Содержание:

ВВЕДЕНИЕ

Актуальность темы исследования заключается в необходимости исследования специфики различных средств разработки программного обеспечения (ПО), применяемого на практике на рынке информационных технологий. Постоянное повышение уровня спроса на разработку и выпуск готового к эксплуатации ПО стимулирует повышение интереса и рост актуальности исследования данного процесса у обучающихся. Следует отметить, что сфера создания программных продуктов различной направленности является одной из наиболее перспективных и активно развивающихся на отечественном рынке труда.

Объект исследования: языки программирования высокого уровня.

Предмет исследования: особенности практического использования языков программирования высокого уровня.

Цель работы заключается в закреплении, расширении, обобщении и систематизации знаний в рамках изучаемой предметной дисциплины, посредством анализа тематических положений в сфере информационных технологий по языкам программирования высокого уровня.

Задачи исследования:

1. Анализ специфики исторического развития языков программирования высокого уровня.

2. Анализ языков написания программного кода.

3. Анализ особенностей классификации высокоуровневых языков программирования.

4. Разработка программного обеспечения с помощью языка C#.

5. Описание структуры проекта и интерфейса пользователя разработанного программного продукта.

В рамках первой главы приведены результаты проведенного анализа специфики исторического развития средства разработки программ, описаны предпосылки развития современных средств создания вычислительных программ, проведен обзор этапа создания первых языков программирования.

В рамках второй главы данной работы описаны результаты анализа языков разработки программного кода. Описан ряд существующих классификаций высокоуровневых языков программирования, приведено описание существующих типов трансляторов, а именно, интерпретаторов, компиляторов и ассемблеров. Кратко обозначены ключевые отличия между декларативными и императивными языками разработки ПО, описаны логические и функциональные подходы к созданию программного кода, приведен рейтинг популярности средств разработки на 2017-й год. Проведен анализ возможностей современных языков программирования C#, Python и C++, кратко обозначены их основные преимущества, недостатки и особенности.

В рамках данной главы приведено описание основных возможностей использования разработанного программного обеспечения с помощью таких средств разработка, как язык программирования C#, интегрированная среда разработки ПО Visual Studio 2010, фреймворка .NET, технологии Windows Forms. Приведены результаты создания пользовательского интерфейса разработанного ПО, отражены основные компоненты, классы, модули и описаны ключевые функциональные возможности по обработке созданной системой данных, хранимых в созданной БД. Обеспечена возможность авторизации в созданной системе. Для исключения возможности ложных срабатываний операций удаления данных из БД предусмотрена валидационная проверка в виде сообщения о подтверждении действий пользователя.

В процессе написания данной работы использовались литературные источники в количестве 20 книг, служащие информационной основой для решения поставленных задач исследования.

ГЛАВА 1. АНАЛИЗ СПЕЦИФИКИ ИСТОРИЧЕСКОГО РАЗВИТИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ ВЫСОКОГО УРОВНЯ

1.1. Предпосылки развития современных средств программирования

Исторически, процесс эволюции в области вычислительной техники и средств разработки ПО носил неравномерный характер. Это проявлялось в том, что периоды накопления теоретических положений и знаний сменялись технологическими прорывами в разработках.

После этого часто наступал период стабилизации, характеризовавшийся использованием полученных результатов исследований на практике, позволяя накопить опыт, выявить слабые стороны для новых инновационных исследований. После каждого такого витка процесс протекания компьютерной эволюции выходил на более высокую ступень развития [7].

Принято полагать, что начало развития информатики положено В. Шикардом в 1623 году, когда он создал машину, позволяющую осуществлять сложение и вычитание чисел. Однако, первым полноценным арифмометром, стала модель знаменитого француза Б.Паскаля. Основным элементом в нем было зубчатое колесо [13].

Первые технические идеи, которые привели к разработке вычислительных цифровых машин, сформированы еще в начале 20-х годов девятнадцатого века Бэббиджем. А именно в 1823 году Бэббидж начал работать над машиной для вычисления полиномов, причем, планировалась, чтобы эта машина выдавала результаты вычислений на печатать в негативной пластине Таким образом впервые возникла идея практического использования внешнего (периферийного) приспособления для итоговой выдачи результатов проведенных вычислений.

Главной была мысль о возможности предварительной фиксации перечня операций вычислительной машины для дальнейшей реализации вычислений в автоматическом режиме, т.е. в виде программы.

Аналитическая машина, проект которой разработал Бебидж, послужила механическим прототипом первых электронно-вычислительных машин (ЭВМ). В ней закладывался принцип совместной работы пяти основных модулей: арифметического, запоминающего, управляющего, ввода и вывода. Программа записывалась на листка - перфокартах, исходные данные и результаты полученных вычислений также располагались на данном носителе

информации. Ключевой особенностью данной машины является программный принцип работы. Он состоит в том, что программа вычислений размещается в памяти ЭВМ и хранится, а исполняемые команды программы также выражены в числовом коде [19].

Идеи Ч.Бэббиджа развивались и использовались рядом других ученых. В частности, в 1890 году Г. Холлерит разработал машину, которая могла работать с данными, представленными в виде таблиц. Данная разработка также управлялась программой, располагаемой на перфокартах и широко применялась для автоматизации проведения переписи населения в 1890 году. Уже в 1896 году ученый основал компанию, ставшую в последствии родителем корпорации IBM.

Не смотря на то, что использованная Бэббиджем реализация данной мысли на базе перфокарт, которые в свое время были придуманы изобретателем из Франции Мари-Жаккаром, технически давно устарела и имеет мало общего с современными методами обеспечения хранения данных в электронных вычислительных машинах (ЭВМ), концептуальный принцип остался прежним. Можно утверждать, что с данного момента ведет свой отсчет эра программирования [11].

Аду Лавлейс, известную ученую, часто называют первым программистом. Ада разработала концептуальные приемы контроля и управления создания последовательностей вычислительных процессов, используемых в области программировании и в наши дни, а также идентифицировала такую незаменимую конструкцию любого языка программирования как цикл [13].

Принципиальным этапом в развитии языков программирования явилось создание системы кодирования команд ЭВМ посредством использования специальных символов.

Данная схема была предложена сотрудником Пенсильванского университета – Моучли.

В процессе работы на ЭВМ «Марк-1» последовательница Моучли - Хоппер столкнулась с многими техническими проблемами управления данными, однако, эксперименты Хоппер и ее команды способствовали развитию методов решения существующих проблем (например, были разработаны концепции использования подпрограмм).

Также, Хоппер, совместно с коллегами впервые ввела такое понятие осуществления техники программирования как программная отладка.

Необходимость в разработке такого механизма возникла еще в 1945 году, когда произошла аварийная остановка работы ЭВМ «Марк-1».

В процессе диагностики была выявлена неисправность одного из управляющих реле, работа которого была заблокирована бабочкой-мотыльком (bug) [7].

После этого случая стал широко применяться термин «отладка» или debuging для обозначения технического процесса проведения тестирования возможных неисправностей в разработанной программе.

В конце 40-х годов 20 века машинный код являлся, фактически, единственным способом организации взаимодействия человека-пользователя с ЭВМ. Важнейшим достижением разработчиков языков программирования тех времен являлась реализация алгоритмов и механизмов использования ЭВМ для автоматической интерпретации языка программирования в машинный код.

В 1949 году, некий Моучли разработал информационную систему, которую назвал «Short Code». Фактически, данная система представляла собой один из первых высокоуровневых языков программирования. Данная система предоставляла возможности описания задачи в виде перечня различных математических формул, после чего, на базе использования специальной таблицы соответствия, производился перевод символов формул в необходимые двухлитерные коды [11].

После этого специальный программный модуль конвертировал полученные коды непосредственно в машинный двоичный код. Фактически, данная система являлась первым примитивным интерпретатором программного кода.

В середине 1951 года, широко известной в научных кругах Хоппер был разработан первый компилятор, который позволял осуществлять функции по объединению заданных команд и поддерживал организацию подпрограмм в процессе трансляции кода [19].

На базе этих возможностей обеспечивалось выделение отдельных фрагментов памяти ЭВМ и осуществлялось преобразование программных команд (тогда их называли псевдокодом) непосредственно в машинный код. Общая структура ЭВМ приведена на рис.1.

