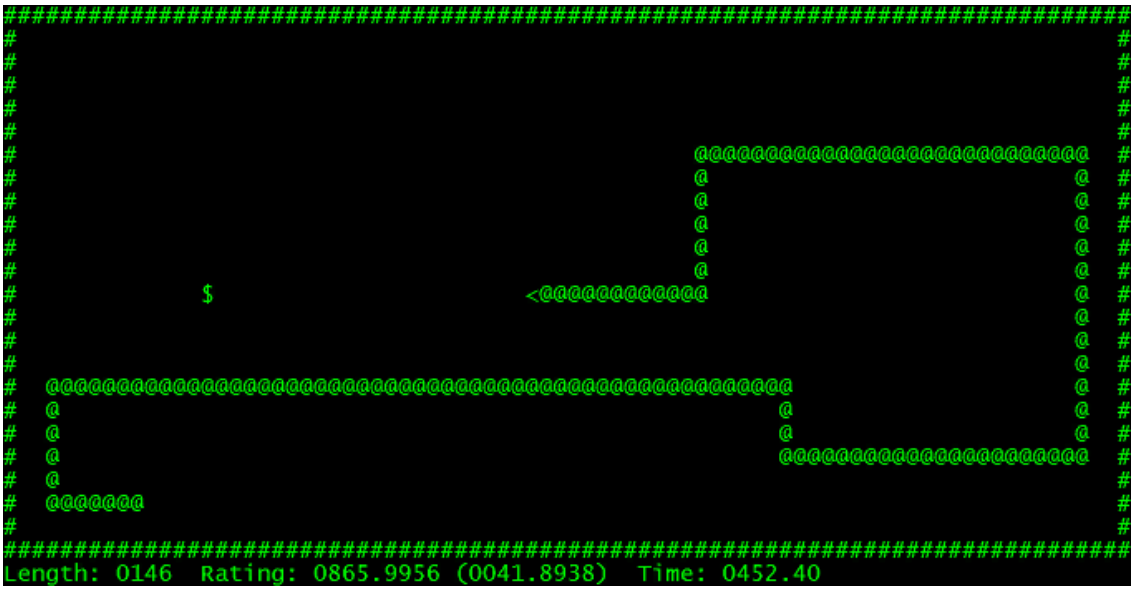
Пишем змейку на C++

Cranium • C/C++ • 2 сентября 2014 • 69 комментариев

Давным-давно, когда мониторы были зелёными, а 64Кб оперативы на борту считалось нормой, существовала игрушка под названием *Snake*. Она также была известна под названиями *Змейка*, *Удав*, *Питон* и даже *Червяк*.
По прошествии времени появилось множество клонов этой игры под различные платформы: от Flash до мобильных телефонов и смартфонов.
Но вот та реализация, работающая в текстовом режиме, видимо умерла вместе с теми компьютерами, для которых она была написана.

И вот, за пару свободных вечеров был написан очередной клон легендарного Snake, который я и представляю вашему вниманию: **Oldschool Snake**.



Как играть

Управление змейкой клавишами управления курсором. Esc — завершение игры. Для выхода из игры надо нажать *Esc* или клавишу *N* на вопрос *«Once more?»*. Змейка не должна натыкаться на стенки и на собственный хвост. Это — смерть. Змейка не умеет ползать хвостом вперёд. Попытаться заставить её это сделать — верная смерть. Кормить змейку надо, естественно, долларами. Когда змейка ест, она растёт.

Top 10 определяется по рейтингу. Общий рейтинг складывается из суммы рейтинговых очков, полученных за каждую съеденную еду. Рейтинговые очки прямо пропорциональны длине змейки и обратно пропорциональны времени, затраченному на достижение очередной порции еды.

Окончание игры

Лицензия

GNU GPL. То есть можно свободно распространять, изучать исходный текст, вносить изменения в исходный текст, использовать в своих некоммерческих проектах.

Замечания по реализации

Игрушка очень простая. Основа игрушки была написана за пару вечеров. Правда потом, наверное, неделя ушла на отладку и, главным образом, на тестирование и допиливание. Вполне возможно, что где-то затаились недобитые баги. Отстрел разрешён.

В этот вариант программы уже заложены некоторые возможности по усовершенствованию игры. Но при этом теряется аутентичность.

Программа написана для **Windows 2000 Professional** (и выше). Для переноса под другие операционки необходимо переписать реализацию класса `CScreen` и иметь порт библиотеки `conio.h`.

