# Life

Generated by Doxygen 1.8.18

# Chapter 1

# Implementation details

## 1.1 Program outline

### 1.1.1 Initialisation

- open the configuration file
- read the configuration into the state matrix
- close the file
- write the initial configuration

### 1.1.2 Main program

Wait for input: number g of generation to process.

- If g=0 terminate
- otherwise: for g generations
  - for each cell calculate the number n of neighbours
  - for each cell:
    - if n==3 set state s=1
    - if n!=2 set state s=0
- Write out the map for the user;

## 1.2 Program implementation

### 1.2.1 Main loop

```
    int main(void) {

    int s[MI][MK];  // state matrix
    int n[MI][MK];  // number of neighbours
    FILE *fp = fopen("life.txt","r");
    if (fp == NULL) { error handling }
    read(fp,s);
    fclose(fp);
    write(s,t);
    while (scanf("%u",&g)) {        // read the number of generations to process
        if (g == 0) break;          // terminate the program
        for (h=0; h<g; h++) {
            // calculate neighbours
            for (i=0; i<MI; i++)
                for (k=0; k<MK; k++){ // pay attention to calculate index modulo MI resp. MK
                    n[i][k] = s[i-1][k-1] + s[i-1][k] + s[i-1][k+1] + s[i][k-1] + s[i][k+1] + s[i+1][k-1] +
   s[i+1][k] + s[i+1][k+1]
                }
            // set new state
            for (i=0; i<MI; i++)
                for (k=0; k<MK; k++)
                if (n[i][k]==3)
                        s[i][k] = 1;
                    else if (n[i][k]!=2)
                        s[i][k]=0;
            t++;
        }
        write(s,t);
    }
    return EXIT_SUCCESS;
}
```

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 3

# File Documentation

## 3.1 life.c File Reference

implements John H. Conway's Game of Life

```
#include <stdio.h>
#include <stdlib.h>
```

### Macros

- #define MI 50
- #define MK 80

### Functions

- void read (FILE *, int[MI][MK])
- void write (int[MI][MK], int)
- int main (void)

  *main logic of the life program*

### 3.1.1 Detailed Description

implements John H. Conway's Game of Life

This C program implements Conway's Game of Life on a torus

**Author**

berndt

**Version**

1.0

**Date**

2020-06-04

**Copyright**



### 3.1.2 Macro Definition Documentation

#### 3.1.2.1 MI

```
#define MI 50
```

number of rows

#### 3.1.2.2 MK

```
#define MK 80
```

number of columns

### 3.1.3 Function Documentation

#### 3.1.3.1 main()

```
int main (
            void  )
```

main logic of the life program

The main routine takes no parameters and returns EXIT_SUCCESS on normal exit or EXIT_FAILURE on failure to read the input file (named "life.txt"). The life map is initialized and written to the screen. The program waits for the number of generations to be processed (integer g). These number of generations is then calculated and the result displayed. If g=0 the program ends.

The program keeps a generation counter (variable t for time). The basic play ground is a grid (map) that keeps the state of each cell in the area of a torus determined by two coordinates i,k that run from 0 to MI, resp. MK (maximum i, resp. maximum k). The MI resp. MK are the fundamental periods of the torus. The variable s (for state) is kept for each cell (i,k) in the array s[i][k]. The number of neighbours is kept in a similar array n[i][k].

The number of neighbours is calculated by

n[i][k] = s[i-1][k-1] + s[i-1][k] + s[i-1][k+1] + s[i][k-1] + s[i][k+1] + s[i+1][k-1] + s[i+1][k] + s[i+1][k+1]

and in dependence of this number the next generation states are

s[i][k] = 1 if ( n[i][k] == 3 )

s[i][k] = 0 if ( n[i][k] != 2 )

Recall that i is taken mod MI and k is taken mod MK.

#### 3.1.3.2 read()

```
void read (
            FILE * fp,
            int s[MI][MK] )
```

reads the input file and populates the state matrix

**Parameters**

| | |
|---|---|
| *fp* | file pointer |
| *s* | matrix of size MI x MK |

**Returns**

void

#### 3.1.3.3 write()

```
void write (
            int s[MI][MK],
            int t )
```

write the state matrix to stdout

**Parameters**

| | |
|---|---|
| *s* | matrix of size MI x MK |
| *t* | generation number |

**Returns**

void

# Index