Pointerworkshop

Tentamen PM

- Opgave 1: sorteren van een (1D) array
- Opgave 2: stukje code evalueren, parameters
- Opgave 3: 2D array
- Opgave 4: pointers

Pointers

- Maken van een pointer:

```
int* p;
```

- Een int aanmaken

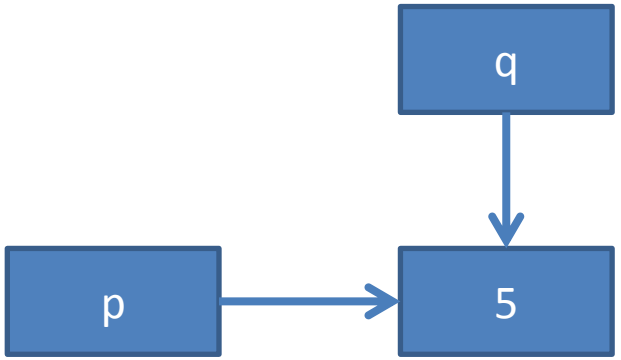
```
p = new int();
```

- Een waarde instellen

```
*p = 5;
```

Twee pointers

```
int* p, int* q;
p = new int();
*p = 5;
```



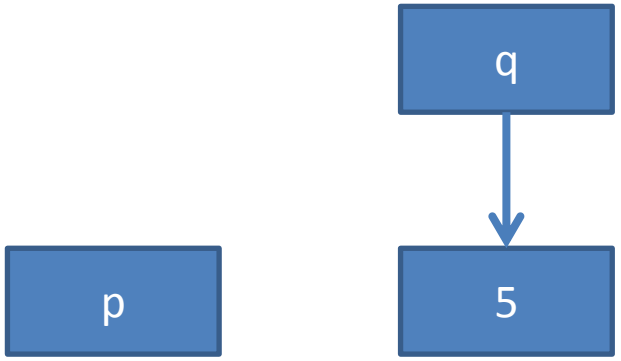
- Verander het adres van q:

```
q = p;
cout << *q;
```

Verwijderen

- P naar niets laten wijzen:

```
p = NULL;
```



- De int verwijderen:

```
delete q;
q = NULL;
```

Verwijderen

- P naar niets laten wijzen:

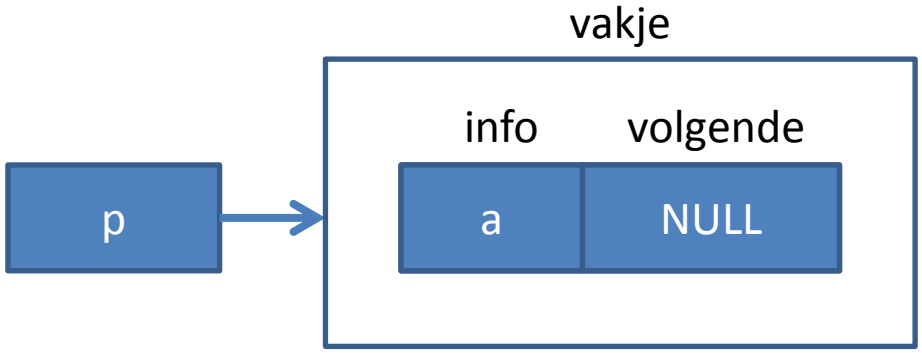
```
p = NULL;
```

- De int verwijderen:

```
delete q;
q = NULL;
```

Klasses en pointers

```
class vakje {
public:
char info;
vakje* volgende;
};//vakje
```