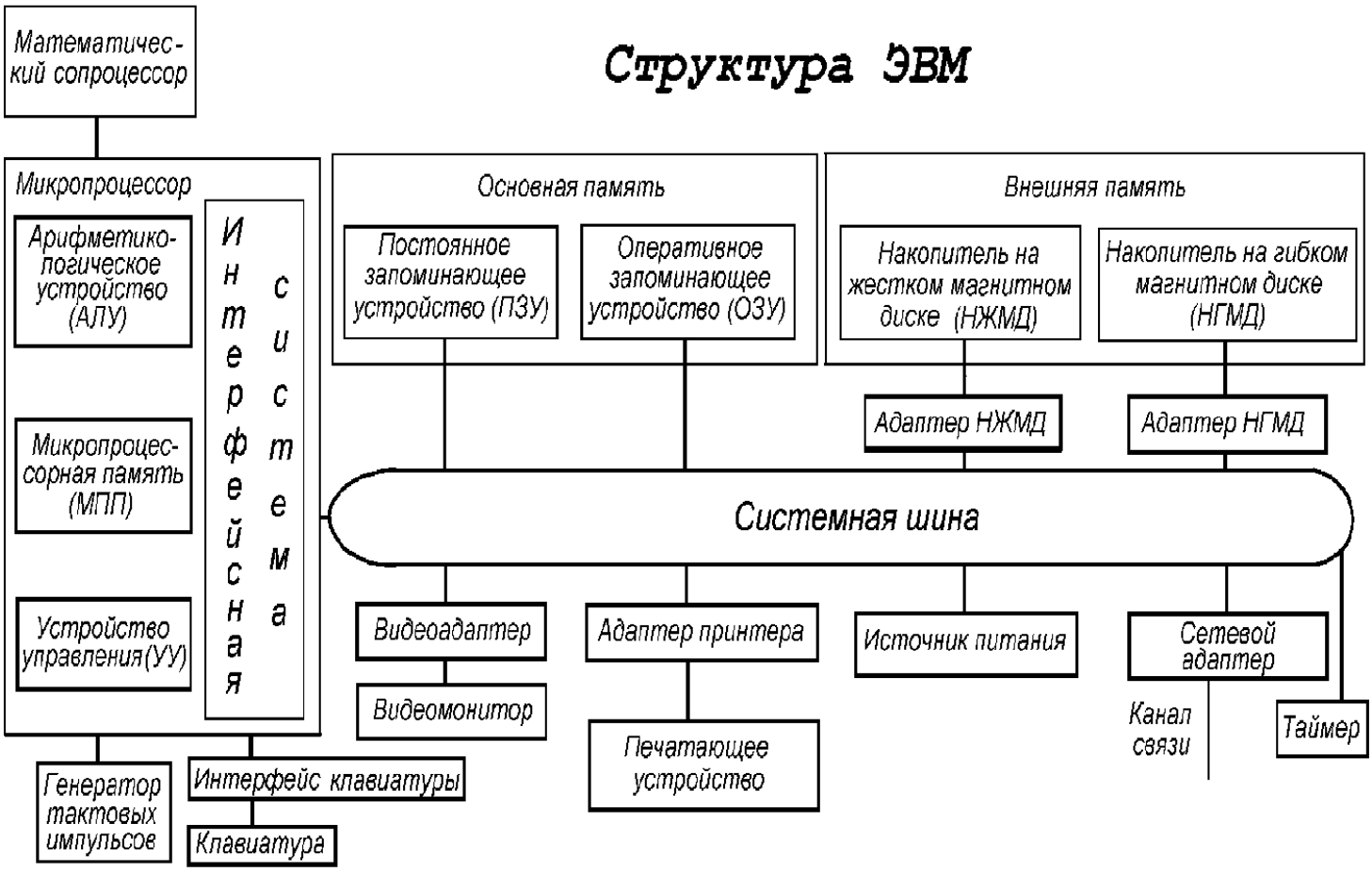

Рисунок 1 – Общая структура ЭВМ

1.2. Обзор этапа создания первых языков программирования

Уже в 1954 году команда Хоппер реализовала систему, которая одновременно интегрировала компилятор и язык программирования, которую назвали MATH-MATIC. Затем была поставлена задача реализовать подобную систему, которая поддерживала бы написание программ на близком к родному, английскому, языку исследователей команды Хоппер. Таким образов в 1958 году появился язык программирования и компилятор, который поддерживал написание программ в таком виде, его назвали FLOW-MATIC.

Данный язык программирования стал первым языком, который получил применение в решении задач обработки различных коммерческих данных. Развитие данного языка привело к разработке популярного в свое время бизнес-

ориентированного зыка COBOL - Common Business Oriented Language. Существенный вклад в разработку данного языка сделала также команда Хоппер [2].

Пример структуры проекта на языке COBOL в консольном окне приведена на рис.2.

```
EDIT       MTH.COBOL.SRCLIB(PERFUNTI) - 01.00            Columns 00001 00072
Command ===>                                              Scroll ===> CSR
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
****** ***************************** Top of Data ******************************
000100         IDENTIFICATION DIVISION.
000200         PROGRAM-ID. PERFUNTI.
000300         ENVIRONMENT DIVISION.
000400         DATA DIVISION.
000500         WORKING-STORAGE SECTION.
000600         01 STD-MARKS                  PIC 9(03).
000700         01 I                          PIC 9(01).
000800         01 TOTAL-MARKS                PIC 9(03).
000900         01 STD-PERCENT                PIC 9(03).9(02).
001000         PROCEDURE DIVISION.
001100             MOVE ZEROES     TO TOTAL-MARKS.
001200             MOVE 1       TO I.
001300             PERFORM UNTIL I > 6
001400                ACCEPT STD-MARKS
001500                ADD STD-MARKS  TO TOTAL-MARKS
001600                COMPUTE I = I + 1
001700             END-PERFORM.
001800             COMPUTE STD-PERCENT = TOTAL-MARKS/6.
001900             DISPLAY 'STUDENT PERCENTAGE : ' STD-PERCENT.
002000             STOP RUN.
****** ************************** Bottom of Data ***************************
```


Рисунок 2 – Пример структуры проекта на языке COBOL

Уже с начала 60-х годов XX века популярность языков программирования начинает стремительно возрастать и общий прогресс в разработке методов и средств программирования с использованием ЭВМ набирается все большие обороты. При этом, вследствие развития технологий постоянно уменьшается использование подхода к написанию программ в машинных командах.

Разрабатываются и выводятся на коммерческий и научный рынок языки программирования более совершенного типа, которые используются в качестве своеобразного посредника между ЭВМ и программистом [1].

Первым и из таких языков стал FORTRAN (FORmula TRANslator, т.е переводчик формул), который был разработан командой программистов из молодой фирмы IBM. Пример кода на данном языке приведен на рис.3. С данного языка наступает эра развития и совершенствования языков программирования высокого уровня.

```
 PROGRAM TPK
! The TPK Algorithm
! Fortran 90 style
  IMPLICIT NONE
  INTEGER                  :: I
  REAL                     :: Y
  REAL, DIMENSION(0:10)    :: A
  READ (*,*) A
  DO I = 10, 0, -1         ! Backwards
    Y = FUN(A(I))
    IF ( Y < 400.0 ) THEN
      WRITE(*,*) I, Y
    ELSE
      WRITE(*,*) I, ' Too large'
    END IF
  END DO
  CONTAINS                  ! Local function
    FUNCTION FUN(T)
      REAL  :: FUN
      REAL, INTENT(IN) :: T
      FUN = SQRT(ABS(T)) + 5.0*T**3
    END FUNCTION FUN
  END PROGRAM TPK
```

Рисунок 3 – Пример кода на FORTRAN

По степени детализации алгоритма языки программирования в настоящее время делятся на [16]:

- Низкоуровневые - языки программирования, близкие к программированию в машинных кодах, на базе использования виртуального или реального вычислительного процессора. При обозначении низкоуровневых команд часто используется мнемонические методы и механизмы. Это делает возможным оперировать командами не в виде последовательностей единиц и нулей, а в форме смысловых сокращений слов, используемых в естественных языках. Примером языка такого типа является ассемблер, представляющий собой целый спектр групп языков, реализованных для разных архитектур, т.к. для одного процессора может существовать несколько видов ассемблера. Они могут быть идентичными в машинных командах, однако, часто различаются макросами и директивами.

- Высокоуровневые (C#, Java)– языки, которые разработаны для обеспечения платформенной независимости создаваемых алгоритмов. В данном случае имеется в вижу, что различные платформенные зависимости перекладываются на программы-трансляторы, которые осуществляют компиляцию текста, который создан на высокоуровневом языке, к виду машинных инструкции (команд). С этим связана необходимость разработки уникальных трансляторов высокоуровневого языка для каждой платформы.
- Сверх высокоуровневые (Алгол-68) — языки, обладающие еще большим уровнем абстракции, чем высокоуровневые языки программирования, в связи с чем их используют для разработки и решения специфических предметно-ориентированных приложений и задач. В таких языках программирования часто реализован синтаксис, не используемый в других языках. В сверх высокоуровневых языках не описывают детали реализации («как делать»), реализуется концепция (принцип «что делать»).

Все языки программирования часто классифицируют по следующим поколениям [6]:

– первое поколение: применение на ЭВМ первого поколения с машинно–ориентированным использованием и ручным процессом управлением памятью;

– второе поколение: применение автокодов или мнемонических символов для представления команд программы;

– третье поколение: языки программирования высокого уровня общего типа и назначения, которые использовались в первую очередь для разработки и реализации пакетов прикладных программ в нужной предметной области (Паскаль, Бейсик, Си);

– четвертое поколение: высокоуровневые языки программирования, использованные для разработки специализированных прикладных программ и приложений, которые поддерживали механизмы использования систем управления базами данных.

– пятое поколение: объектно–ориентированные, декларативные и визуальные высокоуровневые языки программирования. Данные языки применяются в настоящее время и используются для разработки клиент-серверных приложений, мобильных систем и распределенных веб-сайтов. Примерами таких языков являются Си++, Java, C#, Visual Basic, Delphi.

Обобщенная схема классификации программирования [18] приведена на рис.4.

Рисунок 4 – Схема классификации языков программирования

Выводы по главе 1

В данной главе приведены результаты проведенного анализа специфики исторического развития средства разработки программ, описаны предпосылки развития современных средств создания вычислительных программ, проведен обзор этапа создания первых языков программирования. Приведена общая схема классификации языков программирования.

