

# Praktikumstermin Nr. 06, INF:

## Worte spiegeln interaktiv inkl. Unit Tests, Sudoku prüfen, Animation

### **(Pflicht-) Aufgabe INF-06.01:**

#### **Worte spiegeln interaktiv inkl. Unit Tests**

#### **(Schleifen, `if`, Strings, Funktionen, Unit Tests)**

Schreiben Sie ein C++ Programm, welches auf Benutzeranforderung einzelne Worte in einer einzeiligen Zeichenkette `text` spiegeln kann. Die Zeichenkette soll vom Benutzer eingelesen werden und darf auch Leerzeichen sowie Punkte (als Satzzeichen) enthalten. Ihr Programm kann davon ausgehen, dass der Benutzer nur korrekte Eingaben macht.

Wortgrenzen werden durch Leerzeichen, Punkt sowie Anfang bzw. Ende der Zeichenkette gebildet.

Ihr Quelltext muss dabei wie im Folgenden beschrieben modularisiert sein.

Die Datei `wortfunktionen.cpp` soll die „Wortspiegelfunktion“ ...

```
void wortspiegel(string &text, int pos)
```

... enthalten, welche im Text `text` genau das Wort spiegelt, welches sich rund um die Position `pos` befindet (siehe Position des Sternchens im Testlauf).

Die Zählung der Positionen im Text beginnt bei 0.

Sie können gerne auch weitere „textverarbeitungs-bezogene“ Hilfsfunktionen in dieser Datei definieren.

Schreiben Sie auch die zugehörige Headerdatei `wortfunktionen.h`.

Die Datei `bildschirm.cpp` soll die Eingabe- und Ausgabefunktionen ...

```
void ausgabe(string text, int pos);  
char eingabe();
```

... enthalten. Sie können gerne auch weitere „Eingabe/Ausgabe-bezogene“ Hilfsfunktionen in dieser Datei definieren.

Schreiben Sie auch die zugehörige Headerdatei `bildschirm.h`.

Die Datei `test_wortspiegel.cpp` enthält die Testfälle für ihre `wortspiegel()` Funktion. Legen Sie dazu eine leere

`test_wortspiegel.cpp` Datei in ihrem Visual Studio Projekt an und übertragen Sie per Copy-Paste die Inhalte der `test_wortspiegel.cpp` Datei in Ilias in diese Datei.

Die Datei `catch.h` enthält den Code des verwendeten Unit Test Frameworks (Quelle: <https://github.com/catchorg/Catch2/tree/v2.x>), welches auch in der Vorlesung vorgestellt wurde. Legen Sie dazu eine leere `catch.h` Datei in ihrem Visual Studio Projekt an (unter Headerdateien) und übertragen Sie per Copy-Paste die Inhalte der `catch.h` Datei in Ilias in diese Datei.

Die Datei `wortspiegel_main.cpp` soll das Hauptprogramm enthalten. Dieses soll zuerst die Unit Tests ausführen, dann den Text einlesen und anschließend in einer Endlosschleife immer wieder zuerst die Ausgabefunktion aufrufen, dann die Kommando-Eingabe-Funktion aufrufen und das eingegebene Kommando auswerten bzw. zur Ausführung bringen. Beim „Spiegelkommando“ wird die Funktion `wortspiegel()` aufgerufen.

*Programngerüst in der Datei `main.cpp`:*

```
// Datei: wortspiegel_main.cpp
#define CATCH_CONFIG_RUNNER
#include "catch.h"

#include <iostream>
#include <string>
using namespace std;

#include "wortfunktionen.h"
#include "bildschirm.h"

int main()
{
    // Aufruf der Unit Tests ...
    Catch::Session().run();

    // Ab hier ihr Code ...
    //
    // 1. Eingabe der Textzeile
    // 2. Endlosschleife:
    //    2a. Ausgabe der Textzeile und der Zeile mit dem Sternchen
    //    2b. Abfrage der Kommandoeingabe
    //    2c. Auswertung des eingegebenen Kommandos
    //    Insbes. Aufruf der wortspiegel() Funktion beim Kommando 's'

    system("PAUSE");
    return 0;
}
```

## Testlauf: (Benutzereingaben sind zur Verdeutlichung unterstrichen)

---

=====

**All tests passed (22 assertions in 22 test cases)**

Bitte geben Sie den Text ein: ? Dies ist ein Satz. Und noch ein Satz.

