Санкт-Петербургский Государственный

Политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра «Информационные и управляющие системы»

**Курсовой проект**

По дисциплине «Объектно-ориентированное программирование»

Разработка программы «Игра «Snake»

Выполнила студентка гр. 13534/30

Ситникова В. А.

Руководитель доцент Круглов С. К.

Санкт-Петербург

2017 год.

# Содержание

[Содержание 2](#_Toc484979093)

[Введение 3](#_Toc484979094)

[1. Среда разработки приложений Microsoft Visual C++ 4](#_Toc484979095)

[1.1 Основы языка С++ 4](#_Toc484979096)

[1.2 Библиотека стандартных шаблонов (STL) 5](#_Toc484979097)

[1.3 Пакет Microsoft Foundation Classes (MFC) 6](#_Toc484979098)

[1.3 Элементы управления в Microsoft Visual C, MFC 7](#_Toc484979099)

[2. Описание программы 9](#_Toc484979100)

[2.1 Описание компонентов программы 10](#_Toc484979101)

[2.1.1. Описание глобальных переменных, констант и функций. 10](#_Toc484979102)

[2.1.2. Описание переменных, констант и функций класса My\_Snake. 11](#_Toc484979103)

[2.1.3 Иерархия классов: 13](#_Toc484979104)

[2.1.4 Классы 14](#_Toc484979105)

[Открытые члены: 16](#_Toc484979106)

[2.2 Правила игры 19](#_Toc484979107)

[2.3 Демонстрация работы программы 20](#_Toc484979108)

[Заключение 23](#_Toc484979109)

[Приложение А 24](#_Toc484979110)

[Содержание заголовочного файла «My\_Snake.h» 24](#_Toc484979111)

[Содержание описания класса My\_Snake «My\_Snake.CPP» 26](#_Toc484979112)

[Содержание заголовочного файла «SnakeDlg.h» 30](#_Toc484979113)

[Содержание основного файла «SnakeDlg.CPP» 32](#_Toc484979114)

# Введение

Целью курсовой работы является получение опыта самостоятельной разработки пользовательского приложения, работающего под управлением ОС Microsoft Windows, в среде разработки приложений Microsoft Visual C++, c использованием библиотек STL и MFC.

В первой главе курсовой работы приводится краткое описание возможностей интегрированной среды разработки приложений Microsoft Visual C++, а так же краткое описание используемых ресурсов: языка программирования С++ и библиотек STL и MFC.

Во второй главе приводится краткое описание программы и демонстрирование ее работы.

В заключении приводится перечень изученных вопросов и основных результатов курсовой работы.

В приложение вынесены некоторые части программного кода с комментариями.

# 1. Среда разработки приложений Microsoft Visual C++

Microsoft Visual C++ (MSVC) — интегрированная среда разработки приложений на языке C++, разработанная фирмой Microsoft и поставляемая либо как часть комплекта Microsoft Visual Studio, либо отдельно в виде бесплатного функционально ограниченного комплекта Visual C++ Express Edition.

## 1.1 Основы языка С++

C++ — [компилируемый](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) [статически типизированный](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [язык программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения.

Поддерживает такие [парадигмы программирования](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D1%8B_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как [высокоуровневых](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), так и [низкоуровневых языков](http://ru.wikipedia.org/wiki/%D0%9D%D0%B8%D0%B7%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). В сравнении с его предшественником — языком [C](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), — наибольшее внимание уделено поддержке [объектно-ориентированного](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [обобщённого программирования](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

## 1.2 Библиотека стандартных шаблонов (STL)

Библиотека стандартных шаблонов (STL) (англ. Standard Template Library) — набор согласованных обобщённых алгоритмов, контейнеров, средств доступа к их содержимому и различных вспомогательных функций в C++.

Библиотека стандартных шаблонов до включения в стандарт C++ была сторонней разработкой, вначале — фирмы HP, а затем SGI. Стандарт языка не называет её «STL», так как эта библиотека стала неотъемлемой частью языка, однако многие люди до сих пор используют это название, чтобы отличать её от остальной части стандартной библиотеки (потоки ввода-вывода (iostream), подраздел Си и др.).

Проект под названием STLPort, основанный на SGI STL, осуществляет постоянное обновление STL, iostream и строковых классов. Некоторые другие проекты также занимаются разработкой частных применений стандартной библиотеки для различных конструкторских задач. Каждый производитель компиляторов C++ обязательно поставляет какую-либо реализацию этой библиотеки, так как она является очень важной частью стандарта и широко используется.

## 1.3 Пакет Microsoft Foundation Classes (MFC)

Пакет Microsoft Foundation Classes (MFC) — библиотека на языке C++, разработанная Microsoft и призванная облегчить разработку GUI-приложений для Microsoft Windows путём использования богатого набора библиотечных классов.

Библиотека MFC, как и её основной конкурент, Borland VCL, облегчает работу с GUI путём создания каркаса приложения — «скелетной» программы, автоматически создаваемой по заданному макету интерфейса и полностью берущей на себя рутинные действия по его обслуживанию (отработка оконных событий, пересылка данных между внутренними буферами элементов и переменными программы и т. п.). Программисту после генерации каркаса приложения необходимо только вписать код в места, где требуются специальные действия. Каркас должен иметь вполне определенную структуру, поэтому для его генерации и изменения в Visual C++ предусмотрены мастера.

Кроме того, MFC предоставляет объектно-ориентированный слой обёрток (англ. wrappers) над множеством функций Windows API, делающий несколько более удобной работу с ними. Этот слой представляет множество встроенных в систему объектов (окна, виджеты, файлы и т. п.) в виде классов и опять же берёт на себя рутинные действия вроде закрытия дескрипторов и выделения/освобождения памяти.