ГЛАВА 2 АНАЛИЗ ЯЗЫКОВ НАПИСАНИЯ ПРОГРАММНОГО КОДА

2.1. Особенности классификации высокоуровневых языков программирования

К высокоуровневым языкам программирования уровня исторически принято относить следующие.

1.Фортран, использовался преимущественно для написания программ, которые выполняли научные и математические расчеты.

2. Алгол – один из первых коммерческих языков программирования, применялся для решения различных экономических задач).

3. Паскаль, Бейсик – использовались для решения несложных задач и математических расчетов, позже стали применяться для обучения азам программирования.

4. Си – язык разработки первых сетевых протоколов и операционных систем, популярных в ряде отраслей и в настоящее время.

5. Пролог – язык, используемый в области искусственного интеллекта, в силу поддержки гибкого аппарата математической логики и предикатов [5].

Согласно специфике использования высокоуровневых языков программирования используется три типа трансляторов [2]:

1. Интерпретатор, представляющий собой транслятор, выполняющий пооператорную обработку и реализацию исходного программного кода.

2. Компилятор – это транслятор, который осуществляет преобразование всей программы в отдельный модуль машинного языка, а затем программа записывается непосредственно в оперативную или постоянную память ЭВМ и выполняется.

3. Ассемблер, предназначенный для корректного перевода разработанной программы, которая записана на языке ассемблера, в соответствующую программу машинного языка.

Языки программирования часто классифицируют на процедурные и декларативные.

В первом типе разработанная программа явно идентифицирует перечень действий, которые должны быть выполнены. Результат работы программы, при этом, определяется лишь способом использования помощи конкретной процедуры, представляющей собой заданную последовательность действий или алгоритм [6].

Среди таких языков выделяют структурные и операционные языки. В структурных языках запись целых алгоритмических структур (ветвление, циклы и др.) выполняются одним оператором. В операционных языках для решения данной задачи применяется уже несколько операций. Исторически, структурными языками являются Паскаль, Ада, Си. Операционными являются Фортран, Фокал, Бейсик.

К декларативным языкам программирования часто относят логические и функциональные языки.

В логических языках программирования код программы не описывает никаких действий. Программа лишь задает данные и взаимосвязи (соотношения) между ними. Затем, поддерживается возможность задания системе различные вопросы. ЭВМ осуществляет перебор доступных и заданных в программе данные, после чего находит нужный ответ и выводит его. Таким образом, непосредственный порядок перебора команд не описывается явно в программе, а задается синтаксисом самого языка. Классическим языком такого типа программирования является Пролог [12].

В функциональных языках написанный код описывает вычисление конкретной функции. Такая функция, как правило, задается в виде некоторой композиции других, являющихся более простыми, функций. Такие функции также могут подразделяться на более простые и т.д. Одной из главных особенностей элементов функциональных языков являются рекурсии. Их наличие обусловлено тем, что в функциональных языках программирования отсутствуют операторы присваивания и циклов.

Наиболее популярным в настоящее время классом языков программирования высокого уровня является объектно–ориентированные языки. Данные языки чаще всего не используются для описания детальной последовательности действий выполнения и решения задачи, однако, они включают соответствующие механизмы и содержат некоторые элементы процедурного программирования.

В настоящее время наиболее востребованными языками программирования высокого уровня являются Java, C#, Python. Данные высокоуровневые языки программирования являлись машинно–независимыми, в силу их ориентации на систему операндов, которые являются характерными для формализации записей имплементированных алгоритмов [20]. Статистика по состоянию на январь 2017 года приведена на рис.5.

Январь 2017	Январь 2016	Изменение	Язык программирования	Рейтинг	Изменение %
1	1		Java	17.278%	-4.19%
2	2		C	9.349%	-6.69%
3	3		C++	6.301%	-0.61%
4	4		C#	4.039%	-0.67%
5	5		Python	3.465%	-0.39%
6	7	^	Visual Basic .NET	2.960%	+0.38%
7	8	^	JavaScript	2.850%	+0.29%
8	11	^	Perl	2.750%	+0.91%
9	9		Assembly language	2.701%	+0.61%
10	6	⌄⌄	PHP	2.564%	-0.14%
11	12	^	Delphi/Object Pascal	2.561%	+0.78%
12	10	⌄	Ruby	2.546%	+0.50%
13	54	^^	Go	2.325%	+2.16%
14	14		Swift	1.932%	+0.57%
15	13	⌄	Visual Basic	1.912%	+0.23%
16	19	^	R	1.787%	+0.73%
17	26	^^	Dart	1.720%	+0.95%
18	18		Objective-C	1.617%	+0.54%
19	15	⌄⌄	MATLAB	1.578%	+0.35%
20	20		PL/SQL	1.539%	+0.52%

Рисунок 5 – Статистика популярности языков программирования высокого уровня

2.2. Специфические черты и особенности ряда языков программирования

2.2.1. Язык программирования C#

Язык C# базировался на C и изначально создавался с целью обеспечения компонентного программирования, поэтому в его ядро закладывались возможности повторной инициализации и интеграции разработанных программистов программных компонентов. Особенностями данного языка являются :

- язык создавался параллельно с технологией .Net, что позволило разработчикам интегрировать все необходимые механизмы обеспечение функциональных взаимосвязей фреймворка, в том числе FCL и CLR;
- это полноценный объектно-ориентированным язык, причем даже примитивные типы данных языка представлены в качестве отдельных классов [4];

- поддержка механизмов наследования, инкапсуляции и полиморфизма;
- разработан на базе использования C и C ++, что позволило интегрировать наиболее функциональные возможности этих высокоуровневых языков программирования;
- с помощью поддержки ряда фреймворков, выступающих в виде некой надстройки над операционной системой пользователя, разработчики C# могут использовать механизмы создания и работы с виртуальной машиной, аналогично существующим технологиям языка Java [10].

При этом существенно повышается эффективность и используемость программного кода.

Это связано с тем, что исполнительная среда CLR позволяет обеспечить работу компилятора промежуточного уровня, что является более эффективным по сравнению с интерпретатором байт-кода в Java Virtual Machine.

В настоящее время весьма популярен среди разработчиков программного обеспечения (ПО) набор продуктов компании Microsoft, включающих, в частности, интегрированную среду разработки (IDE) программ - Microsoft Visual Studio [2].

В настоящее время актуальной версией является Microsoft Visual Studio 2017.

Предлагаемые средства разработки ПО продукты позволяют оперативно и гибко разрабатывать различные типы приложений, в частности, согласно [10]:

- консольные приложения;
- приложения с графическим интерфейсом, на базе использования популярной среди разработчиков десктопных решений технологии Windows Forms;
- веб-сайты, на базе использования ASP.net.

Состав проекта на языке C# в рамках фрейморка .NET приведена на рис. 6 [4].

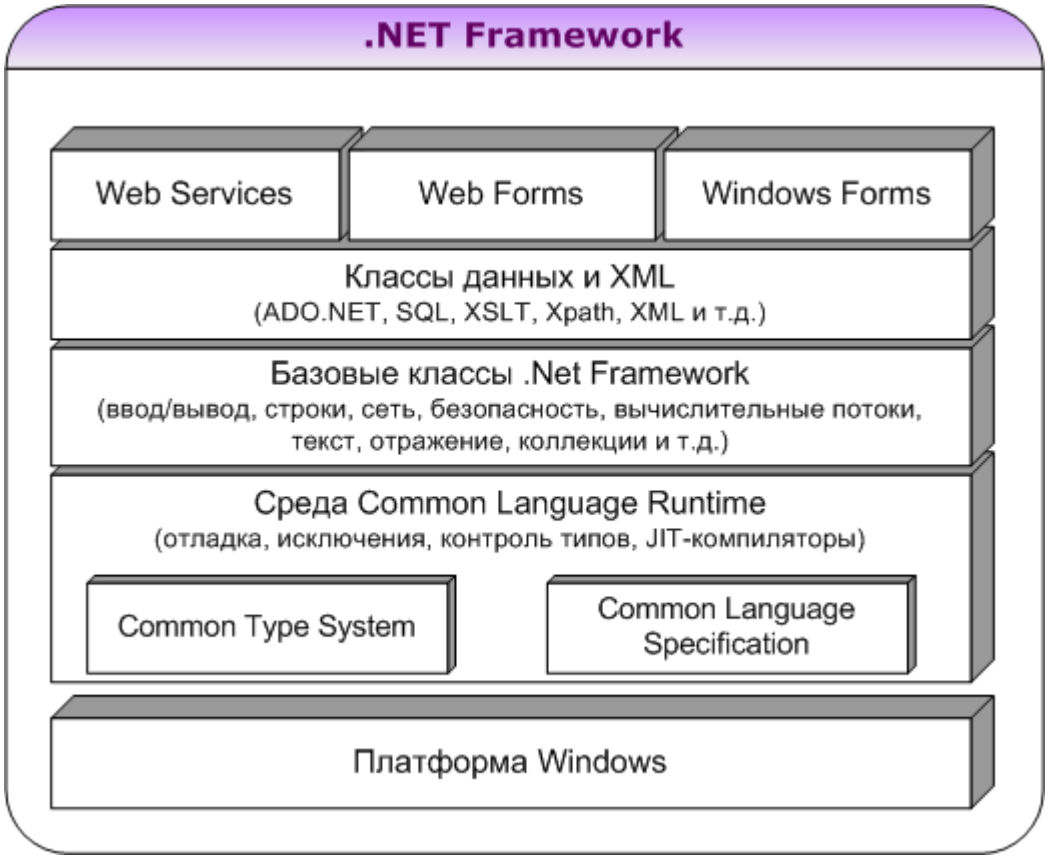

Рисунок 6 - Структура проекта на языке C# в Visual Studio

IDE Visual Studio включает в себя гибкий и современный редактор исходного программного кода, интегрируя поддержку технологии IntelliSense, а также средства оперативного профилирования и рефакторинга кода. Имеющийся в среде разработки отладчик способен функционировать в качестве отладчика на уровне исходного программного кода, а также имеются возможности его использования в качестве отладчика на машинном уровне. К другим встроенным средствам и инструментам среды следует отнести интуитивно понятный редактор форм, который способствует ускорению процесса создания и конфигурирования компонентов графического интерфейса программного приложения, дизайнеры классов, объектов и схем баз данных [14].

