

Министерство науки и высшего образования Российской
Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу

«Программирование»

на тему «Игра

«Змейка»»

20.11.22

Выполнил:

студент группы
21BO2 Сетяев А.Н.

Принял:

Митрохин М.А.

Пенза 2022

«УТВЕРЖДАЮ»
Зав. кафедрой В.
Митрохин М. А.
«___» _____ 2022

ЗАДАНИЕ
на курсовое проектирование по курсу

Программирование
Студенту Сетяеву Артему Николаевичу Группа 21ВО2
Тема проекта: Компьютерная игра «Змейка»

Исходные данные (технические требования) на проектирование
Наименование программы: Компьютерная игра «Змейка».

Задание. Программа должна выполнять следующие функции:

- Подсказки, пауза, выход из игры по нажатию клавиш.
- Наличие заставки.
- Наличие графического интерфейса.

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных — списков, структур, файлов.
3. Режим работы видеосистемы — текстовый/графический.
4. Устройство ввода информации — клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система — Windows.
8. Система программирования — C/C++ на базе среды Microsoft Visual Studio 2019.
9. Язык программирования — Си.

Объем работы по курсу

1. Расчетная часть

2. Графическая часть

3. Экспериментальная часть

Срок выполнения проекта по разделам

1 _____

2 _____

3 _____

4 _____

5 _____

6 _____

7 _____

8 _____

Дата выдачи задания «13» Сентября 2022

Дата защиты проекта «29» 12 22

Руководитель _____

Задание получил «___» _____ 20__ г.

Студент Сетяев Артем Николаевич

Оглавление

Введение.....	4
1. Постановка задачи.....	5
2. Выбор решения.....	6
2.1 Этапы разработки программы.....	6
2.2 Схема данных.....	7
3. Описание разработки программы.....	8
3.1 Спецификация на программные модули	8
4. Отладка и тестирование	9
5. Описание программы.....	10
5.1 Схема работы системы	10
5.2 Схема ресурсов системы	11
5.3 Состав модулей и функций	11
Barfic.h.....	11
Data.h	10
Date.h	10
5.4 Схема взаимодействия модулей	21
5.5 Иерархическая структура программы	22
5.6 Схема программы (функция main)	23
6. Руководство пользователя.....	27
Заключение.....	28
Приложение1. Листинг программы	29
Файл Barfic.h.....	29
Файл Console.cpp	35
Файл Cursor.cpp	36
Файл.cpp.....	36
Файл Dialog.cpp	37
Файл Draw.cpp	47
Файл InputRecord.cpp	53
Файл MainMenu.cpp.....	53
Файл Snake_by_setyaev.cpp.....	57
Приложение 2. Результаты работы программы	
Приложение 3. Список используемой литературы	70

Введение

В наш век бурно-развивающихся информационных технологий всё большее и большее значение приобретают специалисты в компьютерной области знаний. Путь к становлению себя как такого специалиста пролегает через выполнение множества учебно-практических задач, полезных и не очень в практическом смысле, но, безусловно, полезных в смысле получения опыта, проектов.

Одной из таких задач для всех студентов, стремящихся получить большое количество опыта и практики в сфере информационных технологий и поступивших для реализации своих стремлений на специальность «Вычислительные Машины, Комплексы, Системы и Сети» Пензенского государственного Университета, стала курсовая работа по дисциплине «Алгоритмические Языки и Программирование».

В этой пояснительной записке я постараюсь максимально полно раскрыть все этапы, шаги и нюансы выполнения мной этой курсовой работы.

История игры «Змейка» насчитывает более тридцати лет. Первая игра змейка была игровым автоматом Hustle, выпущенный фирмой Gremlin в 1977 г. Возможно, первая реализация на домашнем компьютере была сделана в 1979 году на компьютере TRS-80. После неё вышла версия на Commodore VIC-20, под названием Worms. Микрокомпьютерная версия игры Hustle была выпущена в 1980 году. При написании кода было предварительно подробно изучена предметная область, а также основные требования к новой программе. Интерфейс пользователя разработан так, что бы пользование программой было максимально быстрым и удобным.

Я считаю, что выполнение курсового задания принесло мне огромное количества незаменимого опыта в алгоритмизации поставленных целей, оптимизации кода, знании тонкостей языка C.

1. Постановка задачи

Программа «Змейка» представляет собой с детства знакомую всем игру, в которой маленькая змейка собирает яблоки и с каждым яблоком тело змейки становится больше.

Запуск программы должен осуществляться с помощью нажатия кнопки «Новая игра» на панели меню, которая изображена на рисунке 1.

Управление данной игры интуитивно простое, направление движения змейки осуществляется с помощью клавиш «Вперёд», «Назад» , «Влево» и «Вправо».

«Смерть» змейки или конец игры наступает в том случае, если змейка столкнётся со стенкой ,либо врежется в себя. «Смерть» змейки изображена на рисунке 4.

Выход из игры осуществляется при помощи кнопки «Выход» (на панели меню), либо при нажатии клавиши “Esc” (непосредственно во время игры).

2. Выбор решения

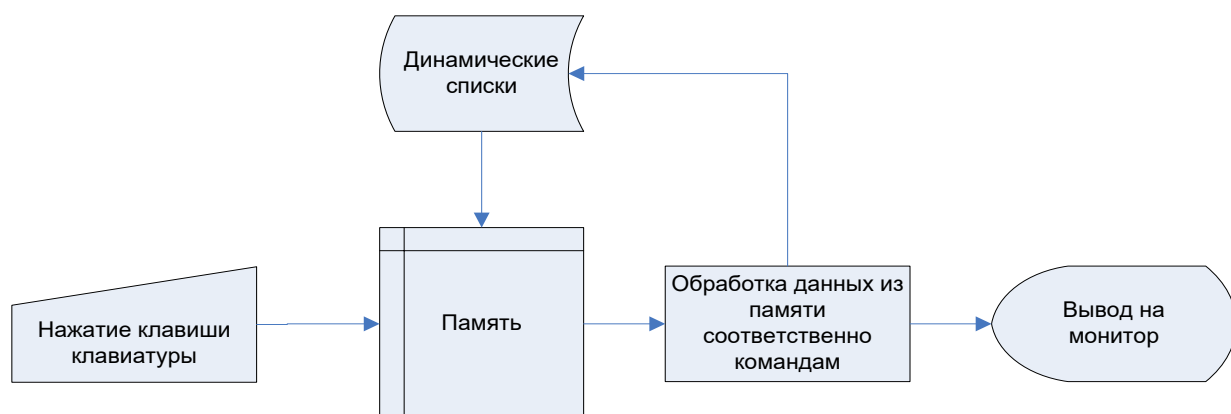
2.1 Этапы разработки программы

Условно процесс подготовки курсового проекта можно разделить на следующие этапы:

1. Постановка задачи, осмысление программы, определение её функционала, деление программы на подпрограммы, отдельные меню, составление структуры программы.
2. Планирование схем подпрограмм, определение их функций.
3. Написание кода программы без отображения графических элементов, проверка корректной работы функций обработки данных, отслеживание исключительных ситуаций.
4. Поиск справочной литературы, изучение найденного материала, применение полученных навыков создания пользовательского интерфейса с использованием псевдографики, отрисовка основных элементов интерфейса
5. Отладка, тестирование, верификация программы.

Основной цикл программы работает событийно, исполнение программы происходит по командам пользователя. В зависимости от нажатой клавиши мыши или клавиатуры будут выполняться те или иные операции. Выход из общего цикла также осуществляется по нажатию определенной клавиши (escape).

2.2 Схема данных



По осуществлении обработки событий клавиатуры, выполняется определенный набор операций, связанных либо с обработкой данных пользователя (добавление новой, изменение уже существующей, удаление ненужной записи), либо с изменением текущих параметров программы (выбираемой даты и пр.). При выполнении операций над данными, изменяются соответствующие файлы.

3. Описание разработки программы

3.1 Спецификация на программные модули

Для упорядочивания логических связей, программа была разделена на ряд тематических модулей, реализующих ту или иную часть программы:

Console.cpp – функции инициализации работы с консолью вывода

Cursor.cpp – функции для работы с курсором

Data.cpp – функции обработки данных

Date.cpp – функции обработки даты

Dialog.cpp – функции диалогов

Draw.cpp – функции отрисовки

InputRecord.cpp – функции ввода

MainMenu.cpp – функции обработки главного меню

Snake_by_setyaev.cpp – основной файл, где содержатся функции игрового процесса и функция main.

Функции реализовывались в последовательности, обозначенной ранее. В первую очередь были написаны и отлажены функции обработки данных: удаления, добавление, поиска нужной записи в файле. Важной особенностью программы является тот факт, что пользователь имеет возможность делать записи данных практически любой длины. Соответственно при записи данных требуется отдельно хранить значение длины записываемых данных.

Вместе с этим реализовывался механизм обработки событий клавиатуры и мыши, что в итоге привело к организации вполне удобного и интерактивного интерфейса.

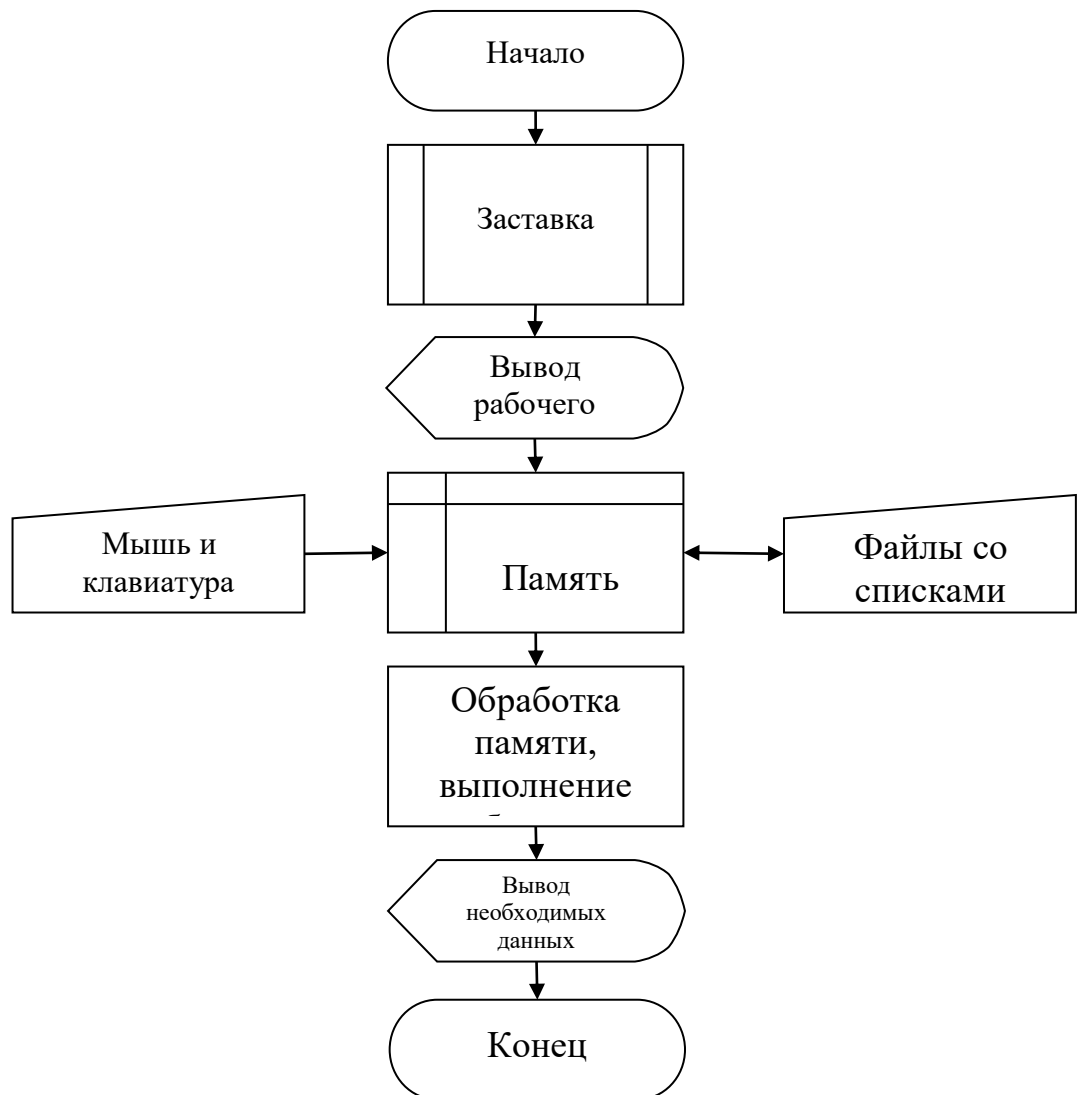
4. Отладка и тестирование

В ходе отладки использовалась возможность пошаговой отладки с целью отслеживания изменения параметров и данных системы. Этот метод позволял наиболее быстро и удобно находить и исправлять ошибки.

Одним из основных требований к любой программе является ее способность корректно решать задачу при любых корректных наборах условий. Поэтому тестирование игры проводилось несколькими лицами. Это позволило выявить ошибки в программе и исправить алгоритмическую составляющую программы с целью получения универсального кода. Кроме того предусмотрена обработка событий, несвязанный с алгоритмическими особенностями программы.

5. Описание программы

5.1 Схема работы системы



5.2 Схема ресурсов системы



5.3 Состав модулей и функций

В ходе разработки этой программы был введен ряд функций. Их условно можно разделить на 3 вида: функции визуализации и интерфейса, функции обработки данных и функции игрового процесса.

Barfic.h

Содержит некоторые именованные константы и глобальные переменные, а также прототипы всех функций интерфейса.

Console.cpp

`BOOL ConsoleSetScreeSize(unsigned nWidth, unsigned nHeight).`

Изменяет размер консоли. Ширина становится равной `nWidth`, высота – `nHeight`. Возвращает `TRUE` при успешном завершении и `FALSE` при ошибке.

`BOOL ConsoleInitialize(void).` Инициализирует консоль вывода. Внутри – вызывается `ConsoleSetScreeSize()`, устанавливает размеры экрана 155 на 70.

InputRecord.cpp

`BOOL InputRecordRead(OUT INPUT_RECORD& rRecord).` Заполняет структуру `INPUT_RECORD`, указатель на которую передан на вход, информацией о событии ввода. Возвращает `TRUE`, если операция прошла успешно.

