```
> class ADS_set;
```

`ADS_set` ist ein Container, der Elemente speichert, die über einen Schlüssel identifiziert werden. `Key` ist der Typ des Schlüssels und zugleich auch der Typ des Elements. D.h., das gesamte gespeicherte Element dient als Schlüssel. `ADS_set` stellt eine Vereinfachung und Verallgemeinerung der STL-Typen «std::set» und «std::unordered_set» dar.

Jeder Schlüssel kann in einem Container höchstens ein Mal auftreten (Mengeneigenschaft). Es kann also in keinem Container zwei Elemente mit demselben Schlüssel geben. Soweit nicht bei einzelnen Methoden/Funktionen anders festgelegt, wird das versuchte Einfügen von bereits vorhandenen Werten stillschweigend ignoriert. Dasselbe gilt, wenn versucht wird, nicht vorhandene Werte zu löschen.

Im Container werden grundsätzlich immer Kopien der an die entsprechenden Methoden übergebenen Elemente gespeichert. Der Container ist in der Folge für die ordnungsgemäße Zerstörung dieser Elemente verantwortlich.

Zur Realisierung ist eine der folgenden Datenstrukturen heranzuziehen:

- B+-Baum (60 Punkte)[17]
- Extendible Hashing (50 Punkte)
- Linear Hashing (50 Punkte)
- Statische Hashtabelle mit Coalesced Chaining (30 Punkte)
- Statische Hashtabelle mit Separate Chaining (30 Punkte)

Die Datenstrukturen sind entsprechend den Spezifikationen aus dem Vortrag (siehe Folien) zu implementieren. Abweichungen bitte vorab mit der LV-Leitung (im Forum) klären.

Der Vollständigkeit halber wird darauf hingewiesen, dass die Datenstrukturen und Algorithmen selbst zu implementieren sind und nicht auf vorhandene Implementierungen (auch nicht in Standard-Bibliotheken) zurückgegriffen werden darf. Für die Realisierung der Datenstruktur sind auch keine Datentypen aus der STL zulässig. Wenn Sie verfügbare Datenstrukturen aus der STL als temporäre **Hilfs**datenstrukturen einsetzen wollen, dann fragen Sie bitte im Zweifelsfall vorher bei der LV-Leitung nach (im Forum).

Die Datenstrukturen müssen - von Systembeschränkungen abgesehen - prinzipiell beliebig viele Werte aufnehmen können. Wenn bei Datenstrukturen mit fixer Größe (zB. statische Hashverfahren) kein Platz mehr vorhanden ist oder die vordefinierte maximale Belegung überschritten wird, so ist die Datenstruktur entsprechend zu vergrößern. Bei statischen Hashverfahren also zB.: neue (größere) Tabelle anlegen und Werte neu hashen. Die Datenstruktur soll mit der Anzahl der gespeicherten Werte "wachsen".

Damit Elemente des Typs `Key` in einem `ADS_set` gespeichert werden können, muss `Key` die folgenden Eigenschaften haben:

- Erzeugung von Objekten ohne Initialisierung ist möglich (bei Klassen: Default-Konstruktor). Achtung: Wenn der Elementdatentyp ein primitiver Datentyp (keine Klasse) ist, dann sind nicht initialisierte Elemente unter Umständen in einem nicht definierten Zustand. Die *Verwendung* nicht initialisierter Elemente (zB in Vergleichen) ist daher verboten.[16]
- Erzeugung von Objekten mit Initialisierung und Übergabe *by value* ist möglich (bei Klassen: Kopier-Konstruktor)
- Zuweisung ist möglich (bei Klassen: Kopierzuweisungsoperator)
- Vergleich mittels `std::less<Key>` (siehe «std::less») ist möglich (Ordnungsrelation für B+-Baum).
- Vergleich mittels `std::equal_to<Key>` (siehe «std::equal_to») ist möglich (Äquivalenzrelation für Hashverfahren)[3].
- Berechnung eines Hashwertes mittels `std::hash<Key>` (siehe «std::hash») ist möglich (für Hashverfahren).

Darüber hinausgehende Eigenschaften von `Key` dürfen nicht vorausgesetzt werden. Andere als die oben genannten Operationen sind für die im Container gespeicherten Elemente vom Typ `Key` daher nicht zulässig!

`N` ist ein datenstrukturspezifischer Templateparameter[1][4], konkret

- **k** (*Ordnung*) beim B+-Baum
- die **Anfangsgröße der Hashtabelle** bei statischen Hashverfahren (Separate Chaining, Coalesced Chaining).
- die **Bucketgröße** bei dynamischen Hashverfahren (Extendible Hashing, Linear Hashing)

Weitere datenstrukturspezifische Parameter können bei Bedarf als Template-Parameter oder als Konstruktor-Parameter realisiert werden. Es muss für alle Template-Parameter (mit Ausnahme von `Key`) Defaultwerte geben, die für alle Parameter sinnvolle Werte vorgeben. Es empfiehlt sich unbedingt, die Datenstruktur mit unterschiedlichen Parameterwerten zu