IDE Visual Studio, также, позволяет разрабатывать и интегрировать в проект сторонние плагины и функциональные расширения, которые обеспечивают новые возможности разработки приложений на различных уровнях. В частности, широко используются плагины добавления функций использования современных систем контроля версий (Subversion, Git), интеграции новых пакетов инструментов для визуального редактирования проектирования кода на UML-языке, создания диаграмм сценариев использования, разработки алгоритмов.

2.2.2. Язык программирования Python

Python это современный объектно-ориентированный язык с поддержкой динамической типизации, автоматического процесса управления памятью, высокоуровневых гибких структур данных (словари, кортежи, списки). Python поддерживает создание классов, связи модулей, гибкую и удобную обработку исключительных ситуаций и многопоточные методы вычислений. Кроме ООП данный язык структурное, функциональное и аспектно-ориентированное программирование [15].

Все объекты в Python подразделяются на атомарные и ссылочные. К первым относятся int, long, complex. При присвоении подобного рода объектов происходит копирование их значений, а в ссылочных объектах осуществляется копирование лишь указателя на объект, поэтому обе переменные после выполнения операции присваивания используют одинаковое значение.

Python позволяет определять тип переменной на этапе исполнения программы. В связи с этим вместо присваивания переменной определенного значения более корректным является использование фразы «связывания определенного значения с конкретным именем» [8]. Структура проекта Python в среде разработки PyCharm приведена на рис.7.

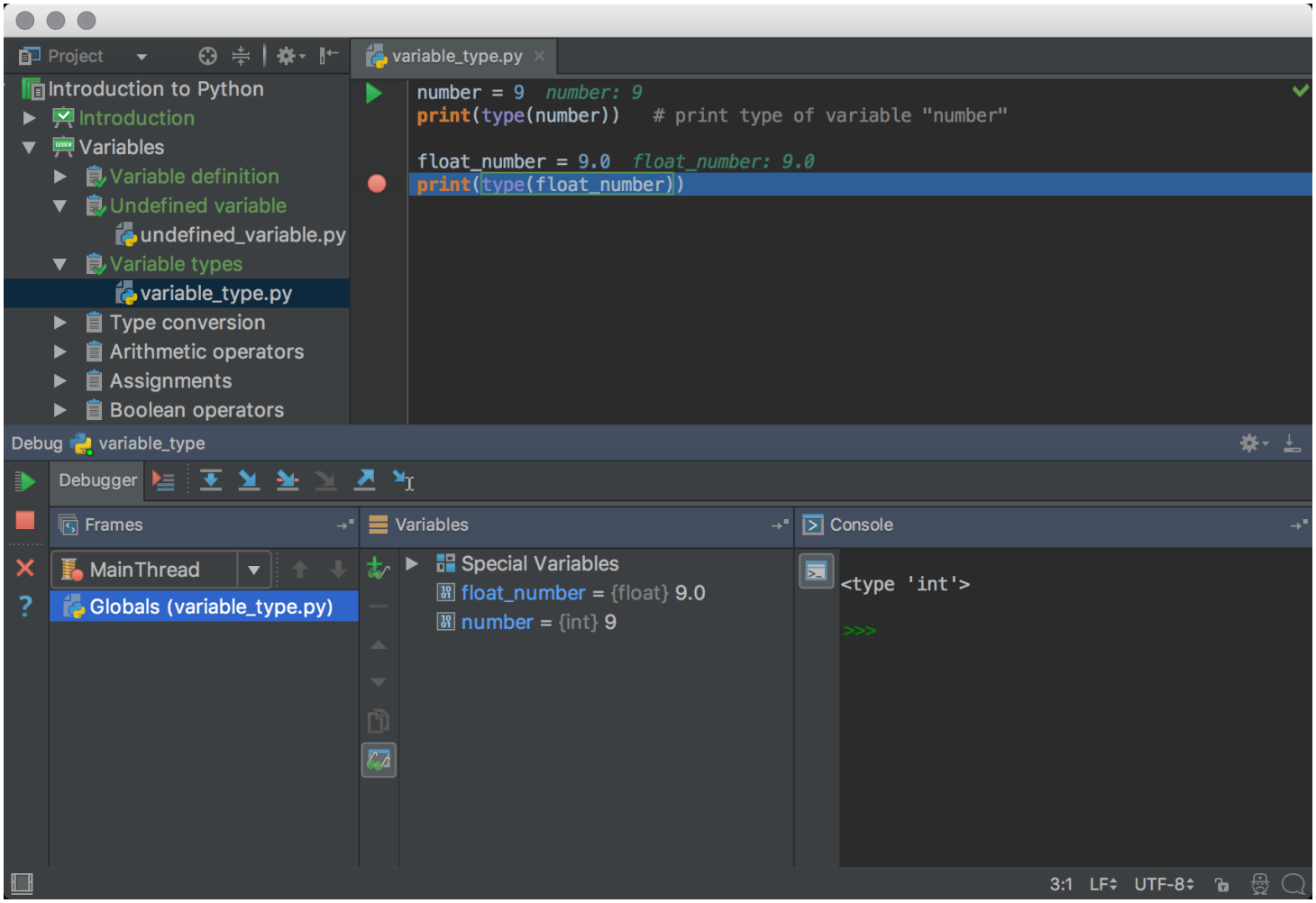

Рисунок 7 – Пример структуры языка программирования Python [19]

В языке поддерживаются такие встроенные типы данных как: бинарный, строковый, Unicode, целочисленный с произвольно заданной степень точности, число с плавающей запятой, и ряд других. Из современных коллекций язык обладает поддержкой таких встроенных структур как: список, словарь, кортеж (модифицируемый список), множество и др. Следует отметить, что все значения, также как и в Ruby, являются объектами, причем это характерно и для функций, модулей, методов и классов [15].

Добавить в программный код новый тип можно путем написания нового класса или определения нового типа в специальном модуле расширения, который может быть разработан на другом языке. Система классов поддерживает полноценные механизмы множественного и единичного наследования, а также функции метапрограммирования. Возможным, также является прямое наследование от подавляющего большинства встроенных типов расширений.

Строки и кортежи, относящиеся к встроенным коллекциям, являются не изменяемыми, а словари и списки – переменными, их можно модифицировать динамически. Поэтому кортежи в данном языке работают существенно быстрее чем списки.

Преимущества языка следующие.

1. Интерпретируемость и динамическая типизация. Это повышает общую скорость разработки кода и позволяет существенно упростить и облегчить отладку программ, однако при этом теряется скорость выполнения приложения.

2. Поддержка модульности и Unicode. Это позволяет разработчику использование в программах символов родного языка, а не только английского. Разработанный модуль может быть легко интегрирован в код произвольных приложений [17].

3. Поддержка работы с объектами в рамках концепции ООП и автоматическая «сборка мусора» (очистка памяти) . Планирование структуры приложения становиться проще и удобнее и разработчику нет необходимости в слежении за различными утечками памяти.

4. Интеграция с многими другими языками программирования для повышения скорости работы программных приложений. Также простота и гибкость синтаксиса языка позволяет разработчику лучше писать и читать созданный программный код. Это помогает сэкономить время разбора в коде одного разработчика другому.

5. Кроссплатформенность и поддержка большого количества модулей, интегрированных и сторонних. В ряде распространенных случаев для разработки программы достаточно просто найти готовые модули и должным образом их связать между собой. Это позволяет запускать программное приложения, написанное на языке Python, на практически любой операционной системе, где установлен интерпретатор языка нужной версии и сборки. Поэтому создатель может сэкономить время и сосредоточить внимание на более высоком уровне разработки и проектирования, работая с готовыми элементами [8].

2.2.3. Язык программирования C++

C++ в настоящее время считается одним из наиболее сложных и функциональных языков, которые применяются при разработке программных приложений в сфере коммерции. В последние годы степень востребованности языка несколько

изменилась, по причине того, что стал активно развиваться язык Java, поэтому некоторые программисты перешли с C++ на Java. Эти языки имеют много общего, поэтому зная один из них досконально для разработчика не составит большого друга оперативно перейти на другой язык.

Изначально язык C++ чаще всего применялся для программных задач в открытой операционной системе семейства UNIX, совместно с системой и средствами языка C. Впоследствии, распространение языка и под другие платформы стало возможно, благодаря реализации в механизмах языка таким функциональных преимуществ, как динамическая загрузка, развитая система проведения трансляции, использование базы данных [3].

