Aufgabe 03

Abgabe bis 15.11

Aufgabe 2.1

Exercise·2.1¶

Re-implement·your·code·from·the·previous·exercise,·so·that·dynamic·C-Arrays·are·possible·inside·the·Levenshtein·distance·class.·Do·not·use·the·STL-template·class·`vector`·for·the·matrix.·For·the·rest·of·the·code·you·may·use·the·vector·template·class.¶

new und delete und new[] und delete[] verwenden.
Entsprechend der Vorlesung für Vektor

Bitte immer noch nicht die Klasse vector<T> zu diesem Zweck verwenden.
map, vector etc. darf natürlich weiterhin an anderer Stelle verwendet werden (aber nicht für die Matrix der LD-Berechnung).

Aufgabe 2.1

Exercise·2.1¶

Re-implement·your·code·from·the·previous·exercise,·so·that·dynamic·C-Arrays·are·possible·inside·the·Levenshtein·distance·class.·Do·not·use·the·STL-template·class·`vector`·for·the·matrix.·For·the·rest·of·the·code·you·may·use·the·vector·template·class.¶

```
class Levenshtein{
public:
Levenshtein(
      const std::vector<std::string>& astr1,
      const std::vector<std::string>& astr2);
/* … */
private:
    // contains the matrix, used for LD calculation as a vector
    int* mpi_mat; // Use here a pointer to int and not a C-Array
    /*… */
};
```

Aufgabe 2.2

- Exercise·2.2¶

Split·your·code·into·different·files,·e.g.¶

- → One·main·file,·which·executes·the·tests¶
- → Header·and·Source-Code·file·for·the·class·Levenshtein·distance·calculation¶
- → Header·and·Source-Code·files·for·the·different·tests¶
- → …¶

Use·include·guards·(#ifndef,···#define···or/and··#pragma·once).¶

Aufgabe 2.3

Exercise 2.3

Split your main program into two parts. If it is called with the command line parameter `--test` all the tests are executed.

Otherwise a test file (you decide which) with some ATC utterances is read in and the contents is output. Later you will also output the extracted callsigns. Currently it is enough to just use the `--test` parameter in main.

Your `main` could look like this:

```
int main(int argc, char* argv[])
{
   if (argc > 1 && string(argv[1]) == "--test")
   {
      bool result = true;
```

Aufgabe 03

1. Replace the global variables, if not already implemented.
2. Use classes instead of the structures, if not already implemented before (no public attributes, access only via Get/Set etc.)
3. Implement also a minimal standard interface for your Levenshtein class, which allocates memory on the heap (Copy-Constructor, Assignment-Operator)
   Do not only show, that DoNothing does not crash, but show that the instances are really copied (a deep copy).
4. Visualize that you have no memory leaks.