Cursor.cpp

`BOOL SetCursorVisible(IN BOOL fVisible).` Устанавливает видимость курсора в соответствии с переданным флагом `fVisible`. Возвращает `TRUE`, если операция прошла успешно.

Dialog.cpp

```
EMessageBoxId
DialogMessageBox(
    IN EColor eDialogForeColor,
    IN EColor eDialogBackColor,
    IN unsigned nX,
    IN unsigned nY,
    IN unsigned nWidth,
    IN unsigned nHeight,
    IN EColor eTextForeColor,
    IN EColor eTextBackColor,
    IN const wchar_t* szText,
    IN EAlignment eTextAlignment,
    IN EMessageBoxType eMessageBoxType
);
```

Отрисовывает диалговое окно в соответствии с переданной информацией о его цвете, размере и типе. Начинает цикл опроса устройств ввода, по завершении которого возвращает результат – код нажатой пользователем кнопки.

Draw.cpp

```
void
SetCharInfo(
    OUT CHAR_INFO* pCharInfo,
    IN wchar_t wcChar,
    IN EColor eForeColor,
    IN EColor eBackColor
);
```

Заполняет структуру CHAR_INFO по адресу pCharInfo переданной в функцию информацией о цвете переднего и заднего плана и начертании символа (кодировка – Юникод).

```
BOOL
DrawPoint(
    IN CHAR_INFO* pCharInfo,
    IN unsigned nX,
    IN unsigned nY
);
```

Выводит символ, информация о котором находится в структуре CHAR_INFO по адресу pCharInfo в позиции с координатами (nX, nY).

```
BOOL
DrawHorLine(
    IN CHAR_INFO* pCharInfo,
    IN unsigned nX,
    IN unsigned nY,
    IN unsigned nLength
);
```

);

Заполняет символом, информация о котором находится в структуре CHAR_INFO по адресу pCharInfo, горизонтальную линию длины nLength начиная с позиции (nX,nY) .

```
void
DrawRectangle(
    IN  CHAR_INFO*  pCharInfo,
    IN  unsigned    nX,
    IN  unsigned    nY,
    IN  unsigned    nWidth,
    IN  unsigned    nHeight
);
```

Заполняет символом, информация о котором находится в структуре CHAR_INFO по адресу pCharInfo, прямоугольную область шириной nWidth и высотой nHeight начиная с позиции (nX,nY) .

```
void
DrawWindow(
    IN  EColor      eForeColor,
    IN  EColor      eBackColor,
    IN  unsigned    nX,
    IN  unsigned    nY,
    IN  unsigned    nWidth,
    IN  unsigned    nHeight
);
```

Заполняет цветом eBackColor прямоугольную область шириной nWidth и высотой nHeight, начиная с позиции (nX,nY), отрисовывает по краям рамку цвета eForeColor.

```
void
DrawSingleLineText(
    IN  const wchar_t*  szText,
    IN  EColor          eForeColor,
    IN  EColor          eBackColor,
    IN  unsigned        nX,
    IN  unsigned        nY,
    IN  unsigned        nWidth,
    IN  unsigned        nHeight,
    IN  EAlignment      eAlignment
);
```

Заполняет цветом eBackColor прямоугольную область шириной nWidth и высотой nHeight начиная с позиции (nX,nY), выводит текст из строки szText поверх неё в соответствии с выравниванием.

```

void
DrawButton(
    IN EColor      eForeColor,
    IN EColor      eBackColor,
    IN EColor      eBack2Color,
    IN unsigned    nX,
    IN unsigned    nY,
    IN unsigned    nWidth,
    IN unsigned    nHeight,
    IN const wchar_t* szText
);

```

Отрисовывает кнопку с текстом (по аналогии с DrawSingleLineText) и тенью цвета eBack2Color.

```

void InputData(
    const wchar_t* Ask,
    wchar_t* buf,
    size_t buf_sz
);

```

Вывод на экран окошка для ввода данных. Ask – приглашение, buf – юникодовый буфер, buf_sz – размер буфера.

```

void OutputSingleList(
    Output* List,
    int shift_down
);

```

Вывод на экран одной записи из сформированного списка вывода List со смещением вниз shift_down.

Группа функций ирового процесса

Snake_by_golyshevsky.cpp

```

struct Game //данные-точности: змейки, яблок, передвижение по X
и Y, задержка, направление
{
    Zmeja gaduka; //змейка
    COORD jabloko; //яблоко
    int dx,dy, //передвижение
    pause; //задержка
    int nap; //направление
};

```

Структура описывает данные точности.

```

void PlusJabloko(Game &g)
{

```

```

int i,x,y;
int n = g.gaduka.PCount;
do
{
    x = rand() % 56+3; //
    y = rand() % 19+3; //координаты яблока
    for(i = 0; i < n; i++)
    {
        if(x == g.gaduka.t[i].X && y == g.gaduka.t[i].Y)
// проверка чтоб яблоко не бросить на змею
        break;
    }
}
while(i < n);
g.jabloko.X = x; //
g.jabloko.Y = y; //запоминаем позицию яблока
SetConsoleCursorPosition(hConsole, g.jabloko); //переносим
курсор в эту позицию
SetConsoleTextAttribute(hConsole, 0x0c); //цвет яблока
printf("%c", 4); //рисует яблоко каким хотим символом

}

```

Функция разброски яблок по игровому полю.

```

void skorostGame(Game &g)
{
    system("cls");
    g.gaduka.PCount = 3; //сколько точек в змейки
    g.gaduka.t = new COORD [3]; //создали точки
    for(int i = 0; i < 3; i++)
    {
        g.gaduka.t[i].X = 20 + i;
        g.gaduka.t[i].Y = 20;
    }
    g.dx = 1;
    g.dy = 0;
    g.pause = 100; //скорость передвижение змеи
    PlusJabloko(g); //рисует яблока
}

```

Функция старта змейки, её координат и скорости.

```

void STENA_2()
{
    SetColor(LightBlue , Black); GotoXY(20,0); cout << "Game
\"Snake by Setyaev\""" << endl;
    GotoXY(64,2); cout << "Behaviours:" << endl ;
    GotoXY(64,3); cout << "Apples:0" << endl ;
    GotoXY(64,4); cout << "Length:3" << endl;
    GotoXY(64,5); cout << "Speed:0" << endl;
    GotoXY(64,7); cout << "Controls:" << endl;
    GotoXY(64,8); cout << "Esc:Exit" << endl;
    GotoXY(64,9); cout << "P:Pause" << endl;
}

```

```

        GotoXY(64,10); cout <<"S:Start" << endl;
        GotoXY(64,11); cout <<"L:Level" << endl;
        GotoXY(64,13);printf("%c",24);cout
<<":Up"<<endl;

        GotoXY(64,14);printf("%c",25);cout<<":Down"<<endl;

        GotoXY(64,15);printf("%c",27);cout<<":Left"<<endl;

        GotoXY(64,16);printf("%c",26);cout<<":Right"<<endl;
        {SetColor(LightMagenta , Black);
        GotoXY(2,2);
        int m = 0;
        for(m = 0; m < 60; m++)
        {
            printf("*");
        }

        {
            GotoXY(2,24);
            int m = 0;
            for(m = 0; m < 60;m++)
            {
                printf("*");
            }
        }

        {
            GotoXY(2,3); cout << "*"<<endl;
            GotoXY(2,4); cout << "*"<<endl;
            GotoXY(2,5); cout << "*"<<endl;
            GotoXY(2,6); cout << "*"<<endl;
            GotoXY(2,7); cout << "*"<<endl;
            GotoXY(2,8); cout << "*"<<endl;
            GotoXY(2,9); cout << "*"<<endl;
            GotoXY(2,10); cout << "*"<<endl;
            GotoXY(2,11); cout << "*"<<endl;
            GotoXY(2,12); cout << "*"<<endl;
            GotoXY(2,13); cout << "*"<<endl;
            GotoXY(2,14); cout << "*"<<endl;
            GotoXY(2,15); cout << "*"<<endl;
            GotoXY(2,16); cout << "*"<<endl;
            GotoXY(2,17); cout << "*"<<endl;
            GotoXY(2,18); cout << "*"<<endl;
            GotoXY(2,19); cout << "*"<<endl;
            GotoXY(2,20); cout << "*"<<endl;
            GotoXY(2,21); cout << "*"<<endl;
            GotoXY(2,22); cout << "*"<<endl;
            GotoXY(2,23); cout << "*"<<endl;
        }

        {
            GotoXY(61,3); cout << "*"<<endl;
            GotoXY(61,4); cout << "*"<<endl;

```

```

        GotoXY(61,5); cout << "*"<<endl;
        GotoXY(61,6); cout << "*"<<endl;
        GotoXY(61,7); cout << "*"<<endl;
        GotoXY(61,8); cout << "*"<<endl;
        GotoXY(61,9); cout << "*"<<endl;
        GotoXY(61,10); cout << "*"<<endl;
        GotoXY(61,11); cout << "*"<<endl;
        GotoXY(61,12); cout << "*"<<endl;
        GotoXY(61,13); cout << "*"<<endl;
        GotoXY(61,14); cout << "*"<<endl;
        GotoXY(61,15); cout << "*"<<endl;
        GotoXY(61,16); cout << "*"<<endl;
        GotoXY(61,17); cout << "*"<<endl;
        GotoXY(61,18); cout << "*"<<endl;
        GotoXY(61,19); cout << "*"<<endl;
        GotoXY(61,20); cout << "*"<<endl;
        GotoXY(61,21); cout << "*"<<endl;
        GotoXY(61,22); cout << "*"<<endl;
        GotoXY(61,23); cout << "*"<<endl;
    }
}

```

Функция создаёт ограждение игрового поля (стену) и выводит на экран характеристики игры и параметры управления.

```

int Move(Game &g)
{
    int & n = g.gaduka.PCount;
    COORD head = g.gaduka.t[n - 1]; //голова
    COORD tail = g.gaduka.t[0]; //хвост
    COORD next;
    next.X = head.X + g.dx;
    next.Y = head.Y + g.dy; //проверка следующей точки по
направлению
    if(next.X < 3 || next.Y < 3 || next.X > 60 || next.Y >
23) //не уперлась ли в стену?

        return STENA;
    if(n > 4)
    {
        for(int i = 0; i < n; i++)
            if(next.X == g.gaduka.t[i].X && next.Y ==
g.gaduka.t[i].Y) //не наехали ли на себя?
                return KONEC;
    }

    if(next.X == g.jabloko.X && next.Y == g.jabloko.Y)
    {
        COORD*temp = new COORD[ ++n ]; //новый массив больший на
1
        for(int i = 0; i < n; i++)
            temp[i] = g.gaduka.t[i]; //перекопируем

```

```

        temp[n - 1] = next; //добавляем одну
        delete [] g.gaduka.t;
        g.gaduka.t = temp;

        SetConsoleCursorPosition(hConsole, head);
        SetConsoleTextAttribute(hConsole, 0x0a); //закрашиваем
яблоко которое съели
        printf("*");
        SetConsoleCursorPosition(hConsole, next);
        SetConsoleTextAttribute(hConsole, 0x0a);
        printf("%c", 1);
        PlusJabloko(g);
        return PLUS;
    }

    for(int i = 0; i < n - 1; i++)
        g.gaduka.t[i] = g.gaduka.t[i + 1];
    g.gaduka.t[n - 1] = next;
    SetConsoleCursorPosition(hConsole, tail); //закрашиваем
хвостик
    printf(" ");

    SetConsoleCursorPosition(hConsole, head);
    SetConsoleTextAttribute(hConsole, 0x0a); //красим хвост змеи
в зеленсй цвет
    printf("*");
    SetConsoleCursorPosition(hConsole, next);
    SetConsoleTextAttribute(hConsole, 0x0f); //красим курсор в
белый цвет (голову змеи)
    printf("%c", 1);

    return MOVE;
}

```

Функция, благодаря которой змейка движется по игровому полю.

```

void StartGame()
{
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE); //получаем
дескриптор консоли
    int key = 0, count = 0;
    bool Pause=false;
    Game g;
    skorostGame(g);
    STENA_2();
    srand(time(0));
    bool pause = false;
    while(key != 27)
    {
        while(!kbhit()) //ждет пока нажмем
        {
            if(Pause==true)
            {
                Sleep(1);
            }
        }
    }
}

```

```

        continue;
    }

    switch (Move(g)) //движение
    {

        case PLUS:
            ++count;
            g.pause-=1;
            GotoXY(64,2);cout << "Behaviours:" << endl ;
            GotoXY(64,3); cout << "Apples:" <<count <<
endl;
            GotoXY(64,4); cout << "Length:"
<<g.gaduka.PCount << endl;
            GotoXY(64,5); cout << "Speed:" <<g.pause<<
endl;
            GotoXY(64,7); cout << "Controls:" << endl;
            GotoXY(64,8); cout << "Esc:Exit" << endl;
            GotoXY(64,9); cout << "P:Pause" << endl;
            GotoXY(64,10); cout <<"S:Start" << endl;
            GotoXY(64,11); cout <<"L:Level" << endl;
            GotoXY(64,13);printf("%c",24);cout
<<":Up"<<endl;

            GotoXY(64,14);printf("%c",25);cout<<":Down"<<endl;

            GotoXY(64,15);printf("%c",27);cout<<":Left"<<endl;

            GotoXY(64,16);printf("%c",26);cout<<":Right"<<endl;
            if (count == 75)
            {
                GotoXY(15,1); cout <<
"Congratulations!!!You win!!!!" << endl;
                getch();
                getch();
                getch();
                getch();
                getch();
                return;
            }
            break;

        case STENA:

        case KONEC:
            GotoXY(23,1); printf("Ohhh!!You loose...it's
loss of losses!\n\n\t\t\t");
            getch();
            return;
            break;
    }

    Sleep(g.pause); //Задержка

