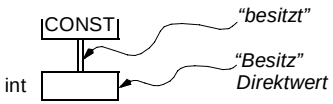
PARAMETERORGANISATION

DREI ARTEN DER DATENHALTUNG

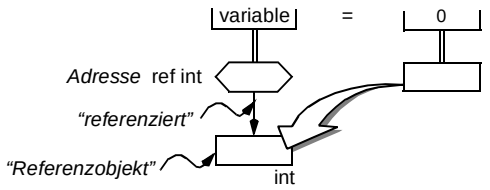
❑ **Konstante**

final int CONST = 10 ;



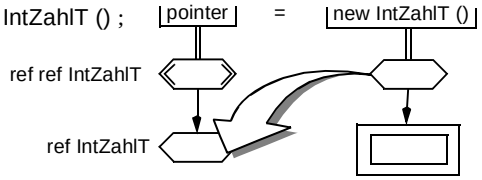
❑ **Variable**

int variable = 0 ;



❑ **Pointer** (Zeigervariable, Adreßvariable)

class IntZahlT { int i } ;
IntZahlT pointer = new IntZahlT () ;



Anm.:
- ein auf NULL gesetzter Pointer kann offensichtlich nicht (sinnvoll) entreferenziert werden;
- der Versuch des Entreferenzierens führt zu einem Laufzeitfehler.

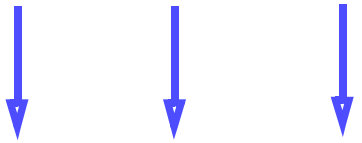
PARAMETERORGANISATION

DEMO

void test (int fpar1, int fpar2, IntZahlT fpar3) {

. . .

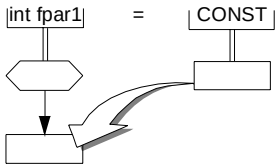
} // test

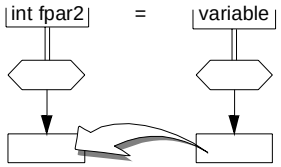


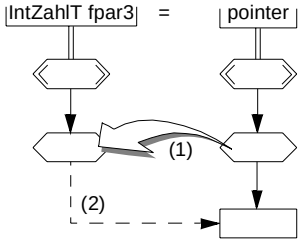
-->> test (CONST, variable, pointer)

Parameterarten, Sicht der Programmiersprache

Call by Value

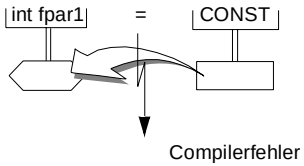


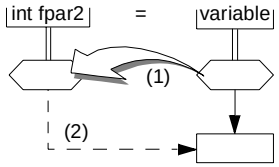


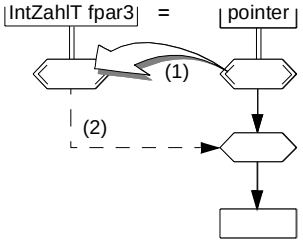


Ziel der "Zuweisung" ist jeweils das *Referenzobjekt* des formalen Parameters

Call by Reference





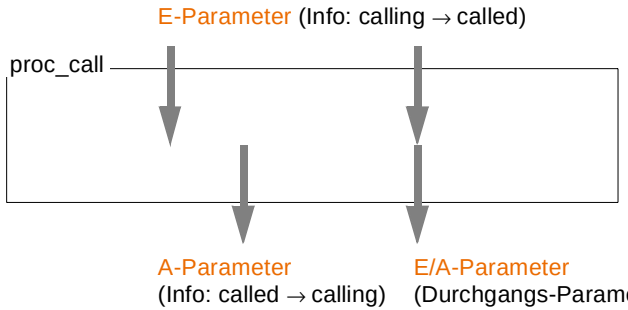


Ziel der "Zuweisung" ist jeweils der *Besitz* des formalen Parameters

Parameterarten, pragmatische Sicht

Hauptprogramm

.
.
.



Anm.:

- das Ergebnis einer Funktion ist ein (syntaktisch ausgezeichneter) A-Parameter;

Realisierung der pragmatischen Sicht mit Mitteln der PS

formaler Parameter	call by value	call by reference
Konstante	E	-
Variable	E	E / A
Pointer	E / A	E / A

Aufrufmechanismus einer Prozedur

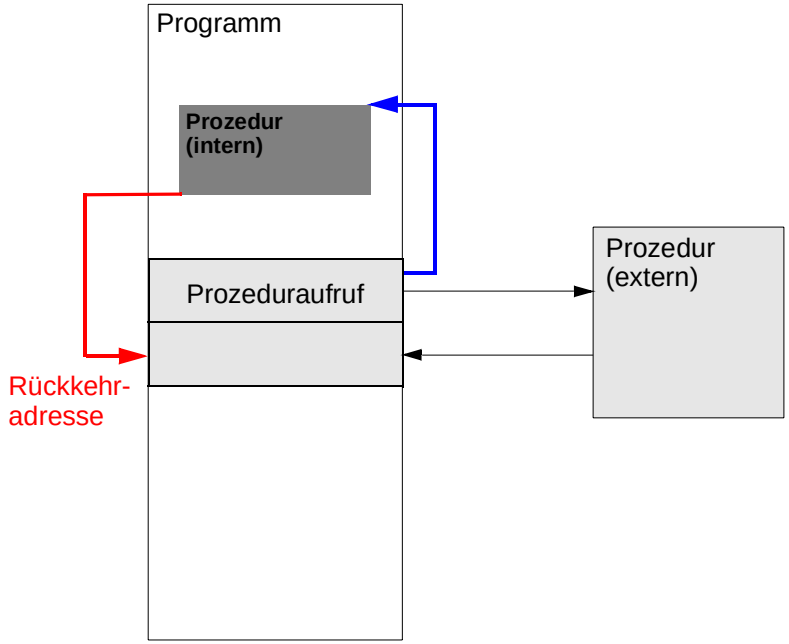


Abbildung eines Programms in den Hauptspeicher