Компилировал **TDM-GCC 4.8.1 64-bit**. С другими компиляторами не проверял.

Вопросы, замечание, предложения, ошибки — пожалуйста в комментарии. Но, скажу сразу, не судите строго за стиль — написано было быстро и, что называется, «для себя».

Исходный код

Полный архив с исходниками игры.

main.cpp

```
/*
 * (c) Cranium, aka Черен. 2014
 * GNU GPL
 *
 * Game "Oldschool Snake
 *
 */

#include <iostream>
#include <conio.h>

#include "CScreen.h"
#include "CGame.h"

using namespace std;


int main() {

    setlocale(LC_ALL, "Russian");

    try {
        CScreen screen;
        screen.cursor_show(false);
        screen.text_attr((WORD)0x0a);
        screen.cls();
        CGame game(screen, 80, 24, 120);

        game.logo();

        game.read_top10();
        game.top10(false);
        game.pak(18);

        do {
            game.game_loop();
            game.top10(true);

        } while (game.once_more());

        game.goodbye();
    }
    catch(CSScreenException& ex) {
        cerr << "*** " << ex.what() << endl;
    }

    return 0;
}
```

CScreen.h

```
/*
 * (c) Cranium, aka Черep. 2014
 * GNU GPL
 *
 */

#ifndef __CSCREEN_H__
#define __CSCREEN_H__

#include <windows.h>

/*
Класс исключения для класса CScreen
*/

class CSScreenException {
public:
    CSScreenException(int _err) : err(_err) {}
    const char *what();         // возвращает строку с описанием
ошибки
    int err;                    // код ошибки
};


/*
Класс CScreen содержит системозависимые вызовы для вывода на консоль.

Данная реализация предназначена для ОС MS Windows 2000 Professional
и более поздних.

Система координат:
    (0, 0)  - левый верхний угол экрана
    ось X   - горизонтально вправо
    ось Y   - вертикально вниз (положительное направление)
*/

class CScreen {
public:
    CScreen();
    ~CScreen();
    void cursor_show(bool visible);                 // показать/скрыть
курсор
    void text_attr(WORD attr);                      // установить цвет
текста/фона
    void pos(int x, int y, char ch = 0);            //
позиционирование курсора и
                                                    // вывод символа,
если ch != 0
    void pos_str(int x, int y, const char *str);    //
позиционирование курсора
                                                    // и вывод строки
```

```
    void cls();                                         // очистка экрана

private:
    HANDLE hConsoleOutput;
    CONSOLE_CURSOR_INFO oldCursorInfo, curCursorInfo;
    WORD oldTextAttr;
};


#endif // __CSCREEN_H__
```

CScreen.cpp

```
/*
 * (c) Cranium, aka Черен. 2014
 * GNU GPL
 *
 */

#include "CScreen.h"

#include <conio.h>

const char *msgs[] = {
    "",
    "Failed GetStdHandle(): INVALID_HANDLE_VALUE",
    "Failed GetConsoleCursorInfo()",
    "Failed SetConsoleCursorInfo()",
    "Failed SetConsoleCursorPosition()"
};

const char *CSScreenException::what() {
    return msgs[err];
}

CScreen::CScreen() {
    hConsoleOutput = GetStdHandle(STD_OUTPUT_HANDLE);
    if (hConsoleOutput == INVALID_HANDLE_VALUE)
        throw CSScreenException(1);    // "INVALID_HANDLE_VALUE"

    if (!GetConsoleCursorInfo(hConsoleOutput, &oldCursorInfo))
        throw CSScreenException(2);
    curCursorInfo.dwSize = oldCursorInfo.dwSize;
    curCursorInfo.bVisible = oldCursorInfo.bVisible;

    CONSOLE_SCREEN_BUFFER_INFO csbi;
    GetConsoleScreenBufferInfo(hConsoleOutput, &csbi);
    oldTextAttr = csbi.wAttributes;
}

CScreen::~CScreen() {
    SetConsoleCursorInfo(hConsoleOutput, &oldCursorInfo);
    SetConsoleTextAttribute(hConsoleOutput, oldTextAttr);
```

```
}

void CScreen::cursor_show(bool visible) {
    curCursorInfo.bVisible = visible;
    if (!SetConsoleCursorInfo(hConsoleOutput, &curCursorInfo))
        throw CSScreenException(3);
}

void CScreen::text_attr(WORD attr) {
    SetConsoleTextAttribute(hConsoleOutput, attr);
}

void CScreen::pos(int x, int y, char ch) {
    COORD point;
    point.X = static_cast<SHORT>(x);
    point.Y = static_cast<SHORT>(y);
    if (!SetConsoleCursorPosition(hConsoleOutput, point))
        throw CSScreenException(4);
    if (ch > 0)
        _putch(ch);
}

void CScreen::pos_str(int x, int y, const char *str) {
    pos(x, y);
    _cprintf("%s", str);
}

void CScreen::cls() {
    system("cls");
}
```