```

```

    }
    key = getch();

    if(key=='P' || key=='p')
        Pause=!Pause;
    else if(key=='S' || key=='s')
        ZmejaStart();
    else if(key=='L' || key=='l')
        Level();
    else if(key==0 || key==224)
    {
        key=getch();

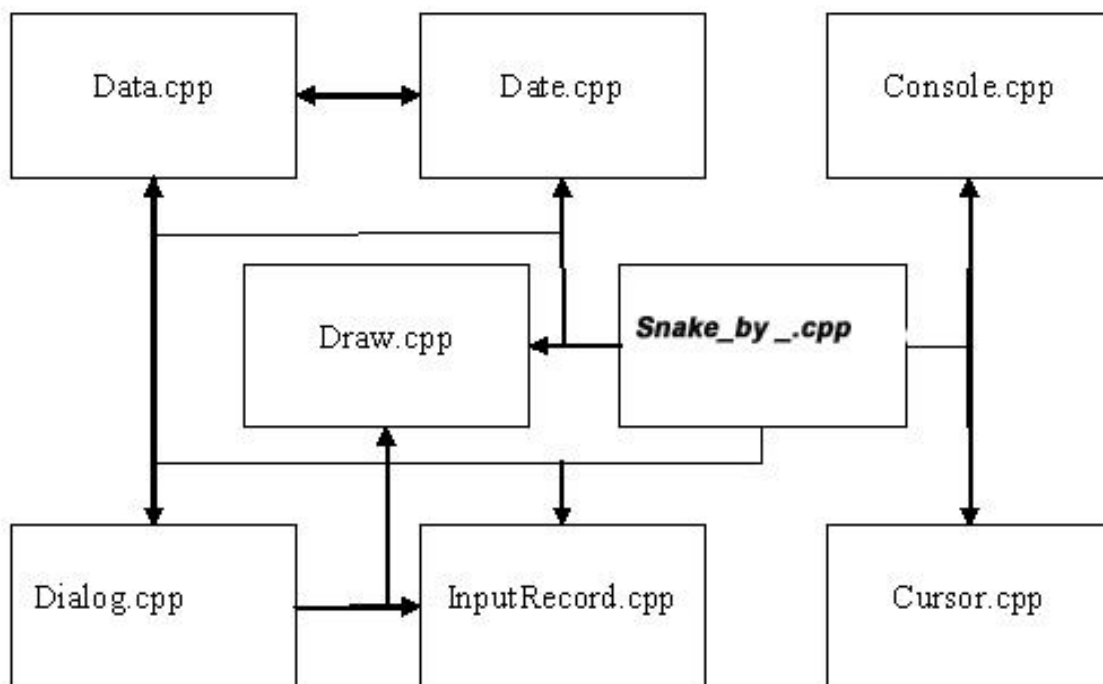
        if(key == 72 && g.nap != DOWN)
        {
            g.nap = UP;
            g.dx = 0;
            g.dy = -1;
        }
        else if(key == 80 && g.nap != UP)
        {
            g.nap = DOWN;
            g.dx = 0;
            g.dy = 1;
        }
        else if(key == 75 && g.nap != RIGHT)
        {
            g.nap = LEFT;
            g.dx = -1;
            g.dy = 0;
        }
        else if(key == 77 && g.nap != LEFT)
        {
            g.nap = RIGHT;
            g.dx = 1;
            g.dy = 0;
        }
    }
}

}

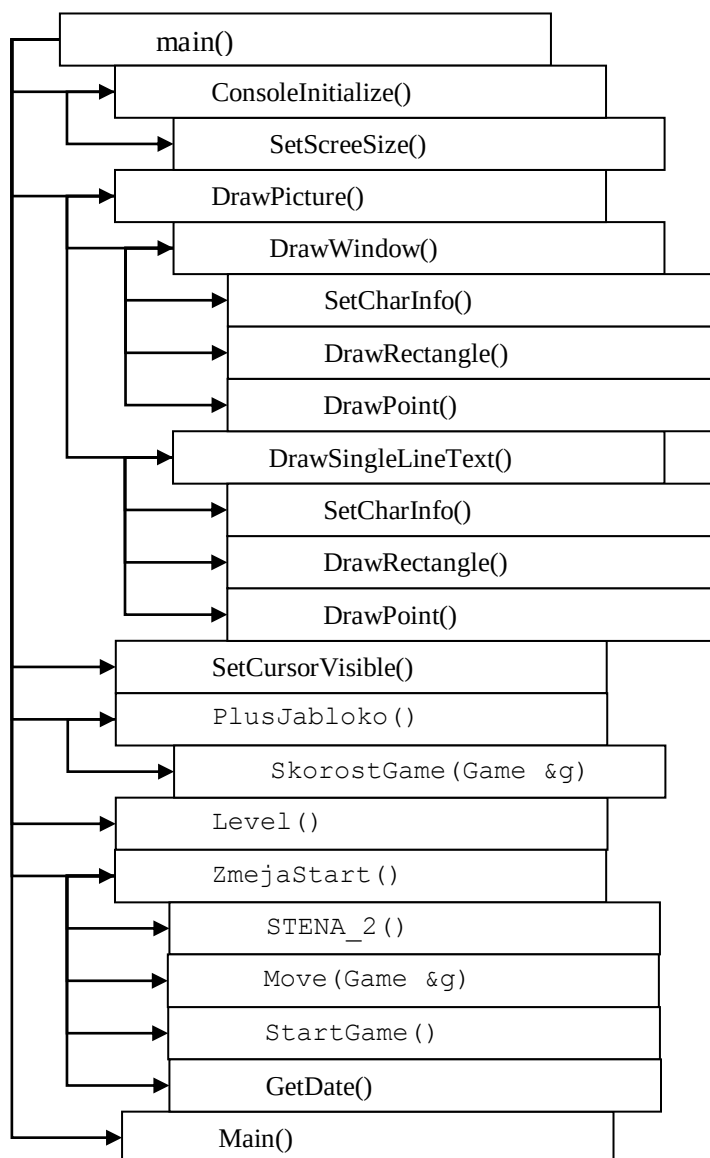
```

Функция осуществляет запуск игры.

