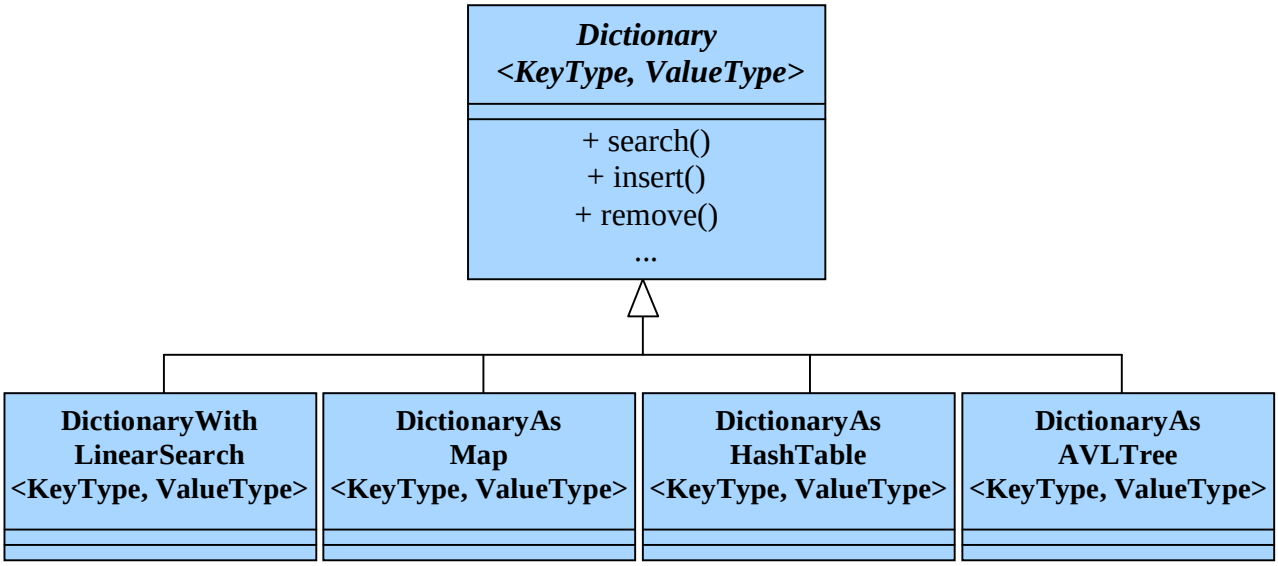
Dictionary

Eine Dictionary-Klasse unterstützt die Verwaltung von Datensätzen. Typische Operationen sind

- Suchen
- Einfügen und
- Löschen

von Datensätzen. Ein Datensatz besteht aus einem Schlüssel (Key) und Nutzdaten (Value), wobei der Schlüssel eindeutig sein soll. Es gibt zu einem Schlüssel also höchstens einen Datensatz.

Die Schnittstelle einer Dictionary-Klasse soll als abstrakte Basisklasse definiert werden. Die Schnittstelle ist parametrisiert mit dem Schlüsseltyp *KeyType* und dem Nutzdatentyp *ValueType* (template-Mechanismus). Durch Schnittstellenvererbung sollen Klassen mit unterschiedlichen Datenstrukturen und Algorithmen realisiert werden.



In einer Personaldaten- und einer Wörterbuchanwendung soll die Funktionalität der einzelnen Klassen getestet werden. Mit einem größeren Wörterbuch soll außerdem die Performance der Klassen durch Laufzeitmessungen untersucht werden.

Im Einzelnen sind die folgenden Programmteile zu erstellen.

1. Dictionary-Schnittstelle (Dictionary.h)

Es kann davon ausgegangen, dass die Nutzdaten eines Datensatzes sehr groß werden können und daher das Kopieren von Nutzdaten sowohl an der Schnittstelle als auch in der Implementierung von Dictionaries vermieden werden soll. Es wird daher mittels Referenzen der direkte Zugriff auf die Nutzdaten gestattet.

Die Schnittstelle besteht aus den folgenden Operationen:

- bool search(const KeyType& k);

  Sucht den Schlüssel k und liefert true zurück, falls der Schlüssel gefunden wird und sonst false. Falls der Schlüssel vorhanden ist, merkt sich das Dictionary die Speicheradresse des Datensatzes, um dann mit einem get-Aufruf auf die Nutzdaten zugreifen zu können.

- bool insert(const Key& k);

  Fügt einen neuen Schlüssel k ein. Falls bereits ein Datensatz mit diesem Schlüssel existiert, wird nicht eingefügt und false zurückgeliefert und ansonsten true. Dictionary merkt sich in jedem Fall die Speicheradresse des Datensatzes, um dann über einen get-Aufruf die Nutzdaten schreiben bzw. ändern zu können.