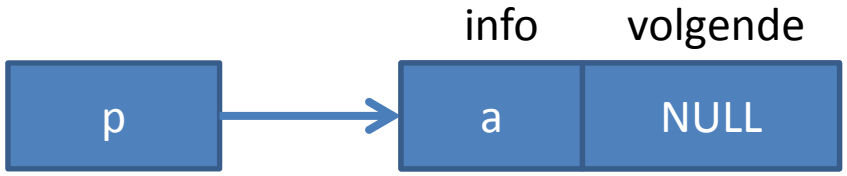

```
vakje* p = new vakje();
```

- p kan wijzen naar een vakje
- Info kan gewijzigd worden via p:

```
p->info = 'a';
p->volgende = NULL;
```

Klasses en pointers

```
class vakje {
public:
char info;
vakje* volgende;
};//vakje
```



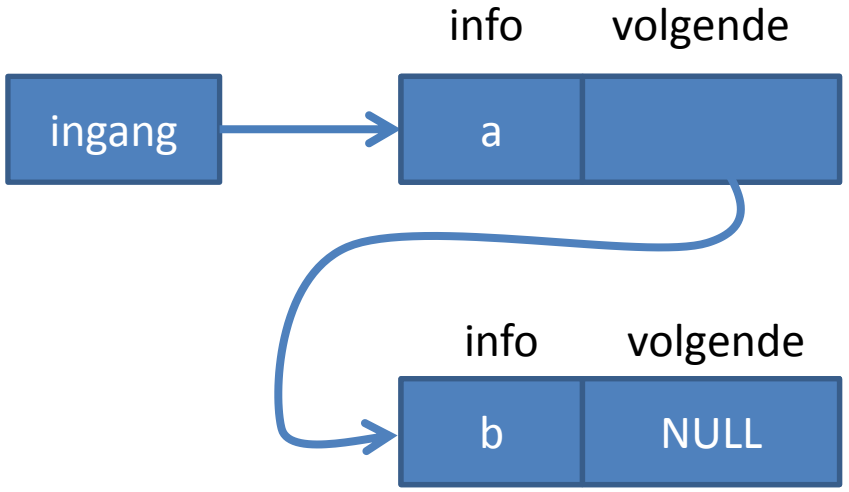

```
vakje* p = new vakje();
```

- p kan wijzen naar een vakje
- Info kan gewijzigd worden via p:

```
p->info = 'a';
p->volgende = NULL;
```

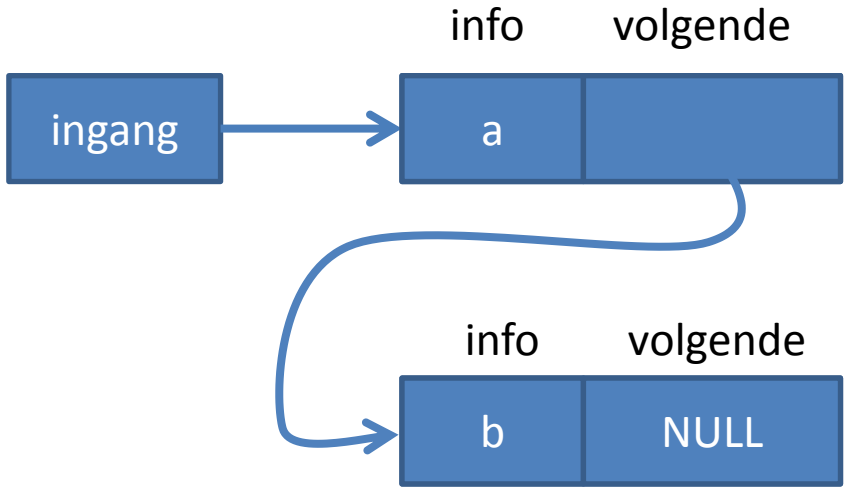
Vakje erbij achteraan

```
ingang->volgende = new vakje();
ingang->volgende->info = ‘b’;
ingang->volgende->volgende = NULL;
```



