Stack

Daten werden gestapelt, der ältere Wert wird vom neuern verdeckt. Es kann immer ein Wert hinzugefügt werden, beim Entfernen wird der neueste Wert entfernt. First In Last Out (FILO). Diese Vorgänge sind sehr schnell und sind generell sicher.

Schritt	0x01	0x02	0x03	0x04	0x05	…	Eingabe	Ausgabe
0 – Voher	Null						-	-
1 – Allocation	\0	\0	\0	\0	\0	…	-	-
2 – Push	A	\0	\0	\0	\0	…	A	-
3 – Push	A	B	\0	\0	\0	…	B	-
4 – Push	A	B	C	\0	\0	…	C	-
5 – Pull	A	B	\0	\0	\0	…	-	C
6 – Pull	A	\0	\0	\0	\0	…	-	B
7 – Pull	Null						-	A

Queue

Daten werden in einer Liste gespeichert, der älteste Wert wird hier entnommen. First in First out (FIFO). Da Daten von Vorne entnommen werden entsteht ungenutzter Speicher. Hier ist zu Achten, dass dieser Speicher möglichst freigegeben wird, natürlich ist eine umbauen des Speichers bei jedem Zugriff nicht unbedingt Sinnvoll.

Schritt	0x01	0x02	0x03	0x04	0x05	…	Eingabe	Ausgabe
0 – Voher	Null						-	-
1 – Allocation	\0	\0	\0	\0	\0	…	-	-
2 – Push	A	\0	\0	\0	\0	…	A	-
3 – Push	A	B	\0	\0	\0	…	B	-
4 – Push	A	B	C	\0	\0	…	C	-
5 – Pull	A	B	C	\0	\0	…	-	A
6 – Pull	A	B	C	\0	\0	…	-	B
7 – Pull	Null						-	C

LinkedList

Daten werden in Ketten-Elementen gespeichert. Jedes zwischen Element kennt seinen nächsten Nachbarn. Durch diese Kette kann man jedes Element ansprechen. Das Letze Element hat immer einen Null Wert, da dieser der Letze Wert ist und keinen nächsten Wert besitzt.

Schritt	Nr.1	Nr.2	Nr.3	Nr.4	Nr.5	…	Eingabe	Ausgabe
1	Null						-	-
	Null					…		
2 - Push	A					…	A	-
	Null					…		
3 - Push	A					…	B	-
	->	B				…		
		Null				…		
4 - Push	A					…	C	-
	->	B				…		
		->	C			…		
			Null			…		
5 – Pull(1)	A					…	-	B
	->	C				…		
		Null				…		