- ValueType& get();

  Liefert eine Referenz auf die Nutzdaten des aktuellen Datensatzes zurück. Damit können die Nutzdaten gelesen, gesetzt und verändert werden. Vor einem get-Aufruf muss ein insert-Aufruf bzw. ein erfolgreicher search-Aufruf (d.h. mit Rückgabewert true) erfolgt sein. Ist kein aktueller Datensatz vorhanden, erscheint eine Fehlerausgabe an der Konsole.

- bool remove(const Key& k);

  Löscht den Datensatz mit Schlüssel k. Falls der Datensatz gelöscht werden konnte (d.h. ein Datensatz mit diesem Schlüssel war vorhanden) wird true und ansonsten false zurückgeliefert.

- int getNumber()

  Liefert die Anzahl der gespeicherten Datensätze.

- void print()

  Gibt alle Datensätze aus.

Folgendes Beispiel illustriert die Funktionsweise der Methoden. Es wird eine Klasse Person vorausgesetzt, die get- und set-Methoden anbietet, um den Familiennamen, den Vornamen bzw. das Geburtsjahr einer Person lesen bzw. schreiben zu können.

```
int main()
{
   Dictionary<int,Person>* persDat;

   persDat = new DictionaryWithLinearSearch<int,Person>;

   if (persDat->insert(12))
   {
       persDat->get().setName("Maier");
       persDat->get().setVorname("Ute");
       persDat->get().setGeb(1982);
   }
   if (personal->insert(21))
   {
       persDat->get().setName("Mueller");
       persDat->get().setVorname("Peter");
       persDat->get().setGeb(1980);
   }

   persDat->print();

   if (persDat->search(12))
      persDat->get().setName("Maier-Mueller");

   persDat->print();

   delete persDat;
   return 0;
}
```

Die Datensätze der Personaldaten persDat besteht aus einem int-Wert als Schlüssel und den oben beschreiben Personendaten als Nutzdaten.

Als Implementierung wird eine Datenstruktur mit linearer Suche gewählt.

Füge einen Datensatz mit Schlüssel 12 ein.

Mit persDat->get() erhält man eine Referenz auf ein Person-Objekt. Damit lassen sich die Nutzdaten in das Dictionary schreiben.

Es wird ausgegeben:
12: Maier, Ute; geb. in 1982
21: Mueller, Peter; geb. in 1980

So lässt sich der Familienname mit Schlüssel 12 in „Maier-Mueller“ ändern.

Es wird ausgegeben:
12: Maier-Mueller, Ute; geb. in 1982
21: Mueller; geb. in 1980

Es sei nochmals betont, dass in der oben beschriebenen Schnittstelle als auch in den weiter unten betrachteten Implementierungen kein Kopieren und Zuweisen der Nutzdaten notwendig ist. D.h. es wird für ValueType kein Kopierkonstruktor und kein Zuweisungsoperator gefordert. Die Nutzdaten werden nur über Referenzen gelesen bzw. beschrieben. Man spricht in diesem Zusammenhang auch von einem **Referenz-Container**. Im Gegensatz dazu wurde in der Vorlesung Dictionaries besprochen, die einen **Werte-Container** darstellen: die Nutzdaten werden kopiert und daher wird ein Zuweisungsoperator benötigt.

Schreiben Sie die oben beschriebene Schnittstelle als abstrakte Klassenschablone und halten Sie außerdem als Kommentar folgende Anforderungen an die Schablonenparameter fest:

KeyType:

Da in Dictionaries die Daten häufig nach dem Schlüssel sortiert sind, müssen für KeyType die Vergleichsoperatoren operator< und operator== definiert sein. Außerdem wird ein Ausgabeoperator operator<< erwartet. Falls der Schlüsseltyp beispielsweise int oder string ist, muss also nichts beachtet werden.

ValueType:

Für die Nutzdaten ist lediglich ein Ausgabeoperator operator<< vorzusehen. Ein Kopierkonstruktor oder Zuweisungsoperator wird nicht benötigt.

