ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра вычислительных систем

КУРСОВАЯ РАБОТА

По дисциплине «Технологии разработки программного обеспечения»

на тему «Игра Жизнь»

Выполнил:

Студент ИП-512

Или Д.А.

Проверил:

Фульман В.О

Новосибирск 2016

Оглавление

[ВВЕДЕНИЕ 3](#_Toc452802725)

[Языки программирования 4](#_Toc452802726)

[ОПИСАНИЕ ВЫПОЛНЕННОГО ПРОЕКТА 5](#_Toc452802727)

[ПРИЛОЖЕНИЕ 1. 11](#_Toc452802728)

[Main.c 11](#_Toc452802729)

[ConsoleWrap.c 25](#_Toc452802730)

[ConsoleWrap.h 28](#_Toc452802731)

## ВВЕДЕНИЕ

Основная цель работы — освоить технологию создания полноценного программного обеспечения, итогом которого должна была быть конечная версия для пользователей проекта «Игра Жизнь». Конечная версия должна включать в себя техническую и пользовательскую документацию, Unit-тесты, сборку для пользователей и автоматизированную сборку с помощью TravisCi.

## ТЕХНИЧЕСКОЕ ЗАДАНИЕ

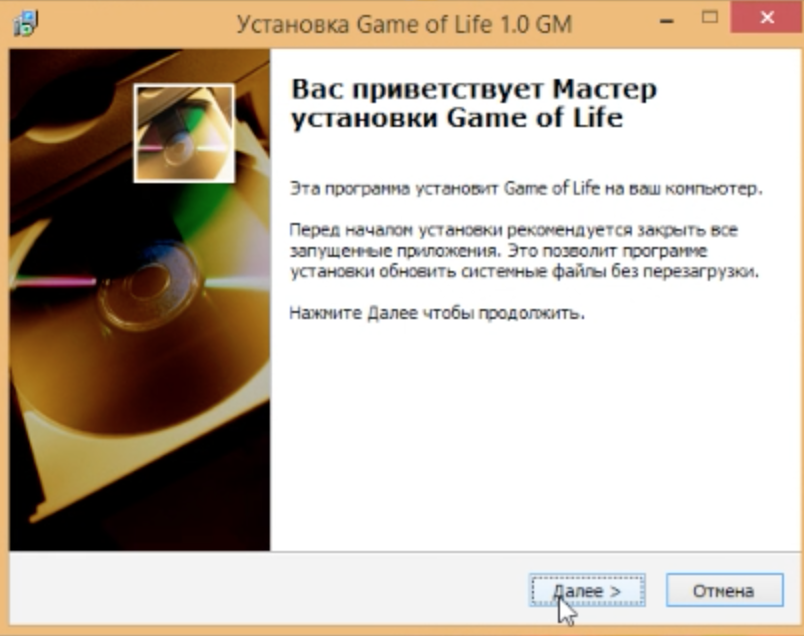
1. Игра «Жизнь» имеет развлекательный характер и имеет множество вариантов использования: от «таймкиллера» до доказательства каких либо теорий.(Игра «Жизнь» и её модификации повлияли (в ряде случаев взаимно) на многие разделы таких точных наук, как математика, информатика, физика)
2. Мир заполняется полностью с помощью Random
3. Приложение будет интерактивным, но участие пользователя будет минимальным: от пользователя требуется только запустить приложение и наблюдать. Предусмотрено ограниченное клеточное поле, верхняя граница поля «соединена» с нижней, а левая граница — с правой, что представляет собой эмуляцию поверхности «тора», но на экране поле всегда отображается в виде равномерной сетки.
4. Аргументы командной строки не будут использованы в данной программе
5. в программе предполагается использование конфигурационного файла типа .ini. ini-файлы — это обычные текстовые файлы, которые можно редактировать и просматривать при помощи любого текстового редактора. INI файл может содержать:
   * пустые строки;
   * комментарии — от символа «;» (точка с запятой) до конца строки;
   * заголовки разделов — строки, состоящие из названия раздела, заключённого в квадратные скобки «[ ]»;
   * значения параметров — строки вида «ключ=значение».

