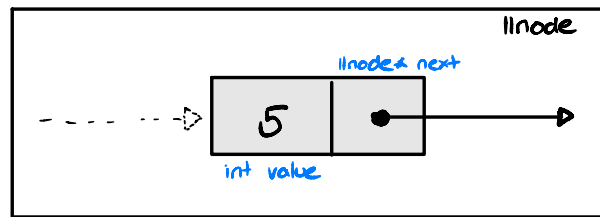


14 Dezember

Verkettete Listen: Ein Element einer verketteten Liste ist wie folgt aufgebaut:



wobei wir `llnode` als Typ in folgendem struct definieren

```
struct llnode {
```

Wert von Node → `int value;`

Zeiger zum nächsten Node → `llnode* next;`

Konstruktor → `llnode (int v, llnode* n): value(v), next(n) { }`  
`};`

Wie ist nun eine linked list definiert?

```
class llnec {
```

```
private:
```

```
    llnode* head;
```

```
public:
```

Konstruktor / Memberfunktionen etc

```
};
```

Der Zeiger "head" zeigt dabei auf den ersten Node.

Das letzte Element zeigt zum nullptr.

Beispiele: siehe Vorlesungsfolien (Lecture 12)

**delete**: Wir wollen nun unserem `llvec` auch die Funktion geben, ein Element (`llnode`) zu löschen. Dies machen wir mit dem **delete** Befehl. **delete** wird auch benutzt um einen temporären Speicherplatz wieder freizugeben. Wenn wir also mit **new** einen Speicherplatz anlegen, müssen explizit gelöscht werden.

2. Fälle:  $p = \text{new Typ}; \Rightarrow \text{delete } p;$   
 $p = \text{new Typ}[]; \Rightarrow \text{delete} [] p;$

**Destruktor**: Wird automatisch aufgerufen, falls der Gültigkeitsbereich eines Objektes vom Typ `T` endet. Deklaration:  
 $\sim T()$  (in Klasse (public))

**Copy - Konstruktor**: Kopiert alle Daten des Typs `T`.

Deklaration:  
 $T (\text{const } T\& x)$

**Zuweisungsoperator**: Gleich wie Copy-Konstruktor, wobei Element schon existiert (operator =)

Bemerkung: weitere Vorteile vom Zuweisungsoperator findet man in den Vorlesungsfolien.