SCoord.h

```
/*
 * (c) Cranium, aka Черен. 2014
 * GNU GPL
 *
 */

#ifndef __SCOORD_H__
#define __SCOORD_H__

struct SCoord {
    int x, y;
    SCoord() : x(0), y(0) {}
    SCoord(int _x, int _y) : x(_x), y(_y) {}
    SCoord& operator +=(const SCoord& op);
};

SCoord operator +(const SCoord& op1, const SCoord& op2);
bool operator ==(const SCoord& op1, const SCoord& op2);

#endif // __SCOORD_H__
```

SCoord.cpp

```
/*
 * (c) Cranium, aka Черен. 2014
 * GNU GPL
 *
 */

#include "SCoord.h"

SCoord& SCoord::operator +=(const SCoord& op) {
    x += op.x;
    y += op.y;
    return *this;
}

SCoord operator +(const SCoord& op1, const SCoord& op2) {
    return SCoord(op1.x + op2.x, op1.y + op2.y);
}

bool operator ==(const SCoord& op1, const SCoord& op2) {
    return op1.x == op2.x && op1.y == op2.y;
}
```

CSnake.h

```
/*
 * (c) Cranium, aka Черен. 2014
 * GNU GPL
 *
 */

#ifndef __CSNAKE_H__
#define __CSNAKE_H__

#include <vector>

#include "SCoord.h"
#include "CScreen.h"

using namespace std;

typedef vector<SCoord> CoordVector;

class CSnake {
public:
    CSnake();
    void reset(SCoord start_pos);                   // "сброс" змеи
    void draw(CScreen& scr);                        // первичная
отрисовка змеи на экране
    void move(const SCoord& delta, CScreen& scr);   // передвижение
змеи на приращение координат
    void grow(const SCoord& pos, int growbits);     // увеличение
```

```
длины змеи
    bool into(const SCoord& pos);                    // проверка
попадания координаты в тело змеи
    SCoord head();                                   // метод
возвращает координаты головы змеи
    int size();                                      // метод
возвращает длину змеи

private:
    CoordVector worm;       // вектор координат сегментов тела змеи
    char head_mark;         // символ, которым отрисовывается голова
змеи
    unsigned int drawn;     // длина отрисованного тела змеи
};

#endif // __CSNAKE_H__
```

