Wegpunkte aus Flugfunk-Wortsequenzen automatisch extrahieren

Max 5 Punkte für die erste Aufgabe für die Klausur.

Was bedeutet automatische Semantik – Interpretation?

Wortsequenz:

austrian five nine eight zulu speed three hundred knots

direct whisky whisky nine eight one

Semantik:

AUA598Z	SPEED	300	kt
AUA598Z	DIRECT_TO	WW981	none

Übungen, die ca. 30% der Klausurnote ausmachen

Aufgaben	Inhalt
Berechnung der Levenshtein-Distanz (05.10.2025)	Die Abgabe der Lüsungen ist für Freitag, den 17.10.2025, vorgesehen. Denken Sie an die Erstellung von au

Alte Klausuren von mir mit Lösungen seit 2001

Nützliche Hilfen

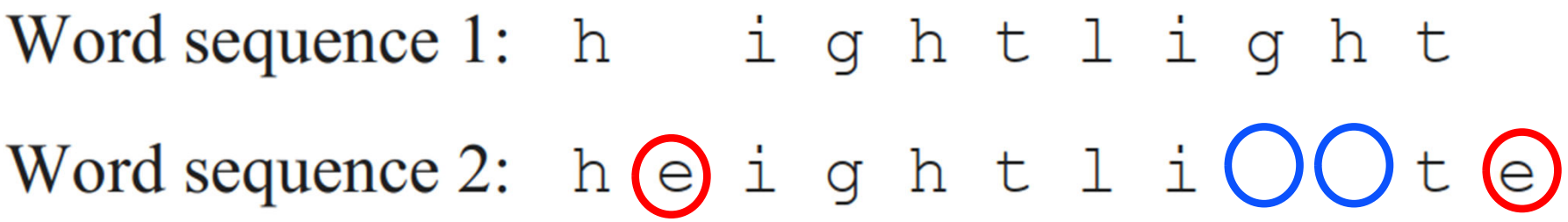
Lehrbücher zu C++ (12.08.2023)

Abgabe Freitag 17.10.2025
Wer will kann ab zwischen 16 Uhr und 17:30 mir und Herr Schmidt die Lösung per BBB vorstellen.

Wer möchte wann? Wenn jeder, finden wir sicherlich auch andere Termine.

Aufgabe 1

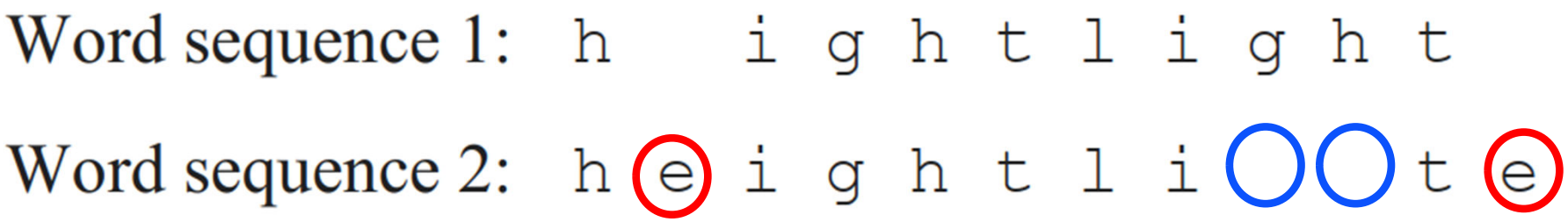
Implement a class Levenshtein to calculate the Levenshtein distance between two, i.e. sequence of characters.



Levenshtein distance is 4.

Aufgabe 1

Implement a class Levenshtein to calculate the Levenshtein distance between two, i.e. sequence of characters.



Levenshtein distance is 4.

Aufgabe 1

```
class Levenshtein
{
public:
   // astr1
   Levenshtein(
      const std::vector<char>& aw1,  // first word aw1 compared to
      const std::vector<char>& aw2)  // i.e. LD between aw1 and aw2

   //! returning the Levenshtein distance between astr1 and astr2
   int CalcLevenshteinDistance();
   std::string backtrace() const;
   std::string GetPrettyPrint(int ai_ld, std::string astr_goldText=
      std::string astr_recognText="") const;
```

Aufgabe 1

```
private:
      // first index runs over size of mstr1
    int Get(int st1Ind, int st2Ind) const {
       return (mpi_mat[st1Ind * mi_spCnt_n2 + st2Ind]);
    }
    std::vector<char> mw1;
    std::vector<char> mw2;
    int mi_zCnt_m1;    // size of mstr1
    int mi_spCnt_n2;   // size of mstr2
    // contains the matrix, used for LD calculation as a
    int mpi_mat[1000]; // later we create a dynamic matr
                       // of mw1 and mw2 are limited to
```