Данная программа будет написана на языке C(си)

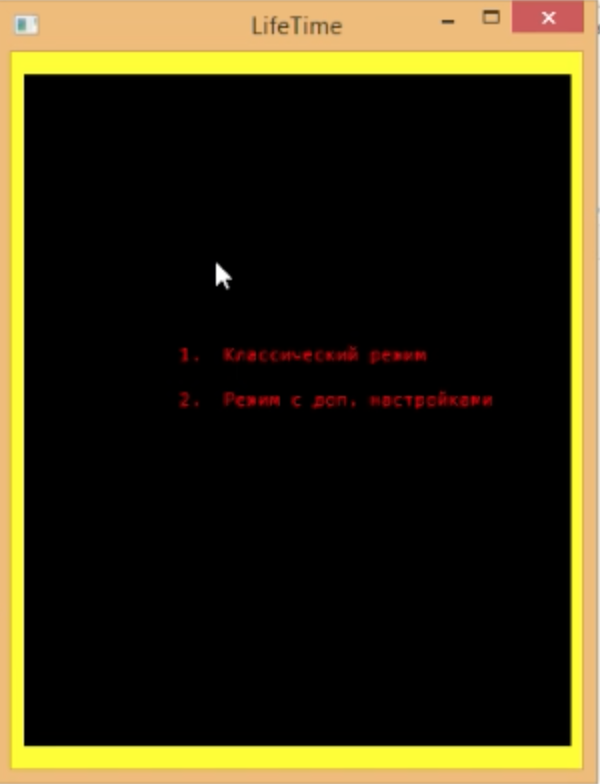
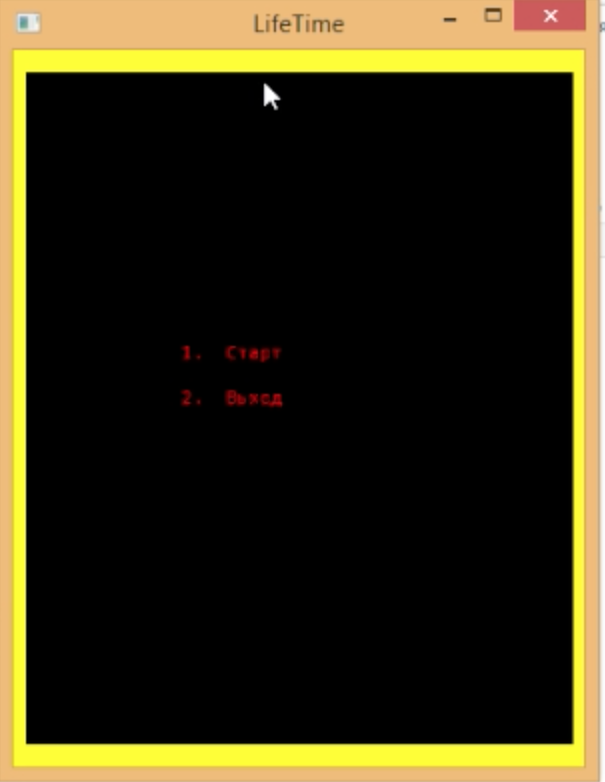
## ОПИСАНИЕ ВЫПОЛНЕННОГО ПРОЕКТА

Проект «Игра Жизнь» создан для операционной системы Windows с помощью среды разработки Visual Studio 2013. Проект представляет из себя графическое представление известного алгоритма математика Джеймса Конвея.

В релизе для пользователей установка программы происходит с помощью интуитивно понятного установщика, для запуска установщика необходимо запустить Setup.exe.



При запуске программы нас встречает меню с выбором действия.



