ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
БЕОГРАД

ПРАКТИКУМ ИЗ ПРОГРАМИРАЊА 2

ОДСЕК ЗА СОФТВЕРСКО ИНЖЕЊЕРСТВО

Пројектни задатак број 9

Упутство за програмере

Пројектни аутори
Алекса Бркић 15/0210 (aleksa.brkic12@gmail.com)
Миљан Зарубица 15/0128 (miljanzarubica1@gmail.com)
Петар Ђекановић 15/0101 (petar.djekanovic@gmail.com)

Београд, јун 2007.

УВОД

- Апликација служи као окружење за решавање судокуа. Корисник има опцију да апликација генерише нерешен судоку или да сам упише свој судоку и да га решава. При решавању, корисник има могућност да тражи помоћ од програма (провера поља, решавање поља, провера целог судокуа). Постоји и могућност сачувавања тренутне игре, заједно са временом.
- У програму су имплементиране многе функције за решавање судокуа које раде у позадини, ради провере судокуа, и ако корисник затражи помоћ.

СПИСАК ФУНКЦИЈА

- **Датотека:** functions.c

Име функције: `isCorrectStart`

Прототип: `int isCorrectStart(int sud[][9]);`

Аргументи: `int sud[][9]:` Матрица која представља судоку

Повратна вредност: `int:` 0 у случају да је судоку контрадикторан, 1 супротно

Опис: Проверава да ли судоку није контрадикторан. Користи функције:

- `int isCorrectStartNumbers(int sud[][9]);`
- `int isCorrectStartRow(int sud[][9]);`
- `int isCorrectStartColumn(int sud[][9]);`
- `int isCorrectStartBox(int sud[][9]);`

Које проверавају да ли су добри бројеви (опсег: од 0 до 9), и да ли се неки број у реду/колони/региону понавља.

Име функције: `isCorrectFinish`

Прототип: `int isCorrectFinish(int sud[][9]);`

Аргументи: `int sud[][9]:` Матрица која представља судоку

Повратна вредност: `int:` 0 у случају да је судоку није решен или је контрадикторан, 1 супротно

Опис: Проверава да ли је судоку решен. Користи функције:

- `int isCorrectFinishNumbers(int sud[][9]);`
- `int isCorrectFinishRow(int sud[][9]);`
- `int isCorrectFinishColumn(int sud[][9]);`
- `int isCorrectFinishBox(int sud[][9]);`

Које проверавају да ли су добри бројеви (опсег: од 1 до 9), и да ли се неки број у реду/колони/региону понавља.

Име функције: `fullHouse`

Прототип: `int fullHouse(int sud[][9]);`

Аргументи: `int sud[][9]`: Матрица која представља судоку

Повратна вредност: `int`: број промена судокуа

Опис: Уписује број у свако поље које је једино непопуњено у свом реду/колони/региону. Користи функције:

- `int fullHouseRow(int sud[][9]);`
- `int fullHouseColumn(int sud[][9]);`
- `int fullHouseBox(int sud[][9]);`

Име функције: `nakedSingles`

Прототип: `int nakedSingles(int sud[][9]);`

Аргументи: `int sud[][9]`: Матрица која представља судоку

Повратна вредност: `int`: број промена судокуа

Опис: Уписује број у свако поље које има само једну могућност када се провере његов ред, колона и регион. Користи функције:

- `int nakedSinglesRow(int sud[][9]);`
- `int nakedSinglesColumn(int sud[][9]);`
- `int nakedSinglesBox(int sud[][9]);`

Функција `int nakedSinglesAlt(int sud[][9], int possibles[][9][9])` извршава исту радњу, користећи тродимензионалну матрицу могућих бројева за свако поље, односно за свако поље које има једну могућност уписује тај број у матрицу.

Име функције: `hiddenSingles`

Прототип: `int hiddenSingles(int sud[][9]);`

Аргументи: `int sud[][9]`: Матрица која представља судоку

Повратна вредност: `int`: број промена судокуа

Опис: Ако је неки број могућ само у једном пољу (за неки ред, колону или регион), уписује тај број. Користи функције:

- `int hiddenSinglesRow(int sud[][9]);`
- `int hiddenSinglesColumn(int sud[][9]);`
- `int hiddenSinglesBox(int sud[][9]);`

Име функције: `nakedDoubles`

Прототип: `int nakedDoubles(int sud[][9]);`

Аргументи: `int sud[][9]`: Матрица која представља судоку

Повратна вредност: `int`: број промена судокуа

Опис: Проверава да ли за нека два поља у врсти/колони/региону важи да су на њима могућа само два броја. Ако јесу, та два броја су немогућа у било ком другом пољу у тој врсти/колони/региону. Након анализе, методом Naked Singles прође кроз цео судоку.

Програм се служи тродимензионалном матрицом која за свако поље означава могуће бројеве. Користи функције:

- `int nakedDoublesRow(int sud[][9]);`
- `int nakedDoublesColumn(int sud[][9]);`
- `int nakedDoublesBox(int sud[][9]);`
- `int nakedSinglesAlt(int sud[][9], int possibles[][9][9]);`
- `void clearPossRow(int possibles[][9][9], int i, int clearPoss[], int j1, int j2)`
- `void clearPossColumn(int possibles[][9][9], int i, int clearPoss[], int j1, int j2)`
- `void clearPossBox(int possibles[][9][9], int i, int clearPoss[], int j1, int j2)`

Име функције: `solveSudoku`

Прототип: `int solveSudoku(int sud[][9]);`

Аргументи: `int sud[][9]:` Матрица која представља судоку

Повратна вредност: `int:` 0 (false) ако се судоку није решио, 1 (true) ако се успешно решио, и -1 (error) ако се не може решити.