1-dim Array verwenden

Aufgabe 1a: CǎľçĽêŵêŋşḥʧêîŋDîşʧǎŋçê□čǎụêŋ

□□□Ẅê□çǎľçụľǎʧê□ʧḥê□Ľêŵêŋşḥʧêîŋ□đîşʧǎŋçê□ộğ□Ţîês□ǎŋđ□Ţộs□
xḥîçḥ□şḥộụľđ□čê□¸
čộộľ□ĽêŵêŋşḥʧêîŋDîşʧŢîêsŢộs□□
□□□□
□□□□□□ŵêçʧộs□şʧsîŋĝ□□ş¸□□□Ţ□□□□î□□□□ê□□□□s□□□□
□□□□□□ŵêçʧộs□şʧsîŋĝ□□ş¸□□□Ţ□□□□ộ□□□□s□□□□
□□□□□□Ľêŵêŋşḥʧêîŋ□đîşʧ□ş¸□□ş¸□□
□□□□□□sêʧụsŋ□¸□□□□đîşʧ□CǎľçĽêŵêŋşḥʧêîŋDîşʧǎŋçê□□□
□□□□□

Aufgabe 1b: Methode backtrace bauen

"
Calling with "Tiere" and "RenTier" as shown in the following test in:

```
bool LevenshteinDistTiereRenTier()
{
    vector<char> s1{ "T", "i", "e", "r", "e" };
    vector<char> s2{ "R", "e", "n", "T", "i", "e", "r" };
    Levenshtein dist(s1, s2);
    cout << "Needed steps: " << dist.backtrace());
    return (4 == dist.CalcLevenshteinDistance());
}
```

should result in

```
…Needed steps: Ins Ins Ins Equ Equ Equ Equ Del
```

Do not forget to implement different tests, which automatically check whether your implementation of backtrace might be correct.

The above code is not enough. It does not automatically test the 8 steps.

Aufgabe 1c: GetPrettyPrint bauen

```
·TEST(LevenshteinTest,·TestPrPrint)¶
·{¶
····vector<char>·s1{·'Z',·'o',·'o',·'t',·'i',·'e',·'r',·'e'·};¶
····vector<char>·s2{·'Z',·'i',·'e',·'g',·'e',·'t',·'i',·'e',·'r'·};¶

····Levenshtein·dist(s1,·s2);¶
····auto·ld·=·dist.CalcLevenshteinDistance();¶
····cout·<<·dist.GetPrettyPrint(ld,·"s1",·"and·s2");¶
·}¶
```

```
s1·····:·Z·o·o····t·i·e·r·e·¶

and·s2·:·Z·i·e·g·e·t·i·e·r····//·LD:·5,·subs:·2·/·ins·:·2·/·del·:·1·gold·words:·8¶
```

Der Code für GetPrettyPrint ist schon angegeben.

Do not forget to implement different tests, which automatically check whether your implementation of backtrace might be correct.

The above code is not enough. It does not automatically test, whether the output is correct.

Regeln

Sie dürfen die Aufgabe auch zu zweit bearbeiten.
Stellen Sie selber sicher, dass jeder aus der Gruppe etwas von der Bearbeitung der Aufgabe hat.

Mögliche weitere Aufgaben

Aufgabe 2:

Verwendung von Arrays dynamischer Größe und Code auf verschiedene Dateien aufteilen

Aufgabe 3:

Klassen anstatt Strukturen nutzen und Kopier-Konstruktur und Zuweisungs-Operator (tiefe und flache Kopie in C++)

Weitere Aufgaben (so der Plan)

Aufgabe 4:
Extraktion von Wegpunktnamen aus Wortfolgen aus falschen Ausgaben eines Spracherkenners

Aufgabe 5:
Effizientere Extraktion der Wegpunktnamen mit der schnellsten und der besten Implementierung

Es ist ausdrücklich erwünscht, dass Sie die Aufgabe in einer größeren Gruppe (z.B. 4 bis 5) GEMEINSAM bearbeiten. Das macht die Sache nicht unbedingt einfacher.

Ich werde (zusammen mit Ihnen) dafür Sorge tragen, dass JEDE(R) aus der Gruppe/Team einen Anteil zur Aufgabelösung beiträgt bzw. nur die Aktiven mit Punkte „versorgen“.

Zugriff auf SVN

URL

Jeder hat Zugriff auf:

- https://code.ostfalia.de/svn/i-wpf-cpp/AlleGruppen (lesend)

Es gibt noch einen privaten Bereich:

- https://code.ostfalia.de/svn/i-wpf-cpp/Studenten/XXX (lesend und schreibend)

XXX steht hier für Ihren Nachnamen (ohne Umlaute (ä = ae etc.),
Bei Doppelnamen mit Blank wurde nur der erste Name gewählt.
(„Müller Meier“ wurde zu „Mueller “).
Groß- und Kleinschreibung bitte beachten,
d.h. das erste Zeichen ist immer in Großbuchstaben.

Wenn Sie noch keinen Zugriff haben, brauche ich Ihre
E-Mail und Vor- und Nachnamen.