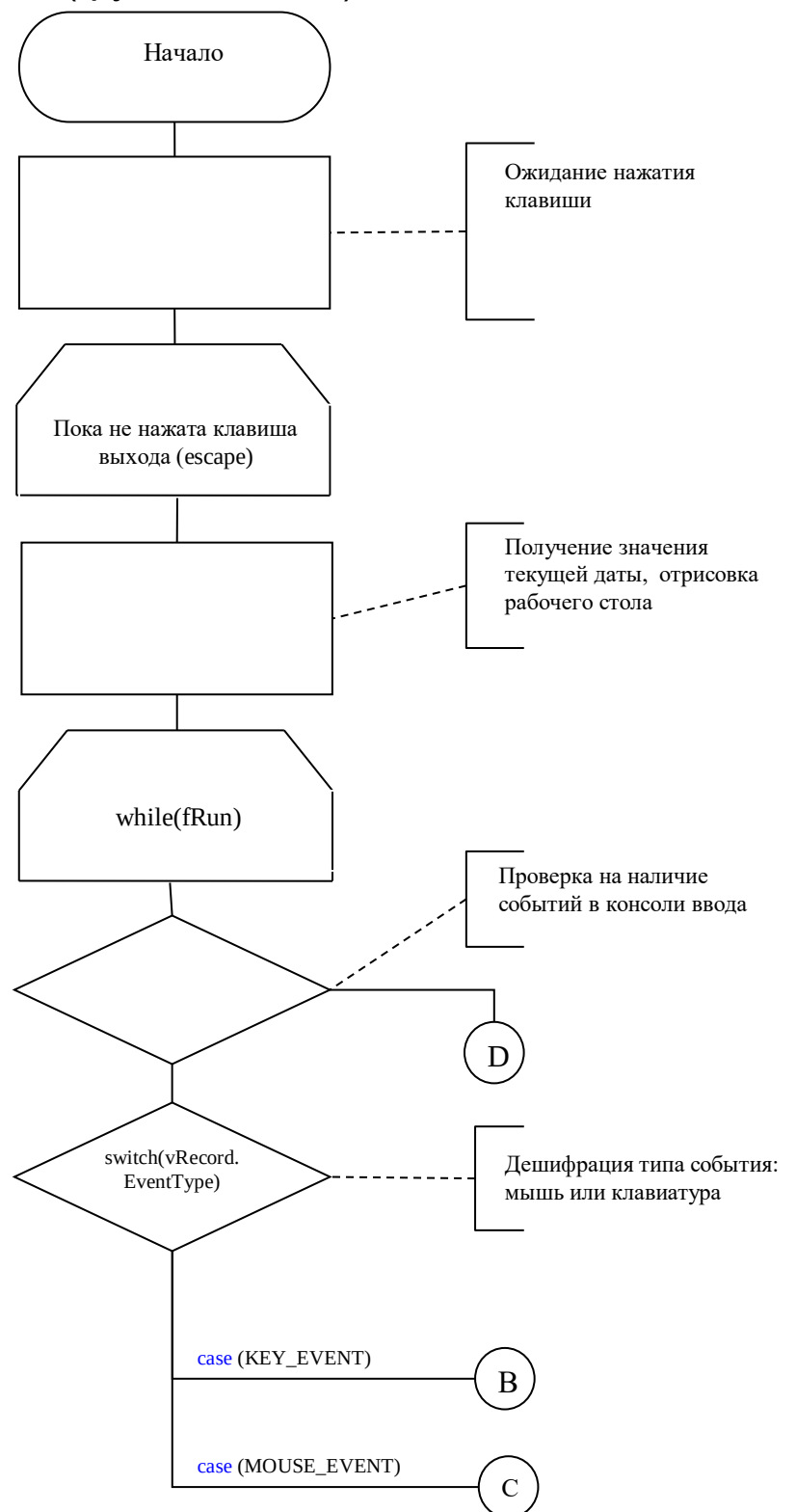
5.4 Схема взаимодействия модулей

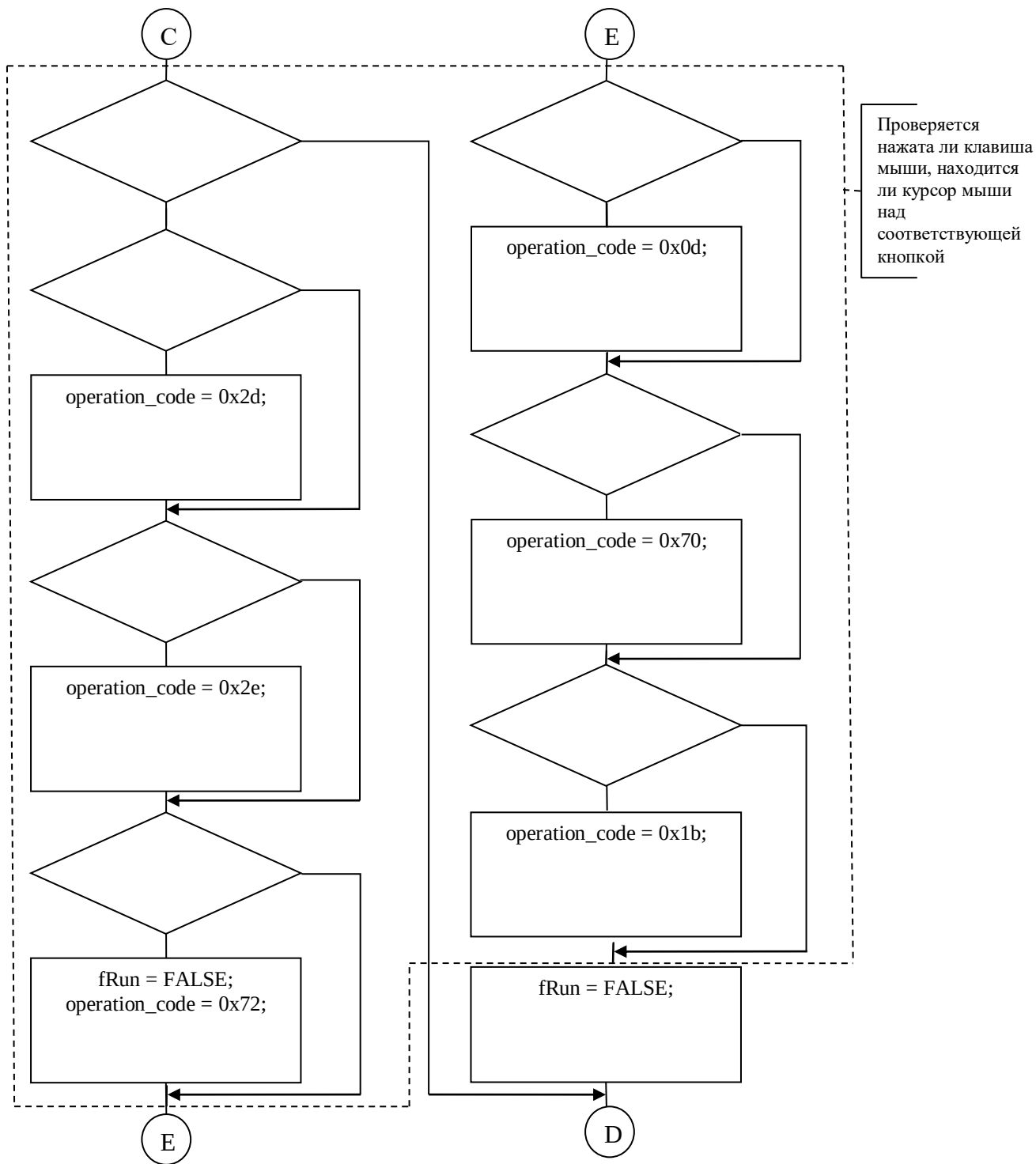


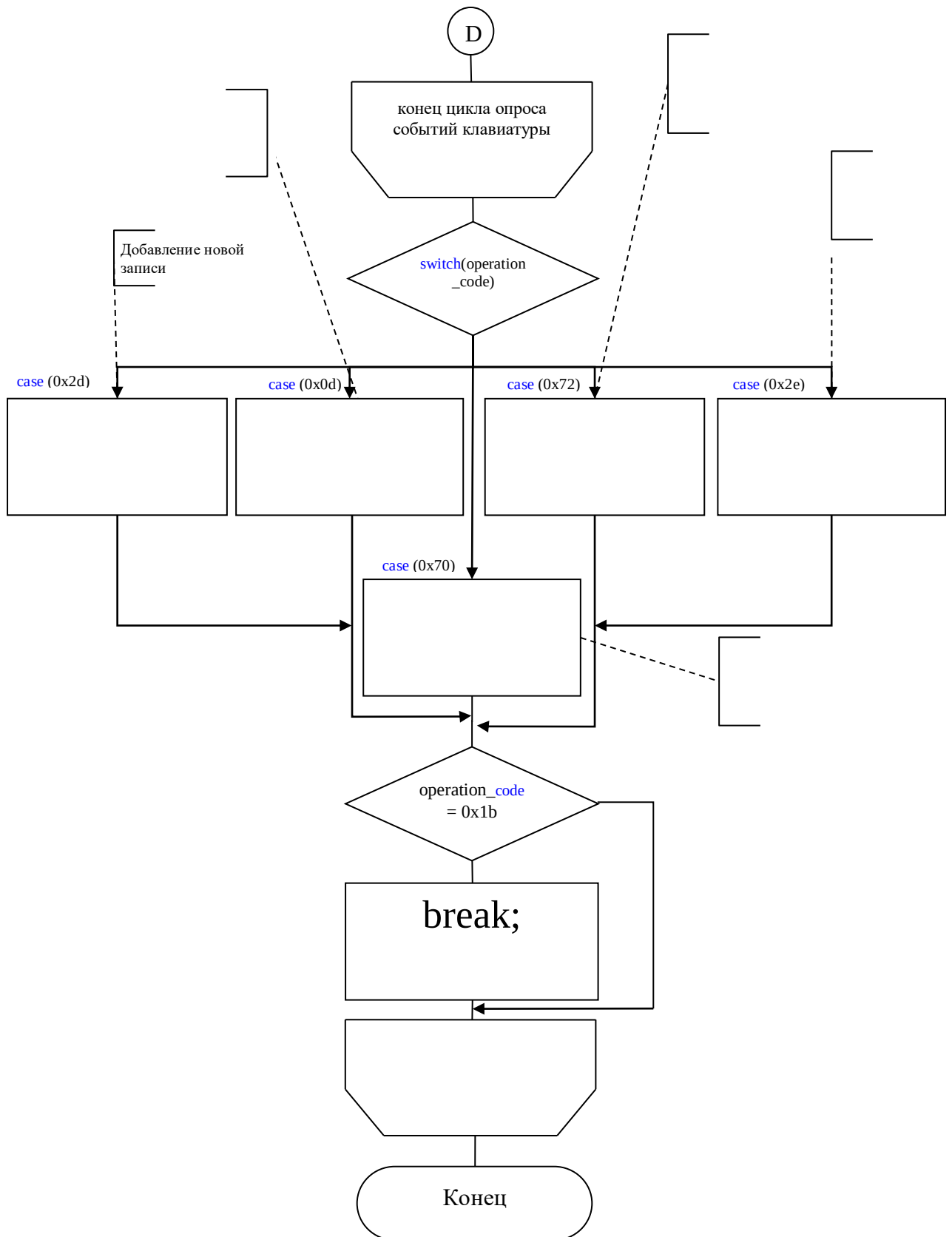
5.5 Иерархическая структура программы



5.6 Схема программы (функция main)







6. Руководство пользователя

Программа «Змейка» представляет собой знакомую всем с детства игру.

После запуска программы в активном окне появляется меню. Снизу вверх расположены три кнопки: «Новая игра», «Об авторе» и «Выход». При нажатии на кнопку «Новая игра», открывается окно с игровым полем и змейкой, собирающей яблоки. Ваша задача: собрать все яблоки и не погибнуть.

Используемые сочетания клавиш:

1. P – Пауза.
2. L – На экране появляется ценная информация о том, как пройти игру.
3. Стрелка «Вверх» – изменяет направление движения змейки вверх.
4. Стрелка «Вниз» – изменяет направление движения змейки вниз.
5. Стрелка «Вправо» – изменяет направление движения змейки вправо.
6. Стрелка «Влево» – изменяет направление движения змейки влево.
7. ESCAPE – для выхода из программы. Также для выхода из режима просмотра записей.

Заключение

В ходе разработки проекта были приобретены и усовершенствованы навыки работы со сложными типами данных: массивами, строками, динамическими одно- и двусвязными списками.

Кроме того, были изучены основы организации текстового псевдографического интерфейса, установки параметров стандартной консоли вывода, а также обработки событий стандартной консоли ввода – клавиатуры и мышки.

Одним из самых важных и полезных приобретенных навыков безусловно является навык работы с динамическими структурами данных.

Помимо этого, были усвоены как общие принципы программирования на языке Си, так и основные парадигмы процедурно-ориентированного программирования, с его методом нисходящего программирования.

Стоит также отметить, что в ходе составления данной пояснительной записки были приобретены некоторые навыки оформления составления технической документации документации.

На момент написания пояснительной записки автор доволен результатом проделанной работы и не видит серьезных несоответствий между поставленной перед ним задачей и её реализацией.

Программа написана так, что оставляет возможность ее доработки и усовершенствования при неизменности основной ее части. Например, можно сделать так, чтобы по игровому полю ходил охотник и сделать возможным сохранение и загрузку игры.

Приложение1. Листинг программы

Файл Barfic.h

```
#pragma once

#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <wchar.h>

//=====
// Функции работы с консолью.
//=====
/*
    GetConsoleScreenBufferInfo
    SetConsoleScreenBufferSize
*/

extern HANDLE      g_hConsoleOutput;
extern unsigned    g_nScreenBufferWidth;
extern unsigned    g_nScreenBufferHeight;

/** Инициализация работы с консолью.
 * Функция устанавливает размер экрана 80x25.
 */
BOOL
ConsoleInitialize(
    void
);

/** Функция проверяет является ли символ введенный запятой */
int iswcomma(wchar_t wch);

/** Изменение размера экрана.
 * @param[in]  nWidth      Ширина экрана.
 * @param[in]  nHeight     Высота экрана.
 */
BOOL
ConsoleSetScreenSize(
    IN  unsigned nWidth,
    IN  unsigned nHeight
);

//=====
// Функции обработки событий клавиатуры и мыши.
//=====
/*
    GetNumberOfConsoleInputEvents
    WriteConsoleOutputW
    ReadConsoleInputW
*/

/** Чтение событий клавиатуры и мыши.
 */
BOOL
InputRecordRead(
    OUT INPUT_RECORD&    rRecord
)
```

```

    );

/** Печать на экран информации о событии клавиатуры и мыши.
 */
void
InputRecordParse(
    IN INPUT_RECORD&    rRecord
    );

//=====
//
// Функции работы с курсором.
//=====
//
/*
    SetConsoleCursorPosition
    SetConsoleCursorInfo
    GetConsoleCursorInfo
*/

/** Скрытие/Отображение курсора.
 */
BOOL
SetCursorVisible(
    IN  BOOL    fVisible
    );

/** Изменение положения курсора.
 */
BOOL
SetCursorPosition(
    IN  unsigned nX,
    IN  unsigned nY
    );

//=====
//
// Функции рисования.
//=====
//
/** Перечисление цветов. */
enum EColor {
    ColorBlack          = 0,
    ColorBlue           = 1,
    ColorGreen          = 2,
    ColorCyan           = 3,
    ColorRed            = 4,
    ColorMagenta        = 5,
    ColorBrown          = 6,
    ColorWhite          = 7,
    ColorGrey           = 8,
    ColorPaleBlue       = 9,
    ColorChlorine        = 10,
    ColorLightGreen     = 10,
    ColorLightCyan      = 11,
    ColorLightBlue      = 11,
    ColorLightRed       = 12,
    ColorLightMagenta   = 13,
    ColorYellow         = 14,
    ColorBrightWhite    = 15,
};

/** Заполнение данных по рисуемому символу.

```

```

    * @param[out] pCharInfo    Заполняемая структура данных по рисуемому
символу.
    * @param[in]   wcChar      Unicode символ.
    * @param[in]   eForeColor  Цвет символа.
    * @param[in]   eBackColor  Цвет фона.
    */
void
SetCharInfo(
    OUT    CHAR_INFO* pCharInfo,
    IN     wchar_t     wcChar,
    IN     EColor       eForeColor,
    IN     EColor       eBackColor
    );

/** Рисование символа на экране.
    * @param[in] pCharInfo    Структура данных по рисуемому символу.
    * @param[in] nX           Координата X.
    * @param[in] nY           Координата Y.
    */
BOOL
DrawPoint(
    IN     CHAR_INFO* pCharInfo,
    IN     unsigned    nX,
    IN     unsigned    nY
    );

/** Рисование горизонтальной линии символов на экране.
    * @param[in] pCharInfo    Структура данных по рисуемому символу.
    * @param[in] nX           Координата X.
    * @param[in] nY           Координата Y.
    * @param[in] nLength      Длина линии.
    */
BOOL
DrawHorLine(
    IN     CHAR_INFO* pCharInfo,
    IN     unsigned    nX,
    IN     unsigned    nY,
    IN     unsigned    nLength
    );

/** Рисование вертикальной линии символов на экране.
    * @param[in] pCharInfo    Структура данных по рисуемому символу.
    * @param[in] nX           Координата X.
    * @param[in] nY           Координата Y.
    * @param[in] nLength      Длина линии.
    */
BOOL
DrawVerLine(
    IN     CHAR_INFO* pCharInfo,
    IN     unsigned    nX,
    IN     unsigned    nY,
    IN     unsigned    nLength
    );

/** Рисование прямоугольника символов на экране.
    * @param[in] pCharInfo    Структура данных по рисуемому символу.
    * @param[in] nX           Координата X.
    * @param[in] nY           Координата Y.
    * @param[in] nWidth       Ширина прямоугольника.
    * @param[in] nHeight      Высота прямоугольника.
    */
void
DrawRectangle(
    IN     CHAR_INFO* pCharInfo,

```

```

    IN unsigned    nX,
    IN unsigned    nY,
    IN unsigned    nWidth,
    IN unsigned    nHeight
);

/** Горизонтальная двойная линия. */
#define DEF_BOX_DRAWINGS_DOUBLE_HOR 0x2550
/** Вертикальная двойная линия. */
#define DEF_BOX_DRAWINGS_DOUBLE_VER 0x2551
/** Угол верхний левый. Двойные линии. */
#define DEF_BOX_DRAWINGS_DOUBLE_DOWN_RIGHT 0x2554
/** Угол верхний правый. Двойные линии. */
#define DEF_BOX_DRAWINGS_DOUBLE_DOWN_LEFT 0x2557
/** Угол нижний левый. Двойные линии. */
#define DEF_BOX_DRAWINGS_DOUBLE_UP_RIGHT 0x255A
/** Угол нижний правый. Двойные линии. */
#define DEF_BOX_DRAWINGS_DOUBLE_UP_LEFT 0x255D
/** Блок. Верхний половинный блок. */
#define DEF_UPPER_HALF_BLOCK 0x2580
/** Блок. Нижний половинный блок. */
#define DEF_LOWER_HALF_BLOCK 0x2584
/** Блок. Полный блок. */
#define DEF_FULL_BLOCK 0x2588

/** Рисование окна с рамкой.
 * @param[in] eForeColor Цвет символов (цвет рамки).
 * @param[in] eBackColor Цвет фона.
 * @param[in] nX Координата X.
 * @param[in] nY Координата Y.
 * @param[in] nWidth Ширина окна.
 * @param[in] nHeight Высота окна.
 */
void
DrawWindow(
    IN EColor    eForeColor,
    IN EColor    eBackColor,
    IN unsigned  nX,
    IN unsigned  nY,
    IN unsigned  nWidth,
    IN unsigned  nHeight
);

/** Перечисление вариантов выравнивания текста. */
enum EAlignment {
    BottomCenter,
    BottomLeft,
    BottomRight,
    MiddleCenter,
    MiddleLeft,
    MiddleRight,
    TopCenter,
    TopLeft,
    TopRight
};

/** Перечисление разрешённых символов ввода. */
enum TypeIn {
    Double,          // 0-9 : ,
    Int,             // 0-9
    String           // 0-9 : a-z : a-я
};

/** Рисование однострочного текста в прямоугольнике.

```

```

* @param[in] eForeColor      Цвет символов.
* @param[in] eBackColor     Цвет фона.
* @param[in] nX              Координата X.
* @param[in] nY              Координата Y.
* @param[in] nWidth          Ширина прямоугольника.
* @param[in] nHeight         Высота прямоугольника.
* @param[in] szText          Строка текста.
* @param[in] eTextAlignment  Выравнивание текста.
*/
void
DrawSingleLineText(
    IN EColor      eForeColor,
    IN EColor      eBackColor,
    IN unsigned    nX,
    IN unsigned    nY,
    IN unsigned    nWidth,
    IN unsigned    nHeight,
    IN const wchar_t* szText,
    IN EAlignment  eTextAlignment
);

/** Рисование кнопки.
 * Минимальные размеры кнопки: ширина (nWidth) 3, высота (nHeight) 2.
 * @param[in] eForeColor      Цвет символов.
 * @param[in] eBackColor     Цвет фона.
 * @param[in] eBack2Color    Цвет подложки.
 * @param[in] nX              Координата X.
 * @param[in] nY              Координата Y.
 * @param[in] nWidth          Ширина кнопки.
 * @param[in] nHeight         Высота кнопки.
 * @param[in] szText          Строка текста.
 */
void
DrawButton(
    IN EColor      eForeColor,
    IN EColor      eBackColor,
    IN EColor      eBack2Color,
    IN unsigned    nX,
    IN unsigned    nY,
    IN unsigned    nWidth,
    IN unsigned    nHeight,
    IN const wchar_t* szText
);

//=====
//
// Функции диалогов.
//=====
/** Перечисление типов MessageBox. */
enum EMessageBoxType {
    MessageBoxOk,
    MessageBoxOkCancel,
    MessageBoxYesNo
};

/** Перечисление возвращаемых значений MessageBox. */
enum EMessageBoxId {
    MessageBoxIdNone,
    MessageBoxIdOk,
    MessageBoxIdCancel,
    MessageBoxIdYes,
    MessageBoxIdNo
};