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- l

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- s

seiD ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- l

seiD ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- s

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- s

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- s

Dies ist ein ztaS. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- s

Dies ist ein Satz. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- s

Dies ist ein ztaS. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein ztaS. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein ztaS. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- s

Dies ist ein ztaS. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein ztaS. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein ztaS. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein ztaS. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein ztaS. Und noch ein Satz.

\*

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r

Dies ist ein ztaS. Und noch ein Satz.

\*

```

Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r
Dies ist ein ztaS. Und noch ein Satz.
      *
Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r
Dies ist ein ztaS. Und noch ein Satz.
      *
Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r
Dies ist ein ztaS. Und noch ein Satz.
      *
Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r
Dies ist ein ztaS. Und noch ein Satz.
      *
Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r
Dies ist ein ztaS. Und noch ein Satz.
      *
Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r
Dies ist ein ztaS. Und noch ein Satz.
      *
Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r
Dies ist ein ztaS. Und noch ein Satz.
      *
Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r
Dies ist ein ztaS. Und noch ein Satz.
      *
Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- r
Dies ist ein ztaS. Und noch ein Satz.
      *
Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- s
Dies ist ein ztaS. Und noch ein ztaS.
      *
Befehl (l: links, r: rechts, s: spiegeln, q: Ende) ?- q
Drücken Sie eine beliebige Taste . . .

```

---

## **(Pflicht-) Aufgabe INF-06.02: Sudoku auf Gültigkeit prüfen (Schleifen, if, Strings, Funktionen)**

Kopieren Sie Ihr Programm der vorherigen Sudoku Praktikumsaufgabe (aus dem vorigen GIP-INF Praktikum) in ein neues Projekt und ändern / erweitern Sie das Programm.

Schreiben Sie ein C++ Programm, welches ein „gelöstes“ Sudoku einliest und in einem Array ...

```
int sudoku[9][9] = {0};
```

... abspeichert (dies haben Sie in der Sudoku-Aufgabe im vorherigen Praktikum realisiert). Anschließend soll das Programm prüfen, ob das Sudoku den Regeln genügt: *„... jede Ziffer in jeder Spalte, in jeder Zeile und in jedem Block (3×3-Unterquadrat) genau einmal vorkommt“*.

Das Ergebnis der Prüfung soll ausgegeben werden mitsamt Hinweisen auf mehrfach oder gar nicht vorkommenden Zahlen, siehe Testläufe.

*Die Zeilen, Spalten und Blöcke sind dabei von 0 an durchnummeriert.*

*Innerhalb der Prüfung einer Zeile/Spalte/Blocks werden „Regelverletzungen“ in aufsteigender Zeilen-/Spalten-/Block- und aufsteigender Zahlen-Reihenfolge ausgegeben.*

*Hinweis (muss aber nicht unbedingt benutzt werden):*

Für Sudoku-Blöcke 0...8 liegen ...

... die Zeilen des Blocks im Bereich

von `zeile = block/3*3` bis `zeile <= block/3*3+2`

... die Spalten des Blocks im Bereich

von `spalte = block%3*3` bis `spalte <= block%3*3+2`

Die Sudokus der beiden Testläufe finden Sie auch in einer Datei in Ilias.

