Einführung in C++ – Übung 2
Testatgruppe A (Isaak)

Rasmus Diederichsen   Simon Kern

24. Oktober 2014

Aufgabe 2.1

Listing 1: easter.c

```c
#include <stdio.h>
#include <stdlib.h>
/**
 * Reads a year from stdin and calculates the date of easter for
     that year,
 * which it prints on stdout. This is done until EOF is read.
 *
 * @return 0 on success
 */
int main(void)
{
   int year, golden_year, century,
       leap_years_skipped, moon_coor_fact,
       D, epact, day;
   char * month = malloc(6 * sizeof(char));
   /* User interaction */
   /* printf("Please enter year: "); */

   while (scanf("%d", &year) != EOF)
   {

      if (year < 0)
      {
         printf("Can't␣work␣with␣years␣B.C.\n");
         continue;
      }

      golden_year = (year % 19) + 1;
      century = (year / 100) + 1;
      leap_years_skipped = (3 * century / 4) - 12;
      moon_coor_fact = ((8 * century + 5) / 25) - 5;
      D = (5 * year / 4) - leap_years_skipped - 10; /* what should
          I call this? */
```

```
        epact = (11 * golden_year + 20 + moon_coor_fact -
            leap_years_skipped) % 30;
        if ((epact == 25 && golden_year > 11) || epact == 24) epact
            ++;
        day = 44 - epact;
        if (day < 21) day += 30;
        day += 7 - ((D + day) % 7);
        if (day > 31) {
            month = "April";
            day = day - 31;
        }
        else month = "March";
        printf("%d␣-␣%s␣%d\n", year, month, day);
    }
    return 0;
}
```

Listing 2: makefile for easter.c

```
#
# Einfuehrung in die Programmiersprache C++
#
# Makefile fuer Aufgabe 2 (easter)
#

CC              = gcc
INFILE          = years.in
OUTFILE         = easter_dates.out
CORRECT_OUTFILE = correct_easter_dates.out


easter: easter.o
        $(CC) easter.o -o bin/easter

easter.o: src/easter.c
        $(CC) -Wall -Wstrict-prototypes -ansi -pedantic -c src/
          easter.c

test:
        ./easter < $(INFILE) > $(OUTFILE)
        ./run_test $(OUTFILE) $(CORRECT_OUTFILE)

check:
        ./c_style_check.py easter.c
```

Aufgabe 2.2

Listing 3: sudoku.c

```
#include <stdio.h>
#include "include/sudoku.h"


int is_valid(int z, int i, int j, int sudoku[9][9])
{
    int x, y;
```

```c
   int subfield_x_ind = (i / 3) * 3; /* check the 3x3 field containing (i,j) */
   int subfield_y_ind = (j / 3) * 3;
   if ((z < 1 || z > 9) || (sudoku[i][j] != 0)) return 0; /* if out of bounds or not free */
   for (x = 0; x < 9; x++) /* check row and column intersecting at (i,j) */
      if (sudoku[x][j] == z || sudoku[i][x] == z) return 0;
   for (x = subfield_x_ind; x < subfield_x_ind + 3; x++){
      for (y = subfield_y_ind; y < subfield_y_ind + 3; y++){
         if(sudoku[x][y] == z) return 0;
      }
   }
   return 1; /* if no probs found, success! */
}

int solve_sudoku(int i, int j, int sudoku[9][9])
{
   int z; /* the number to try */
   while (i < 9 && (sudoku[i][j] != 0))
   { /* find next free field */
      j++;
      if (j >= 9)
      {
         j = 0;
         i++;
      }
   }

   if (i >= 9) return 1; /* no free fields? Already done. */
   else
   {
      for (z = 1; z <= 9; z++) /* try all numbers */
      {
         if (is_valid(z, i, j, sudoku)) /* if assignment valid ... */
         {
            sudoku[i][j] = z;
            if (!solve_sudoku(i, j, sudoku)) sudoku[i][j] = 0; /* ... continue and try with next field */
            else return 1; /* wenn the last field is to be set and it works, the function will jump here and bubble up true */
         }
      }
   }
   return 0; /* if control jumps here, it means no number was valid, so return false */
}

void print_sudoku(int sudoku[9][9])
{
   int i,j;
   printf("+-----+-------+-----+\n");
   for (i = 0; i < 9; i++) {
      for (j = 0; j < 9; j++) {
         if (j == 3 || j == 6)
```

```c
            printf("|␣%d␣",sudoku[i][j]);
         else
            printf("%d␣",sudoku[i][j]);
      }
      printf("\n");
      if (i == 2 || i == 5)
         printf("+-----+-------+-----+\n");
   }
   printf("+-----+-------+-----+\n");
}
/**
 * Main function to start the solver.
 * The sudoku is currently hardwired in the code.
 */
int main(void)
{

   if(solve_sudoku(0,0,field1)) print_sudoku(field1);
   else printf("Unsolvable.\n");
   return 0;
}
```

Listing 4: sudoku.h

```c
int field1[][9] = {
   {0, 0, 0,  0, 0, 8,  0, 3, 0},
   {0, 3, 0,  5, 0, 0,  4, 7, 1},
   {2, 0, 0,  1, 0, 0,  6, 9, 0},

   {5, 0, 0,  0, 0, 2,  1, 0, 0},
   {1, 2, 4,  0, 0, 0,  9, 6, 3},
   {0, 0, 6,  4, 0, 0,  0, 0, 2},

   {0, 8, 9,  0, 0, 5,  0, 0, 7},
   {3, 5, 2,  0, 0, 9,  0, 4, 0},
   {0, 1, 0,  3, 0, 0,  0, 0, 0}
};

/**
  * Function to check whether an assignment for a given field is
      legal.
  *
  * @param z The number to be assigned.
  * @param i The row number of the field to be assigned.
  * @param j The column number of the field to be assigned.
  * @param sudoku the Sudoku field against which to check.
  * @return 1 on success, 0 on failure.
  */
int is_valid(int z, int i, int j, int sudoku[9][9]);

/**
 * Function to attempt solving a sudoku.
 *
 * @param i The row number at which to set the first number.
 * @param j The column number at which to set the first number.
 * @return 1 if a silution was found, 0 otherwise.
 */
int solve_sudoku(int i, int j, int sudoku[9][9]);
```

```c

/**
 * Function to print a sudoku array on stdout.
 *
 * @param sudoku The puzzle to be printed.
 */
void print_sudoku(int sudoku[9][9]);
```

Listing 5: makefile for sudoku.c

```makefile
hello: sudoku
	gcc -Wall -Wstrict-prototypes -ansi -pedantic src/sudoku.c
	    -o sudoku
clean:
	rm sudoku
test:
	./sudoku
```