## 1.3 Элементы управления в Microsoft Visual C, MFC

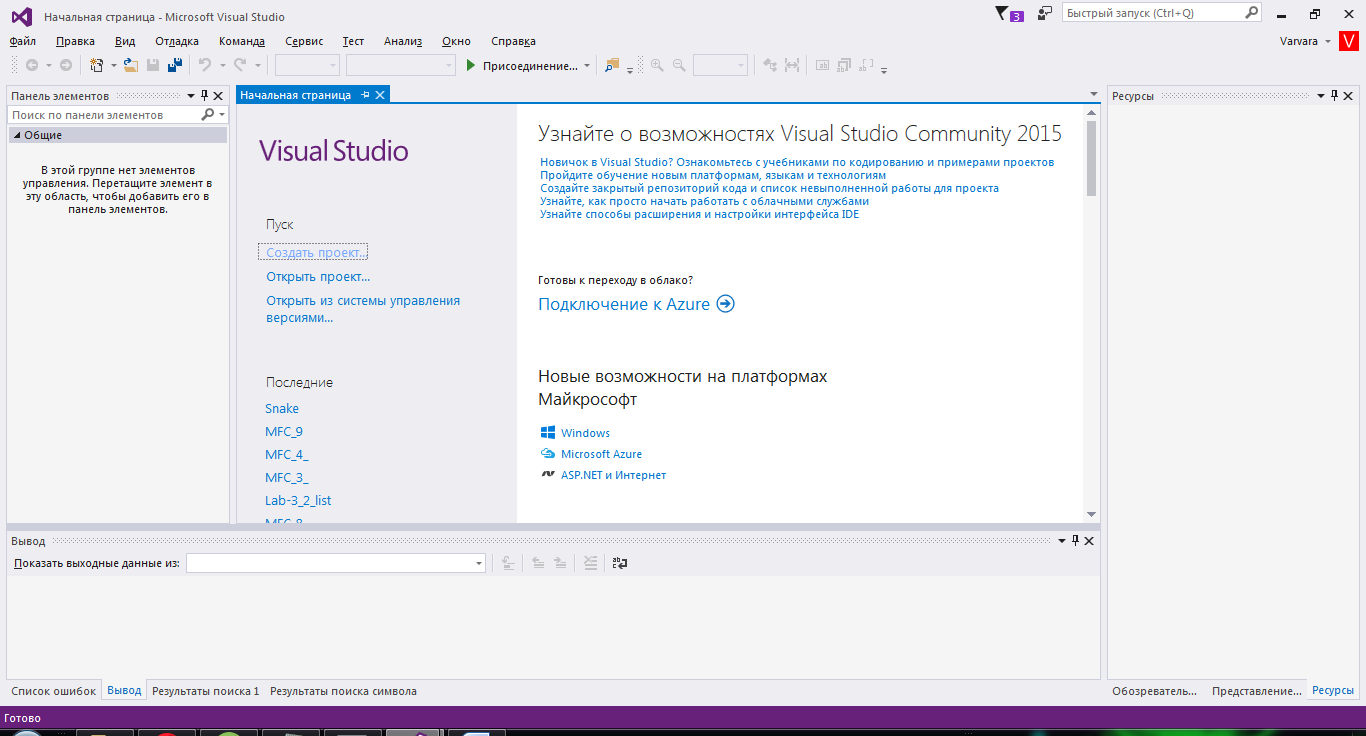


Рисунок 1.1 – Главное окно Microsoft Visual Studio

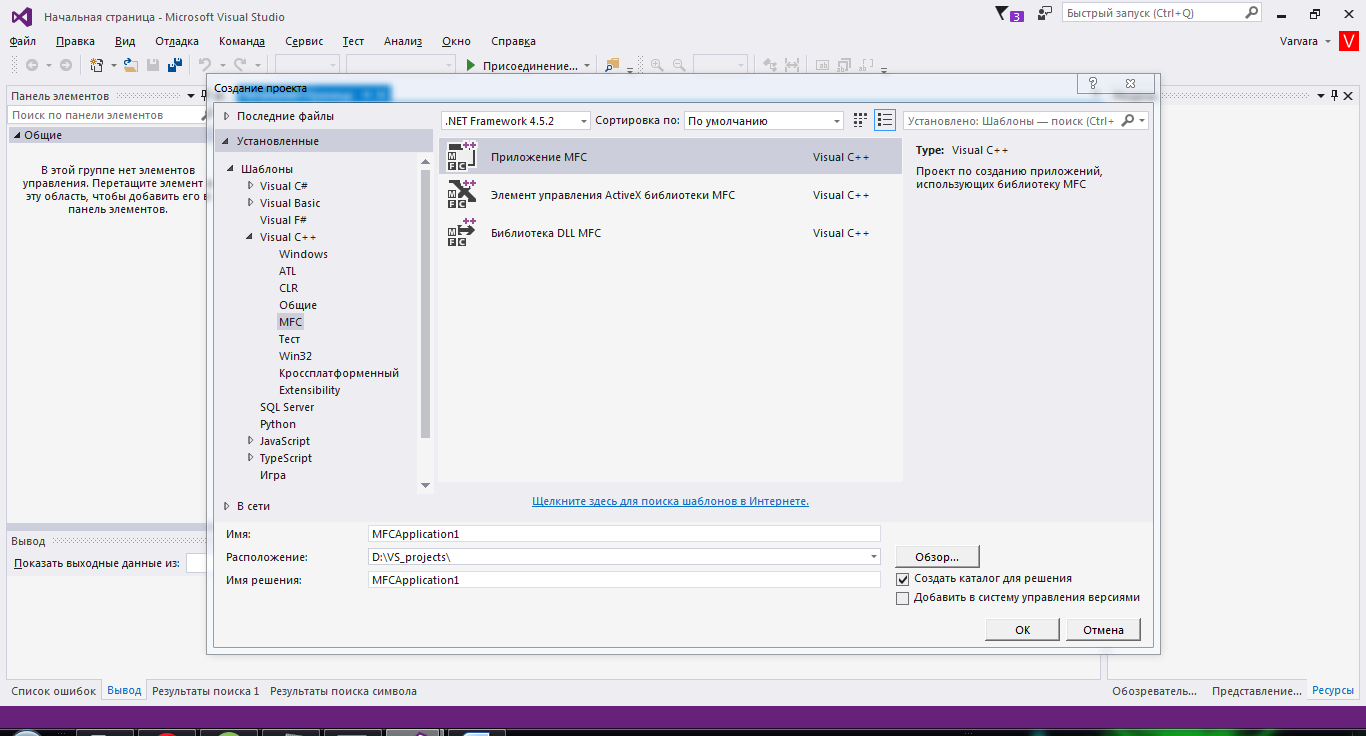


Рисунок 1.2 – Окно создания проекта

Добавление кода приложения к каркасу реализовано двумя способами. Первый использует механизм наследования: основные программные структуры каркаса представлены в виде классов, наследуемых от библиотечных. В этих классах предусмотрено множество виртуальных функций, вызываемых в определенные моменты работы программы. Путём доопределения (в большинстве случаев необходимо вызвать функцию базового класса) этих функций программист может добавлять выполнение в эти моменты своего кода.

Второй способ используется для добавления обработчиков оконных событий. Мастер создает внутри каркасов классов, связанных с окнами, специальные массивы — карты (оконных) сообщений (англ. message map), содержащие пары «ID сообщения — указатель на обработчик». При добавлении/удалении обработчика мастер вносит изменения в соответствующую карту сообщений.