```

```

};

/** Диалог MessageBox.
 * Минимальные размеры кнопки: ширина (nWidth) 38, высота (nHeight) 5.
 * Диалог завершает работу после нажатия на любую кнопку.
 * @param[in] eDialogForeColor Цвет символов диалога.
 * @param[in] eDialogBackColor Цвет фона диалога.
 * @param[in] nX Координата X диалога.
 * @param[in] nY Координата Y диалога.
 * @param[in] nWidth Ширина диалога.
 * @param[in] nHeight Высота диалога.
 * @param[in] eTextForeColor Цвет символов текста.
 * @param[in] eTextBackColor Цвет фона текста.
 * @param[in] szText Строка текста.
 * @param[in] eTextAlignment Выравнивание текста.
 * @param[in] eMessageBoxType Тип MessageBox.
 */
EMessageBoxId
DialogMessageBox(
    IN EColor eDialogForeColor,
    IN EColor eDialogBackColor,
    IN unsigned nX,
    IN unsigned nY,
    IN unsigned nWidth,
    IN unsigned nHeight,
    IN EColor eTextForeColor,
    IN EColor eTextBackColor,
    IN const wchar_t* szText,
    IN EAlignment eTextAlignment,
    IN EMessageBoxType eMessageBoxType
);

/** Диалог ввода текста.
 * Минимальные размеры кнопки: ширина (nWidth) 2, высота (nHeight) 1.
 * Диалог позволяет вводить только символы.
 * Диалог завершает работу после нажатия Enter или Esc.
 * После нажатия Enter введенный текст сохраняется в szText.
 * После нажатия Esc в szText записывается пустая строка.
 * @param[in] eForeColor Цвет символов.
 * @param[in] eBackColor Цвет фона.
 * @param[in] nX Координата X диалога.
 * @param[in] nY Координата Y диалога.
 * @param[in] nWidth Ширина диалога.
 * @param[in] nHeight Высота диалога.
 * @param[out] szText Строка текста.
 * @param[in] nMaxTextSize Размер буфера строки в символах.
 */
void
DialogTextBox(
    IN EColor eForeColor,
    IN EColor eBackColor,
    IN unsigned nX,
    IN unsigned nY,
    IN unsigned nWidth,
    IN unsigned nHeight,
    OUT wchar_t* szText,
    IN size_t nMaxTextSize,
    IN TypeIn mode
);

```

Файл head.h

```
#include <conio.h>
#include <stdlib.h>
#include <windows.h>
#include <locale.h>
#include <malloc.h>
#include <memory.h>
#include "Barfic.h"

void DrawMainMenu(int w, int h);
int ChoiceButton(void);
```

Файл Console.cpp

```
/**
 * @file Console.cpp
 * Функции работы с консолью.
 */

#include "StdAfx.h"

HANDLE      g_hConsoleOutput = INVALID_HANDLE_VALUE;
unsigned     g_nScreenBufferWidth = 0;
unsigned     g_nScreenBufferHeight = 0;

BOOL
ConsoleInitialize(
    void
)
{
    BOOL fResult;

    // Получение идентификатора стандартной консоли вывода
    g_hConsoleOutput = GetStdHandle(STD_OUTPUT_HANDLE);
    if (g_hConsoleOutput == INVALID_HANDLE_VALUE)
    {
        return FALSE;
    }

    fResult = ConsoleSetScreeSize(80, 25);
    if (fResult == FALSE)
    {
        return FALSE;
    }

    return TRUE;
}

BOOL
ConsoleSetScreeSize(
    IN unsigned nWidth,
    IN unsigned nHeight
)
```

```

{
    BOOL    fResult;
    COORD    vSize;

    // Установка размера экранного буфера 80x30
    vSize.X = nWidth;
    vSize.Y = nHeight;
    fResult = SetConsoleScreenBufferSize(g_hConsoleOutput,
vSize);
    if (fResult == FALSE)
    {
        return FALSE;
    }

    // Сохранение текущего размера экранного буфера
    g_nScreenBufferWidth = nWidth;
    g_nScreenBufferHeight = nHeight;

    return TRUE;
}

```

Файл Cursor.cpp

```

/**
 * @file Cursor.cpp
 * Функции работы с курсором.
 */

#include "StdAfx.h"

BOOL
SetCursorVisible(
    IN  BOOL    fVisible
)
{
    BOOL    fResult;
    CONSOLE_CURSOR_INFO vCursorInfo;

    // Получение информации по курсору
    fResult =
        GetConsoleCursorInfo(
            g_hConsoleOutput,
            &vCursorInfo);
    if (fResult == FALSE)
    {
        return FALSE;
    }

    // Заполнение данных структуры CONSOLE_CURSOR_INFO
    vCursorInfo.bVisible = fVisible;

    // Запись информации по курсору
    fResult =

```

```

        SetConsoleCursorInfo(
            g_hConsoleOutput,
            &vCursorInfo);
    if (fResult == FALSE)
    {
        return FALSE;
    }

    return TRUE;
}

BOOL
SetCursorPosition(
    IN  unsigned nX,
    IN  unsigned nY
)
{
    BOOL    fResult;
    COORD    vCoord;

    // Проверка координат
    if ((nX >= g_nScreenBufferWidth) ||
        (nY >= g_nScreenBufferHeight))
    {
        return FALSE;
    }

    // Заполнение данных по местоположению курсора
    vCoord.X = nX;
    vCoord.Y = nY;

    // Запись информации по курсору
    fResult =
        SetConsoleCursorPosition(
            g_hConsoleOutput,
            vCoord);
    if (fResult == FALSE)
    {
        return FALSE;
    }

    return TRUE;
}

```

Файл Dialog.cpp

```

/**
 * @file Draw.cpp
 * Функции диалогов.
 */

#include "StdAfx.h"

```

```

int iswcomma(wchar_t wch)
{
    if( (int)wch == (int)',')
        return 1;
    else
        return 0;
};

```

```

EMessageBoxId
DialogMessageBox(
    IN EColor          eDialogForeColor,
    IN EColor          eDialogBackColor,
    IN unsigned        nX,
    IN unsigned        nY,
    IN unsigned        nWidth,
    IN unsigned        nHeight,
    IN EColor          eTextForeColor,
    IN EColor          eTextBackColor,
    IN const wchar_t*  szText,
    IN EAlignment      eTextAlignment,
    IN EMessageBoxType eMessageBoxType
)
{
    // Локальная структура описания массива кнопок
    struct {
        EColor          m_eForeColor;
        EColor          m_eBackColor;
        unsigned        m_nX;
        unsigned        m_nY;
        unsigned        m_nWidth;
        unsigned        m_nHeight;
        EMessageBoxId   m_eId;
    } aButtonCoord[3];
    // Курс отображаемых кнопок
    size_t nButtonAmount = 1;
    // Текущая активная кнопка
    size_t nButtonSelected = 0;
    // Признак необходимости перерисовки кнопок
    BOOL fUpdate = TRUE;
    // Признак работы цикла опроса событий клавиатуры и мыши
    BOOL fRun = TRUE;
    // Возвращаемое значение
    EMessageBoxId eId = MessageBoxIdNone;

    size_t nCount;
    INPUT_RECORD vRecord;

    // Проверка высоты и ширины
    if ((nHeight < 5) || (nWidth < 38))
    {
        return MessageBoxIdNone;
    }
}

```

```

    }

    // Рисование окна с рамкой
    DrawWindow(eDialogForeColor, eDialogBackColor, nX, nY,
nWidth, nHeight);
    // Рисование однострочного текста в прямоугольнике
    DrawSingleLineText(
        eTextForeColor,
        eTextBackColor,
        nX + 2,
        nY + 1,
        nWidth - 4,
        nHeight - 5,
        szText,
        eTextAlignment);

    // Цикл опроса событий клавиатуры и мыши
    while (fRun)
    {
        // Проверка необходимости перерисовки кнопок
        if (fUpdate)
        {
            // Сброс признака необходимости перерисовки кнопок
            // Сброс необходим для того, чтобы не было мерцания
экрана
            fUpdate = FALSE;

            // Заполнение данных по кнопкам
            for (nCount = 0; nCount < 3; nCount++)
            {
                if (nButtonSelected == nCount)
                    aButtonCoord[nCount].m_eForeColor =
ColorBrightWhite;
                else
                    aButtonCoord[nCount].m_eForeColor =
ColorGrey;

                aButtonCoord[nCount].m_eBackColor = ColorGreen;
                aButtonCoord[nCount].m_nX = nX + 2 + (nCount *
11);

                aButtonCoord[nCount].m_nY = nY + nHeight - 3;
                aButtonCoord[nCount].m_nWidth = 10;
                aButtonCoord[nCount].m_nHeight = 2;
            }

            // Рисование кнопок
            switch (eMessageBoxType)
            {
                case MessageBoxOk:
                    nButtonAmount = 1;
                    aButtonCoord[0].m_eId = MessageBoxIdOk;
                    DrawButton(
                        aButtonCoord[0].m_eForeColor,
aButtonCoord[0].m_eBackColor, eDialogBackColor,

```

```

        aButtonCoord[0].m_nX,
aButtonCoord[0].m_nY, aButtonCoord[0].m_nWidth,
aButtonCoord[0].m_nHeight,
        L"Ok");
        break;
    case MessageBoxOkCancel:
        nButtonAmount = 2;
        aButtonCoord[0].m_eId = MessageBoxIdOk;
        aButtonCoord[1].m_eId = MessageBoxIdCancel;
        DrawButton(
            aButtonCoord[0].m_eForeColor,
aButtonCoord[0].m_eBackColor, eDialogBackColor,
            aButtonCoord[0].m_nX,
aButtonCoord[0].m_nY, aButtonCoord[0].m_nWidth,
aButtonCoord[0].m_nHeight,
            L"OK");
        DrawButton(
            aButtonCoord[1].m_eForeColor,
aButtonCoord[0].m_eBackColor, eDialogBackColor,
            aButtonCoord[1].m_nX,
aButtonCoord[0].m_nY, aButtonCoord[0].m_nWidth,
aButtonCoord[0].m_nHeight,
            L"Отмена");
        break;
    case MessageBoxYesNo:
        nButtonAmount = 2;
        aButtonCoord[0].m_eId = MessageBoxIdYes;
        aButtonCoord[1].m_eId = MessageBoxIdNo;
        DrawButton(
            aButtonCoord[0].m_eForeColor,
aButtonCoord[0].m_eBackColor, eDialogBackColor,
            aButtonCoord[0].m_nX,
aButtonCoord[0].m_nY, aButtonCoord[0].m_nWidth,
aButtonCoord[0].m_nHeight,
            L"Да");
        DrawButton(
            aButtonCoord[1].m_eForeColor,
aButtonCoord[0].m_eBackColor, eDialogBackColor,
            aButtonCoord[1].m_nX,
aButtonCoord[0].m_nY, aButtonCoord[0].m_nWidth,
aButtonCoord[0].m_nHeight,
            L"Нет");
        break;
    }
}

// Чтение событий клавиатуры и мыши
if (InputRecordRead(vRecord))
{
    // Дешифрация типа события
    switch (vRecord.EventType)
    {
        // Клавиатура

```

```

        case KEY_EVENT:
            // Если нажата кнопка
            if (vRecord.Event.KeyEvent.bKeyDown)
            {
                switch
                (vRecord.Event.KeyEvent.wVirtualKeyCode)
                {
                    // Tab
                    case 0x09:
                        // Переключение активной кнопки
                        nButtonSelected++;
                        nButtonSelected =
nButtonSelected % nButtonAmount;
                        fUpdate = TRUE;
                        break;

                    // Enter
                    case 0x0D:
                        // Возвращаемое значение
                        eId =
aButtonCoord[nButtonSelected].m_eId;
                        // Выход
                        fRun = FALSE;
                        break;

                    // Esc
                    case 0x1B:
                        // Возвращаемое значение
                        eId = MessageBoxIdCancel;
                        // Выход
                        fRun = FALSE;
                        break;
                }
            }
            break;

        // Мышь
        case MOUSE_EVENT:
            // Если нажата ЛЕВАЯ кнопка мыши
            if (vRecord.Event.MouseEvent.dwButtonState &
0x01)
            {
                unsigned nMouseX =
vRecord.Event.MouseEvent.dwMousePosition.X;
                unsigned nMouseY =
vRecord.Event.MouseEvent.dwMousePosition.Y;
                // Если курсор находится над одной из
кнопок
                for (nCount = 0; nCount < nButtonAmount;
nCount++)
                {
                    if ((nMouseX >=
aButtonCoord[nCount].m_nX)

```

```

        &&
        (nMouseX <
(aButtonCoord[nCount].m_nX + aButtonCoord[nCount].m_nWidth - 1))
        &&
        (nMouseY >=
aButtonCoord[nCount].m_nY
        &&
        (nMouseY <
(aButtonCoord[nCount].m_nY + aButtonCoord[nCount].m_nHeight -
1)))
    {
        // Найдена кнопка над которой
нажата ЛЕВАЯ кнопка мыши
        // Возвращаемое значение
eId =
aButtonCoord[nCount].m_eId;
        // Выход
fRun = FALSE;
    }
}
break;
    }
}
Sleep(20);
}

// Возвращаемое значение
return eId;
}

void
DialogTextBox(
    IN   EColor      eForeColor,
    IN   EColor      eBackColor,
    IN   unsigned    nX,
    IN   unsigned    nY,
    IN   unsigned    nWidth,
    IN   unsigned    nHeight,
    OUT  wchar_t*    szText,
    IN   size_t      nMaxTextSize,
    IN   TypeIn      mode
)
{
    // Признак необходимости перерисовки кнопок
    BOOL fUpdate = TRUE;
    // Признак работы цикла опроса событий клавиатуры и мыши
    BOOL fRun = TRUE;
    // Максимальная длина строки текста
    size_t nMaxTextLength;
    // Длина строки текста
    size_t nTextLength;

```

```

// Положение курсора в тексте
size_t nTextCursor;
// Смещение от начала текста для рисования текста
size_t nTextDraw;
// Разрешённые символы
int m;
// Если символ ',' введён только один раз
BOOL Comma = FALSE;

INPUT_RECORD vRecord;
CHAR_INFO vPoint;

// Проверка высоты и ширины
if ((nHeight < 1) || (nWidth < 2))
{
    return;
}
// Проверка максимальной длины строки
if (nMaxTextSize == 0)
{
    return;
}

// Рисование прямоугольника
SetCharInfo(&vPoint, L' ', eForeColor, eBackColor);
DrawRectangle(&vPoint, nX, nY, nWidth, nHeight);

// Начальные значения
nMaxTextLength = nMaxTextSize - 1;
nTextLength = 0;
nTextCursor = 0;
nTextDraw = 0;
szText[0] = 0;

// Установка положения курсора
SetCursorPosition(nX, nY);
// Отображение курсора
SetCursorVisible(TRUE);

// Цикл опроса событий клавиатуры и мыши
while (fRun)
{
    // Проверка необходимости перерисовки кнопок
    if (fUpdate)
    {
        // Сброс признака необходимости перерисовки кнопок
        // Сброс необходим для того, чтобы не было мерцания
        экрана
        fUpdate = FALSE;

        // Корректировка смещения от начала текста для
        рисования текста