При выборе «классического» режима клетки размножаются и умирают по стандартным условиям автомата – при количестве соседей меньше 2 клетка умирает, а при 3 соседях - рождает новую.

Режим “Classic” включает в себя 3 видов клетки:

• клетки готовые к смерти

• клетки 1-го поколения

• клетки 2-го поколения



Режим “Custom” включает 6 видов клетки:

• Клетки из “Classic” режима

o клетки готовые к смерти

o клетки 1-го поколения

o клетки 2-го поколения

• Бессмертные клетки

• Клетки-любовники

• Клетки-смертники



Функции клеток:

• Бессмертные клетки, клетки под номером 4 – не умирают, не изменяются в зависимости от времени, может умереть только от клетки-смертника, способны к размножению

• Клетки-любовники, клетки под номером 6 — растет, как обычная клетка, при рождении следущего поколения заполняет все клетки вокруг себя «детьми»

1. Клетки-смертники, клетки под номером 5 – появляется как обычная клетка, но при вырастании превращается в клетку смертника и убивает все клетки вокруг себя

• Мертвые клетки, клетки под номером 0 — нет функций, только что умершая клетка клетка

• Клетки 1-го поколения, клетки под номером 1 – взрослые клетки, способны к размножению

• Клетки 2-го поколения, клетки под номером 2 – «дети», только что «родившиеся» клетки, вырастут в дальнейшем.

## Реализация индивидуальной части в проекте

## ПРИЛОЖЕНИЕ 1.

### Main.c

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <time.h>

#include <locale.h>

#include "ConsoleWrap.c"

void clear(int \*\*&info, int height, int width)

{

for (int i = 0; i < height + 1; i++)

{

for (int j = 0; j < width + 1; j++)

{

info[j][i] = 3;

}

}

}

void clear\_Scope(int \*\*&info, int height, int width)

{

int j, i;

for (i = 0; i < height + 1; i++)

{

info[0][i] = 3;

info[width][i] = 3;

}

for (j = 0; j < width + 1; j++)

{

info[j][0] = 3;

info[j][height];

}

FillPos(0, 0, f\_darkyellow, '#', width + 1);

FillPos(0, height, f\_darkyellow, '#', width + 1);

FillPos(0, height + 6, f\_darkyellow, '#', width + 1);

for (i = 1; i < height + 6; i += 1)

{

SetChar(0, i, f\_darkyellow, '#');

SetChar(width, i, f\_darkyellow, '#');

}

}

void ch(int \*\*&info, int height, int width)

{

int i, j, k = 0, ch = 0;

for (i = 1; i < height; i++)

{

for (j = 1; j < width; j++)

{

if (info[j][i] == 1)

{

ch++;

}

if (info[j][i] == 3)

{

SetChar(j, i, c\_black, ' ');

}

}

}

SetPos(38, height + 1);

printf(" ");

SetPos(40, height + 1);

SetColor(f\_red);

printf("%d", ch);

}

void classic\_RandomFirstGen(int \*\*&info, int height, int width)

{

int i, j, k;

for (k = 1; k < 101; k++)

{

i = rand() % 24 + 1;

j = rand() % 49 + 1;

if (info[j][i] != 1)

{

info[j][i] = 1;

SetChar(j, i, f\_red, '1');

SetPos(0, 0);

}

else

{

k--;

}

}

}

void rename(int \*\*&info, int height, int width)

{

int i, j, k = 0;

for (i = 1; i < height; i++)

{

for (j = 1; j < width; j++)

{

if (info[j][i] == 0)

{

info[j][i] = 3;

SetChar(j, i, c\_black, ' ');

}

else

if (info[j][i] == 2)

{

info[j][i] = 1;

SetChar(j, i, f\_red, '1');

}

}

}

}

void deathGen(int \*\*&info, int height, int width)

{

int i, j, k = 0, dth = 0, ch = 0, death\_ch = 0;

for (i = 1; i < height; i++)

