Pointers

& = adresoperator

\* = dereferentieoperator

%p = hexadecimale pointer

# Algemeen

**Declaratie**

int g; // een gewone integer met als naam g

int \*pg // een pointer naar een integer met als naam pg

**Nullpointer**

* int \*pg = 0;
* int \*pg = NULL;

**Voidpointer**

Voidpointer kan niet gederefereerd worden.

# Pointers en arrays

double eenArray[5] = {3};

p = &eenArray[0] // pointer naar het eerste element van de array

q = eenArray// pointer naar het eerste element van de array

🡺 p == q

\*eenArray // De eerste waarde van de array

eenArray[0] // De eerste waarde van de array

🡺 \*eenArray == eenArray[0]

int \*a // pointer naar een integer

int \*\*b // pointer naar een pointer naar een integer

int \*x[5] // array van pointers naar integers

int (\*y)[10] // pointer naar array van integers

# Pointer naar const

const int \*pg

Je kan de waarde van de inhoud van de pointer niet veranderen.

De pointer zelf kan je wel veranderen.

# Bewerkingen op pointers

int t[] = {1, 2, 3, 4, 5};

int \*p = t;

printf("%d \n", \*(p++)); // 1

printf("%d \n", \*(p)); // 2

printf("%d \n", \*(p--)); // 2

printf("%d \n", \*(p)); // 1

**Let op:**

p++ ≠ (\*p)++ ≠ \*p++

int t[] = {1, 2, 3, 4, 5};

int \*q = &t[0];

int \*p = &t[4];

printf("%ld \n", p-q); // 4

printf("%ld \n", q-p); // -4

printf("%d \n", p < q); // 0 -> p staat voor q

printf("%d \n", p > q); // 1 -> q staat voor p

# Pointers als functieresultaat

Als je een return type met “const” hebt, is het beter om overal de “const” weg te doen.

# Constante pointer

int a = 1;

const int b = 1; // Read only

int const c = 1; // Read only

int \* const d = &a; // Variable is writeable, Pointer is const

const int \* const e = &a; // Variable is read only, Pointer is const

a++; // ok

\*d = a++; //ok

b++; // variable 'b' declared const here

c++; // variable 'c' declared const here

e++; // variable 'e' declared const here

d = &b // variable 'd' declared const here

e = &b; // variable 'e' declared const here

\*e = a++; // read-only variable is not assignable