```

```

        // Корректировка необходима, чтобы текст не выходил
за границы элемента
        if (nTextCursor < nTextDraw)
        {
            nTextDraw = nTextCursor;
        }
        if ((nTextCursor - nTextDraw + 1) > nWidth)
        {
            nTextDraw = nTextCursor - nWidth + 1;
        }
        if (nTextDraw > 0)
        {
            if ((nTextLength - nTextDraw + 1) < nWidth)
            {
                nTextDraw = nTextLength - nWidth + 1;
            }
        }

        // Рисование однострочного текста в прямоугольнике
DrawSingleLineText(
    eForeColor,
    eBackColor,
    nX,
    nY,
    nWidth,
    nHeight,
    &szText[nTextDraw],
    MiddleLeft);
    // Установка положения курсора
    SetCursorPosition(nX + nTextCursor - nTextDraw, nY);
}

// Чтение событий клавиатуры и мыши
if (InputRecordRead(vRecord))
{
    // Дешифрация типа события
    switch (vRecord.EventType)
    {
        // Клавиатура
        case KEY_EVENT:
            // Если нажата кнопка
            if (vRecord.Event.KeyEvent.bKeyDown)
            {
                switch
(vRecord.Event.KeyEvent.wVirtualKeyCode)
                {
                    // Enter
                    case 0x0D:
                        if((nTextLength == 0) &&
(nTextCursor == 0))
                            break;

                    // Выход
                    fRun = FALSE;
                }
            }
        }
    }
}

```

```

        break;

// Esc
case 0x1B:
    szText[0] = 0;
    // Выход
    fRun = FALSE;
    break;

// Backspace
case 0x08:
    if (nTextCursor > 0)
    {
        memmove(
            &szText[nTextCursor -
1],
            &szText[nTextCursor],
            (nTextLength -
nTextCursor + 1) * sizeof(wchar_t));
        nTextLength--;
        nTextCursor--;
        fUpdate = TRUE;
        if (wcschr(szText,
',') != NULL)
            Comma = TRUE;
        else
            Comma = FALSE;
    }
    break;

// Del
case 0x2e:
    if (nTextCursor < nTextLength)
    {
        memmove(
            &szText[nTextCursor],
            &szText[nTextCursor +
1],
            (nTextLength -
nTextCursor + 1) * sizeof(wchar_t));
        nTextLength--;
        fUpdate = TRUE;
        if (wcschr(szText,
',') != NULL)
            Comma = TRUE;
        else
            Comma = FALSE;
    }
    break;

// Left
case 0x25:
    if (nTextCursor > 0)

```

```

        {
            nTextCursor--;
            fUpdate = TRUE;
        }
        break;

// Right
case 0x27:
    if (nTextCursor < nTextLength)
    {
        nTextCursor++;
        fUpdate = TRUE;
    }
    break;

default:
    switch (mode)
    {
        // ВВОДИМ ВЕЩЕСТВЕННОЕ
        case Double:

            if ((iswcomma(vRecord.Event.KeyEvent.uChar.UnicodeChar)) && (Comma == FALSE))
            {
                Comma = TRUE;
                m = 1;
            }
            else
                m =
iswdigit(vRecord.Event.KeyEvent.uChar.UnicodeChar);
                break;
                // ВВОДИМ ЦЕЛОЕ ЧИСЛО
            case Int:
                m =
iswdigit(vRecord.Event.KeyEvent.uChar.UnicodeChar);
                break;
                // ВВОДИМ СТРОКУ
            case String:
                m =
iswalpha(vRecord.Event.KeyEvent.uChar.UnicodeChar) || iswdigit(vRecord.Event.KeyEvent.uChar.UnicodeChar);
                break;
        }
        if (m)
        {
            if (nTextLength <
nMaxTextLength)
            {
                memmove(
                    &szText[nTextCursor
+ 1],

```

```

&szText[nTextCursor],
                                                    (nTextLength -
nTextCursor + 1) * sizeof(wchar_t));
                                                    szText[nTextCursor] =
vRecord.Event.KeyEvent.uChar.UnicodeChar;
                                                    nTextLength++;
                                                    nTextCursor++;
                                                    fUpdate = TRUE;
                                                    }
                                                    }
                                                    break;
                                                    }
                                                    break;

// МЫШЬ
case MOUSE_EVENT:
    break;

    }
    }
    Sleep(20);
}

// Скрытие курсора
SetCursorVisible(FALSE);
}

```

Файл Draw.cpp

```

/**
 * @file Draw.cpp
 * Функции рисования.
 */

#include "StdAfx.h"

void
SetCharInfo(
    OUT CHAR_INFO*      pCharInfo,
    IN  wchar_t         wcChar,
    IN  EColor          eForeColor,
    IN  EColor          eBackColor
)
{
    pCharInfo->Char.UnicodeChar = wcChar;
    pCharInfo->Attributes = 0;
    pCharInfo->Attributes |= eForeColor & 0x000F;
    pCharInfo->Attributes |= (eBackColor & 0x000F) << 4;
}

BOOL
DrawPoint(

```

```

    IN  CHAR_INFO*   pCharInfo,
    IN  unsigned     nX,
    IN  unsigned     nY
    )
{
    BOOL          fResult;
    COORD         vSize;
    COORD         vCoord;
    SMALL_RECT    vRegion;

    // Проверка координат
    if ((nX >= g_nScreenBufferWidth) ||
        (nY >= g_nScreenBufferHeight))
    {
        return FALSE;
    }

    // Заполнение данных для отрисовки 1 точки
    vSize.X = 1;
    vSize.Y = 1;
    vCoord.X = 0;
    vCoord.Y = 0;
    vRegion.Top = nY;
    vRegion.Bottom = nY;
    vRegion.Left = nX;
    vRegion.Right = nX;

    // Запись данных в стандартную консоль вывода
    fResult =
        WriteConsoleOutputW(
            g_hConsoleOutput,
            pCharInfo,
            vSize,
            vCoord,
            &vRegion);
    if (fResult == FALSE)
    {
        return FALSE;
    }

    return TRUE;
}

BOOL
DrawHorLine(
    IN  CHAR_INFO*   pCharInfo,
    IN  unsigned     nX,
    IN  unsigned     nY,
    IN  unsigned     nLength
    )
{
    BOOL fResult;

```

```

while (nLength > 0)
{
    fResult = DrawPoint(pCharInfo, nX, nY);
    if (fResult == FALSE)
    {
        return FALSE;
    }
    nX++;
    nLength--;
}

return TRUE;
}

BOOL
DrawVerLine(
    IN  CHAR_INFO*  pCharInfo,
    IN  unsigned     nX,
    IN  unsigned     nY,
    IN  unsigned     nLength
)
{
    BOOL fResult;

    while (nLength > 0)
    {
        fResult = DrawPoint(pCharInfo, nX, nY);
        if (fResult == FALSE)
        {
            return FALSE;
        }
        nY++;
        nLength--;
    }

    return TRUE;
}

void
DrawRectangle(
    IN  CHAR_INFO*  pCharInfo,
    IN  unsigned     nX,
    IN  unsigned     nY,
    IN  unsigned     nWidth,
    IN  unsigned     nHeight
)
{
    BOOL fResult;
    size_t nCount;

    for (nCount = 0; nCount < nHeight; nCount++)
    {
        fResult = DrawHorLine(pCharInfo, nX, nY, nWidth);
    }
}

```

```

        if (fResult == FALSE)
        {
            return;
        }
        nY++;
    }
}

void
DrawWindow(
    IN EColor      eForeColor,
    IN EColor      eBackColor,
    IN unsigned    nX,
    IN unsigned    nY,
    IN unsigned    nWidth,
    IN unsigned    nHeight
)
{
    size_t nCount;
    CHAR_INFO vPoint;

    // Рисование прямоугольника
    SetCharInfo(&vPoint, L' ', eForeColor, eBackColor);
    DrawRectangle(&vPoint, nX, nY, nWidth, nHeight);

    // Рисование рамки
    if ((nWidth > 2) && (nHeight > 2))
    {
        // Рисование горизонтальных полос рамки
        SetCharInfo(&vPoint, DEF_BOX_DRAWINGS_DOUBLE_HOR,
eForeColor, eBackColor);
        for (nCount = 1; nCount < nWidth - 1; nCount++)
        {
            DrawPoint(&vPoint, nX + nCount, nY);
            DrawPoint(&vPoint, nX + nCount, nY + nHeight - 1);
        }
        // Рисование вертикальных полос рамки
        SetCharInfo(&vPoint, DEF_BOX_DRAWINGS_DOUBLE_VER,
eForeColor, eBackColor);
        for (nCount = 1; nCount < nHeight - 1; nCount++)
        {
            DrawPoint(&vPoint, nX, nY + nCount);
            DrawPoint(&vPoint, nX + nWidth - 1, nY + nCount);
        }
        // Угол верхний левый
        SetCharInfo(&vPoint, DEF_BOX_DRAWINGS_DOUBLE_DOWN_RIGHT,
eForeColor, eBackColor);
        DrawPoint(&vPoint, nX, nY);
        // Угол верхний правый
        SetCharInfo(&vPoint, DEF_BOX_DRAWINGS_DOUBLE_DOWN_LEFT,
eForeColor, eBackColor);
        DrawPoint(&vPoint, nX + nWidth - 1, nY);
        // Угол нижний левый

```

```

        SetCharInfo(&vPoint, DEF_BOX_DRAWINGS_DOUBLE_UP_RIGHT,
eForeColor, eBackColor);
        DrawPoint(&vPoint, nX, nY + nHeight - 1);
        // Угол нижний правый
        SetCharInfo(&vPoint, DEF_BOX_DRAWINGS_DOUBLE_UP_LEFT,
eForeColor, eBackColor);
        DrawPoint(&vPoint, nX + nWidth - 1, nY + nHeight - 1);
    }
}

void
DrawSingleLineText(
    IN EColor          eForeColor,
    IN EColor          eBackColor,
    IN unsigned        nX,
    IN unsigned        nY,
    IN unsigned        nWidth,
    IN unsigned        nHeight,
    IN const wchar_t*  szText,
    IN EAlignment      eTextAlignment
)
{
    size_t nLength;
    size_t nCount;
    size_t nTextX;
    size_t nTextY;
    CHAR_INFO vPoint;

    // Проверка высоты и ширины
    if ((nHeight == 0) || (nWidth == 0))
    {
        return;
    }

    // Длина строки
    nLength = wcslen(szText);
    if (nLength > nWidth)
    {
        nLength = nWidth;
    }

    // Рисование прямоугольника
    SetCharInfo(&vPoint, L' ', eForeColor, eBackColor);
    DrawRectangle(&vPoint, nX, nY, nWidth, nHeight);

    // Определение координат текста
    switch (eTextAlignment)
    {
        case BottomCenter:
            nTextX = nX + (nWidth - nLength) / 2;
            nTextY = nY + nHeight - 1;
            break;
        case BottomLeft:

```

```

        nTextX = nX;
        nTextY = nY + nHeight - 1;
        break;
    case BottomRight:
        nTextX = nX + nWidth - nLength;
        nTextY = nY + nHeight - 1;
        break;
    case MiddleCenter:
        nTextX = nX + (nWidth - nLength) / 2;
        nTextY = nY + (nHeight - 1) / 2;
        break;
    case MiddleLeft:
        nTextX = nX;
        nTextY = nY + (nHeight - 1) / 2;
        break;
    case MiddleRight:
        nTextX = nX + nWidth - nLength;
        nTextY = nY + (nHeight - 1) / 2;
        break;
    case TopCenter:
        nTextX = nX + (nWidth - nLength) / 2;
        nTextY = nY;
        break;
    case TopLeft:
        nTextX = nX;
        nTextY = nY;
        break;
    case TopRight:
        nTextX = nX + nWidth - nLength;
        nTextY = nY;
        break;
    }

    // Рисование текста
    for (nCount = 0; nCount < nLength; nCount++)
    {
        SetCharInfo(&vPoint, szText[nCount], eForeColor,
eBackColor);
        DrawPoint(&vPoint, nTextX + nCount, nTextY);
    }
}

void
DrawButton(
    IN    EColor          eForeColor,
    IN    EColor          eBackColor,
    IN    EColor          eBack2Color,
    IN    unsigned        nX,
    IN    unsigned        nY,
    IN    unsigned        nWidth,
    IN    unsigned        nHeight,
    IN    const wchar_t*  szText
)

```

```

{
    size_t nCount;
    CHAR_INFO vPoint;

    // Проверка высоты и ширины
    if ((nHeight < 2) || (nWidth < 4))
    {
        return;
    }

    // Рисование прямоугольника
    SetCharInfo(&vPoint, L' ', eForeColor, eBackColor);
    DrawRectangle(&vPoint, nX, nY, nWidth - 1, nHeight - 1);
    // Рисование однострочного текста в прямоугольнике
    DrawSingleLineText(
        eForeColor,
        eBackColor,
        nX + 1,
        nY,
        nWidth - 3,
        nHeight - 1,
        szText,
        MiddleCenter);

    // Рисование тени
    // Угол верхний правый
    SetCharInfo(&vPoint, DEF_LOWER_HALF_BLOCK, ColorBlack,
eBack2Color);
    DrawPoint(&vPoint, nX + nWidth - 1, nY);
    // Рисование вертикальной полосы тени
    SetCharInfo(&vPoint, DEF_FULL_BLOCK, ColorBlack,
eBack2Color);
    for (nCount = 1; nCount < nHeight - 1; nCount++)
    {
        DrawPoint(&vPoint, nX + nWidth - 1, nY + nCount);
    }
    // Рисование горизонтальных полосы тени
    SetCharInfo(&vPoint, DEF_UPPER_HALF_BLOCK, ColorBlack,
eBack2Color);
    for (nCount = 1; nCount < nWidth; nCount++)
    {
        DrawPoint(&vPoint, nX + nCount, nY + nHeight - 1);
    }
}