CSnake.cpp

```
/*
 * (c) Cranium, aka Черен. 2014
 * GNU GPL
 *
 */

#include "CSnake.h"

const char SNAKE_TAIL = '@';        // символ для отрисовки сегментов
тела змеи, кроме головы


CSnake::CSnake() {
    head_mark = '<';
}

void CSnake::reset(SCoord start_pos) {
    worm.clear();
    worm.reserve(1000);         // зарезервировть память
    worm.push_back(start_pos);  // добавить координаты головы
    worm.push_back(start_pos);  // добавить координаты хвоста
    worm[0].x++;                // координата x хвоста - на 1 правее
}

void CSnake::draw(CScreen& scr) {
    unsigned int wsize = worm.size() - 1;
    for (unsigned int i = 0; i < wsize; i++)
        scr.pos(worm[i].x, worm[i].y, SNAKE_TAIL);
    scr.pos(worm[wsize].x, worm[wsize].y, head_mark);
    drawn = worm.size();
}

void CSnake::move(const SCoord& delta, CScreen& scr) {
    // При перемещении змеи перерисовывается только положение головы
```

```
(и первого сегмента)
    // и положение хвоста. Остальные сегменты змеи не
перерисовываются.

    // Перерисовка хвоста.
    // Длина змеи увеличивается, когда змея растёт (метод grow()),
    // но змея на экране не изменяется. Поэтому, если отрисованная
длина змеи
    // совпадает с реальной длиной, то на экране затирается последний
сегмент змеи (хвост).
    // В противном случае, хвост остаётся на месте, голова сдвигается
на единицу,
    // а отрисованная длина увеличивается.
    if (drawn == worm.size())
        scr.pos(worm[0].x, worm[0].y, ' ');
    else
        drawn++;

    // сдвиг координат в векторе без отрисовки
    for (unsigned int i = 1; i < worm.size(); i++)
        worm[i - 1] = worm[i];

    worm[worm.size()-1] += delta;       // координата головы
изменяется на приращение

    // выбор символа для отрисовки головы в зависимости от направления
движения
    if (delta.x < 0)
        head_mark = '<';
    else if (delta.x > 0)
        head_mark = '>';
    else if (delta.y < 0)
        head_mark = 'A';
    else // (delta.y > 0)
        head_mark = 'V';

    // Перерисовка головы и первого сегмента змеи.
    scr.pos(worm[worm.size() - 1].x, worm[worm.size() - 1].y,
head_mark);
    scr.pos(worm[worm.size() - 2].x, worm[worm.size() - 2].y,
SNAKE_TAIL);
}

void CSnake::grow(const SCoord& pos, int growbits) {
    for (int i = 0; i < growbits; ++i)
        worm.insert(worm.begin(), pos);
}

bool CSnake::into(const SCoord& pos) {
    for (unsigned int i = 0; i < worm.size(); i++)
        if (worm[i].x == pos.x && worm[i].y == pos.y)
            return true;
    return false;
}
```

```
SCoord CSnake::head() {
    return worm[worm.size() - 1];
}

int CSnake::size() {
    return static_cast<int>(worm.size());
}
```

CGame.h

```
/*
 * (c) Cranium, aka Черep. 2014
 * GNU GPL
 *
 */

#ifndef __CGAME_H__
#define __CGAME_H__

#include <time.h>
#include <fstream>
#include <utility>

#include "CScreen.h"
#include "CSnake.h"
#include "SCoord.h"

using namespace std;

const int NAMELENGTH = 16;      // размер буфера для имени игрока

// Структура для хранения результата игры

struct SRecord {
    char name[NAMELENGTH];      // имя игрока
    double rating;              // рейтинг
    int length;                 // длина змеи
    double game_time;           // время игры
    time_t date;                // дата и время окончания игры

    SRecord();
    void as_string(char *buffer);   // отформатированная строка
результата
};


class CGame {
public:
    CGame(CScreen& _scr, int _width = 80, int _height = 24, int
_latency = 100);
    void game_loop();           // основной цикл игры
    void top10(bool after_game); // работа с таблицей 10 лучших
```

```
результатов
    bool once_more();           // вывод запроса и приём ответа от
игрока
    void pak(int y);            // "Press any key for continue..."
    void read_top10();          // чтение из файла таблицы 10 лучших
результатов
    void write_top10();         // запись в файл таблицы 10 лучших
результатов
    void logo();                // вывод заставки игры
    void goodbye();             // вывод копирайта по окончании игры

private:
    enum Command { CMD_NOCOMMAND = 0, CMD_EXIT, CMD_LEFT, CMD_RIGHT,
CMD_UP, CMD_DOWN };
    enum State { STATE_OK, STATE_EXIT, STATE_DIED };

    typedef pair<int, Command> CmdPair;

    int width, height;      // ширина и высота игрового поля
    int latency;            // задержка между изменением позиции в
миллисекундах
    CScreen scr;            // подсистема визуализации
    CSnake snake;           // змейка
    double duration_game;   // длительность игры
    double rating, rating_i; // рейтинг итоговый и частичный

    SRecord ttop10[10];     // таблица 10 лучших результатов

    CmdPair cmd_table[5];   // таблица команд управления игрой

    void draw_field();      // прорисовка игрового поля
    SCoord make_food();     // вычисление позиции для еды
    void print_stat();      // вывод текущей статистики ниже игрового
поля
    Command get_command();  // приём команды с клавиатуры
    void top10_table();     // вывод таблицы 10 лучших результатов
};



#endif // __CGAME_H__
```