*Sollten bei nicht gültigen Sudokus die „Meldungen“ ihres Programms in einer anderen Reihenfolge sein als in den Testläufen oder einzelne „Meldungen“ über mehrfach vorhandene bzw. nicht vorhandene Zahlen bei ihnen mehrfach vorkommen, so ist das auch o.k.*

*Letztendlich müssen die einzelnen Meldungen (mal abgesehen von Wiederholungen und Reihenfolge) aber sein wie in den Testläufen ...*

Testlauf: (Benutzereingaben diesmal **nicht** unterstrichen, da sonst zu unübersichtlich)

---

Bitte geben Sie das Sudoku ein:

```

.5.1.4.|.8.6.9.|.7.2.3
.8.7.2.|.3.4.5.|.6.1.9
.9.6.3.|.2.1.7.|.5.4.8
-----|-----|-----
.6.2.8.|.1.3.4.|.9.5.7
.1.9.7.|.6.5.2.|.8.3.4
.4.3.5.|.7.9.8.|.1.6.2
-----|-----|-----
.2.4.6.|.9.7.1.|.3.8.5
.7.5.1.|.4.8.3.|.2.9.6
.3.8.9.|.5.2.6.|.4.7.1
Das Sudoku ist gueltig.
Drücken Sie eine beliebige Taste . . .

```

---

*Im folgenden Sudoku wiederholen sich bei den „Meldungen“ bestimmte Muster. Dadurch ist es leichter zu kontrollieren. Ich habe durch Farbgebung versucht, die sich wiederholenden Muster hervorzuheben ...*

```

Bitte geben Sie das Sudoku ein:
.1.1.4.|.8.6.9.|.7.2.3
.8.7.2.|.3.4.5.|.6.1.9
.9.6.3.|.2.1.7.|.5.4.8
-----|-----|-----
.6.2.8.|.1.3.4.|.9.5.7
.1.9.7.|.6.5.2.|.8.3.4
.4.3.5.|.7.9.8.|.1.6.2
-----|-----|-----
.2.4.6.|.9.7.1.|.3.8.5
.7.5.1.|.4.8.3.|.2.9.6
.3.8.9.|.5.2.6.|.4.7.9
Spalte 0: Zahl 1 kommt mehrfach vor.
Spalte 0: Zahl 5 kommt nicht vor.
Spalte 8: Zahl 1 kommt nicht vor.
Spalte 8: Zahl 9 kommt mehrfach vor.
Zeile 0: Zahl 1 kommt mehrfach vor.
Zeile 0: Zahl 5 kommt nicht vor.
Zeile 8: Zahl 1 kommt nicht vor.
Zeile 8: Zahl 9 kommt mehrfach vor.
Block 0: Zahl 1 kommt mehrfach vor.
Block 0: Zahl 5 kommt nicht vor.
Block 8: Zahl 1 kommt nicht vor.
Block 8: Zahl 9 kommt mehrfach vor.
Drücken Sie eine beliebige Taste . . .

```

---

*Die folgende „Aufgabe“ ist gar keine richtige „Challenge“, sondern die Vorbereitung für weitere Nutzungen dieser Möglichkeiten der graphischen Ausgabe (in kommenden GIP-INF Praktika). Die Aufgabe ist Pflicht, so dass ich ab jetzt voraussetzen werde, dass diese Graphik auf ihrem System funktioniert. Sollten Sie Probleme mit der Aufgabe bzw. der „Technik“ haben, so steht das Aufgaben-Tutorium am Freitag 11.12.2020 zur Verfügung. Dort wird die Aufgabe behandelt.*

## **(Pflicht-) Aufgabe INF-06.03: „Animation“**

In dieser Aufgabe verwenden wir die Grafikbibliothek von <http://cimg.eu> in einer für GIP-Zwecke angepassten Form. Diese Bibliothek sollte auch auf Linux- und Mac-Rechnern funktionieren (dort bei g++/gcc die Compiler-Optionen `-std=c++11 -lX11 -pthread` hinzufügen). Wir sichern Ihnen aber nur zu, dass es auf den FH Terminalrechnern funktioniert.

Legen Sie in einem neuen Projekt eine leere Headerdatei `CImgGIP05.h` an (siehe ggfs. Vorlesungsunterlagen, wie man Headerdateien anlegt).

Tragen Sie in diese Datei per copy-paste den Inhalt der gleichnamigen Datei `CImgGIP05.h` ein, die Sie im GIP-INF Praktikumsordner in Ilias finden.