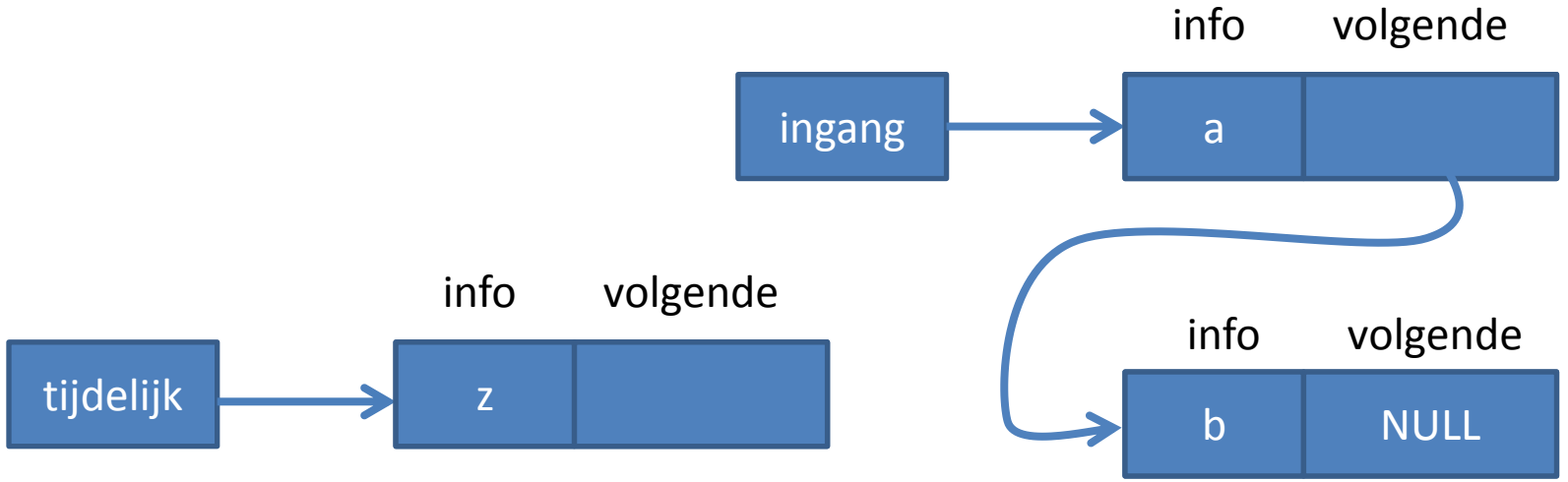
Pointerlijst afdrukken

```
vakje* hulp = ingang;
while ( hulp != NULL ) {
cout << hulp->info << endl;
hulp = hulp->volgende;
}//while
```



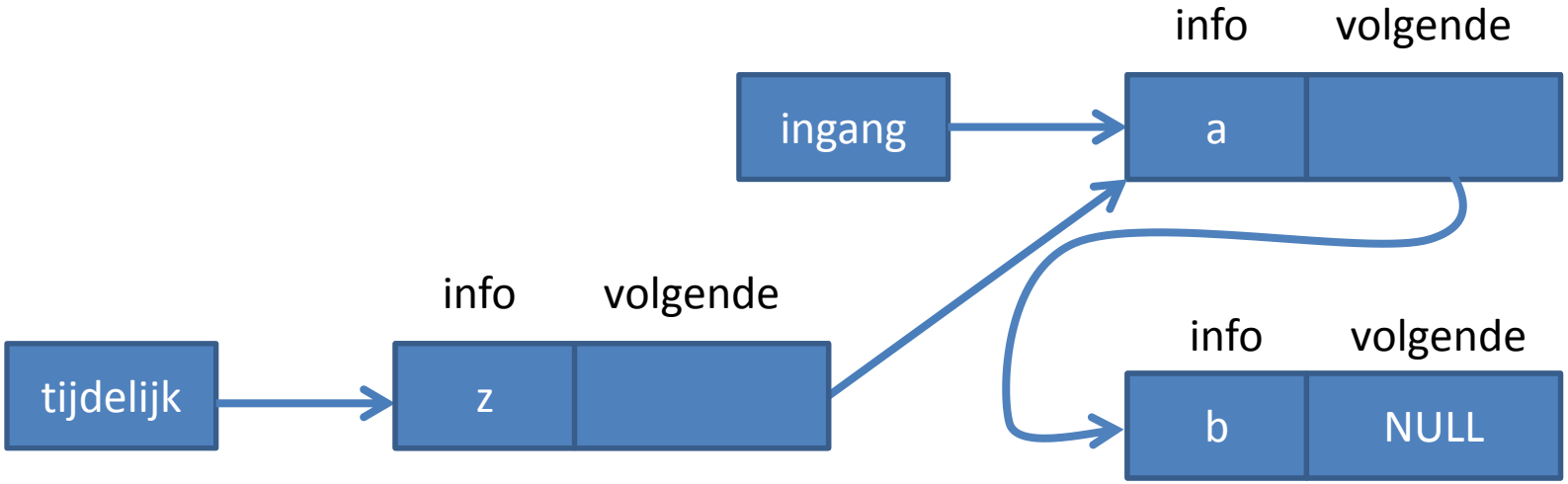
Vakje vooraan erbij

```
vakje* tijdelijk;
tijdelijk = new vakje;
tijdelijk->info = 'z';
tijdelijk->volgende = ingang;
ingang = tijdelijk;
```



