① # Stapel als Datenstrukturen (Stack)

Implementierung als Vektor oder
als Liste (einfach verkettet)

LIFO - Last in First out

als Vektor mit $n > 0$ Elementen:

$s[0], s[1], \ldots, s[n-1]$ z.B. von Typ int

sp als Stapelindex ist immer Index des ersten freien Elements

$sp <= n-1$ muß beim Vektor immer gelten

Wenn $sp == 0$, dann Stapel leer, ist auch Anfangswert

int isEmpty() Stapel leer?

int isFull() Stapel voll?

void push(int wert) Neues Element oben auf den Stapel legen
(Voraussetzung: isFull() == 0 (false))

void pop() Entferne oberstes Element, sp--, falls Stapel nicht bereits leer

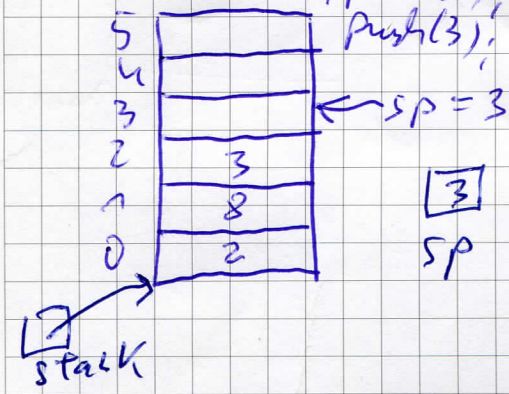
int top() Zeige oberstes Element, nicht entfernen

int poptop() liefert oberstes Element und entfernt es

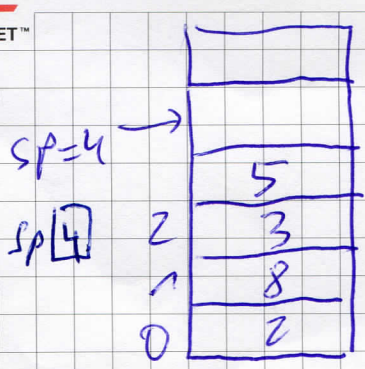
1a

stack_init(6)

push(2); push(8); push(3);

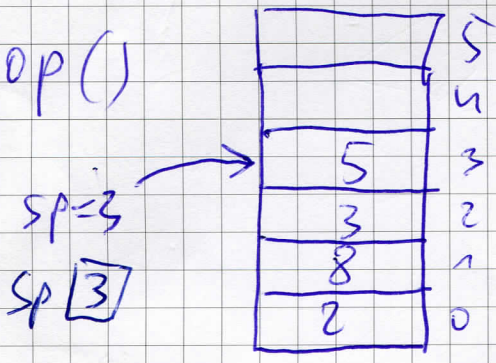


push(5)

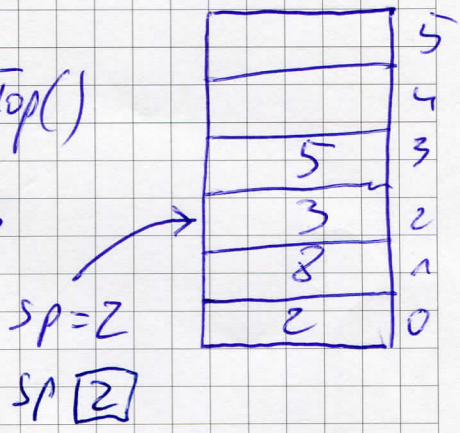


top()
↳ 5

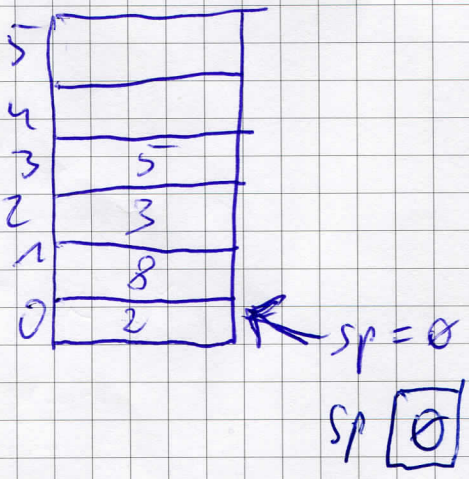
pop()



pop top()
↳ 3



pop top(); pop top()



isEmpty() ≠ 0

(2) # Stapel als Liste:

Anzahl der Elemente wird durch RAM begrenzt

Listenzeiger auf obersten Element:

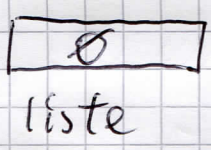
```
struct elem *liste = 0; // am Anfang
```

(ersetzt sp im Vergleich zu vector)

Elementtyp im Stapel für z.B. int-Werte:

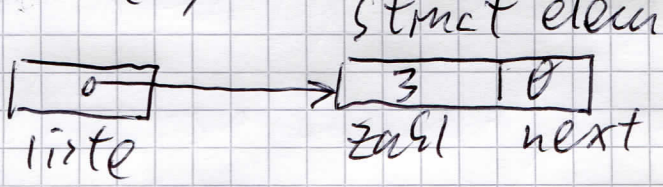
```
struct elem { int zahl; // Wert
              struct elem *next; // Verkettung
};
```

int isFull() ist immer falsch, restliche Funktionen werden über der Liste implementiert

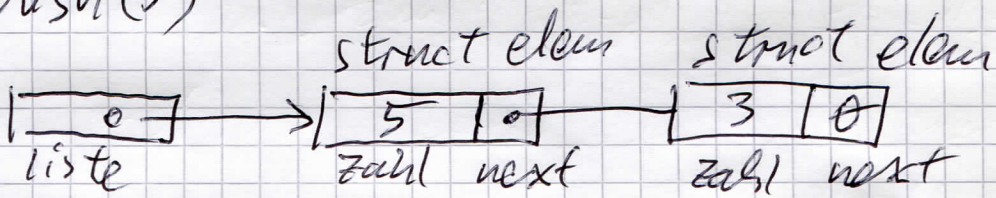


leerer Stapel

push(3)



push(5)



popTop() → 5