2. Klasse DictionaryWithLinearSearch (DictionaryWithLinearSearch.h und .cpp)

a) Diese Klasse verwaltet alle Datensätze in einem dynamischen Feld, wobei jedes Feldelement jeweils aus einem Schlüssel und einem Zeiger auf die Nutzdaten besteht. Die Feldgröße wird bei Bedarf (falls Feld gefüllt ist) verdoppelt. Für die Suche nach einem Schlüssel wird einfache lineare Suche verwendet.

b) Testen Sie Ihre Klasse mit dem oben angegebenen Hauptprogramm, das jedoch für eine ausreichende Testabdeckung noch um einige Methodenaufrufe erweitert werden sollte. Das Hauptprogramm und die darin benutzte Klasse Person finden Sie auf der Homepage.

c) Testen Sie Ihre Klasse nun mit einer kleinen Wörterbuchanwendung. Ein Wörterbuch besteht aus eine Menge von Einträgen, wobei sich jeder Eintrag aus einem deutschen Wort als Schlüssel und einer Liste von englischen Wörtern als Nutzdaten zusammensetzt. Für die Liste der englischen Wörter kann der STL-Container set verwendet werden. Falls Wörter durch den Datentyp string dargestellt werden, lässt sich ein Wörterbuch wie unten dargestellt definieren. Ein Hauptprogramm mir einer kleinen Wörterbuchanwendung finden Sie auf der Homepage. Fügen Sie zu Testzwecken noch einige Methodenaufrufe dazu.

```
typedef string Deutsch;
typedef set<string> Englisch;

Dictionary< Deutsch, Englisch >* woerterBuch;

woerterBuch = new DictionaryWithLinearSearch< Deutsch, Englisch >;
```

3. Klasse DictionaryAsMap (DictionaryAsMap.h und .cpp)

Diese Klasse implementiert ein Dictionary mit dem STL-Container map.

4. Klasse DictionaryAsHashTable (DictionaryAsHashTable.h und .cpp)

Für die Implementierung wird eine Hashtabelle mit Verkettung der Datensätze mit gleichem Schlüssel eingesetzt. Definieren Sie eine Hashfunktion für int-Schlüssel und eine Hashfunktion für string-Schlüssel. Es ist zu beachten, dass bei anderen Schlüsseltypen weitere Hash-Funktionen geschrieben werden müssen.

Es genügt, wenn die gewünschte Feldgröße im Konstruktor angegeben wird. Beachten Sie, dass die tatsächliche Feldgröße gegebenenfalls auf eine Primzahl vergrößert werden sollte.

Freiwillig: Dynamisches Hash-Vefahren:

Die Feldgröße wird nicht beim Konstruktor angegeben. Stattdessen wird mit einer Feldgröße von initialSize ≈ 1000 Elementen gestartet. Das Feld wird dann bei Bedarf (falls das Feld gefüllt ist) in etwa verdoppelt. Dabei ist ein Umkopieren der Hashtabelle notwendig.

5. Klasse DictionaryAsAVLTree (DictionaryAsAVLTree.h und .cpp)

Für die Implementierung wird ein ausgeglichener, binärer Suchbaum (AVL-Baum) verwendet. Verwenden Sie als Ausgangspunkt den auf der Homepage (→Programmquellen) vorhandenen Programm-Code für einen binären Suchbaum. Passen Sie die Schnittstellen der Methoden an und erweitern Sie die Methoden insertR und removeR um die in der Vorlesung besprochenen Balancieroperationen

6. Performance-Untersuchung

Auf meiner Homepage finden Sie eine Wörterbuch-Datei mit einer Folge von n ≈ 16000 Deutsch-Englisch-Wortpaaren. Messen Sie die CPU-Zeiten (in sec) für die implementierten Klassen in vier Anwendungsfällen und tragen Sie die Zeiten in folgende Tabelle ein:

	DictionaryWithLinearSearch	**DictionaryAsMap**	**DictionaryAsHashTable**	**DictionaryAsAVLTree**
Aufbau eines Dictionary: alle n Wortpaare in ein leeres Dictionary eintragen; d.h. n insert-Aufrufe.				
Erfolgreiche Suche: alle n deutschen Wörter im Dictionary nachschlagen; d.h. n search-Aufrufe mit Rückgabewert true.				
Nicht erfolgreiche Suche: alle n englischen Wörter im Dictionary nachschlagen [1)]; d.h. n search-Aufrufe mit Rückgabewert false.				
Abbau eines Dictonary: alle n Wortpaare löschen; d.h. n remove-Aufrufe				

[1)] Beachten Sie, dass beim Nachschlagen eines englischen Wortes in einem Deutsch-Englisch-Wörterbuch dieses üblicherweise nicht vorkommt. Ausnahmen sind Wörter, die es sowohl im Englischen wie im Deutschen gibt (nicht notwendigerweise mit der gleichen Bedeutung); Beispiele: an, die, Kindergarten, gut, etc. Die Ausnahmen fallen jedoch aufgrund ihrer geringen Anzahl nicht ins Gewicht.