{

for (j = 1; j < width; j++)

{

if (info[j][i] == 1)

{

if (info[j + 1][i + 1] == 1 || info[j + 1][i + 1] == 0)

{

ch++;

}

if (info[j - 1][i - 1] == 1 || info[j - 1][i - 1] == 0)

{

ch++;

}

if (info[j + 1][i - 1] == 1 || info[j + 1][i - 1] == 0)

{

ch++;

}

if (info[j - 1][i + 1] == 1 || info[j - 1][i + 1] == 0)

{

ch++;

}

//

if (info[j + 1][i] == 1 || info[j + 1][i] == 0)

{

ch++;

}

if (info[j][i + 1] == 1 || info[j][i + 1] == 0)

{

ch++;

}

if (info[j - 1][i] == 1 || info[j - 1][i] == 0)

{

ch++;

}

if (info[j][i - 1] == 1 || info[j][i - 1] == 0)

{

ch++;

}

if (ch < 2 || ch > 3)

{

info[j][i] = 0;

SetChar(j, i, c\_black, ' ');

death\_ch++;

}

ch = 0;

}

}

}

SetPos(38, height + 3);

printf(" ");

SetPos(40, height + 3);

SetColor(f\_cyan);

printf("%d", death\_ch);

}

void secondGen(int \*\*&info, int height, int width)

{

int i, j, k = 0, dth = 0, ch = 0;

for (i = 1; i < height; i++)

{

for (j = 1; j < width; j++)

{

if (info[j][i] == 3)

{

if (info[j + 1][i + 1] == 1)

{

ch++;

}

if (info[j - 1][i - 1] == 1)

{

ch++;

}

if (info[j + 1][i - 1] == 1)

{

ch++;

}

if (info[j - 1][i + 1] == 1)

{

ch++;

}

//

if (info[j + 1][i] == 1)

{

ch++;

}

if (info[j][i + 1] == 1)

{

ch++;

}

if (info[j - 1][i] == 1)

{

ch++;

}

if (info[j][i - 1] == 1)

{

ch++;

}

if (ch == 3 || ch > 3)

{

info[j][i] = 2;

SetChar(j, i, b\_blue, '2');

k++;

}

ch = 0;

}

}

}

SetColor(f\_blue);

SetPos(38, height + 2);

printf(" ");

SetPos(40, height + 2);

printf("%d", k);

}

void interface(int \*\*&info, int height, int width)

{

SHORT i;

// In case of changed background color

SetChar(0, 0, c\_black, ' ');

system("cls");

// Up row

FillPos(0, 0, f\_darkyellow, '#', width + 1);

// Down row

FillPos(0, height, f\_darkyellow, '#', width + 1);

FillPos(0, height + 6, f\_darkyellow, '#', width + 1);

// Side columns

for (i = 1; i < height + 6; i += 1)

{

SetChar(0, i, f\_darkyellow, '#');

SetChar(width, i, f\_darkyellow, '#');

}

WritePos(2, height + 1, f\_red, "Первое поколение: [ ] ");

WritePos(2, height + 2, f\_blue, "Второе поколение: [ ] ");

WritePos(2, height + 3, f\_cyan, "Мертвые: [ ] ");

}

void custom\_RandomFirstGen(int \*\*&info, int height, int width)

{

int i, j, k;

for (k = 1; k < 101; k++)

{

i = rand() % 24 + 1;

j = rand() % 49 + 1;

if (info[j][i] != 1)

{

info[j][i] = 1;

SetChar(j, i, b\_red, ' ');

SetPos(0, 0);

}

else

{

k--;

}

}

}

void custom\_Rename(int \*\*&info, int height, int width)

{

int i, j, custom;

for (i = 1; i < height; i++)

{

for (j = 1; j < width; j++)

{

if (info[j][i] == 0)

{

info[j][i] = 3;

SetChar(j, i, b\_black, ' ');

}

else

if (info[j][i] == 2)