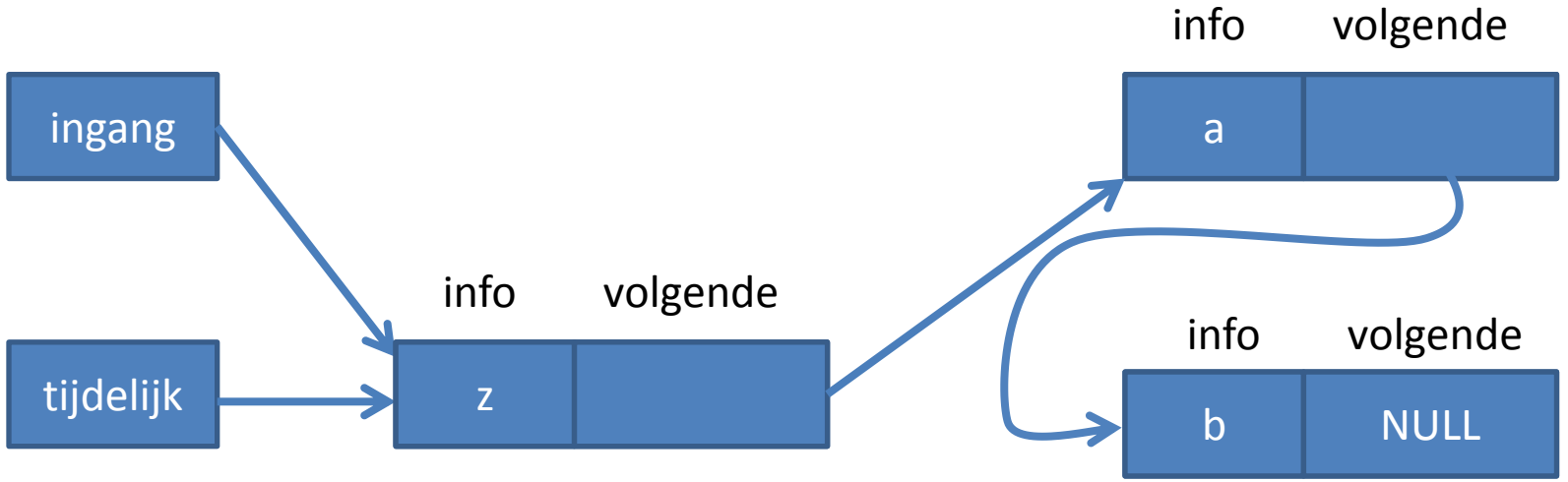
Vakje vooraan erbij

```
vakje* tijdelijk;
tijdelijk = new vakje;
tijdelijk->info = 'z';
tijdelijk->volgende = ingang;
ingang = tijdelijk;
```