Опис: Комбиновањем претходних функција за решавање покушава да реши судоку. Ако у једној итерацији не дође до промена, прекида извршавање. Користи функције:

- `int fullHouse(int sud[][9]);`
- `int nakedSingles(int sud[][9]);`
- `int hiddenSingles(int sud[][9]);`
- `int nakedDoubles(int sud[][9]);`
- `int isCorrectFinish(int sud[][9]);`

Име функције: `bruteForce`

Прототип: `int bruteForce(int sud[][9]);`

Аргументи: `int sud[][9]:` Матрица која представља судоку

Повратна вредност: `int:` 0 (false) ако судоку још није решен, 1 (true) ако јесте.

Опис: Пролази кроз судоку и на прво празно место ставља све могуће бројеве, и за сваки рекурзивно позива `bruteForce`. Користи функције:

- `int isCorrectLocal(int sud[][9]);`
- `int isCorrectFinish(int sud[][9]);`

Име функције: `smartBruteForce`

Прототип: `int smartBruteForce(int sud[][9]);`

Аргументи: `int sud[][9]:` Матрица која представља судоку

Повратна вредност: `int:` 1 (true) ако се успешно решио, и -1 (error) ако се не може решити.

Опис: Пролази кроз судоку и на прво празно место ставља све могуће бројеве, ствара локалну судоку матрицу и за њу позива функцију `solveSudoku`, и затим позива себе за ту матрицу, док локална матрица не буде решена. Користи функције:

- `int isCorrectLocal(int sud[][9]);`
- `int solveSudoku(int sud[][9]);`
- `int isCorrectFinish(int sud[][9]);`

Име функције: `oneSolution`

Прототип: `int smartBruteForce(int sud[][9], int* numOfSolutions);`

Аргументи: `int sud[][9]`: Матрица која представља судоку
`int* numOfSolutions`: Број који се повећава налажењем новог решеног судокуа

Повратна вредност: `int:` 1 ако судоку има тачно једно решење, 0 ако нема ниједно, и 2 ако има више решења.

Опис: Проверава да ли судоку има једно решење. Ако има више од једног, завршава се извршавање. Користи функције:

- `int isCorrectLocal(int sud[][9]);`
- `int isCorrectFinish(int sud[][9]);`

Функцију позива функција:

- `int hasOneSolution(int sud[][9]);`

која враћа 1 (true) ако судоку има једно решење, и 0 (false) супротно.

Функције које нису наведене:

```
void buildSudoku(int sud[][9]);

int fullPos(int pos[9]);

int generateSudoku(int sud[][9]);
int generateSudokuLevel(int sud[][9], int level);

int x(int i, int j);
int y(int i, int j);

void clearPossRow(int possibles[][9][9], int i, int clearPoss[], int j1, int j2);
void clearPossColumn(int possibles[][9][9], int j, int clearPoss[], int i1, int i2);
void clearPossBox(int possibles[][9][9], int sub, int clearPoss[], int pos1, int
pos2);
```

- **Датотека:** graphics.c

Име функције: `game`

Прототип: `void game(char *file_name);`

Опис: Главни екран игре за решавање судокуа. У функцији су имплементиране све помоћи

за особу која решава (провера поља, решавање поља, провера целог судокуа), могућност сачувавања игре и завршавање судокуа.

Име функције: `start_menu`
Прототип: `int start_menu();`
Повратна вредност: `int:` вредност која означава мени у који се улази
Опис: Приказује главни мени игре, са опцијама нова игра, учитавање старе игре, инструкције и излаз из програма.

Функције које нису наведене:

```
void start_scr();
void sudoku_logo();

int exit_menu();

void instructions();

void finish_scr(float time);

void loading();
void init_load();
```

- **Датотека:** save-load.c

Име функције: `loadGame`
Прототип: `char *load_game()`
Повратна вредност: `char*:` Име датотеке која се учитава
Опис: При избору опције учитавања фајла, учитава изабрани фајл.

Име функције: `makeBin`
Прототип: `void makeBin(char* name, int sudoku[][9], int bool_sudoku[][9], int time);`
Аргументи: `char* name`: Име датотеке која се прави
`int sud[][9]`: Матрица која представља судоку
`int bool_sudoku[][9]`: Матрица која представља у ком је стању свако поље
`int time`: Време које је досад протекло у решавању тренутног судокуа
Опис: Стварање бинарног фајла који служи као сачуван судоку.

Име функције: `readBin`

Прототип: `void readBin(char* name, int sudoku[][9], int bool_sudoku[][9], int*time);`
Аргументи: `char* name`: Име датотеке која се прави
`int sud[][9]`: Матрица која представља судоку
`int bool_sudoku[][9]`: Матрица која представља у ком је стању свако поље
`int* time`: Време које је досад протекло у решавању тренутног судокуа
Опис: Отварање бинарног фајла у коме је сачуван судоку.

Име функције: `makeSVG`
Прототип: `void makeSVG(char* name, int sudoku[][9], int bool_sudoku[][9]);`
Аргументи: `char* name`: Име датотеке која се прави
`int sud[][9]`: Матрица која представља судоку
`int bool_sudoku[][9]`: Матрица која представља у ком је стању свако поље
Опис: Стварање .svg фајла за одређену сачувану игру.

Функције које нису наведене:

```
char *enter_name();

 int save_scr();
```