В целом данный язык является программным средством широкого спектра действий, начиная от системного программирования и заканчивая разработкой распределенных и масштабируемых систем и приложений.

В языке фактически отсутствуют элементарные операций и типы данных. В частности нет типа матрица с логической операцией обращения или элемента «строка» с поддержкой операций конкатенации. В случае, когда программисту необходимо использовать такие типы данных он способен самостоятельно инициализировать их в коде. Таким образом, процесс написания кода на C++ представляет собой процесс инициализации и описания логики взаимодействия нужных типов данных и операций. Следует отметить, что грамотно реализованный пользовательский тип данных отличен от встроенного лишь способом определения, а способ использования может остаться прежним [6].

При разработке программного обеспечения на базе использования языка часто применяют интегрированную среду разработки Borland C++ Builder (рис.8).

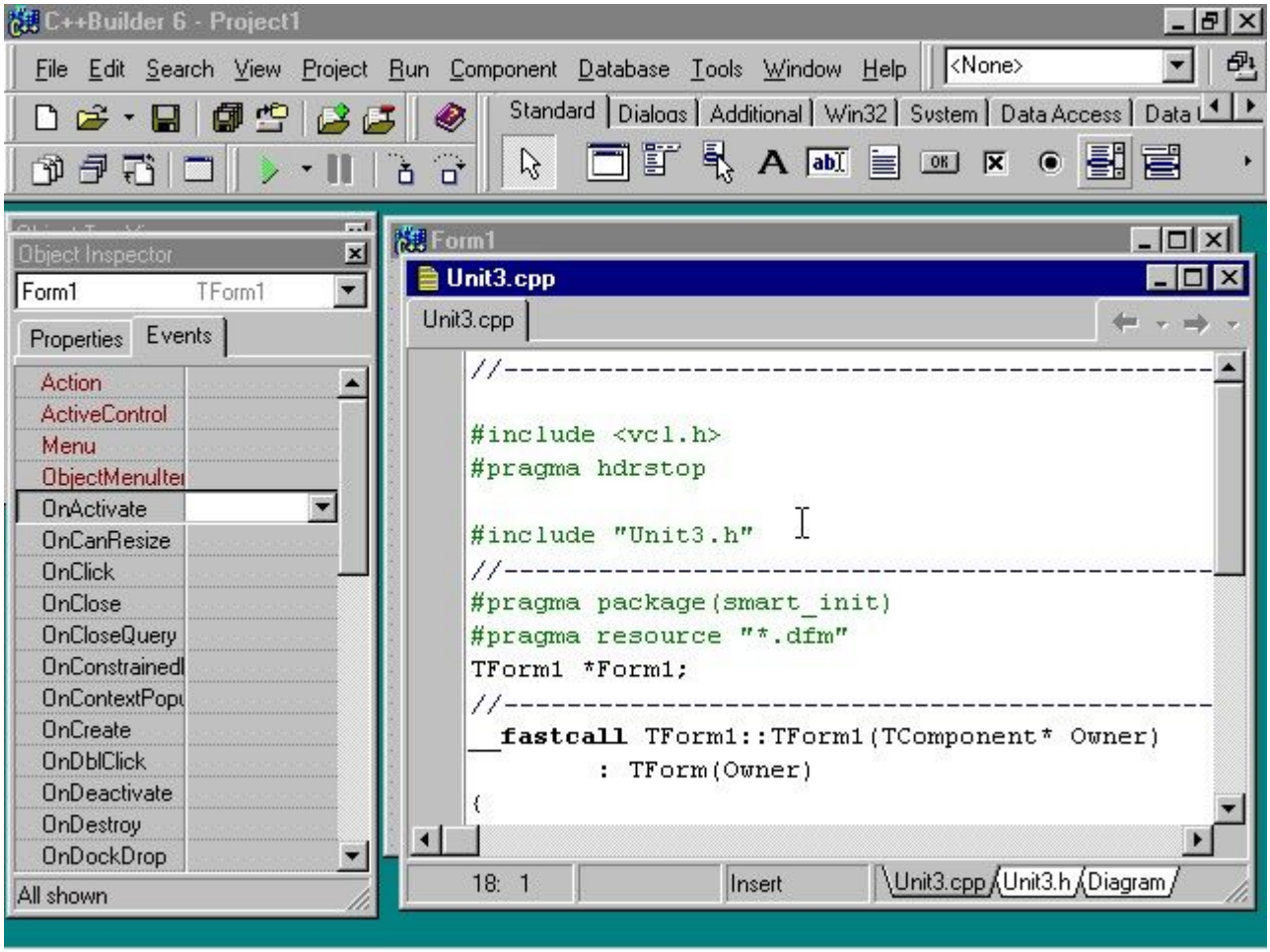

Рисунок 8 – Пример интерфейса среды Borland C++ Builder

С целью оптимизации времени на компиляцию и исполнение написанного программного кода в языке C++ убраны возможности, потенциально приводящие к различным не оптимальным расходам памяти, в том числе и когда описанные операции не будут использованы в программе. Для этих целей разработчиками языка было принято решение не хранить объектах специальную служебную информацию. Т.е., в случае, когда программист инициализировал структуру, которая содержит две величины, каждая из которых занимает по 16 разрядов, то данная структура уместиться в рамках 32-х разрядного регистра [9].

Типы данных в языке C++ включают в себя возможности гибкого синтаксического анализа, который обеспечивается путем интеграции средств транслятора с целью идентификации возможной порчи используемых данных. Данные средства могут использоваться программистом в полной мере, благодаря поддержке оперативной скорости выполнения разработанной программы [3].

Язык в полной мере C++ поддерживает основные принципы объектно-ориентированной парадигмы разработки программного кода, в том числе наследование, инкапсуляцию и полиморфизм.

Выводы по главе 2

В рамках второй главы данной работы описаны результаты анализа языков разработки программного кода. Описан ряд существующих классификаций высокоуровневых языков программирования, приведено описание существующих типов трансляторов, а именно, интерпретаторов, компиляторов и ассемблеров. Кратко обозначены ключевые отличия между декларативными и императивными языками разработки ПО, описаны логические и функциональные подходы к созданию программного кода, приведен рейтинг популярности средств разработки на 2017-й год. Проведен анализ возможностей современных языков программирования C#, Python и C++, кратко обозначены их основные преимущества, недостатки и особенности.

ГЛАВА 3 СОЗДАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ПОМОЩЬЮ ЯЗЫКА ПРОГРАММИРОВАНИЯ C#

3.1. Разработка и описание структуры проекта создаваемого ПО

Для программной реализации разрабатываемой системы необходимо определить структуру и логику работы системы.

Она состоит из формы авторизации, где пользователь, в зависимости от введенного логина и пароля входит в соответствующую учетную запись, которая открывает доступ к форме управления данными по информационному обеспечению, программно-техническому обеспечению, организационному обеспечению, финансовой документации и кадровой документации.

Разработанная схема основных форм и компонентов системы приведена на рис.9.