{

info[j][i] = 1;

SetChar(j, i, b\_red, ' ');

custom = rand() % 100000 + 1;

if (custom > 0 && custom <= 10)

{

info[j][i] = 4;

SetChar(j, i, b\_green, ' ');

}

if (custom > 10 && custom <= 110) {

info[j][i] = 5;

SetChar(j, i, b\_yellow, ' ');

}

if (custom > 110 && custom <= 210) {

info[j][i] = 6;

SetChar(j, i, b\_magenta, ' ');

}

//

}

}

}

}

void custom\_SecondGen(int \*\*&info, int height, int width)

{

int i, j, k = 0, dth = 0, ch = 0;

for (i = 1; i < height; i++)

{

for (j = 1; j < width; j++)

{

if (info[j][i - 1] == 6 || info[j][i + 1] == 6 || info[j - 1][i] == 6 || info[j + 1][i] == 6 ||

info[j + 1][i + 1] == 6 || info[j + 1][i - 1] == 6 || info[j - 1][i + 1] == 6 || info[j - 1][i - 1] == 6)

{

info[j][i] = 2;

SetChar(j, i, b\_blue, ' ');

k++;

}

else

if (info[j][i] == 3)

{

if (info[j + 1][i + 1] == 1)

{

ch++;

}

if (info[j - 1][i - 1] == 1)

{

ch++;

}

if (info[j + 1][i - 1] == 1)

{

ch++;

}

if (info[j - 1][i + 1] == 1)

{

ch++;

}

//

if (info[j + 1][i] == 1)

{

ch++;

}

if (info[j][i + 1] == 1)

{

ch++;

}

if (info[j - 1][i] == 1)

{

ch++;

}

if (info[j][i - 1] == 1)

{

ch++;

}

if (ch >= 3)

{

info[j][i] = 2;

SetChar(j, i, b\_blue, ' ');

k++;

}

ch = 0;

}

}

}

SetColor(f\_blue);

SetPos(38, height + 2);

printf(" ");

SetPos(40, height + 2);

printf("%d", k);

}

void custom\_DeathGen(int \*\*&info, int height, int width)

{

int i, j, k = 0, dth = 0, ch = 0, death\_ch = 0;

for (i = 1; i < height; i++)

{

for (j = 1; j < width; j++)

{

if (info[j][i] == 1 || info[j][i] == 6)

{

if (info[j + 1][i + 1] == 1 || info[j + 1][i + 1] == 6 || info[j + 1][i + 1] == 4 || info[j + 1][i + 1] == 0)

{

ch++;

}

if (info[j - 1][i - 1] == 1 || info[j - 1][i - 1] == 6 || info[j - 1][i - 1] == 4 || info[j - 1][i - 1] == 0)

{

ch++;

}

if (info[j + 1][i - 1] == 1 || info[j + 1][i - 1] == 6 || info[j + 1][i - 1] == 4 || info[j + 1][i - 1] == 0)

{

ch++;

}

if (info[j - 1][i + 1] == 1 || info[j - 1][i + 1] == 6 || info[j - 1][i + 1] == 4 || info[j - 1][i + 1] == 0)

{

ch++;

}

if (info[j + 1][i] == 1 || info[j + 1][i] == 6 || info[j + 1][i] == 4 || info[j + 1][i] == 0)

{

ch++;

}

if (info[j][i + 1] == 1 || info[j][i + 1] == 6 || info[j][i + 1] == 4 || info[j][i + 1] == 0)

{

ch++;

}

if (info[j - 1][i] == 1 || info[j - 1][i] == 6 || info[j - 1][i] == 4 || info[j - 1][i] == 0)

{

ch++;

}

if (info[j][i - 1] == 1 || info[j][i - 1] == 6 || info[j][i - 1] == 4 || info[j][i - 1] == 0)

{

ch++;

}

if (ch < 2 || ch > 3)

{

info[j][i] = 0;