```

Файл InputRecord.cpp

```

/**
 * @file InputRecord.cpp
 * Функции обработки событий клавиатуры и мыши.
 */

#include "StdAfx.h"

```

```

BOOL
InputRecordRead(
    OUT INPUT_RECORD&    rRecord
)
{
    const INPUT_RECORD Empty = { 0 };

    HANDLE    hConsole;
    DWORD     nUnread;
    BOOL      bRet;
    DWORD     dwSize;

    // Получение идентификатора стандартной консоли ввода
    hConsole = GetStdHandle(STD_INPUT_HANDLE);

    // Получение количества событий в буфере ввода
    bRet =
        GetNumberOfConsoleInputEvents(
            hConsole,
            &nUnread);
    if ((bRet == TRUE) && (nUnread > 0))
    {
        // В буфере ввода есть события

        // Чтение одного события из буфера
        bRet =
            ReadConsoleInputW(
                hConsole,
                &rRecord,
                1,
                (DWORD*)&dwSize);
        if ((bRet == TRUE) && (dwSize == 1))
        {
            return TRUE;
        }
    }

    rRecord = Empty;

    return FALSE;
}

void
InputRecordParse(
    IN INPUT_RECORD&    rRecord
)
{
    // Дешифрация типа события
    switch (rRecord.EventType)
    {
        /*/ Клавиатура
        case KEY_EVENT:

```

```

        {
            if(rRecord.Event.KeyEvent.bKeyDown == 27)
                exit(0);
            break;
        }

        */ // МЫШЬ
        case MOUSE_EVENT:
        {
            break;
        }
    }
}

```

Файл MainMenu.cpp

```

#include "head.h"

void DrawMainMenu(int w, int h)
{
    CHAR_INFO vPoint;
    BOOL fResult;

    SetCursorVisible(FALSE);
    // Закрашивание области
    SetCharInfo(&vPoint, L' ', ColorPaleBlue, ColorPaleBlue);
    DrawRectangle(&vPoint, 0, 0, w, h);

    // Рисование главного окна меню
    DrawWindow(ColorWhite, ColorMagenta , 20, 1, 40, 23);

    // Создание кнопок меню
    DrawButton(ColorBrightWhite,ColorChlorine, ColorBlack, 30,
3, 20, 4,  L"New Game"); // Создать
    DrawButton(ColorWhite,ColorLightRed, ColorBlack, 30, 8, 20,
4,  L"Save"); // Открыть
    DrawButton(ColorWhite,ColorBlue, ColorBlack, 30, 13, 20, 4,
L"Load"); // Удалить
    DrawButton(ColorWhite,ColorGreen, ColorBlack, 30, 18, 20, 4,
L"Exit"); // Выход
};

// *****
// ** Если нажата кнопка "down"
// *****
int SelectDown(int& sel)
{

```

```

        switch(sel)
        {
        case 0:                // текущая кнопка "Создать"
            {
                DrawButton(ColorBrightWhite,ColorChlorine,
ColorBlack, 30, 3, 20, 4,  L"New Game"); // Создать
                DrawButton(ColorWhite,ColorLightRed, ColorBlack,
30, 8, 20, 4,  L"Save");    // Сохранить
                return 1;
            }
        case 1:                // текущая кнопка "Открыть"
            {
                DrawButton(ColorWhite,ColorLightRed, ColorBlack,
30, 8, 20, 4,  L"Save");    // Сохранить
                DrawButton(ColorWhite,ColorBlue, ColorBlack, 30,
13, 20, 4, L"Load");        // Загрузить
                return 2;
            }
        case 2:                // текущая кнопка "Удалить"
            {
                DrawButton(ColorWhite, ColorCyan, ColorBlack, 30,
13, 20, 4, L"Load");        // Загрузить
                DrawButton(ColorWhite,ColorGreen, ColorBlack, 30,
18, 20, 4, L"Exit");        // Выход
                return 3;
            }
        case 3:                // текущая кнопка "Выход"
            {
                DrawButton(ColorBrightWhite,ColorChlorine,
ColorBlack, 30, 3, 20, 4,  L"New Game"); // Создать
                DrawButton(ColorWhite,ColorGreen, ColorBlack, 30,
18, 20, 4, L"Exit");        // Выход
                return 0;
            }
        default:
            return sel;
        }
    }
    //*****
    // ** Если нажата кнопка "up"
    // *****
    int SelectUp(int& sel)
    {
        switch(sel)
        {
        case 0:                // текущая кнопка "Создать"
            {
                DrawButton(ColorBrightWhite,ColorChlorine,
ColorBlack, 30, 3, 20, 4,  L"New Game"); // Создать
                DrawButton(ColorWhite,ColorGreen, ColorBlack, 30,
18, 20, 4, L"Exit");        // Выход
                return 3;
            }
        }
    }

```

```

        case 1:                                // текущая кнопка "Открыть"
        {
            DrawButton(ColorWhite,ColorLightRed, ColorBlack,
30, 8, 20, 4, L"Save");    // Сохранить
            DrawButton(ColorBrightWhite,ColorChlorine,
ColorBlack, 30, 3, 20, 4, L"New Game");    // Создать
            return 0;
        }
        case 2:                                // текущая кнопка "Удалить"
        {
            DrawButton(ColorWhite,ColorBlue, ColorBlack, 30,
13, 20, 4, L"Load");    // Загрузить
            DrawButton(ColorWhite,ColorLightRed, ColorBlack,
30, 8, 20, 4, L"Save");    // Сохранить

            return 1;
        }
        case 3:
        {
            DrawButton(ColorWhite,ColorGreen, ColorBlack, 30,
18, 20, 4, L"Exit"); // Выход
            DrawButton(ColorWhite,ColorBlue, ColorBlack, 30,
13, 20, 4, L"Load");    // Загрузить
            return 2;
        }
        default:
            return sel;
    }
}
//*****
// ** Функция возвращает номер выбранной кнопки
// *****
int ChoiceButton(void)
{
    int iSelectButton = 0;    // по умолчанию выбрана кнопка
"Создать"
    char ch;                // код нажатой клавиши

    while(true)
    {
        ch = getch();

        switch(ch)
        {
            case 80:        // нажата кнопка "down"
            {
                iSelectButton = SelectDown(iSelectButton);
                break;
            }
            case 72:        // нажата кнопка "up"
            {
                iSelectButton = SelectUp(iSelectButton);
                break;
            }
        }
    }
}

```

```

        }
        case 13:           // нажата кнопка "enter"
        {
            return iSelectButton;
        }
    }

    return 1;
}

```

Snake_by_setyaev.cpp

```

#include <iostream> //стандартная библиотека
#include <time.h> //случайные числа
#include <stdio.h> //для printf
#include <windows.h> // для HANDLE, курсора, цвета
#include <conio.h> //для kbhit
#include "head.h"

using namespace std;

HANDLE hConsole;
HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);

void GotoXY(int X, int Y)
{
    COORD coord = { X, Y };
    SetConsoleCursorPosition(hStdOut, coord);
}
//Цвет
enum ConsoleColor
{
    Black          = 0,
    Blue           = 1,
    Green          = 2,
    Cyan           = 3,
    Red            = 4,
    Magenta        = 5,
    Brown          = 6,
    LightGray      = 7,
    DarkGray       = 8,
    LightBlue      = 9,
    LightGreen     = 10,
    LightCyan      = 11,
    LightRed       = 12,
    LightMagenta   = 13,
    Yellow         = 14,
    White          = 15
}

```

```

};

void SetColor(ConsoleColor text, ConsoleColor background)
{
    SetConsoleTextAttribute(hStdOut, (WORD)((background << 4) |
text));
}

struct Zmeja // структура змейка
{
    COORD *t; //точки
    int PCount; //количество яблок
};
enum uprawlenie{LEFT,UP,RIGHT,DOWN}; //направление змейки
struct Game //данные-точности: змейки, яблок, передвижение по X
и Y, задержка, направление
{
    Zmeja gaduka; //змейка
    COORD jabloko; //яблоко
    int dx,dy, //передвижение
        pause; //задержка
    int nap; //направление
};

//Функция разброски яблок
void PlusJabloko(Game &g)
{
    int i,x,y;
    int n = g.gaduka.PCount;
    do
    {
        x = rand() % 56+3; //
        y = rand() % 19+3; //кординаты яблока
        for(i = 0; i < n; i++)
        {
            if(x == g.gaduka.t[i].X && y == g.gaduka.t[i].Y)
// проверка чтоб яблоко не бросить на змею
                break;
        }
    }
    while(i < n);
    g.jabloko.X = x; //
    g.jabloko.Y = y; //запоминаем позицию яблока
    SetConsoleCursorPosition(hConsole, g.jabloko); //переносим
курсор в эту позицию
    SetConsoleTextAttribute(hConsole,0x0c); //цвет яблока
    printf("%c", 4); //рисует яблоко каким хотим символом
}
// Функция старта змейки, ее координат и скорости

```

```

void skorostGame(Game &g)
{
    system("cls");
    g.gaduka.PCount = 3; //сколько точек в змейки
    g.gaduka.t = new COORD [3]; //создали точки
    for(int i = 0; i < 3; i++)
    {
        g.gaduka.t[i].X = 20 + i;
        g.gaduka.t[i].Y = 20;

    }
    g.dx = 1;
    g.dy = 0;
    g.pause = 100; //скорость передвижение змеи
    PlusJabloko(g); //рисует яблока
}

void Level()
{
    GotoXY(10,10); cout << "Wu nikogda ne wuigraete " << endl;
    GotoXY(10,11); cout << "eslu ne bydete bduetelnu!!!" << endl;
}
void ZmejaStart()
{
    GotoXY(10,15); cout << "Soberite 75 Jablok togda
posmotrim" << endl;
}
void STENA_2()
{
    SetColor(LightBlue , Black); GotoXY(20,0); cout << "Game
\"Snake by Golyshevsky\" " << endl;
    GotoXY(64,2); cout << "Behaviours:" << endl ;
    GotoXY(64,3); cout << "Apples:0" << endl ;
    GotoXY(64,4); cout << "Length:3" << endl;
    GotoXY(64,5); cout << "Speed:0" << endl;
    GotoXY(64,7); cout << "Controls:" << endl;
        GotoXY(64,8); cout << "Esc:Exit" << endl;
        GotoXY(64,9); cout << "P:Pause" << endl;
        GotoXY(64,10); cout << "S:Start" << endl;
        GotoXY(64,11); cout << "L:Level" << endl;
        GotoXY(64,13); printf("%c",24); cout
<< ":Up" << endl;

        GotoXY(64,14); printf("%c",25); cout << ":Down" << endl;

        GotoXY(64,15); printf("%c",27); cout << ":Left" << endl;

        GotoXY(64,16); printf("%c",26); cout << ":Right" << endl;
        {SetColor(LightMagenta , Black);
        GotoXY(2,2);
        int m = 0;
        for(m = 0; m < 60; m++)
        {

```

```

        printf("*");
    }
}

{
    GotoXY(2,24);
    int m = 0;
    for(m = 0; m < 60;m++)
    {
        printf("*");
    }
}

{
    GotoXY(2,3); cout << "*"<<endl;
    GotoXY(2,4); cout << "*"<<endl;
    GotoXY(2,5); cout << "*"<<endl;
    GotoXY(2,6); cout << "*"<<endl;
    GotoXY(2,7); cout << "*"<<endl;
    GotoXY(2,8); cout << "*"<<endl;
    GotoXY(2,9); cout << "*"<<endl;
    GotoXY(2,10); cout << "*"<<endl;
    GotoXY(2,11); cout << "*"<<endl;
    GotoXY(2,12); cout << "*"<<endl;
    GotoXY(2,13); cout << "*"<<endl;
    GotoXY(2,14); cout << "*"<<endl;
    GotoXY(2,15); cout << "*"<<endl;
    GotoXY(2,16); cout << "*"<<endl;
    GotoXY(2,17); cout << "*"<<endl;
    GotoXY(2,18); cout << "*"<<endl;
    GotoXY(2,19); cout << "*"<<endl;
    GotoXY(2,20); cout << "*"<<endl;
    GotoXY(2,21); cout << "*"<<endl;
    GotoXY(2,22); cout << "*"<<endl;
    GotoXY(2,23); cout << "*"<<endl;
}

{
    GotoXY(61,3); cout << "*"<<endl;
    GotoXY(61,4); cout << "*"<<endl;
    GotoXY(61,5); cout << "*"<<endl;
    GotoXY(61,6); cout << "*"<<endl;
    GotoXY(61,7); cout << "*"<<endl;
    GotoXY(61,8); cout << "*"<<endl;
    GotoXY(61,9); cout << "*"<<endl;
    GotoXY(61,10); cout << "*"<<endl;
    GotoXY(61,11); cout << "*"<<endl;
    GotoXY(61,12); cout << "*"<<endl;
    GotoXY(61,13); cout << "*"<<endl;
    GotoXY(61,14); cout << "*"<<endl;
    GotoXY(61,15); cout << "*"<<endl;
    GotoXY(61,16); cout << "*"<<endl;
    GotoXY(61,17); cout << "*"<<endl;
    GotoXY(61,18); cout << "*"<<endl;
    GotoXY(61,19); cout << "*"<<endl;
}