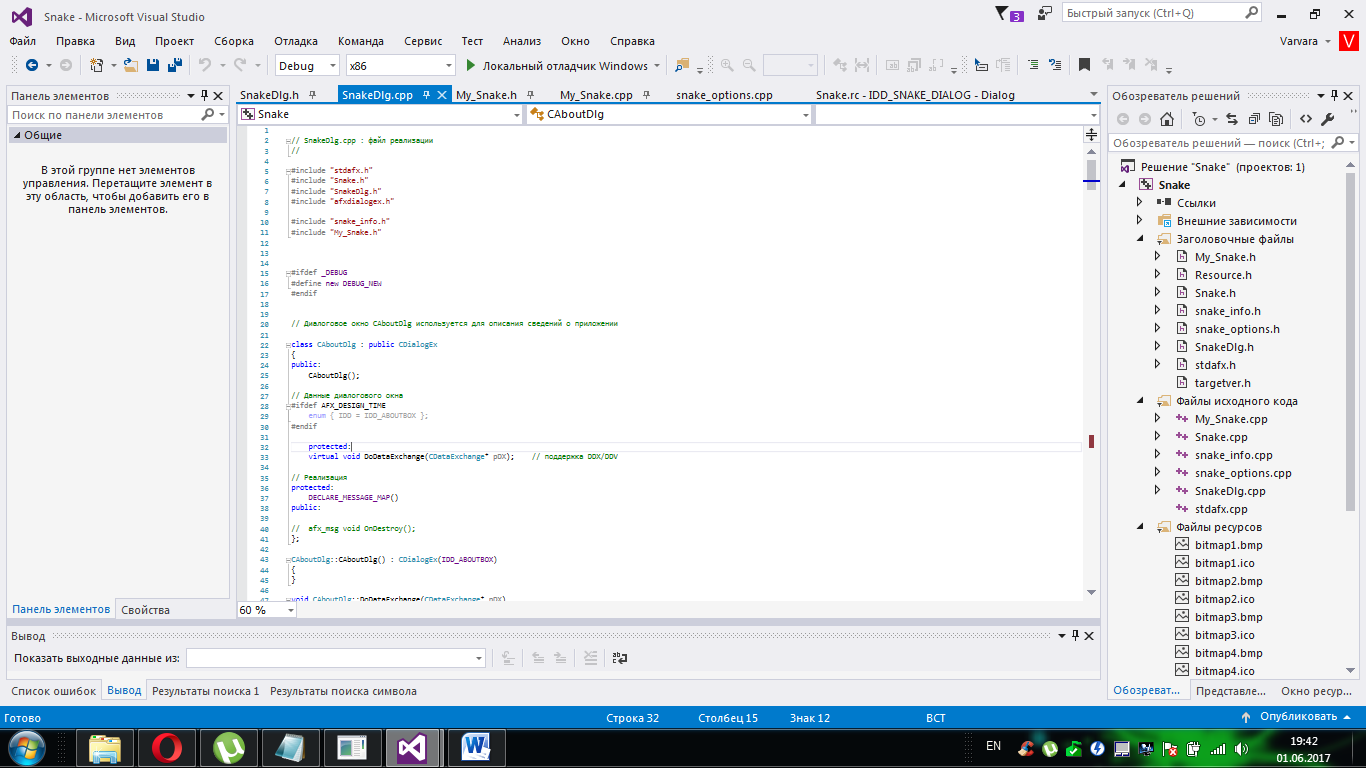


Рисунок 1.3 – Окно разработки программы

# 2. Описание программы

Приложение представляет собой прямоугольное окно, содержащее меню, игровое поле и две кнопки: «новая игра» и «выход». Меню, в свою очередь содержит справку об игре и приложении, настройки для изменения скорости и цвета змейки.

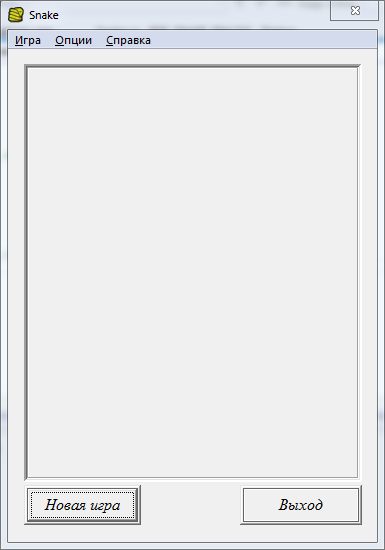


Рисунок 2.1 – Внешний вид приложения

## 2.1 Описание компонентов программы

### 2.1.1. Описание глобальных переменных, констант и функций.

В таблице 2.1 перечислены используемые в программе глобальные константы и переменные, в таблице 2.2 – функции.

Таблица 2.1 – Глобальные константы и переменные

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| my\_game\_field | CRect | Переменная игрового поля |
| \_point | Int | Учет игровых очков |
| Snake | My\_Snake | Собственный класс для работы со змейкой |
| InstallTimer | Int | Проверка установки таймера |

Таблица 2.2 – функции

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| OnPaint | void | Стандартная функция – перерисовывает приложение |
| OnInitDialog | BOOL | Инициализирует начальные условия визуальзации |
| OnTimer | void | Функция срабатывает через определенные промежутки времени, задает основные игровые действия |
| OnDestroy | Void | Функция корректного завершения приложения |
| PreTranslateMessage | BOOL | Обеспечивает управление стрелками |
| Snake\_take\_snake | Void | Функция проверки съела ли змея сама себя (конец игры) |
| set\_timer\_game | Void | Функция устанавливает таймер |
| Snake\_take\_food | Void | Функция проверки съела ли змея еду (рост змеи) |

Так же в программе используются стандартные функции, созданные мастером классов, такие как: нажатия на кнопки или использования элементов меню.

### 2.1.2. Описание переменных, констант и функций класса My\_Snake.

В таблице 2.3 перечислены константы и переменные класса My\_Snake (отвечает за змейку, ее координаты, цвет и т.д.), в таблице 2.4 – функции.