CGame.cpp

```
/*
 * (c) Cranium, aka Черен. 2014
 * GNU GPL
 *
 */

#include "CGame.h"

#include <iostream>
```

```
#include <cstring>
#include <conio.h>

// форматная строка для форматирования результата игры
const char *recordFormatStr = "%-15s  %9.4f  %4u  %7.2f  %s";

SRecord::SRecord() {
    name[0] = '\0';
    rating = 0.0;
    length = 0;
    game_time = 0;
    date = static_cast<time_t>(0);
}

void SRecord::as_string(char *buffer) {
    sprintf(buffer, recordFormatStr, name, rating, length, game_time,
ctime(&date));
}

ostream& operator << (ostream& os, const SRecord& rec) {
    os
        << rec.rating << ' '
        << rec.length << ' '
        << rec.game_time << ' '
        << rec.date << ' '
        << rec.name << endl;
    return os;
}

istream& operator >> (istream& is, SRecord& rec) {
    is
        >> rec.rating
        >> rec.length
        >> rec.game_time
        >> rec.date;
    is.ignore(1);
    is.getline(&rec.name[0], 16);
    return is;
}

// Функция сравнения результатов по рейтингу.
// Необходима для работы qsort() для сортировки по убыванию.
int rec_compare(const void *_op1, const void *_op2) {
    const SRecord *op1 = reinterpret_cast<const SRecord *>(_op1);
    const SRecord *op2 = reinterpret_cast<const SRecord *>(_op2);
    return static_cast<int>(op2->rating - op1->rating);
}

/*----- end of struct SRecord -----------------------------------*/


// очистка буфера клавиатуры
void clearkeys() {
    while (_kbhit())
```

```
        _getch();
}

// Конструктор
// _scr     - объект, осуществляющий вывод на консоль
// _width   - ширина игрового поля (x)
// _height  - высота игрового поля (y)
// _latency - задержка между изменением позиции в миллисекундах

CGame::CGame(CScreen& _scr, int _width, int _height, int _latency) :
    width(_width), height(_height), latency(_latency), scr(_scr) {

    srand(static_cast<unsigned int>(time(NULL)));

    duration_game = 0;
    rating = 0.0;

    // инициализация таблицы команд управления игрой
    cmd_table[0] = CmdPair(27, CMD_EXIT);   // escape key
    cmd_table[1] = CmdPair('K', CMD_LEFT);  // стрелка влево
    cmd_table[2] = CmdPair('M', CMD_RIGHT); // стрелка вправо
    cmd_table[3] = CmdPair('H', CMD_UP);    // стрелка вверх
    cmd_table[4] = CmdPair('P', CMD_DOWN);  // стрелка вниз
}

CGame::Command CGame::get_command() {
    int ch;

    ch = _getch();
    if (ch == 0 || ch == 0xe0) {
        ch = _getch();
    }

    for (int i = 0; i < 5; i++) {
        if (cmd_table[i].first == ch) {
            return cmd_table[i].second;
        }
    }
    return CMD_NOCOMMAND;
}

// Координата еды вычисляется случайным образом.
// Ограничение: координата не должна попадать в тело змеи.
SCoord CGame::make_food() {
    SCoord food;
    do {
        food.x = rand() % (width - 2) + 1;
        food.y = rand() % (height - 2) + 1;
    } while (snake.into(food));

    return food;
}
```

```
const char BORDER = '#';    // символ для рисования рамки игрового
поля


void CGame::draw_field() {

    scr.cls();

    for (int y = 0; y < height; y++) {
        if (y == 0 || y == height - 1) {
            for (int x = 0; x < width; x++)
                scr.pos(x, y, BORDER);
        }
        else {
            scr.pos(0, y, BORDER);
            scr.pos(width - 1, y, BORDER);
        }
    }
    scr.pos(0, height);
    _cprintf("Length: ****  Rating: ****.**** (****.****)  Time:
****.**");
}


void CGame::print_stat() {
    scr.pos(8, height);
    _cprintf("%04u", snake.size());
    scr.pos(22, height);
    _cprintf("%09.4f", rating);
    scr.pos(33, height);
    _cprintf("%09.4f", rating_i);
    scr.pos(51, height);
    _cprintf("%07.2f", duration_game);
}

void CGame::top10_table() {
    scr.cls();
    char buf[80];

    scr.pos_str(width / 2 - 12, 2, "***** T O P    1 0 *****");
    scr.pos_str(5, 4, "Name              Rating    Length  Time
Date");


    for (int i = 0; i < 10; i++) {
        ttop10[i].as_string(buf);
        scr.pos_str(5, 5 + i, buf);
    }
}

void CGame::top10(bool after_game) {

    char buf[80];
    char buf_encoded[NAMELENGTH];
```

```
    top10_table();      // показать таблицу 10 лучших результатов

    time_t date = time(NULL);
    if (after_game) {
        // если игра была сыграна, то показать текущий результат
        scr.pos(5, 16);
        _cprintf(recordFormatStr, "Your result", rating, snake.size(),
duration_game, ctime(&date));
    }

    if (rating > ttop10[9].rating) {    // если рейтинг игры больше,
чем меньший из 10 лучших...
        // запросить имя игрока
        scr.pos_str(5, 20, "Your name: _");
        scr.pos(16, 20);
        cin.getline(&buf[0], NAMELENGTH);
        clearkeys();
        OemToCharBuff(buf, buf_encoded, static_cast<DWORD>
(NAMELENGTH));
        // заменить последнюю запись в таблице 10 лучших результатов
        strcpy(ttop10[9].name, buf_encoded);
        ttop10[9].date = date;
        ttop10[9].game_time = duration_game;
        ttop10[9].length = snake.size();
        ttop10[9].rating = rating;