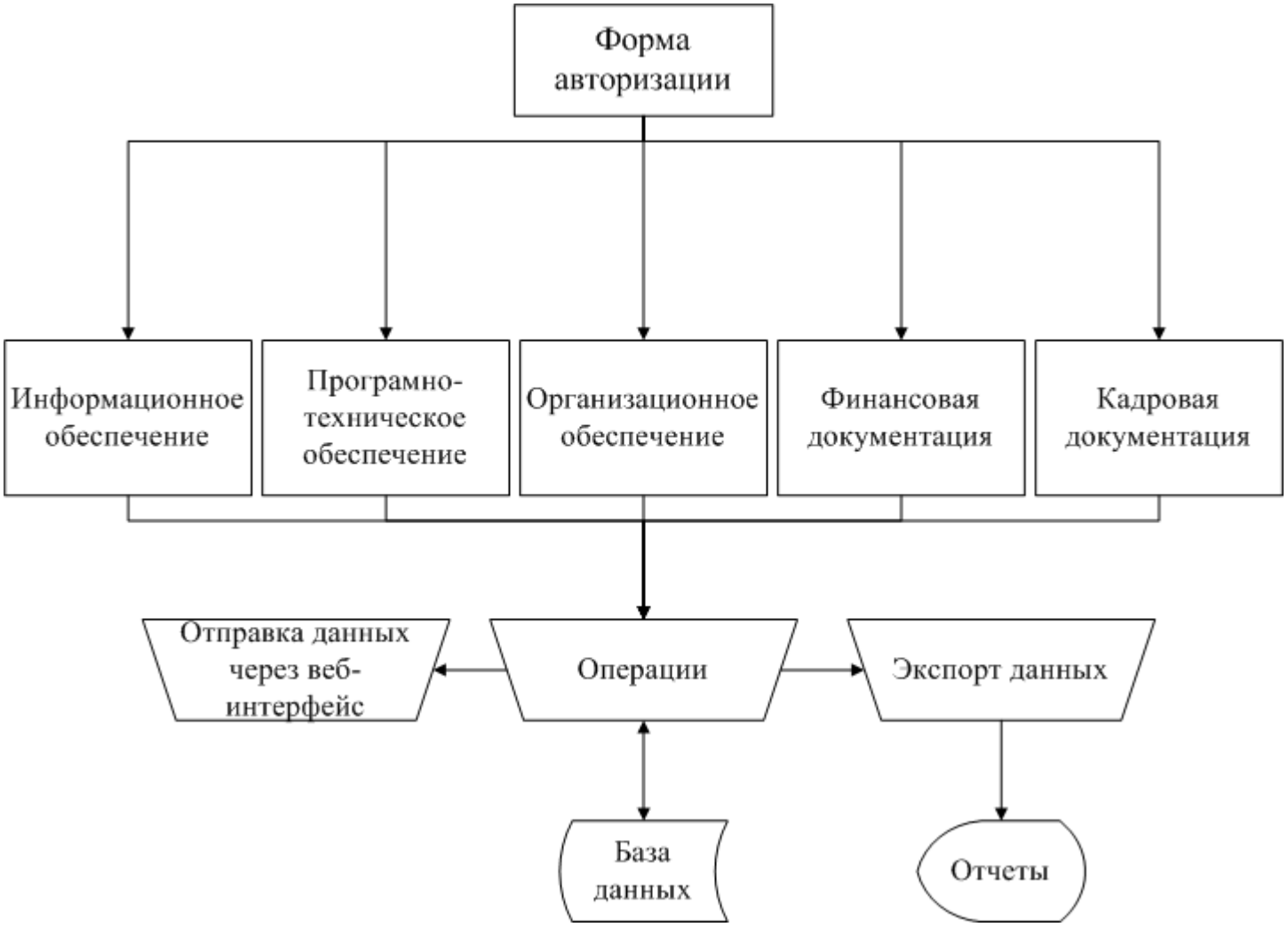

Рисунок 9 – Схема основных форм и компонентов системы

На каждой из этих форм поддерживаются возможности обработки данных в соответствующих таблицах базы данных (БД), экспорта данных в отдельных отчетов и отправки данных через веб-интерфейс.

Проект системы в среде разработки MS Visual Studio приведен на рис.10.

Разработанный проект системы использует следующие программные компоненты интеграции работы с разработанным хранилищем данных (БД) в своем составе DataSet, BindingTable, BindingNavigator, BindingSource, TableAdapter. Они используются в совокупности для обеспечения возможностей построения автоматических связей БД с пользовательским интерфейсом.

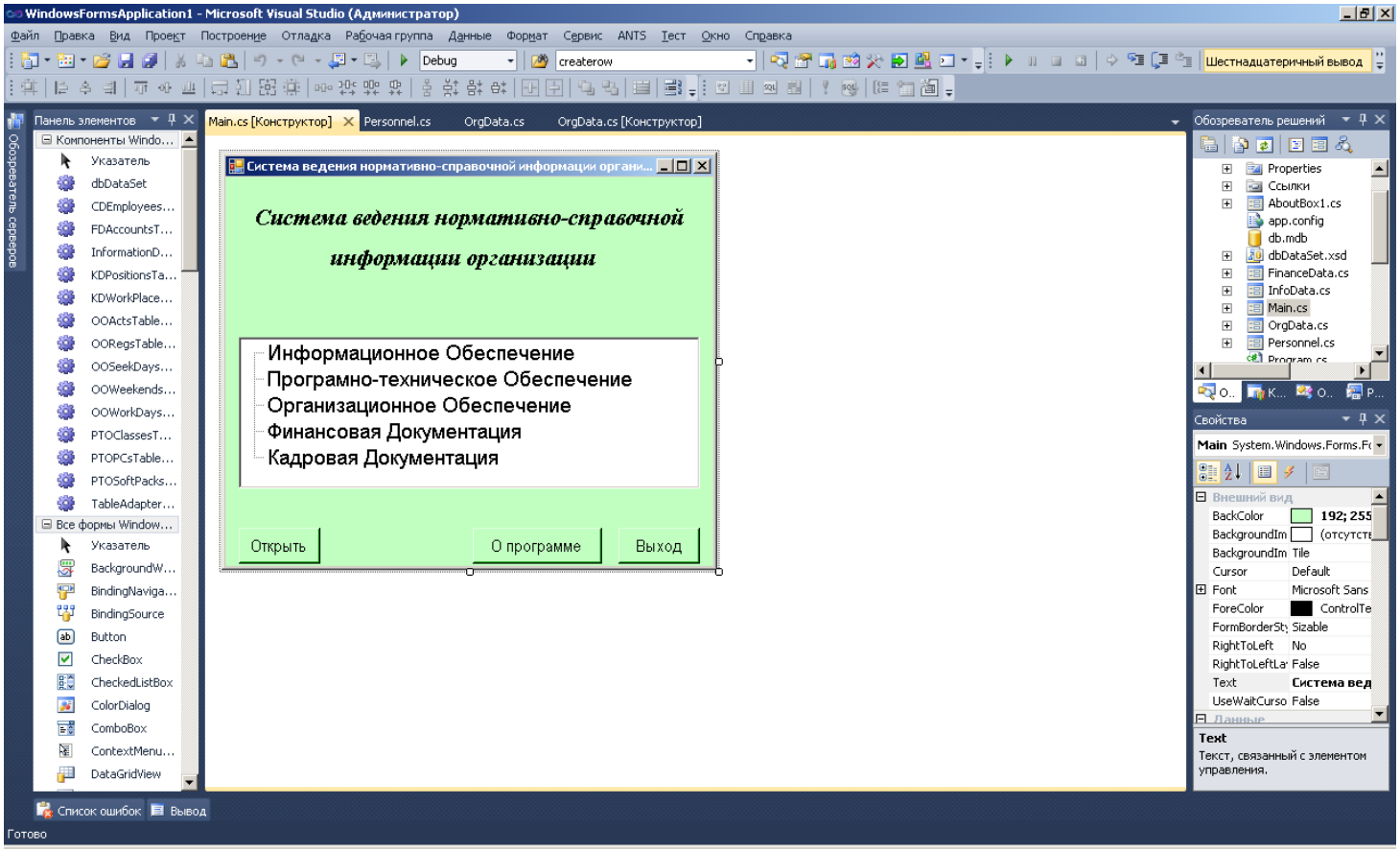

Рисунок 10 – Проект системы в среде разработки MS Visual Studio

Структура разработанных классов обработки данных в БД приведена на рис.11.

Каждый из классов начинается с названия соответствующей таблицы базы данных для организации выгрузки наборов данных в соответствующий компонент формы проекта.

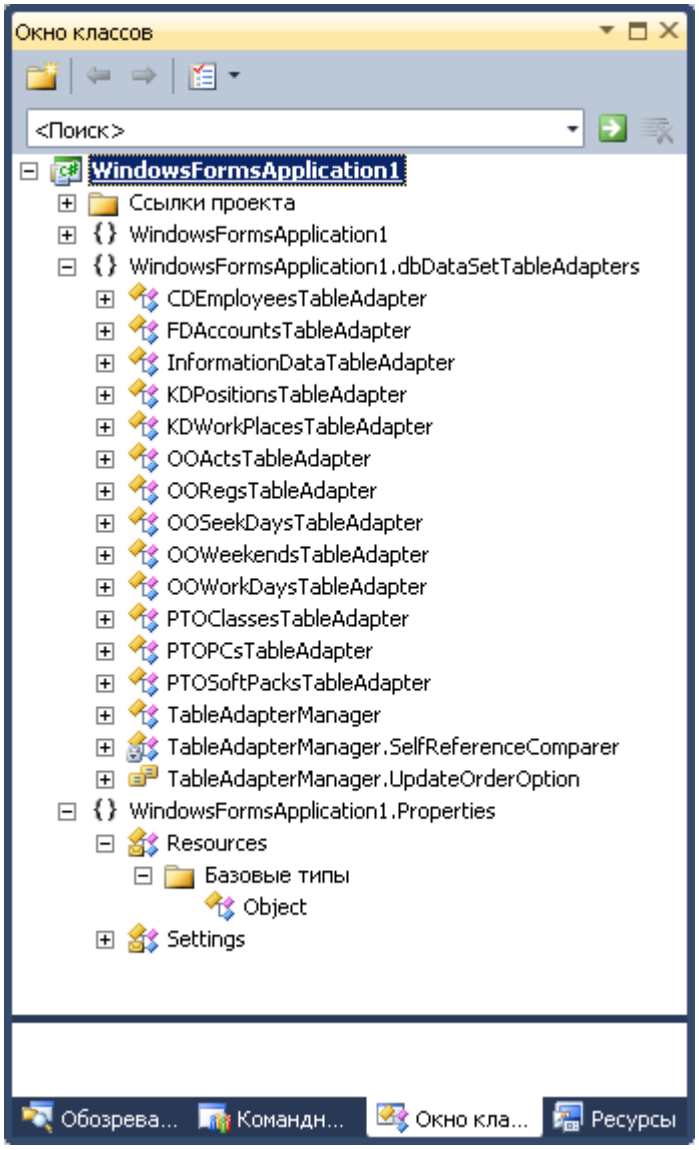

Рисунок 11 – Структура разработанных классов обработки данных в БД

Для обеспечения функциональных возможностей работы программы были созданы следующие формы:

1. Main - главная форма проекта, которая доступна из административного режима пользователя.

2. FinanceData - форма редактирования финансовых данных по рабочим организации, доступная под записью менеджера или в административном режиме.

3. InfoData - форма редактирования информационных данных по регламенту организации, доступная под записью менеджера или в административном режиме.

4. OrgData - форма редактирования организационных данных компании, доступная для использования под записью менеджера или в административном режиме.

5. Personnel - форма редактирования данных по сотрудникам организации, доступная для использования под записью менеджера или в административном режиме.