Таблица 2.1 –константы и переменные класса My\_Snake

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| Privet: |  |  |
| XY | int | Направление движения |
| My\_speed | Int | Скорость |
| My\_game | Int | Процесс игры: 0 – ничего не происходит, 1 – запуск игры, 2 – игра в процессе, 3 – игра закончена |
| pen\_color\_black | CPoint | Черное перо рисования змейки |
| pen\_color\_red | CPoint | Красное перо рисования змейки |
| pen\_color\_blue | CPoint | Синее перо рисования змейки |
| pen\_color\_green | CPoint | Зеленое перо рисования змейки |
| pen\_color\_purple | CPoint | Пурпурное перо рисования змейки |
| pen\_color\_cyan | CPoint | Голубое перо рисования змейки |
| pen\_color\_yelloy | CPoint | Желтое перо рисования змейки |
| Where\_Food | int | Проверка нахождения еды на поле |
| Public: |  |  |
| \*My\_Pen\_Snake | CPoint | Указатель, позволяет изменить цвет змейки |
| Food | CPoint | Координаты еды |
| Coord | vector <CPoint> | Вектор, хранит в себе координаты тела змейки |
| \*const move | const int | Указатель на XY |
| \*const speed | const int | Указатель на My\_speed |
| \*const game | const int | Указатель на My\_game |
| \*const have\_food | const int | Указатель на Where\_Food |

Таблица 2.2 – функции класса My\_Snake

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| My\_Snake | Void | Конструктор |
| ~My\_Snake | Void | Стандартный деконструктор |
| New\_snake(CPoint) | Void | Функция создает новую змейку |
| Nothing | Void | Переводит My\_game в положение 0 (ничего не происходит) |
| New\_game | Void | Переводит My\_game в положение 1 (запускает новую игру) |
| Go\_game | Void | Переводит My\_game в положение 2 (игра в процессе) |
| Game\_over | Void | Переводит My\_game в положение 3 (игра окончена) |
| Go\_riglt | Void | Изменяет перемещение впаво |
| Go\_left | void | Изменяет перемещение влево |
| Go\_up | Void | Изменяет перемещение вверх |
| Go\_bottom | Void | Изменяет перемещение вниз |
| color\_black | Void | Устанавливает черный цвет |
| color\_red | Void | Устанавливает красный цвет |
| color\_blue | Void | Устанавливает синий цвет |
| color\_green | Void | Устанавливает зеленый цвет |
| color\_purple | Void | Устанавливает пурпурный цвет |
| color\_cyan | Void | Устанавливает голубой цвет |
| color\_yelloy | Void | Устанавливает желтый цвет |
| set\_speed(int) | Void | Устанавливает скорость |
| New\_food(CPoint) | CPoint | Устанавливает Where\_Food в положение 1 (еда на поле) |
| Eat\_food() | Void | Устанавливает Where\_Food в положение 0 (еда съедена) |

### 2.1.3 Иерархия классов:

* CDialogEx
  + CAboutDlg
  + CSnakeDlg
  + snake\_info
  + snake\_options
* CWinApp
  + CSnakeApp
* My\_Snake

### 2.1.4 Классы

#### 2.1.4.1 Класс CAboutDlg



Рисунок 2.2 – Граф наследования CAboutDlg

Защищенные члены***:***

* virtual void **DoDataExchange** (CDataExchange \*pDX)

#### 2.1.4.2 Класс CSnakeApp



Рисунок 2.3 – Граф наследования CSnakeApp

Открытые члены:

* virtual BOOL **InitInstance** ()

Объявления и описания членов классов находятся в файлах:

* Snake.h
* Snake.cpp

#### 2.1.4.3. Класс CSnakeDlg



Рисунок 2.4 – Граф наследования :CSnakeDlg

Открытые члены:

* **CSnakeDlg** (CWnd \*pParent=NULL)
* afx\_msg void **OnBnClickedexet** ()
* afx\_msg void **OnBnClickednewgame** ()
* afx\_msg void **OnMy11** ()
* afx\_msg void **OnMy12** ()
* afx\_msg void **OnMy221** ()
* afx\_msg void **OnMy222** ()
* afx\_msg void **OnMy223** ()
* afx\_msg void **OnMy224** ()
* afx\_msg void **OnMy225** ()
* afx\_msg void **OnMy226** ()
* afx\_msg void **OnMy227** ()
* afx\_msg void **OnMy231** ()
* afx\_msg void **OnMy232** ()
* afx\_msg void **OnMy233** ()
* afx\_msg void **OnMy31** ()
* afx\_msg void **OnMy32** ()
* void **set\_timer\_game** ()
* void **Snake\_take\_food** ()
* void **Snake\_take\_snake** ()
* afx\_msg void **OnTimer** (UINT\_PTR nIDEvent)
* afx\_msg void **OnDestroy** ()
* afx\_msg BOOL **PreTranslateMessage** (MSG \*pMsg)

Открытые атрибуты:

* int **\_point**
* **snake\_info** **S\_info**
* CRect **my\_game\_field**
* **My\_Snake** **Snake**
* int **InstallTimer**

Поля данных:

* int **\_point**
* **snake\_info** **S\_info**
* CRect **my\_game\_field**
* **My\_Snake** **Snake**
* int **InstallTimer**

Защищенные члены:

* virtual void **DoDataExchange** (CDataExchange \*pDX)
* virtual BOOL **OnInitDialog** ()
* afx\_msg void **OnSysCommand** (UINT nID, LPARAM lParam)
* afx\_msg void **OnPaint** ()
* afx\_msg HCURSOR **OnQueryDragIcon** ()

Защищенные данные:

* HICON **m\_hIcon**

Объявления и описания членов классов находятся в файлах:

* SnakeDlg.h
* SnakeDlg.cpp

#### 2.1.4.4 Класс My\_Snake

### Открытые члены:

* void **New\_snake** (CPoint)
* void **Nothing** ()
* void **New\_game** ()
* void **Go\_game** ()
* void **Game\_over** ()
* void **Go\_riglt** ()
* void **Go\_left** ()
* void **Go\_up** ()
* void **Go\_bottom** ()
* void **color\_black** ()
* void **color\_red** ()
* void **color\_blue** ()
* void **color\_green** ()
* void **color\_purple** ()
* void **color\_cyan** ()
* void **color\_yelloy** ()
* void **set\_speed** (int)
* CPoint **New\_food** (CPoint)
* void **Eat\_food** ()

Поля данных

* CPen \* **My\_Pen\_Snake**
* CPoint **Food**
* std::vector< CPoint > **coord**
* const int \*const **move** = &XY
* const int \*const **speed** = &My\_speed
* const int \*const **game** = &My\_game
* const int \*const **have\_food** = &Where\_Food

Открытые атрибуты:

* CPen \* **My\_Pen\_Snake**
* CPoint **Food**
* std::vector< CPoint > **coord**
* const int \*const **move** = &XY
* const int \*const **speed** = &My\_speed
* const int \*const **game** = &My\_game
* const int \*const **have\_food** = &Where\_Food

Объявления и описания членов классов находятся в файлах:

* My\_Snake.h
* My\_Snake.cpp

#### 2.1.4.5 Класс snake\_info



Рисунок 2.5 – Граф наследования snake\_info

Открытые члены:

* **snake\_info** (CWnd \*pParent=NULL)
* afx\_msg void **OnStnClickedBitmap2** ()

Защищенные члены:

* virtual void **DoDataExchange** (CDataExchange \*pDX)

Объявления и описания членов классов находятся в файлах:

* snake\_info.h
* snake\_info.cpp

## 2.2 Правила игры

Правила игры "Snake":

* Змейка всё время движется, перемещаясь на заданное расстояние через определённый промежуток времени.
* Управление змейкой происходит через клавиши - стрелки.
* Когда змейка достигает еды, её длина увеличивается на одно звено.
* Когда змейка втыкается в саму себя, вы проигрываете.
* Достигнув края поля, змейка появляется с другой стороны поля.
* Суть игры увеличить змейку на столько, на сколько это возможно.

