

Naslov v pomnilniku

- Naslov spremenljivke pridobimo s pomočjo operatorja &

```
int a=1;  
int b=2;  
std::cout << &a << std::endl;  
std::cout << &b << std::endl;
```

```
0x7ffcdb0e87c  
0x7ffcdb0e878
```

$$0x7ffcdb0e87c - 0x7ffcdb0e878 = 4B$$

Kazalci

- Spremenljivka, ki kaže na neko lokacijo v pomnilniku
 - Tip spremenljivke: kazalec na spremenljivko nekega tipa (int *)
 - Vrednost kazalca je naslov v pomnilniku

```
int a=5;  
int* pa = &a;  
pa[0] = 1;  
// koliko je a?
```

```
int a=5;  
int* pa;  
pa = &a;  
*pa = 1;  
// koliko je a?
```

- Kazalci so tudi spremenljivke! Imajo svoj naslov.

```
int a=1;  
int *pa=&a;  
std::cout << a << std::endl;  
std::cout << pa[0] << std::endl;  
std::cout << *pa << std::endl;  
  
std::cout << &a << std::endl;  
std::cout << pa << std::endl;  
std::cout << &pa << std::endl;
```

1
1
1

0x7fff1f522f2c
0x7fff1f522f2c
0x7fff1f522f20

Funkcija za zamenjavo vsebine od a in b

```
void swap1(int &a, int &b){  
    int tmp=a;  
    a=b;  
    b=tmp;  
}
```

& - prenos po referenci

```
void swap2(int *a, int *b){  
    int tmp=*a;  
    *a=*b;  
    *b=tmp;  
}
```

```
int x=1;  
int y=2;  
swap1(x,y);  
swap2(&x,&y);
```

```
void swap3(int a, int b){  
    int tmp=a;  
    a=b;  
    b=tmp;  
}
```

Dinamična alokacija pomnilnika

- Spremenljivke se nahajajo na skladu (https://en.wikipedia.org/wiki/Call_stack ponavadi ima statično kapaciteto)
 - Prostor za spremenljivke se rezervira že pred zagonom programa → statična velikost polja
- Na kopici (heap, https://en.wikipedia.org/wiki/Memory_management#HEAP) lahko dinamično alociramo pomnilnik (med zagonom). Ne smemo ga pozabiti sprostiti (**delete**)!
 - Za vsak **new** je potrebno klicati **delete**
<http://www.cplusplus.com/reference/new/operator%20delete%5B%5D/>
 - <http://www.cplusplus.com/reference/new/operator%20delete/>

```
int a[10];  
int b = 5;
```

```
int x = 100;  
int *pa = new int[x];  
pa[0] = 5;  
pa[1] = 3;  
...  
delete []pa;
```

Primer uporabe kazalcev

Brez kazalcev

```
#include <string>
using namespace std;
struct Proizvajalec{
    string ime;
};
struct Avto{
    string ime;
    Proizvajalec proizvajalec;
};
int main(){
    Proizvajalec p;
    p.ime="Tesla";
    Avto a;
    a.ime="S";
    a.proizvajalec=p;
    Avto b;
    b.ime="3";
    b.proizvajalec=p;

    a.proizvajalec.ime="Nikola";
    // vrednost od b.proizvajalec.ime ?
    // vrednost od p.ime ?
    // a.proizvajalec je kopija od p!!!
}
```

Uporaba kazalcev

```
#include <string>
using namespace std;
struct Proizvajalec{
    string ime;
};
struct Avto{
    string ime;
    Proizvajalec *proizvajalec;
};
int main(){
    Proizvajalec* p=new Proizvajalec;
    p->ime="Tesla";
    Avto *a=new Avto;
    a->ime="S";
    a->proizvajalec=p;
    Avto *b=new Avto;
    b->ime="3";
    b->proizvajalec=p;

    a->proizvajalec->ime="Nikola";
    // vrednost od b->proizvajalec->ime ?
    // vrednost od p->ime ?
    // a->proizvajalec kaže na
    // enako mesto kot p!}
}
```

Preverite z razhroščevalnikom!