6. ReadDoc - форма организации внешнего и внутреннего просмотра информационных или нормативных данных.

7. TechData - форма редактирования данных по программно-техническому составу организации, доступная для использования под записью менеджера или в административном режиме.

8. WebPage - форма организации возможностей коммуникации и отправки данных через веб-интерфейс.

Общий состав созданных форм решения WindowsFormsApplication1 разработанного проекта системы приведен в обозревателю решений MS Visual Studio на рис.12.

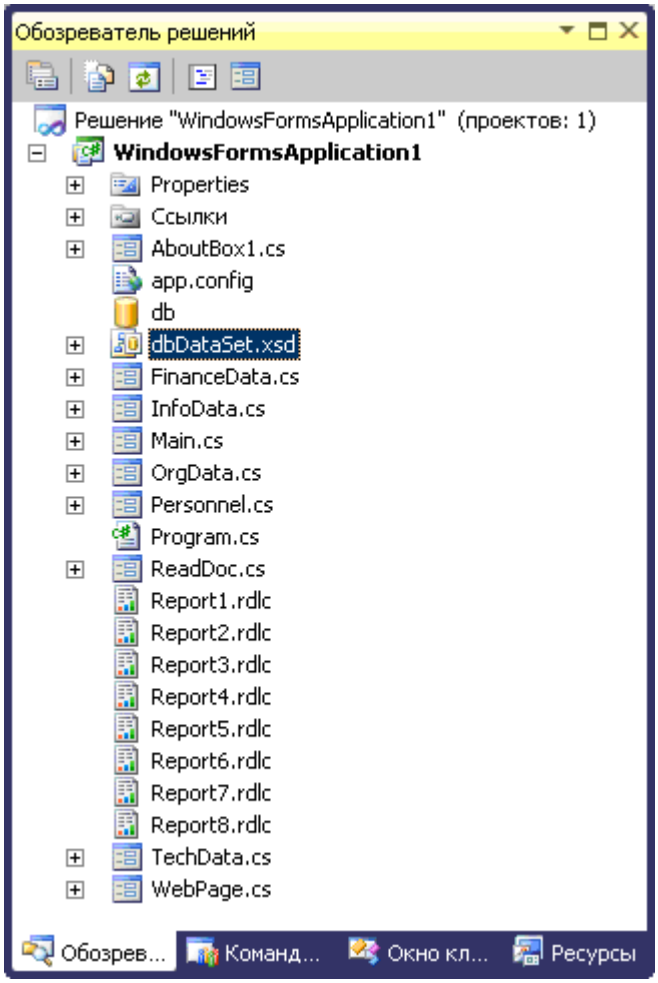

Рисунок 12 - Содержание форм проекта в обозревателе решений

Также, в состав решения WindowsFormsApplication1 входят сгенерированные файлы отчетов в формате rdlc.

3.2. Описание внешнего вида форм и возможностей разработанного ПО

Внешний вид созданной формы авторизации пользователя в системе приведен на рис.13. Для авторизации в системе предусмотрен режим администратора и режимы менеджеров из соответствующих подразделений (кадрового, программно-технического, финансового, организационного и информационного).

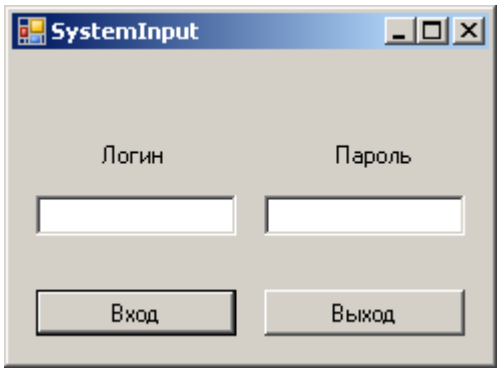

Рисунок 13 - Внешний вид созданной формы авторизации пользователя в системе

Информационное сообщение о некорректном вход в систему приведено на рис.14. Оно появляется в случае введения некорректных данных с авторизации (логина или пароля).

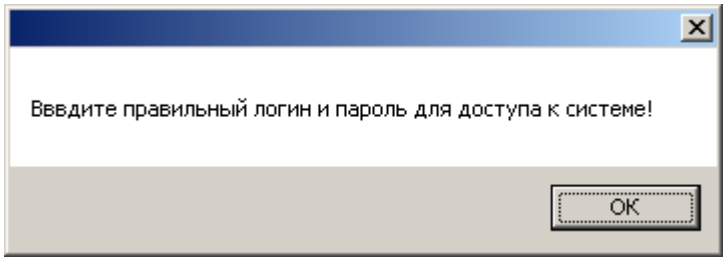

Рисунок 14 - Информационное сообщение о некорректном вход

Внешний вид созданной главной формы системы под записью администратора приведен на рис.15. Данная форма позволяет перейти к соответствующим форм

управления данными БД (выбор соответствующей записи с помощью компонента TreeView), осуществить просмотр информации о разработчике программного продукта (нажатие кнопки «О программе») или завершить работу программы (нажатие на кнопку «Выход»).

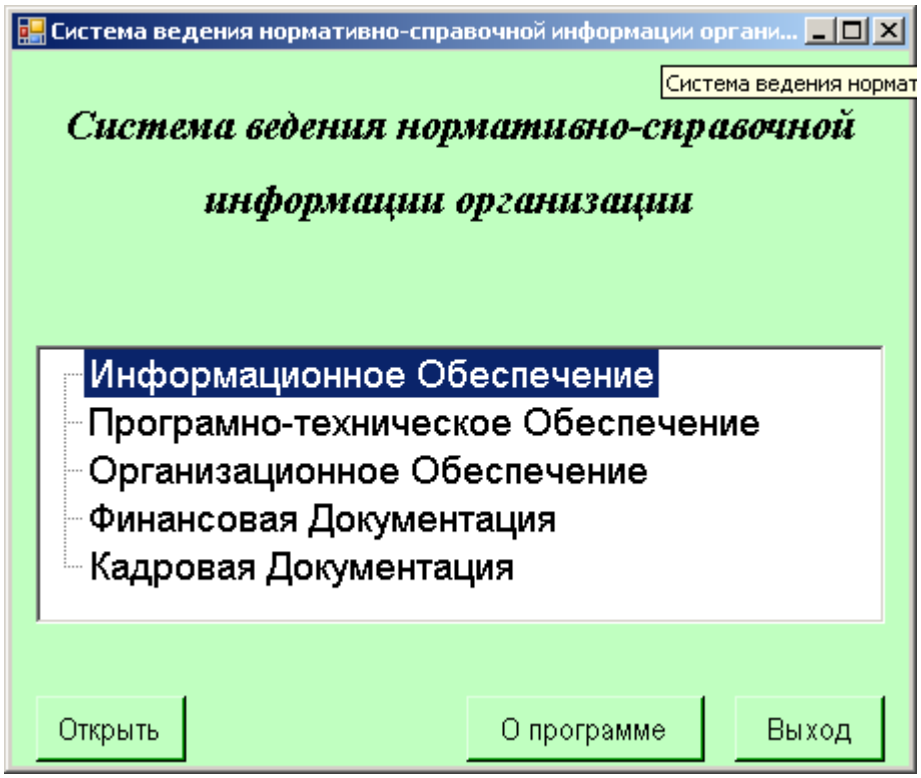

Рисунок 15 - Внешний вид созданной главной формы системы под записью администратора

Внешний вид созданной формы по управлению данными по стандартам во вкладке компонента TabContol «Редактирование» приведен на рис.16. Для обеспечения необходимых возможностей по управлению данными в таблице БД предусмотрено компоненты:

1. TextBox - для связи соответствующих полей БД с данной формой пользователя системы.

2. Button - для обеспечения возможностей добавления данных, перехода к вкладке просмотра данных по стандартам, перехода к форме поиска данных, формирования отчета и закрытие данной формы.