```

```

        GotoXY(61,20); cout << "*"<<endl;
        GotoXY(61,21); cout << "*"<<endl;
        GotoXY(61,22); cout << "*"<<endl;
        GotoXY(61,23); cout << "*"<<endl;
    }
}

//Функция которая двигает и рисует
enum {KONEC, STENA, PLUS, MOVE};
int Move(Game &g)
{
    int & n = g.gaduka.PCount;
    COORD head = g.gaduka.t[n - 1]; //голова
    COORD tail = g.gaduka.t[0]; //хвост
    COORD next;
    next.X = head.X + g.dx;
    next.Y = head.Y + g.dy; //проверка следующей точки по
направлению
    if(next.X < 3 || next.Y < 3 || next.X > 60 || next.Y >
23) //не уперлась ли в стену?

        return STENA;
    if(n > 4)
    {
        for(int i = 0; i < n; i++)
            if(next.X == g.gaduka.t[i].X && next.Y ==
g.gaduka.t[i].Y) //не наехали ли на себя?
                return KONEC;
    }

    if(next.X == g.jabloko.X && next.Y == g.jabloko.Y)
    {
        COORD*temp = new COORD[ ++n ]; //новый массив больший на
1
        for(int i = 0; i < n; i++)
            temp[i] = g.gaduka.t[i]; //перекопируем
        temp[n - 1] = next; //добавляем одну
        delete [] g.gaduka.t;
        g.gaduka.t = temp;

        SetConsoleCursorPosition(hConsole,head);
        SetConsoleTextAttribute(hConsole, 0x0a); //закрашиваем
яблоко которое съели
        printf("*");
        SetConsoleCursorPosition(hConsole,next);
        SetConsoleTextAttribute(hConsole,0x0a);
        printf("%c",1);
        PlusJabloko(g);
        return PLUS;
    }
}

```

```

        for(int i = 0; i < n - 1; i++)
            g.gaduka.t[i] = g.gaduka.t[i + 1];
        g.gaduka.t[n - 1] = next;
        SetConsoleCursorPosition(hConsole,tail); //закрашиваем
        хВОСТИК
        printf(" ");

        SetConsoleCursorPosition(hConsole,head);
        SetConsoleTextAttribute(hConsole, 0x0a); //красим хвост змеи
        в зеленсй цвет
        printf("*");
        SetConsoleCursorPosition(hConsole,next);
        SetConsoleTextAttribute(hConsole,0x0f); //красим курсор в
        белый цвет (голову змеи)
        printf("%c",1);

        return MOVE;
    }

void StartGame()
{
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    //получаем дескриптор консоли
    int key = 0, count = 0;
    bool Pause=false;
    Game g;
    skorostGame(g);
    STENA_2();
    srand(time(0));
    bool pause = false;
    while(key != 27)
    {
        while(!kbhit()) //ждет пока нажмем
        {
            if(Pause==true)
            {
                Sleep(1);
                continue;
            }

            switch (Move(g)) //движение
            {
                case PLUS:
                    ++count;
                    g.pause-=1;
                    GotoXY(64,2);cout << "Behaviours:" << endl ;
                    GotoXY(64,3); cout << "Apples:" <<count <<
endl;
                    GotoXY(64,4); cout << "Length:"
<<g.gaduka.PCount << endl;
                    GotoXY(64,5); cout << "Speed:" <<g.pause<<
endl;

```

```

        GotoXY(64,7); cout << "Controls:" << endl;
        GotoXY(64,8); cout << "Esc:Exit" << endl;
        GotoXY(64,9); cout << "P:Pause" << endl;
        GotoXY(64,10); cout <<"S:Start" << endl;
        GotoXY(64,11); cout <<"L:Level" << endl;
        GotoXY(64,13);printf("%c",24);cout
<<":Up"<<endl;

        GotoXY(64,14);printf("%c",25);cout<<":Down"<<endl;

        GotoXY(64,15);printf("%c",27);cout<<":Left"<<endl;

        GotoXY(64,16);printf("%c",26);cout<<":Right"<<endl;
        if (count == 75)
        {
            GotoXY(15,1); cout <<
"Congratulations!!!You win!!!!" << endl;
            getch();
            getch();
            getch();
            getch();
            getch();
            return;
        }
        break;

        case STENA:

        case KONEC:
            GotoXY(23,1); printf("Ohhh!!You loose...it's
loss of losses!\n\n\t\t\t");
            getch();
            return;
            break;
    }

    Sleep(g.pause); //Задержка
}
key = getch();

if(key=='P' || key=='p')
    Pause=!Pause;
else if(key=='S' || key=='s')
    ZmejaStart();
else if(key=='L' || key=='l')
    Level();
else if(key==0 || key==224)
{
    key=getch();

    if(key == 72 && g.nap != DOWN)
    {
        g.nap = UP;

```

```

        g.dx = 0;
        g.dy = -1;
    }
    else if(key == 80 && g.nap != UP)
    {
        g.nap = DOWN;
        g.dx = 0;
        g.dy = 1;
    }
    else if(key == 75 && g.nap != RIGHT)
    {
        g.nap = LEFT;
        g.dx = -1;
        g.dy = 0;
    }
    else if(key == 77 && g.nap != LEFT)
    {
        g.nap = RIGHT;
        g.dx = 1;
        g.dy = 0;
    }
    }
}
}
}

```

```

void main()
{
    BOOL fResult;
    BOOL flag = TRUE;
    EMessageBoxId eId;
    int iCommandButton = 0;

    // Инициализация работы с консолью
    fResult = ConsoleInitialize();
    if (fResult == FALSE)
    {
        // return 1;
    }

    // Скрытие курсора
    fResult = SetCursorVisible(FALSE);
    if (fResult == FALSE)
    {
        // return 1;
    }

    DrawMainMenu(80, 25);

    while(flag == TRUE)

```

```

{
    iCommandButton = ChoiceButton();

    switch(iCommandButton)
    {
        case 0:          // Создать
        {
            StartGame();
            continue;
        }
        case 1:          // Открыть
        {
            continue;
        }
        case 2:          //Удалить
        {
            continue;
        }
        case 3: // Выход

            eId =
                DialogMessageBox(
                    ColorBrightWhite, ColorCyan,
                    15, 6, 40, 8,
                    ColorYellow, ColorCyan, L"Вы
действительно хотите выйти ?", MiddleCenter,
                    MessageBoxYesNo);
            if(eId == MessageBoxIdYes)
                flag = FALSE;
            else if(eId == MessageBoxIdNo)
            {
                DrawMainMenu(80, 25);
                continue;
            }
        }
    }

    hConsole = GetStdHandle(STD_OUTPUT_HANDLE); //получаем
дескриптор консоли
    int key = 0, count = 0;
    bool Pause=false;
    Game g;
    skorostGame(g);
    STENA_2();
    srand(time(0));
    bool pause = false;
    while(key != 27)
    {
        while(!kbhit()) //ждет пока нажмем
        {
            if(Pause==true)
            {
                Sleep(1);
                continue;
            }

```

```

    }

    switch (Move(g)) //движение
    {
        case PLUS:
            ++count;
            g.pause-=1;
            GotoXY(64,2);cout << "Behaviours:" << endl ;
            GotoXY(64,3); cout << "Apples:" <<count <<
endl;
            GotoXY(64,4); cout << "Length:"
<<g.gaduka.PCount << endl;
            GotoXY(64,5); cout << "Speed:" <<g.pause<<
endl;
            GotoXY(64,7); cout << "Controls:" << endl;
            GotoXY(64,8); cout << "Esc:Exit" << endl;
            GotoXY(64,9); cout << "P:Pause" << endl;
            GotoXY(64,10); cout <<"S:Start" << endl;
            GotoXY(64,11); cout <<"L:Level" << endl;
            GotoXY(64,13);printf("%c",24);cout
<<":Up"<<endl;

            GotoXY(64,14);printf("%c",25);cout<<":Down"<<endl;

            GotoXY(64,15);printf("%c",27);cout<<":Left"<<endl;

            GotoXY(64,16);printf("%c",26);cout<<":Right"<<endl;
            if (count == 75)
            {
                GotoXY(15,1); cout <<
"Congratulations!!!You win!!!!" << endl;
                getch();
                getch();
                getch();
                getch();
                getch();
                return;
            }
            break;

        case STENA:

        case KONEC:
            GotoXY(23,1); printf("Ohhh!!You loose...it's
loss of losses!\n\n\t\t\t");
            getch();
            return;
            break;
    }

    Sleep(g.pause); //Задержка
}

```

```

key = getch();

if(key=='P' || key=='p')
    Pause=!Pause;
else if(key=='S' || key=='s')
    ZmejaStart();
else if(key=='L' || key=='l')
    Level();
else if(key==0 || key==224)
{
    key=getch();

    if(key == 72 && g.nap != DOWN)
    {
        g.nap = UP;
        g.dx = 0;
        g.dy = -1;
    }
    else if(key == 80 && g.nap != UP)
    {
        g.nap = DOWN;
        g.dx = 0;
        g.dy = 1;
    }
    else if(key == 75 && g.nap != RIGHT)
    {
        g.nap = LEFT;
        g.dx = -1;
        g.dy = 0;
    }
    else if(key == 77 && g.nap != LEFT)
    {
        g.nap = RIGHT;
        g.dx = 1;
        g.dy = 0;
    }
}

}

}

```

Приложение 2. Результаты работы программы



Рисунок 1 – Игровое поле

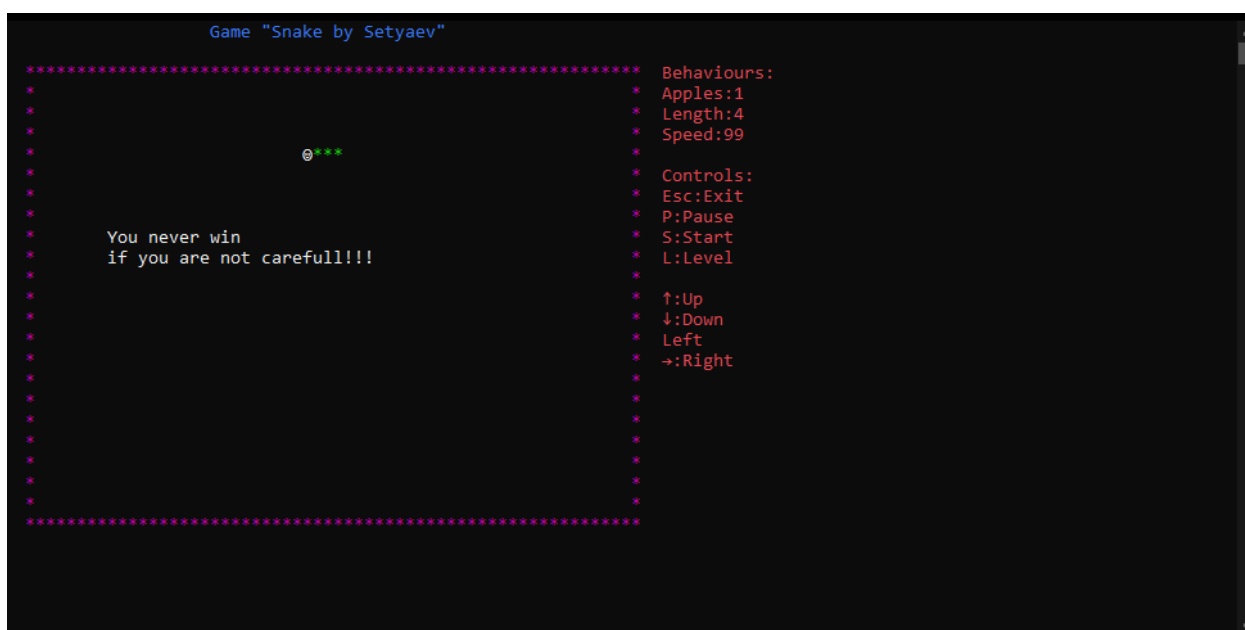


Рисунок 2 – Игровое поле с подсказкой

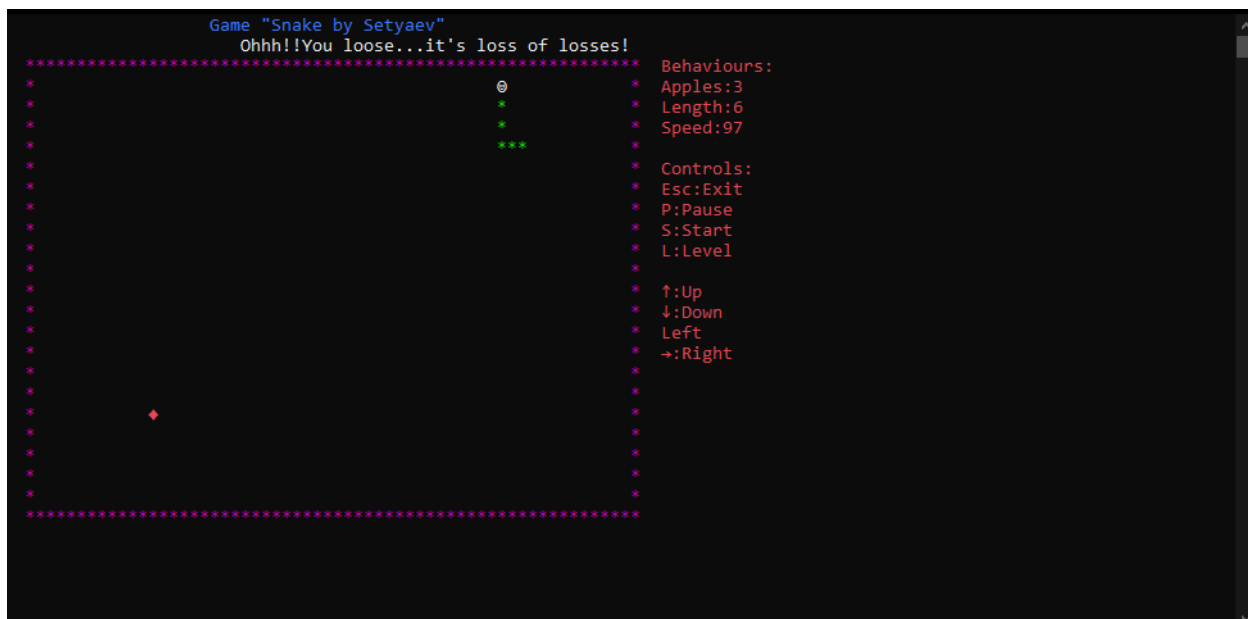


Рисунок 3 – «Смерть» змейки

Приложение 3. Список используемой литературы

1. «ЯЗЫК С» Б.В. Керниган, Д.М. Ритчи.
2. Конспект лекций по курсу «Алгоритмические языки и программирование» ПГУ.
3. Материалы электронных энциклопедий и сообщества MSDN.com
4. «Microsoft Visual Studio 2022» Пауэрс Л., Снелл М.
5. «Сборник по Win32 API» К.Г. Финогенов, П.В. Румянцев, Юрий Щупак, Джеффри Рихтер.