        // отсортировать результаты по убыванию
        qsort(ttop10, 10, sizeof(SRecord), rec_compare);
        // обновить таблицу на экране
        top10_table();

        // обновить файл с 10 лучшими результатами
        write_top10();
    }
}

void CGame::pak(int y) {
    scr.pos_str(width / 2 - 15, y, "Press any key for continue...");
    _getch();
    clearkeys();
}

bool CGame::once_more() {
    scr.pos_str(width / 2 - 12, height - 3, "O n c e    m o r e ?");

    int ch = _getch();
    clearkeys();
    if (ch == 'N' || ch == 'n' || ch == 27)
        return false;
    return true;
}

const char *top10_filename = "snake.dat";   // имя файла для хранения
10 лучших результатов
```

```
void CGame::read_top10() {
    ifstream fin(top10_filename);
    if (fin) {
        for (int i = 0; i < 10; i++)
            fin >> ttop10[i];
    }
    fin.close();
}

void CGame::write_top10() {
    ofstream fout(top10_filename);
    if (fout) {
        for (int i = 0; i < 10; i++)
            fout << ttop10[i];
    }
    fout.close();
}

const char *ver_number = "v 1.1";
const char *copyright = "(c) Cranium, 2014.";

void CGame::logo() {
    scr.pos_str(width / 2 - 9, 10, "O l d s c h o o l");
    scr.pos_str(width / 2 - 7, 12, "S  N  A  K  E");
    scr.pos_str(width / 2 - 3, 16, ver_number);
    scr.pos_str(width / 2 - 9, height, copyright);
    pak(22);
}

void CGame::goodbye() {
    scr.cls();
    _cprintf("Oldschool Snake %s\n%s\n", ver_number, copyright);
}


const char FOOD = '$';      // символ для вывода еды

void CGame::game_loop() {

    duration_game = 0;
    rating = rating_i = 0.0;

    draw_field();           // нарисовать игровое поле

    snake.reset(SCoord(width / 2, height / 2));     // установить
змею: длина 2,
                                                    // положение - в
середине игрового поля,
                                                    // направление -
влево
    Command cmd = CMD_NOCOMMAND;
    State stt = STATE_OK;
    // delta  содержит приращение координат (dx, dy) для каждого
```

```
перемещения змеи по полю
    SCoord delta(-1, 0);                // начальное движение - влево
    SCoord food = make_food();          // вычислить координаты еды
    scr.pos(food.x, food.y, FOOD);      // вывести еду на экран

    snake.draw(scr);                    // первичное рисование змеи

    print_stat();                       // вывести начальную
статистику игры

    clock_t time1, time2, duration;
    time1 = clock();

    do {

        if (_kbhit())                   // если в буфере клавиатуры
есть информация,
            cmd = get_command();        // то принять команду

        // обработка команд
        switch (cmd) {
        case CMD_LEFT:
            delta = SCoord(-1, 0);
            break;
        case CMD_RIGHT:
            delta = SCoord(1, 0);
            break;
        case CMD_UP:
            delta = SCoord(0, -1);
            break;
        case CMD_DOWN:
            delta = SCoord(0, 1);
            break;
        case CMD_EXIT:
            stt = STATE_EXIT;
        default:
            break;
        };

        SCoord hd = snake.head();       // координата головы змеи
        hd += delta;                    // координата головы змеи
после приращения (следующая позиция)
        // если голова змеи столкнулась с границей поля или со телом
змеи, то змея умирает
        if (hd.x == 0 || hd.x == width-1 || hd.y == 0 || hd.y ==
height-1 || snake.into(hd))
            stt = STATE_DIED;

        if (stt == STATE_OK) {          // если змея ещё жива, то
            snake.move(delta, scr);     // сдвинуть змею на delta

            if (snake.head() == food) { // если координата головы змеи
совпадает с координатой еды, то
                snake.grow(food, 3);    // увеличить длину змеи
```

```
                food = make_food();      // вычислить координаты новой
еды
                scr.pos(food.x, food.y, FOOD); // вывести еду на экран

                // Вычисление времени игры, частичного и общего
рейтинга.
                // Частичный рейтинг вычисляется как длина змеи,
делённая на время в секундах,
                // затраченное на подход к еде (время от съедения
предыдущей еды до съедения следующей).
                // Таким образом, чем чаще змея ест и чем она длиннее,
тем выше частичный рейтинг.
                time2 = clock();
                duration = time2 - time1;
                duration_game += static_cast<double>(duration) /
CLOCKS_PER_SEC;
                rating_i = static_cast<double>(snake.size()) /
duration * CLOCKS_PER_SEC;
                rating += rating_i;      // общий рейтинг - сумма
частичных рейтингов за игру
                time1 = time2;

                print_stat();            // вывод текущей статистики
игры
            }

            Sleep(latency);              // задержка перед следующим
изменением позиции
        }

    } while (stt == STATE_OK);           // играем, пока змея жива

    scr.pos_str(width / 2 - 8, 10, " G a m e    o v e r ");
    clearkeys();
    _getch();
    clearkeys();
}
```

