

# Programovanie 10

Pointers (smerník, odkaz, ukazovateľ, ...)

# Adresa

- Každá premenná, pole, funkcia, má adresu (kde sa nachádza v pamäti za behu):
- Premenné: adresu zistíme pomocou &
  - &premenna
- Polia, funkcie: adresu zistíme priamo identifikátorom (bez zátvoriek)
  - fun
  - pole

# Beztypový smerník

- Umožňuje ukladať adresy na neskoršie použitie:

```
void *ptr;
```

```
ptr = &i;
```

```
printf("Adresa i je: %p\n", ptr);
```

```
scanf("%i", ptr); //nacita na  
adresu, ulozenu v ptr
```

- nepodporuje iné operácie ako priradenie

# Typový smerník

- Umožňuje ukladať adresy, aritmetiku adres a sprístupnenie hodnôt na uloženej adrese pomocou sprístupňovacieho operátora \*

```
int i; int *ptr;
ptr = &i;
printf("Adresa i je: %p\n", ptr);
scanf("%i", ptr);
printf("Hodnota na adrese %p je %i\n", ptr,
*ptr);
*ptr = 5;
printf("Hodnota i je: %i\n", i); //malo by byt 5
```

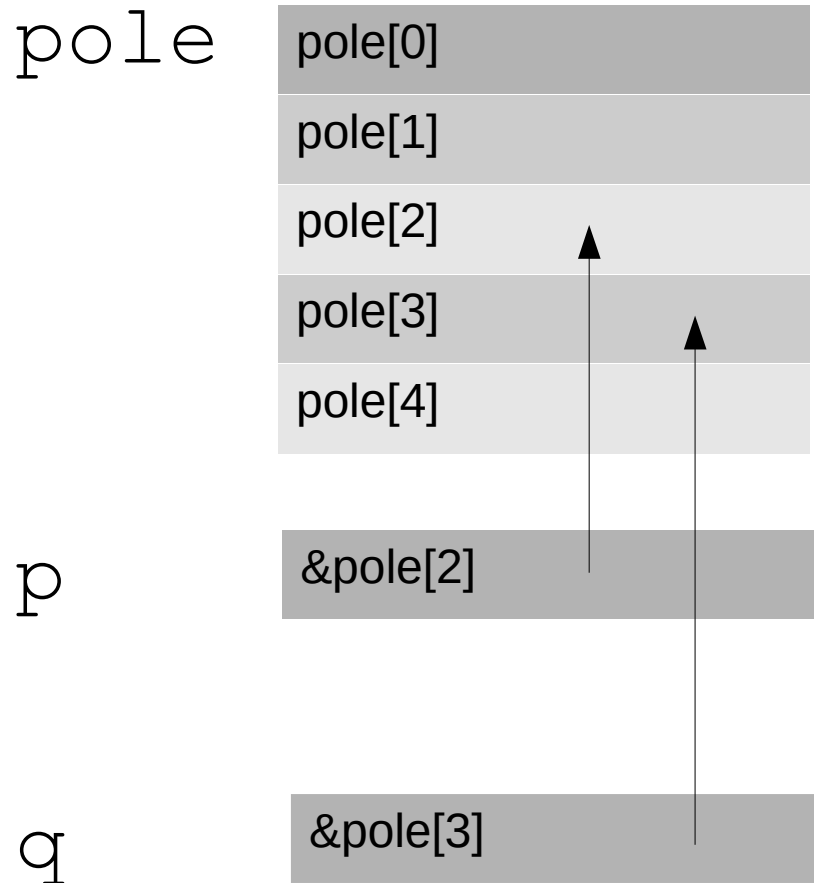
# Smerníková aritmetika

- Typový smerník podporuje operácie + a -
  - smerník +/- int: posunie sa o int pozícií v pamäti dopredu/dozadu
  - veľkosť „pozície“ je daná typom smerníka
  - smerník – smerník: ako vzdialené sú dané údaje (v počtoch pozícií)

# Smerníková aritmetika

```
int pole[5];  
//adresa stredu pola  
int *p = &pole[2];  
int *q= p-1;    //odkaz na pole[1]  
q = p+1;        //odkaz na pole[3]  
printf("%i\n", q-p); //vypise 1
```

# Smerníková aritmetika



# Smerníková aritmetika a operátor indexovania v jazyku C

`pole`        `===`        `&pole[0]`

`pole+1`     `===`        `&pole[1]`

`pole+i`     `===`        `&pole[i]`

`*pole`        `===`        `pole[0]`

`*(pole+i)`   `===`        `pole[i]`

//pozor na prioritu operatora:

`*pole+i`     `===`        `pole[0]+i`



# Smerník na funkciu

- Ukladá adresu funkcie, nedá sa robiť aritmetika, ale dá sa volať funkcia:

```
double (*) (double) fnptr;
```

```
fnptr = sin;
```

```
printf ("%lf\n", fnptr(0));
```

```
fnptr = cos;
```

```
printf ("%lf\n", fnptr(0));
```

# Konverzie smerníkov

- Podobne ako pri premenných: (ciel\_typ)
  - pozor na prioritu pri použití s operátorom \*

```
void fun(void*  adresa) {  
    printf("%0x\n", adresa);  
    printf("%0x\n", *((int*) adresa));  
    printf("%0x\n", *((char*) adresa));  
}
```