SetChar(j, i, b\_cyan, ' ');

death\_ch++;

}

ch = 0;

}

else

if (info[j][i] == 5)

{

SetChar(j, i, b\_cyan, ' ');

info[j][i] = 0;

SetChar(j + 1, i, b\_cyan, ' ');

info[j + 1][i] = 0;

SetChar(j, i + 1, b\_cyan, ' ');

info[j][i + 1] = 0;

SetChar(j - 1, i, b\_cyan, ' ');

info[j - 1][i] = 0;

SetChar(j, i - 1, b\_cyan, ' ');

info[j][i - 1] = 0;

SetChar(j + 1, i + 1, b\_cyan, ' ');

info[j + 1][i + 1] = 0;

SetChar(j + 1, i - 1, b\_cyan, ' ');

info[j + 1][i - 1] = 0;

SetChar(j - 1, i + 1, b\_cyan, ' ');

info[j - 1][i + 1] = 0;

SetChar(j - 1, i - 1, b\_cyan, ' ');

info[j - 1][i - 1] = 0;

clear\_Scope(info, height, width);

}

}

}

SetColor(f\_cyan);

SetPos(38, height + 3);

printf(" ");

SetPos(40, height + 3);

printf("%d", death\_ch);

death\_ch = 0;

}

void classic(int \*\*&info, int height, int width)

{

bool f = 0;

interface(info, height, width);

DWORD key = 0;

classic\_RandomFirstGen(info, height, width);

ch(info, height, width);

do

{

if (\_kbhit())

{

key = \_getch();

f = 0;

}

if (key == 32 && f == 0)

{

key = \_getch();

f = 1;

}

else

{

secondGen(info, height, width);

deathGen(info, height, width);

rename(info, height, width);

ch(info, height, width);

}

} while (key != '0' && key != 27);

}

void custom(int \*\*&info, int height, int width)

{

bool f = 0;

interface(info, height, width);

DWORD key = 0;

custom\_RandomFirstGen(info, height, width);

ch(info, height, width);

do

{

if (\_kbhit())

{

key = \_getch();

f = 0;

}

if (key == 32 && f == 0)

{

key = \_getch();

f = 1;

}

else

{

custom\_SecondGen(info, height, width);

custom\_DeathGen(info, height, width);

custom\_Rename(info, height, width);

ch(info, height, width);

}

} while (key != '0' && key != 27);

}

void start(int height, int width)

{

int i, j, \*\*info;

info = new int\*[width + 1];

for (i = 0; i < width + 1; i++)

{

info[i] = new int[height + 1];

}

DWORD key = 0;

clear(info, height, width);

system("CLS");

for (i = 0; i <= height + 7; i++)

{

for (j = 0; j <= width; j++)

{

SetChar(j, 0, f\_yellow | b\_yellow, ' ');

}

SetChar(0, i, f\_yellow | b\_yellow, ' ');

SetChar(width, i, f\_yellow | b\_yellow, ' ');

}

WritePos(15, 13, f\_red, "1. Старт");

WritePos(15, 15, f\_red, "2. Выход");

do{

if (\_kbhit()) key = \_getch();

switch (key)

{

case '1':

{

key = 0;

WritePos(15, 13, f\_red, " ");

WritePos(15, 13, f\_red, "1. Классический режим");

WritePos(15, 15, f\_red, " ");

WritePos(15, 15, f\_red, "2. Режим с доп. настройками");

do{

if (\_kbhit()) key = \_getch();

switch (key)

{

case '1':

{ classic(info, height, width);

start(25, 50);

}

case '2':

{ custom(info, height, width);

start(25, 50);

}

}

} while (TRUE);

}

case '2': exit(0);

}

} while (TRUE);

}

int main()

{

setlocale(LC\_ALL, "Russian");

srand(time(NULL));

int height = 25, width = 50;

if (InitConsole("LifeTime", width + 1, height + 7, b\_black & f\_white))