Legen Sie dann eine Quelltextdatei `animation.cpp` an und kopieren Sie folgendes Programmgerüst in diese Datei:

```
#include <iostream>
#define CIMGGIP_MAIN
#include "CImgGIP05.h"
using namespace std;

int main()
{
    // Für das "blaue Spielfeld" ...
    const int left_border = 100;
    const int right_border = 500;

    // Für Ausdehnung und Position des weißen Quadrats ...
    int x = 200, y = 300;
    const int rectangle_size = 50;
    int delta = 10;

    gip_white_background();
```

```

while (gip_window_not_closed())
{
    // Blaues "Spielfeld" neu zeichnen
    // (übermalt dann auch das letzte weiße Quadrat) ...
    gip_draw_rectangle(left_border, 100, // linke obere Ecke
                       right_border, 500, // rechte untere Ecke
                       blue);

    // Weißes Quadrat neu zeichnen an der Position x,y ...
    gip_draw_rectangle(x, y, // linke obere Ecke
                       x + rectangle_size, // rechte untere Ecke
                       y + rectangle_size,
                       white);

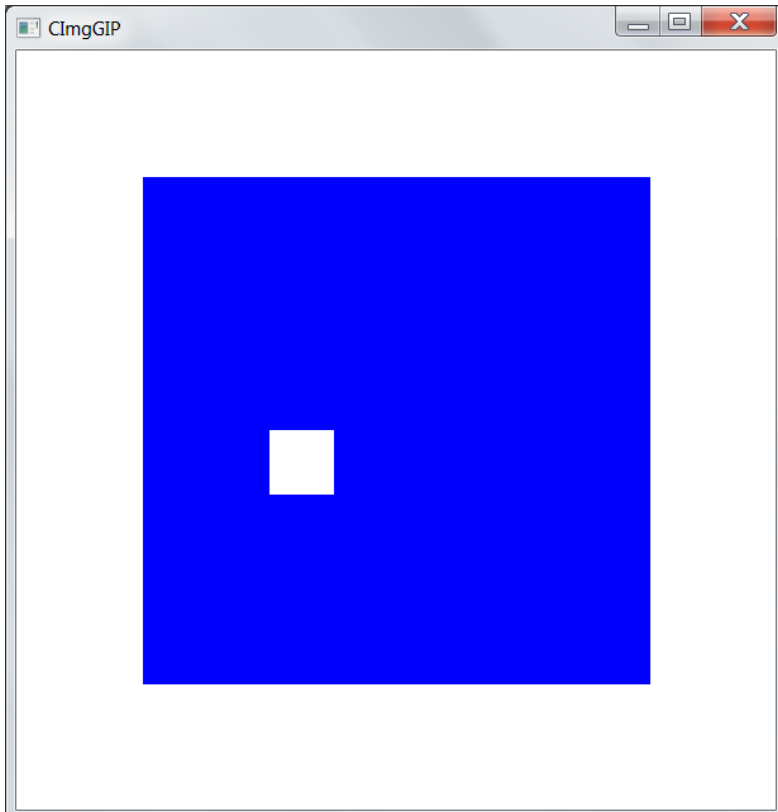
    // Nächste Position des weißen Quadrats berechnen.
    // Dabei berücksichtigen, dass das Quadrat von den Rändern des
    // blauen "Spielfelds" abprallen muss ...

    /* Ihr Code zur Berechnung der neuen Werte von x und y hier */

    gip_wait(100);
}
return 0;
}

```

*Screenshot:*



### *Aufgabe:*

Modifizieren Sie das Hauptprogramm, so dass sich das weiße Quadrat immer zwischen dem linken und rechten Rand des blauen Quadrats hin und her bewegt. Sobald der linke Rand des weißen Quadrats den linken Rand des blauen Quadrats berührt, bzw. der rechte Rand des weißen Quadrats den rechten Rand des blauen Quadrats, soll sich die Bewegungsrichtung umkehren. Bei jedem Schleifendurchlauf soll die x-Koordinate des weißen Quadrats um den Wert `delta` erhöht bzw. erniedrigt werden.