Prenos parametrov v funkcije

Kaj izpiše spodnji program? Popravi ga!

zamenjaj1.c

```c
#include <stdio.h>

void zamenjaj(int x, int y) {
  int t = x;
  x = y;
  y = t;
}

main() {
  int a=10;
  int b=5;

  printf("a=%d, b=%d\n", a, b);
  zamenjaj(a,b);
  printf("a=%d, b=%d\n", a, b);
}
```

# Prenos parametrov v funkcije

Program smo popravili v treh korakih:

1. spremenili smo glavo funkcije

```
zamenjaj(int x, int y)  -> zamenjaj(int *x, int *y)
```

2. popravili smo klic funkcije

```
zamenjaj(a,b);          ->   zamenjaj(&x, &y);
```

3. vsem pojavitvam parametrov v funkcijah smo dodali zvezdico (namesto `x` smo pisali `*x`)

# Tabela kot parameter

Videlo smo: Parameter se lahko znotraj funkcije spremeni le v primeru, da ga vanjo prenesemo kot kazalec!

```
// x se ne more spremeniti          // x se lahko spremeni
bla(x);                             bla(&x);
```

Ima tudi to pravilo kakšno izjemo?

# Tabela kot parameter

- Kaj izpiše spodnji program?

```c
#include <stdio.h>

void zamenjaj(int a[], int i, int j) {
  int t = a[i];
  a[i] = a[j];
  a[j] = t;
}

main() {
  int tab[5] = {0,1,2,3,4};

  // izpisemo drugi in cetrti element
  printf("tab[2]=%d, tab[4]=%d \n", tab[2], tab[4]);

  // zamenjamo vrednost?
  zamenjaj(tab, 2, 4);

  // izpisemo drugi in cetrti element
  printf("tab[2]=%d, tab[4]=%d \n", tab[2], tab[4]);
}
```

# Tabela kot parameter

```
// x se ne more spremeniti          // x se lahko spremeni
bla(x);                             bla(&x);
```

Ima torej to pravilo izjemo?

- Odgovor: NE! (tabela je pravzaprav kazalec!)

# Uporaba funkcije `scanf`

- Videli smo: funkcija `scanf` se uporablja takole:

```
// NAROBE!!!                    // PRAVILNO
scanf("%d", i);                 scanf("%d", &i);
```

- Ima morda to pravilo kakšno izjemo?

```
scanf("%s", niz);
```

- Tudi to NI izjema (niz je tabela znakov, torej kazalec na `char`)