## 2.3 Демонстрация работы программы

Для начала работы с программой нужно запустить файл Snake.exe. из каталога \Snake\Debug. Появится окно программы (рисунок 2.1).

Для запуска новой игры необходимо нажать на соответствующую кнопку или запустить через меню Игра\Новая игра. Управление ведется через клавиши – стрелки, при нажатии на кнопку – змейка будет двигаться в соответствующую сторону.

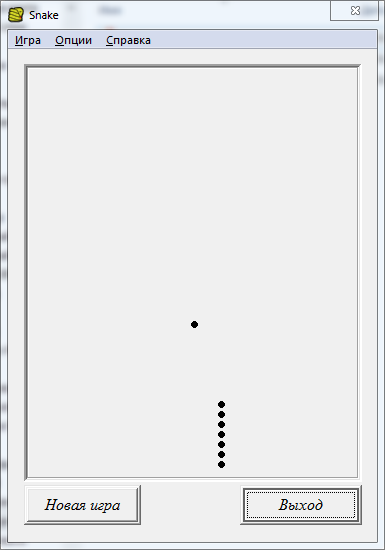


Рисунок 2.2 – Процесс игры

При достижении головой змейки еды, её длина увеличивается. Достигнув края поля, змейка появляется с другой стороны поля. Цвет можно поменять в меню Опции\Цвет змейки. (Рисунок 2.3)

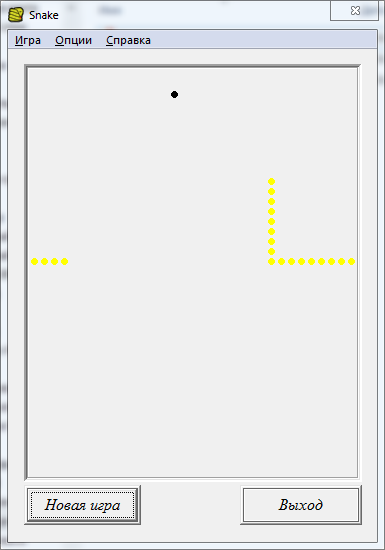


Рисунок 2.3 – Результат достижения края поля и изменения цвета

Скорость змейки может так же меняться, при этом изменяется количество получаемых очков.

Когда змейка втыкается в саму себя, вы проигрываете (рисунок 2.4). На игровом поле выводится надпись «Game over» и количество заработанных очков.



Рисунок 2.7 –Конец игры

# Заключение

В ходе выполнения курсовой работы были более подробно изучены следующие вопросы:

Использование стандартной библиотеки STL и ее контейнера – “Vector”, а так же ее свойств и методов.

Создание собственного класса;

Использование библиотеки MFC и взаимодействие с помощью нее с WinAPI.

Полученные знания использованы на практике. Результатом работы является приложение-игра «Snake».

# Приложение А

Листинг программы

В приложении приведены не все части кода, а только те, которые были написаны, непосредственно, во время создания приложения «Snake»

## Содержание заголовочного файла «My\_Snake.h»

#pragma once

#include "vector"

class My\_Snake

{

int XY; //направление перемещения: 0-вверх, 1-вправо, 2-вниз, 3-влево

int My\_speed; //скорость змейки, 1 - 3, 0 - игра на паузе

int My\_game; //если 0 - ничего не происходит

// 1 - запуск игры

// 2 - игра в процессе

// 3 - игра закончена

CPen pen\_color\_black; // перо рисования

CPen pen\_color\_red;

CPen pen\_color\_blue;

CPen pen\_color\_green;

CPen pen\_color\_purple;

CPen pen\_color\_cyan;

CPen pen\_color\_yelloy;

int Where\_Food; //Проверка: 1 - еда на поле, 0 - еды нет на поле

public:

CPen \*My\_Pen\_Snake;

CPoint Food; //Координаты еды

std::vector <CPoint> coord; //вектор, хранящие точки частей тела змейки, размер вектора определяет длину змейки

const int \*const move = &XY; //константные указатели на переменные

const int \*const speed = &My\_speed;

const int \*const game = &My\_game;

const int \*const have\_food = &Where\_Food;

My\_Snake();

~My\_Snake();

void New\_snake(CPoint); //ф-ция новой змейки

void Nothing(); //ничего не происходит

void New\_game(); //запускает новую игру

void Go\_game(); //игра в процессе

void Game\_over(); //конец игры

void Go\_riglt(); //изменение перемещения

void Go\_left();

void Go\_up();

void Go\_bottom();

void color\_black(); //установка цвета змеи

void color\_red();

void color\_blue();

void color\_green();

void color\_purple();

void color\_cyan();

void color\_yelloy();

void set\_speed(int); //установка скорости змеи

CPoint New\_food(CPoint); //Еда на поле

void Eat\_food(); //Еда съедина

};

## Содержание описания класса My\_Snake «My\_Snake.CPP»

#include "stdafx.h"

#include "My\_Snake.h"

#include "vector"

My\_Snake::My\_Snake()

{

pen\_color\_black.CreatePen(PS\_SOLID, 7, RGB(0, 0, 0));

pen\_color\_red.CreatePen(PS\_SOLID, 7, RGB(255, 0, 0));

pen\_color\_blue.CreatePen(PS\_SOLID, 7, RGB(0, 0, 255));

pen\_color\_green.CreatePen(PS\_SOLID, 7, RGB(0, 255, 0));

pen\_color\_purple.CreatePen(PS\_SOLID, 7, RGB(255, 0, 255));

pen\_color\_cyan.CreatePen(PS\_SOLID, 7, RGB(0, 255, 255));

pen\_color\_yelloy.CreatePen(PS\_SOLID, 7, RGB(255, 255, 0));

My\_Pen\_Snake = &pen\_color\_black; //изначально цвет черный

My\_game = 0; //игра не начата

My\_speed = 1; //скорость=1

Where\_Food = 0; //Изначально еды нет

}

My\_Snake::~My\_Snake()

{

}

//создаем новую змею для игры, длина змеи = 3;

//выбираем направление движения

void My\_Snake::New\_snake(CPoint A)

