IAM – Integer Array Model

*IAM – Integer Array Model* beschreibt das binäre Datenformat zur auslagerung von Listen und Abbildungen in eine Datei. Ziel dieses Formats ist es, entsprechende Dateien per *File-Mapping* in den Arbeitsspeicher abzubilden und daruf sehr effiziente Lese- und Suchoperationen ausführen zu können. Die Modifikation ist nicht vorgesehen.

# Notation

Die folgenden Datenstrukturen werden jeweils als Auflistung ihrer Datenfelder vorgestellt, wobei zu jedem der Felder dessen Datenformat, Anzahl und Beschreibung angegeben sind. Virtuelle Datenfelder, die nicht explizit in der Datei gespeichert werden, werden ohne Format und Anzahl dargestellt. Das Datenformat ***INT*** steht für eine ganze Zahl mit 4 Byte Länge.

# Datenstruktur: IAM\_Index

Kodiert eine Zusammenstellung beliebeg vieler Listen und Abbildungen.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| ID | INT | 1 | Kennzeichnung von Datenstruktur und Bytereihenfolge.  Ist 0x600DF00D. |
| mapCount | INT | 1 | Anzahl der Abbildungen.  Ist größer oder gleich 0. |
| listCount | INT | 1 | Anzahl der Listen.  Ist größer oder gleich 0. |
| mapArray | IAM\_Map | mapCount | Auflistung aller Abbildungen. |
| listArray | IAM\_List | listCount | Auflistung aller Listen. |

# Datenstruktur: IAM\_List

Kodiert eine Liste von Elementen, bei welcher jedes Element eine Liste von Zahlen (INT) ist.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| itemCount | INT | 1 | Anzahl der Elemente.  Ist größer oder gleich 0. |
| rangeArray | INT | rangeCount+1 | Startpositionen der Elemente im Speicherbereich valueArray.  Das i-ten Element beginnt dort an Positionen rangeArray[i] und hat eine Läge von rangeArray[i+1]‑rangeArray[i].  Die Startposition rangeArray[0] ist 0. |
| valueArray | INT | rangeArray[itemCount] | Speicherbereich mit den Zahlen der Elemente. Die erste Zahl jedes Elements kann als Strukturkennung zur Interpretation der restlichen Zahlen des Elements dienen. |

# Datenstruktur: IAM\_Map

Kodiert eine Streuwert-Abbildung von Schlüsseln auf Werte, bei welcher jeder Schlüssel und jeder Wert eine Liste von Zahlen (INT) ist. Die Berechnung des Streuwerts zu einem gesuchten Schlüssel erfolgt über den folgenden Algorithmus:

**int** hash(**int**[] key, **int** keySize) {

**int** hash = 0x811C9DC5;

**for**(**int** **i** = 0; i < keySize; i++) hash = (hash \* 0x01000193) ^ key[i];

**return** hash;

}

Der Streuwert des gesuchten Schlüssels wird über eine Bitmaske in den Index des einzigen Schlüsselbereichs überführt, in dem der gesuchte Schlüssel enthalten sein kann. Die Bitmaske wird über folgenden Algorithmus ermittelt:

**int** mask(**int** entryCount) {

**int** mask = 1;

**while**(mask < entryCount) mask <<= 1;

**return** mask - 1;

}

Der Index des Schlüsselbereichs zu einem gegebenen Schlüssel ergibt sich damit über folgenden Algorithmus:

**int** index(**int**[] key, **int** keySize, **int** entryCount) {

**return** hash(key, keySize) & mask(entryCount);

}

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| keySize | INT | 1 | Anzahl der Zahlen in einem Schlüssel.  Ist größer oder gleich 1. |
| valueSize | INT | 1 | Anzahl der Zahlen in einem Wert.  Ist größer oder gleich 0. |
| entryCount | INT | 1 | Anzahl der Einträge (Schlüssel-Wert-Paare) in der Abbildung.  Ist größer oder gleich 0. |
| rangeMask | - | - | Bitmaske zur Umrechnung des Streuwerts eines gesuchten Schlüssels in den Index des einzigen Schlüsselbereichs, in dem dieser Schlüssel enthalten sein kann.  Ergibt sich aus entryCount. |
| rangeArray | INT | rangeMask+2 | Startpositionen der Schlüsselbereiche im Speicherbereich keyArray.  Der i-te Bereich beginnt dort an Positionen rangeArray[i] und hat eine Läge von rangeArray[i+1]‑rangeArray[i].  Die Startposition rangeArray[0] ist 0. |
| keyArray | INT | keySize \* entryCount | Speicherbereich mit den Zahlen der Schlüssel.  Der i-te Schlüssel beginnt hier an Positionen keySize \* i und hat eine Läge von keySize. |
| valueArray | INT | valueSize \* entryCount | Speicherbereich mit den Zahlen der Werte.  Der i-te Wert beginnt hier an Positionen valueSize \* i und hat eine Läge von valueSize. |