

December 12, 2017 at 15:25

1. Intro. Mike Spivey announced a programming contest in February 2005, asking for a program that solves “sudoku” puzzles (which evidently appear daily in British newspapers). This program takes a sudoku specification in standard input and creates — on standard output — a file that can be piped into DANCE in order to deduce all solutions.

Brief explanation: Each possible placement of a digit corresponds to a row, column, and box where that digit does not yet appear. We want an exact cover of those rows, columns, and boxes.

Apology: I wrote this in a big hurry. But I couldn’t resist the task, because it is such a nice application of exact covering.

```
#include <stdio.h>
char buf[11];
int row[9][10], col[9][10], box[9][10]; /* things to cover */
int board[9][9]; /* positions already filled */

main()
{
    register int j, k, d, x;
    for (k = 0; k < 9; k++) <Input row k 2>;
    <Output the column names needed by DANCE 3>;
    for (j = 0; j < 9; j++)
        for (k = 0; k < 9; k++)
            if (!board[k][j]) <Output the possibilities for filling column j of row k 4>;
}
```

2. In a production system I would of course try to give more informative error messages about malformed input data. Here I simply quit, if the rules haven’t been followed.

```
#define panic(m)
    { fprintf(stderr, "%s!\n%s", m, buf); exit(-1); }

<Input row k 2> ≡
{
    fgets(buf, 11, stdin);
    if (buf[9] != '\n') panic("Input_line_should_have_9_characters_exactly!\n");
    for (j = 0; j < 9; j++)
        if (buf[j] != '.') {
            if (buf[j] < '1' ∨ buf[j] > '9') panic("Illegal_character_in_input!\n");
            d = buf[j] - '0';
            if (row[k][d]) panic("Two_identical_digits_in_a_row!\n");
            row[k][d] = 1;
            if (col[j][d]) panic("Two_identical_digits_in_a_column!\n");
            col[j][d] = 1;
            x = ((int)(k/3)) * 3 + ((int)(j/3));
            if (box[x][d]) panic("Two_identical_digits_in_a_box!\n");
            box[x][d] = 1;
            board[k][j] = 1;
        }
}
```

This code is used in section 1.

3. First we print out all the positions, rows, columns, and boxes that need to be covered.

⟨Output the column names needed by DANCE 3⟩ ≡

```

for ( $k = 0$ ;  $k < 9$ ;  $k++$ )
  for ( $j = 0$ ;  $j < 9$ ;  $j++$ )
    if ( $\neg board[k][j]$ ) printf("_p%d%d",  $k, j$ );
for ( $k = 0$ ;  $k < 9$ ;  $k++$ )
  for ( $d = 1$ ;  $d \leq 9$ ;  $d++$ ) {
    if ( $\neg row[k][d]$ ) printf("_r%d%d",  $k, d$ );
    if ( $\neg col[k][d]$ ) printf("_c%d%d",  $k, d$ );
    if ( $\neg box[k][d]$ ) printf("_b%d%d",  $k, d$ );
  }
printf("\\n");

```

This code is used in section 1.

4. Then we print out all the possible placements.

⟨Output the possibilities for filling column j of row k 4⟩ ≡

```

{
   $x = ((\text{int})(k/3)) * 3 + ((\text{int})(j/3))$ ;
  for ( $d = 1$ ;  $d \leq 9$ ;  $d++$ )
    if ( $\neg row[k][d] \wedge \neg col[j][d] \wedge \neg box[x][d]$ ) printf("p%d%d_r%d%d_c%d%d_b%d%d\\n",  $k, j, k, d, j, d, x, d$ );
}

```

This code is used in section 1.

5. Index.

board: [1](#), [2](#), [3](#).

box: [1](#), [2](#), [3](#), [4](#).

buf: [1](#), [2](#).

col: [1](#), [2](#), [3](#), [4](#).

d: [1](#).

exit: [2](#).

fgets: [2](#).

fprintf: [2](#).

j: [1](#).

k: [1](#).

main: [1](#).

panic: [2](#).

printf: [3](#), [4](#).

row: [1](#), [2](#), [3](#), [4](#).

stderr: [2](#).

stdin: [2](#).

x: [1](#).

⟨Input row k 2⟩ Used in section 1.

⟨Output the column names needed by DANCE 3⟩ Used in section 1.

⟨Output the possibilities for filling column j of row k 4⟩ Used in section 1.

SUDOKU

	Section	Page
Intro	1	1
Index	5	3