{

XY = rand() % 4;

std::vector <CPoint> new\_coord(3);

coord = new\_coord;

coord[0] = A;

if (XY==0)

{

CPoint B(0,10);

A += B; coord[1] = A;

A += B; coord[2] = A;

}

else if (XY==1)

{

CPoint B(-10,0);

A += B; coord[1] = A;

A += B; coord[2] = A;

}

else if (XY==2)

{

CPoint B(0,-10);

A += B; coord[1] = A;

A += B; coord[2] = A;

}

else

{

CPoint B(10,0);

A += B; coord[1] = A;

A += B; coord[2] = A;

}

}

//изменения перемещения

void My\_Snake::Go\_up()

{

XY = 0;

}

void My\_Snake::Go\_riglt()

{

XY = 1;

}

void My\_Snake::Go\_bottom()

{

XY = 2;

}

void My\_Snake::Go\_left()

{

XY = 3;

}

//изменение парамитра игры My\_game:

// 0 - ничего не происходит

// 1 - запуск игры

// 2 - игра в процессе

// 3 - игра на паузе

// 4 - игра закончена

void My\_Snake::Nothing()

{

My\_game = 0;

}

void My\_Snake::New\_game()

{

My\_game = 1;

}

void My\_Snake::Go\_game()

{

My\_game = 2;

}

void My\_Snake::Game\_over()

{

My\_game = 3;

}

//Установка цвета змеи

void My\_Snake::color\_black()

{

My\_Pen\_Snake = &pen\_color\_black;

}

void My\_Snake::color\_red()

{

My\_Pen\_Snake = &pen\_color\_red;

}

void My\_Snake::color\_blue()

{

My\_Pen\_Snake = &pen\_color\_blue;

}

void My\_Snake::color\_green()

{

My\_Pen\_Snake = &pen\_color\_green;

}

void My\_Snake::color\_purple()

{

My\_Pen\_Snake = &pen\_color\_purple;

}

void My\_Snake::color\_cyan()

{

My\_Pen\_Snake = &pen\_color\_cyan;

}

void My\_Snake::color\_yelloy()

{

My\_Pen\_Snake = &pen\_color\_yelloy;

}

//установка скорости змеи

void My\_Snake::set\_speed(int A)

{

My\_speed = A;

}

CPoint My\_Snake::New\_food(CPoint A)

{

if (Where\_Food==0)

{

Food = A;

Where\_Food = 1;

}

return A;

}

void My\_Snake::Eat\_food()

{

Where\_Food = 0;

}

## Содержание заголовочного файла «SnakeDlg.h»

// SnakeDlg.h : файл заголовка

//

#include "snake\_info.h"

#include "My\_Snake.h"

#include "vector"

#pragma once

// диалоговое окно CSnakeDlg

class CSnakeDlg : public CDialogEx

{

// Создание

public:

CSnakeDlg(CWnd\* pParent = NULL); // стандартный конструктор

// Данные диалогового окна

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_SNAKE\_DIALOG };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

// Реализация

protected:

HICON m\_hIcon;

// Созданные функции схемы сообщений

virtual BOOL OnInitDialog();

afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);

afx\_msg void OnPaint();

afx\_msg HCURSOR OnQueryDragIcon();

DECLARE\_MESSAGE\_MAP()

public:

afx\_msg void OnBnClickedexet();

afx\_msg void OnBnClickednewgame();

afx\_msg void OnMy11();

afx\_msg void OnMy12();

afx\_msg void OnMy221();

afx\_msg void OnMy222();

afx\_msg void OnMy223();

afx\_msg void OnMy224();

afx\_msg void OnMy225();

afx\_msg void OnMy226();

afx\_msg void OnMy227();

afx\_msg void OnMy231();

afx\_msg void OnMy232();

afx\_msg void OnMy233();

afx\_msg void OnMy31();

afx\_msg void OnMy32();

void set\_timer\_game(); //ф-ция запуска таймера

void Snake\_take\_food(); //Ф-ция проверки - съела ли змея еду

void Snake\_take\_snake(); //Ф-ция проверки - съела ли змея себя

int \_point; //Учет игровых очков

snake\_info S\_info; //Класс окна информации

CRect my\_game\_field; //Переменная игрового поля

My\_Snake Snake; //Класс для работы со змеей

int InstallTimer; //Переменная проверки установки таймера

afx\_msg void OnTimer(UINT\_PTR nIDEvent);

afx\_msg void OnDestroy();

afx\_msg BOOL PreTranslateMessage(MSG\* pMsg);

};

## Содержание основного файла «SnakeDlg.CPP»

// SnakeDlg.cpp : файл реализации

//

#include "stdafx.h"

#include "Snake.h"

#include "SnakeDlg.h"

#include "afxdialogex.h"

#include "snake\_info.h"

#include "My\_Snake.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// Диалоговое окно CAboutDlg используется для описания сведений о приложении

class CAboutDlg : public CDialogEx

{

public:

CAboutDlg();

// Данные диалогового окна

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_ABOUTBOX };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

// Реализация

protected:

DECLARE\_MESSAGE\_MAP()

public:

// afx\_msg void OnDestroy();

};

CAboutDlg::CAboutDlg() : CDialogEx(IDD\_ABOUTBOX)

{

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)

// ON\_WM\_DESTROY()

END\_MESSAGE\_MAP()

// диалоговое окно CSnakeDlg

CSnakeDlg::CSnakeDlg(CWnd\* pParent /\*=NULL\*/)

: CDialogEx(IDD\_SNAKE\_DIALOG, pParent)

, \_point(0)

{

m\_hIcon = AfxGetApp()->LoadIcon(IDI\_ICON1); //Замена иконки приложения на свою.

}

void CSnakeDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CSnakeDlg, CDialogEx)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_BN\_CLICKED(my\_exet, &CSnakeDlg::OnBnClickedexet)

ON\_BN\_CLICKED(my\_new\_game, &CSnakeDlg::OnBnClickednewgame)

ON\_COMMAND(my\_1\_1, &CSnakeDlg::OnMy11)

ON\_COMMAND(my\_1\_2, &CSnakeDlg::OnMy12)

ON\_COMMAND(my\_2\_2\_1, &CSnakeDlg::OnMy221)

ON\_COMMAND(my\_2\_2\_2, &CSnakeDlg::OnMy222)

ON\_COMMAND(my\_2\_2\_3, &CSnakeDlg::OnMy223)

ON\_COMMAND(my\_2\_2\_4, &CSnakeDlg::OnMy224)

ON\_COMMAND(my\_2\_2\_5, &CSnakeDlg::OnMy225)

ON\_COMMAND(my\_2\_2\_6, &CSnakeDlg::OnMy226)

ON\_COMMAND(my\_2\_2\_7, &CSnakeDlg::OnMy227)

ON\_COMMAND(my\_2\_3\_1, &CSnakeDlg::OnMy231)

ON\_COMMAND(my\_2\_3\_2, &CSnakeDlg::OnMy232)

ON\_COMMAND(my\_2\_3\_3, &CSnakeDlg::OnMy233)

ON\_COMMAND(my\_3\_1, &CSnakeDlg::OnMy31)

ON\_COMMAND(my\_3\_2, &CSnakeDlg::OnMy32)

ON\_WM\_TIMER()

ON\_WM\_DESTROY()

ON\_WM\_KEYDOWN()

END\_MESSAGE\_MAP()

// обработчики сообщений CSnakeDlg

BOOL CSnakeDlg::OnInitDialog()

{

CDialogEx::OnInitDialog(); //----начальные условия

// Добавление пункта "О программе..." в системное меню.