Автор статьи: Череп.

Рейтинг — 3.8 Голосов — 19

← Преобразование времени в устный формат на Javascript

Многомерные массивы в C++ — практическое пособие →

Рекомендованные статьи

- Многомерные массивы в C++ — практическое пособие
- Dev-C++ 5 — легкая среда разработки программ на C++
- Введение в Windows Forms — пишем первую программу
- Русский текст в строках на C++
- Русский язык в консоли C++
- Первая программа на C++ — урок 1
- Метод Зейделя на C++

- **Вычисление факториала на C++**
- **Переменные и типы данных в C++ — урок 2**
- **Продолжаем изучать классы — урок 11**

Обсудите статью на форуме.

Комментарии к статье: 69

Показать предыдущие 10 комментариев

Вадим • 11 ноября 2017

Спасибо!

OneMoreTime • 1 декабря 2017

что мне делать?
[Error] ld returned 1 exit status

Тата • 28 декабря 2017

Может вопрос не по теме, но очень нужна помощь. Надо чтоб игра работала как под виндовз так и под линукс. С виндовз проблемм нет — все работает. В линуксе на строке include <conio.h> выдает ошибку. Код игры не Ваш — другой, компиляция в code::blocks.
Я знаю, вы им не пользуетесь....но хоть какая идея?

MasterOfAlteran • 28 декабря 2017

Тата, надо использовать кроссплатформенные средства.

Типо прграммист • 12 февраля 2018

код довольно прост...
хотя и кажется большим,а так норм сама игра мне влилась в душу и поэтому я и решил воссоздать старую добрую змейку!!!!

Roman • 22 февраля 2018

Выдает ошибку 4996.
Как решил:

```
1) Правой кнопкой по файлу CGame.cpp
2) Свойства
3) c/c++  -> Дополнительно  -> Отключить указанные
предупреждения -> Изменить
4) В пустое окошко вписать 4996 -> OK -> "применить".
```

---

sam • 22 февраля 2018

> Выдает ошибку 4996.

А в каком месте-то выдает?

---

roman • 26 февраля 2018

при кмпиляции

---

sam • 27 февраля 2018

А я, сцуко, всё думал, где ж это выдает ошибку 4996? Ты мне прям глаза открыл.

Aycon • 2 ноября 2018

Я ещё могу представить что змейку надо кормить долларами, но какого чёрта твоя змейка из собак XD

Возможность комментировать эту статью отключена автором. Возможно, во всем виновата её провокационная тематика или большое обилие флейма от предыдущих комментаторов.

Если у вас есть вопросы по содержанию статьи, рекомендуем вам обратиться за помощью на наш форум.