{

start(25, 50);

}

return 0;

}

### ConsoleWrap.c

#ifdef \_MSC\_VER

#define \_CRT\_SECURE\_NO\_WARNINGS

#endif

#define WIN32\_LEAN\_AND\_MEAN

#include <Windows.h>

#include <stdio.h>

#include <stdlib.h>

#include "ConsoleWrap.h"

static DWORD tmpDword;

static struct

{

CONSOLE\_SCREEN\_BUFFER\_INFOEX propEx;

CONSOLE\_SCREEN\_BUFFER\_INFO prop;

CONSOLE\_FONT\_INFOEX font;

CONSOLE\_CURSOR\_INFO cursor;

HANDLE hIn, hOut;

COORD pos;

}

console;

BOOL InitConsole(LPSTR Title, SHORT x, SHORT y, WORD color)

{

BOOL result;

console.hIn = GetStdHandle(STD\_INPUT\_HANDLE);

console.hOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

result = (console.hIn != INVALID\_HANDLE\_VALUE && console.hOut != INVALID\_HANDLE\_VALUE);

result = SetConsoleTitle(Title);

console.font.cbSize = sizeof(console.font); // (!) structure size needs to be specified first

console.font.nFont = 8;

console.font.dwFontSize.X = 7;

console.font.dwFontSize.Y = 14;

console.font.FontFamily = 54;

console.font.FontWeight = 400;

wsprintfW(console.font.FaceName, L"%s", L"Consolas");

result = result && SetCurrentConsoleFontEx(console.hOut, FALSE, &console.font);

console.propEx.cbSize = sizeof(console.propEx); // (!) structure size needs to be specified first

result = result && GetConsoleScreenBufferInfoEx(console.hOut, &console.propEx);

console.propEx.dwSize.X = console.propEx.srWindow.Right = console.propEx.dwMaximumWindowSize.X = x;

console.propEx.dwSize.Y = console.propEx.srWindow.Bottom = console.propEx.dwMaximumWindowSize.Y = y;

console.propEx.wAttributes = color;

result = result && SetConsoleScreenBufferInfoEx(console.hOut, &console.propEx);

CONSOLE\_FULLSCREEN\_MODE, &NNEW);

console.cursor.bVisible = FALSE;

console.cursor.dwSize = 1;

result = result && SetConsoleCursorInfo(console.hOut, &console.cursor);

// The best is yet to come...

/\*\*/

return result;

}

/\*inline\*/ BOOL SetPos(SHORT x, SHORT y)

{

console.pos.X = x;

console.pos.Y = y;

return SetConsoleCursorPosition(console.hOut, console.pos);

}

/\*inline\*/ BOOL SetColor(WORD color)

{

return SetConsoleTextAttribute(console.hOut, color);

}

/\*inline\*/ BOOL SetChar(SHORT x, SHORT y, WORD color, CHAR ch)

{

if (SetPos(x, y) && SetConsoleTextAttribute(console.hOut, color))

{

putchar(ch);

return TRUE;

}

return FALSE;

}

/\*inline\*/ BOOL FillPos(SHORT x, SHORT y, WORD color, CHAR \_ch, DWORD qty)

{

if (SetPos(x, y))

{

return FillConsoleOutputAttribute(console.hOut, color, qty, console.pos, &tmpDword) && FillConsoleOutputCharacter(console.hOut, \_ch, qty, console.pos, &tmpDword);

}

return FALSE;

}

extern /\*inline\*/ BOOL Write(WORD color, LPSTR string)

{

if (SetConsoleTextAttribute(console.hOut, color))

{

printf(string);

return TRUE;

}

return FALSE;

}

/\*inline\*/ BOOL WritePos(SHORT x, SHORT y, WORD color, LPSTR string)

{

if (SetPos(x, y) && SetConsoleTextAttribute(console.hOut, color))

{

printf(string);

return TRUE;

}

return FALSE;