// IDM\_ABOUTBOX должен быть в пределах системной команды.

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != NULL)

{

BOOL bNameValid;

CString strAboutMenu;

bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);

ASSERT(bNameValid);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

// Задает значок для этого диалогового окна. Среда делает это автоматически,

// если главное окно приложения не является диалоговым

SetIcon(m\_hIcon, TRUE); // Крупный значок

SetIcon(m\_hIcon, FALSE); // Мелкий значок

// ввTODO: добавьте дополнительную инициализацию

//////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////////

//---------------------------Игровое поле------------------------//

//------------Инициализируем переменную игрового окна------------//

CWnd \* uk\_my\_game\_field = GetDlgItem(my\_field);

uk\_my\_game\_field->GetWindowRect(&my\_game\_field);

ScreenToClient(&my\_game\_field);

my\_game\_field.DeflateRect(10,10,10,10);

//////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////////

return TRUE; // возврат значения TRUE, если фокус не передан элементу управления

}

void CSnakeDlg::OnSysCommand(UINT nID, LPARAM lParam)

{

if ((nID & 0xFFF0) == IDM\_ABOUTBOX)

{

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else

{

CDialogEx::OnSysCommand(nID, lParam);

}

}

// При добавлении кнопки свертывания в диалоговое окно нужно воспользоваться приведенным ниже кодом,

// чтобы нарисовать значок. Для приложений MFC, использующих модель документов или представлений,

// это автоматически выполняется рабочей областью.

void CSnakeDlg::OnPaint() //Функция перерисовывания окна

{

CPaintDC dc(this); // Получить контекст устройства

if (\*Snake.game == 1) //Создание новой игры

{

dc.SelectObject(Snake.My\_Pen\_Snake); //Цвет змеи

Snake.New\_snake(my\_game\_field.CenterPoint()); //Нарисовать змею по координатам

for (int i = 0; i < Snake.coord.size(); dc.MoveTo(Snake.coord[i]), dc.LineTo(Snake.coord[i]), i++);

Snake.Go\_game();

}

else if (\*Snake.game == 2) //перерисовывание в процессе игры

{

//Рисуем змею

dc.SelectObject(Snake.My\_Pen\_Snake);

for (int i = 0; i < Snake.coord.size(); dc.MoveTo(Snake.coord[i]), dc.LineTo(Snake.coord[i]), i++);

//Рисуем еду

CPen New\_pen(PS\_SOLID, 7, RGB(0, 0, 0));

dc.SelectObject(New\_pen);

if (\*Snake.have\_food == 0) //Проверка наличия еды на поле

{

CPoint B = my\_game\_field.TopLeft();

int X = rand() % 32;

int Y = rand() % 40;

B.x += X \* 10;

B.y += Y \* 10;

dc.MoveTo(Snake.New\_food(B));

dc.LineTo(B);

}

else

{

dc.MoveTo(Snake.Food);

dc.LineTo(Snake.Food);

}

}

else if (\*Snake.game == 3) //конец игры

{

CString S;

S.Format("Вы набрали %d очков", \_point);

dc.TextOut(150, 200, "GAME OVER");

dc.TextOut(125, 230, S);

}

}

// Система вызывает эту функцию для получения отображения курсора при перемещении

// свернутого окна.

HCURSOR CSnakeDlg::OnQueryDragIcon()

{

return static\_cast<HCURSOR>(m\_hIcon);

}

void CSnakeDlg::OnBnClickedexet() //Кнопка - выход

{

int iResults;

iResults = MessageBox("Вы уверены, что хотите выйти?",

"Выход?",

MB\_YESNO + MB\_ICONQUESTION);

if (iResults == IDYES)

{

OnOK();

}

}

void CSnakeDlg::OnBnClickednewgame() //Кнопка - новая игра

{

\_point = 0;

Snake.New\_game();

Snake.Eat\_food();

set\_timer\_game();

Invalidate();

}

void CSnakeDlg::OnMy11() //Новая игра

{ OnBnClickednewgame(); }

void CSnakeDlg::OnMy12() //Выход

{ OnBnClickedexet(); }

void CSnakeDlg::OnMy221() //Цвет черный

{

Snake.color\_black();

Invalidate();

}

void CSnakeDlg::OnMy222() //Цвет синий

{

Snake.color\_blue();

Invalidate();

}

void CSnakeDlg::OnMy223() //Цвет красный

{

Snake.color\_red();

Invalidate();

}

void CSnakeDlg::OnMy224() //Цвет зеленый

{

Snake.color\_green();

Invalidate();

}

void CSnakeDlg::OnMy225() //Цвет пурпурнвй

{

Snake.color\_purple();

Invalidate();

}

void CSnakeDlg::OnMy226() //Цвет голубой

{

Snake.color\_cyan();

Invalidate();

}

void CSnakeDlg::OnMy227() //Цвет желтый

{

Snake.color\_yelloy();

Invalidate();

}

void CSnakeDlg::OnMy231() //Скорость 1

{

Snake.set\_speed(1);

if (\*Snake.game == 2) set\_timer\_game();

}

void CSnakeDlg::OnMy232() //Скорость 2

{

Snake.set\_speed(2);

if (\*Snake.game == 2) set\_timer\_game();

}

void CSnakeDlg::OnMy233() //Скорость 3

{

Snake.set\_speed(3);

if (\*Snake.game == 2) set\_timer\_game();

}

void CSnakeDlg::OnMy31() //Информация о игре

{ S\_info.DoModal(); }

void CSnakeDlg::OnMy32() //Информация о программе

{ //Создать объект класса CAboutDlg

CAboutDlg dlg;

//Вывести диалоговую панель About

dlg.DoModal();

}

void CSnakeDlg::Snake\_take\_food() //Проверка - съела ли змея еду

{

CPoint A;

for (int i = Snake.coord[0].x-5; i < Snake.coord[0].x + 5; i++)