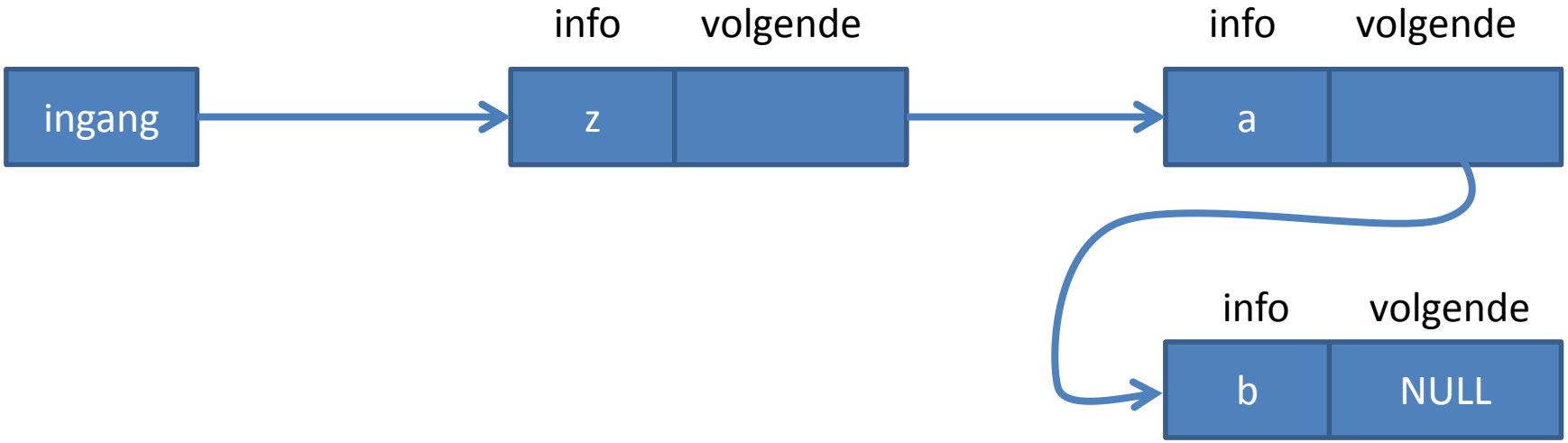
Vakje vooraan erbij

```
vakje* tijdelijk;
tijdelijk = new vakje;
tijdelijk->info = 'z';
tijdelijk->volgende = ingang;
ingang = tijdelijk;
```



Vakje vooraan erbij

```
vakje* tijdelijk;
tijdelijk = new vakje;
tijdelijk->info = 'z';
tijdelijk->volgende = ingang;
ingang = tijdelijk;
```



Functie

```
void plaatsvoor(vakje* & ingang) {
vakje* tijdelijk;
tijdelijk = new vakje;
tijdelijk->info = 'z';
tijdelijk->volgende = ingang;
ingang = tijdelijk;
}
```

- *Call by reference* (&) omdat de ingang aangepast moet worden.

Call by reference en call by value

- Call by value

  De functie krijgt de waarde binnen. Als de waarde van het variable binnen de functie aanroep verandert, blijft de waarde van het variable buiten de functie aanroep nog hetzelfde.

```
void pasAan(int a) {
  a++;
}

int a = 5;
pasAan(a);
cout << a; // a is 5
```

- Call by reference

  De functie krijgt een referentie naar het variable. Als het variable binnen de functie aanroep verandert, verandert dit ook de waarde van het variable buiten de functie aanroep.

```
void pasAan2(int & a) {
  a++;
}

int b = 5;
pasAan(b);
cout << b; // b is nu 6
```