}

### ConsoleWrap.h

#ifndef \_CONSOLE\_WRAP\_H

#define \_CONSOLE\_WRAP\_H

#include <stdio.h>

#define WIN32\_LEAN\_AND\_MEAN

#include <Windows.h>

// All possible console color combinations

enum colors

{

c\_black = 0x0000, // black foreground and background

b\_black = 0xFF0F, // use it with '& f\_color'as a mask instead of '|'

b\_darkred = BACKGROUND\_RED,

b\_darkgreen = BACKGROUND\_GREEN,

b\_darkblue = BACKGROUND\_BLUE,

b\_darkmagenta = BACKGROUND\_RED | BACKGROUND\_BLUE,

b\_darkyellow = BACKGROUND\_RED | BACKGROUND\_GREEN,

b\_darkcyan = BACKGROUND\_BLUE | BACKGROUND\_GREEN,

b\_grey = BACKGROUND\_RED | BACKGROUND\_BLUE | BACKGROUND\_GREEN,

b\_red = BACKGROUND\_INTENSITY | BACKGROUND\_RED,

b\_green = BACKGROUND\_INTENSITY | BACKGROUND\_GREEN,

b\_blue = BACKGROUND\_INTENSITY | BACKGROUND\_BLUE,

b\_magenta = BACKGROUND\_INTENSITY | BACKGROUND\_RED | BACKGROUND\_BLUE,

b\_yellow = BACKGROUND\_INTENSITY | BACKGROUND\_RED | BACKGROUND\_GREEN,

b\_cyan = BACKGROUND\_INTENSITY | BACKGROUND\_BLUE | BACKGROUND\_GREEN,

b\_white = BACKGROUND\_INTENSITY | BACKGROUND\_RED | BACKGROUND\_BLUE | BACKGROUND\_GREEN,

f\_black = 0xFFF0, // use it with '& b\_color'as a mask instead of '|'

f\_darkred = FOREGROUND\_RED,

f\_darkgreen = FOREGROUND\_GREEN,

f\_darkblue = FOREGROUND\_BLUE,

f\_darkmagenta = FOREGROUND\_RED | FOREGROUND\_BLUE,

f\_darkyellow = FOREGROUND\_RED | FOREGROUND\_GREEN,

f\_darkcyan = FOREGROUND\_BLUE | FOREGROUND\_GREEN,

f\_grey = FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN,

f\_red = FOREGROUND\_INTENSITY | FOREGROUND\_RED,

f\_green = FOREGROUND\_INTENSITY | FOREGROUND\_GREEN,

f\_blue = FOREGROUND\_INTENSITY | FOREGROUND\_BLUE,

f\_magenta = FOREGROUND\_INTENSITY | FOREGROUND\_RED | FOREGROUND\_BLUE,

f\_yellow = FOREGROUND\_INTENSITY | FOREGROUND\_RED | FOREGROUND\_GREEN,

f\_cyan = FOREGROUND\_INTENSITY | FOREGROUND\_BLUE | FOREGROUND\_GREEN,

f\_white = FOREGROUND\_INTENSITY | FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN

};

// --- API prototypes

// TODO: Feel free to include new API-functionality for your needs

// TODO: Some implementations better should be revised and refactored to some common style

BOOL InitConsole(LPSTR Title, SHORT x, SHORT y, WORD color);

//

/\*inline\*/ BOOL SetColor(WORD color);

/\*inline\*/ BOOL SetChar(SHORT x, SHORT y, WORD color, CHAR ch);

/\*inline\*/ BOOL SetPos(SHORT x, SHORT y);

/\*inline\*/ BOOL WritePos(SHORT x, SHORT y, WORD color, LPSTR string);

/\*inline\*/ BOOL FillPos(SHORT x, SHORT y, WORD color, CHAR \_ch, DWORD qty);

/\*inline\*/ BOOL Write(WORD color, LPSTR string);

#endif