{

for (int j = Snake.coord[0].y - 5; j < Snake.coord[0].y + 5; j++)

{

A.x = i; A.y = j;

if (Snake.Food == A)

{

Snake.coord.push\_back(Snake.coord.back());

Snake.Eat\_food();

switch (\*Snake.speed)

{

case 1:

\_point += 10;

break;

case 2:

\_point += 50;

break;

case 3:

\_point += 100;

break;

}

break;

}

}

}

}

void CSnakeDlg::Snake\_take\_snake() //Проверка - съела ли змея себя - конец игры

{

CPoint A;

for (int k = 1; k < Snake.coord.size(); k++)

{

for (int i = Snake.coord[0].x - 5; i < Snake.coord[0].x + 5; i++)

{

for (int j = Snake.coord[0].y - 5; j < Snake.coord[0].y + 5; j++)

{

A.x = i; A.y = j;

if (Snake.coord[0] == Snake.coord[k])

{

Snake.Game\_over();

KillTimer(1);

break;

}

}

}

}

}

void CSnakeDlg::set\_timer\_game() //Устанока таймера движения

{

if (InstallTimer == TRUE)

{

KillTimer(1);

}

if (\*Snake.speed == 1)

{

InstallTimer = SetTimer(1, 100, NULL);

}

else if (\*Snake.speed == 2)

{

InstallTimer = SetTimer(1, 60, NULL);

}

else

{

InstallTimer = SetTimer(1, 20, NULL);

}

}

void CSnakeDlg::OnTimer(UINT\_PTR nIDEvent)

{

switch (\*Snake.move) //передвижение змеи

{

case 0: //движение вверх

{

CPoint B(0, -10);

CPoint A = Snake.coord[0] + B;

if (my\_game\_field.PtInRect(A))

{

for (int i = (int)Snake.coord.size() - 1; i > 0; Snake.coord[i] = Snake.coord[i - 1], i--);

Snake.coord[0] = Snake.coord[1] + B;

}

else

{

B = my\_game\_field.BottomRight();

for (int i = (int)Snake.coord.size() - 1; i > 0; Snake.coord[i] = Snake.coord[i - 1], i--);

Snake.coord[0].y = B.y-1;

}

break;

}

case 1: //движение вправо

{

CPoint B(10, 0);

CPoint A = Snake.coord[0] + B;

if (my\_game\_field.PtInRect(A))

{

for (int i = (int)Snake.coord.size() - 1; i > 0; Snake.coord[i] = Snake.coord[i - 1], i--);

Snake.coord[0] = Snake.coord[1] + B;

}

else

{

B = my\_game\_field.TopLeft();

for (int i = (int)Snake.coord.size() - 1; i > 0; Snake.coord[i] = Snake.coord[i - 1], i--);

Snake.coord[0].x = B.x;

}

break;

}

case 2: //движение вниз

{

CPoint B(0, 10);

CPoint A = Snake.coord[0] + B;

if (my\_game\_field.PtInRect(A))

{

for (int i = (int)Snake.coord.size() - 1; i > 0; Snake.coord[i] = Snake.coord[i - 1], i--);

Snake.coord[0] = Snake.coord[1] + B;

}

else

{

B = my\_game\_field.TopLeft();

for (int i = (int)Snake.coord.size() - 1; i > 0; Snake.coord[i] = Snake.coord[i - 1], i--);

Snake.coord[0].y = B.y;

}

break;

}

case 3: //движение влево

{

CPoint B(-10, 0);

CPoint A = Snake.coord[0] + B;

if (my\_game\_field.PtInRect(A))

{

for (int i = (int)Snake.coord.size() - 1; i > 0; Snake.coord[i] = Snake.coord[i - 1], i--);

Snake.coord[0] = Snake.coord[1] + B;

}

else

{

CPoint B = my\_game\_field.BottomRight();

for (int i = (int)Snake.coord.size() - 1; i > 0; Snake.coord[i] = Snake.coord[i - 1], i--);

Snake.coord[0].x = B.x-1;

}

break;

}

}

Snake\_take\_food(); //проверка еды

Snake\_take\_snake(); //проверка конца игры

Invalidate();

CDialogEx::OnTimer(nIDEvent);

}

void CSnakeDlg::OnDestroy()

{

CDialogEx::OnDestroy();

KillTimer(1);

// TODO: добавьте свой код обработчика сообщений

}

BOOL CSnakeDlg::PreTranslateMessage(MSG\* pMsg) //Вместо OnKeyDown - управление змей

{

if (pMsg->message == WM\_KEYDOWN)

{

switch (pMsg->wParam)

{

case VK\_RIGHT:

if (\*Snake.move != 3)

Snake.Go\_riglt();

break;

case VK\_DOWN:

if (\*Snake.move != 0)

Snake.Go\_bottom();

break;

case VK\_LEFT:

if (\*Snake.move != 1)

Snake.Go\_left();

break;

case VK\_UP:

if (\*Snake.move != 2)

Snake.Go\_up();

}

}

return CDialog::PreTranslateMessage(pMsg);

}

Список литературы

1. Википедия, свободная энциклопедия [Электронный ресурс]: С++., режим доступа: https://ru.wikipedia.org/w/index.php?title=C%2B%2B&stable=0&redirect=no
2. Википедия, свободная энциклопедия [Электронный ресурс]: Microsoft Foundation Classes., режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_Foundation_Classes>
3. Википедия, свободная энциклопедия [Электронный ресурс]: Стандартная библиотека шаблонов., режим доступа: https://ru.wikipedia.org/wiki/Стандартная\_библиотека\_шаблонов
4. Заметки по Delphi [Электронный ресурс]:Платонов Андрей Николаевич, Справочник., Режим доступа: http://platonov-andrei.narod.ru/Index.htm
5. Класс CRect[Электронный ресурс]: Microsoft, Developer Network, Документация по Visual Studio 2017 .,Режим доступа: https://msdn.microsoft.com/ru-ru/library/h58f4c7y.aspx#crect\_\_ptinrect
6. Класс CPoint [Электронный ресурс]: Microsoft, Developer Network, Документация по Visual Studio 2017 .,Режим доступа: https://msdn.microsoft.com/ru-ru/library/t792xy69.aspx#cpoint\_\_operator\_-
7. MFC шаг за шагом [Электронный ресурс]: Каев Артем, MFC шаг за шагом, электронный учебник .,Режим доступа: http://www.firststeps.ru/mfc/steps/mfc1.html
8. Справочник С++[Электронный ресурс]: cppreference.com, Справка по С++ .,Режим доступа: http://ru.cppreference.com/w/cpp