3. Label - для текстовых подписей соответствующих компонентов интерфейса.

4. ProgressBar - для отражения процесса добавления данных в таблицу БД.

5. GroupBox - для структурного объединения компонентов формы.

6. BindingNavigator - для перехода по записям в БД.

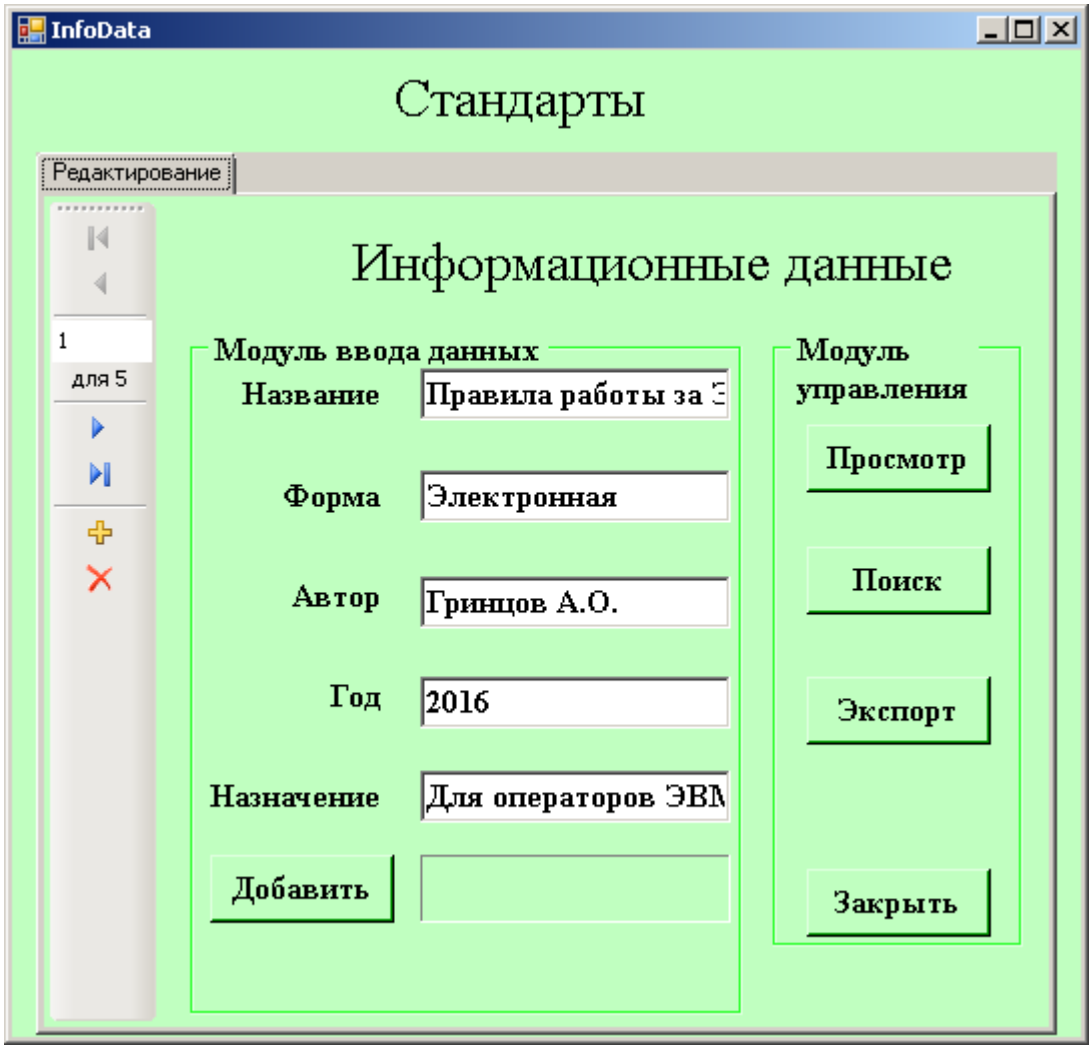

Рисунок 16 - Внешний вид созданной формы по управлению данными по стандартам (вкладка редактирования)

Внешний вид созданной формы по управлению данными по стандартам во вкладке компонента TabContol «Просмотр» приведен на рис.17.

Для обеспечения необходимых возможностей по управлению данными в таблице БД предусмотрено компоненты:

1. TextBox - проведение поиска данных по таблице БД;
2. Button - для инициализации параметры данных, возврат к вкладке редактирования данных, удаление избранное записку в таблице;
3. Label - для текстовых подписей соответствующих компонентов интерфейса;

4. ProgressBar - для отображения процесса удаления данных из таблицы БД;
5. ComboBox - для возможности загон критерия (названия поля таблицы БД), по которому проводит поиск;
6. DataGridView - для структурного просмотра записей БД в интерфейсе пользователя.

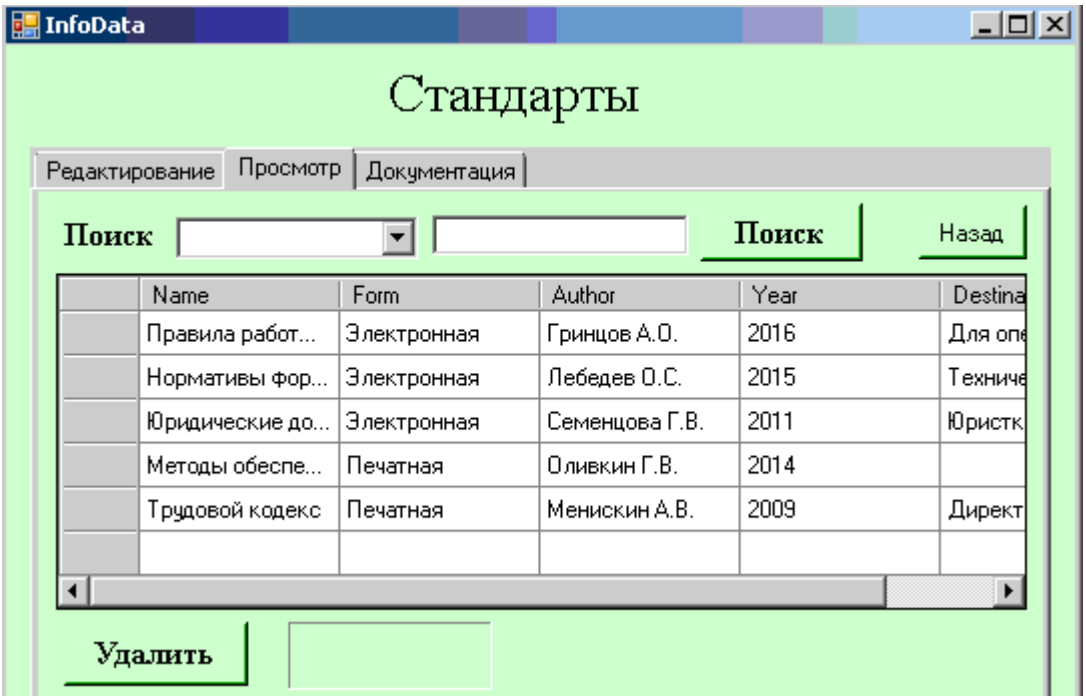

Рисунок 17 - Внешний вид созданной формы по управлению данными по стандартам (вкладка просмотр)

Пример работы функции поиска по таблице по критерию «ФИО Автора» приведен на рис.18. Поиск осуществляется по ячейкам соответствующего столбца DataGridView, результирующий строку выделяется цветом для более удобной ориентации пользователя.

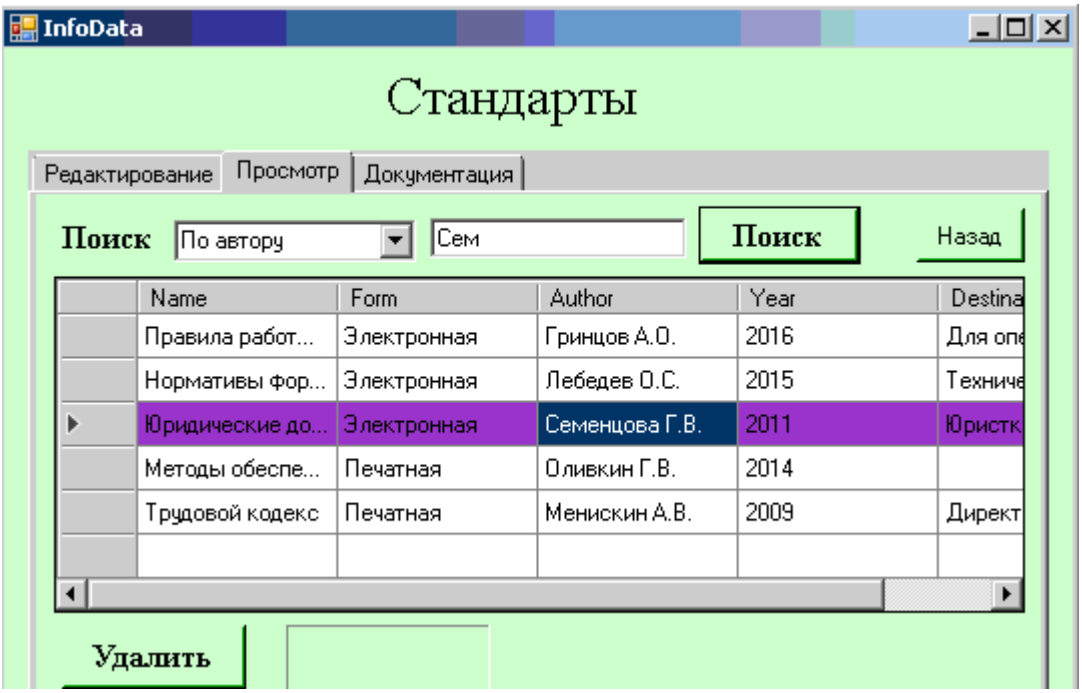

Рисунок 18 - Пример работы функции поиска по таблице по критерию «ФИО Автора»

При выполнении функции удаления данных с выбранной строки DataGridView генерируется соответствующее информационное сообщение (рис.19).

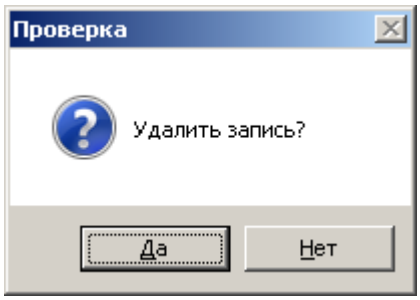

Рисунок 19 - Информационное сообщение о проверке решения по удалению данных в таблице БД

В случае подтверждения пользователем операции удаления данных благодаря нажатию кнопки «Да» в информационном сообщении о проверке решения по удалению данных в таблице БД запись удаляется, о чем свидетельствует надпись в Label, что находится рядом с компонентом ProgressBar.

Пример результата удаления записи из таблицы базы данных приведен на рис.20.

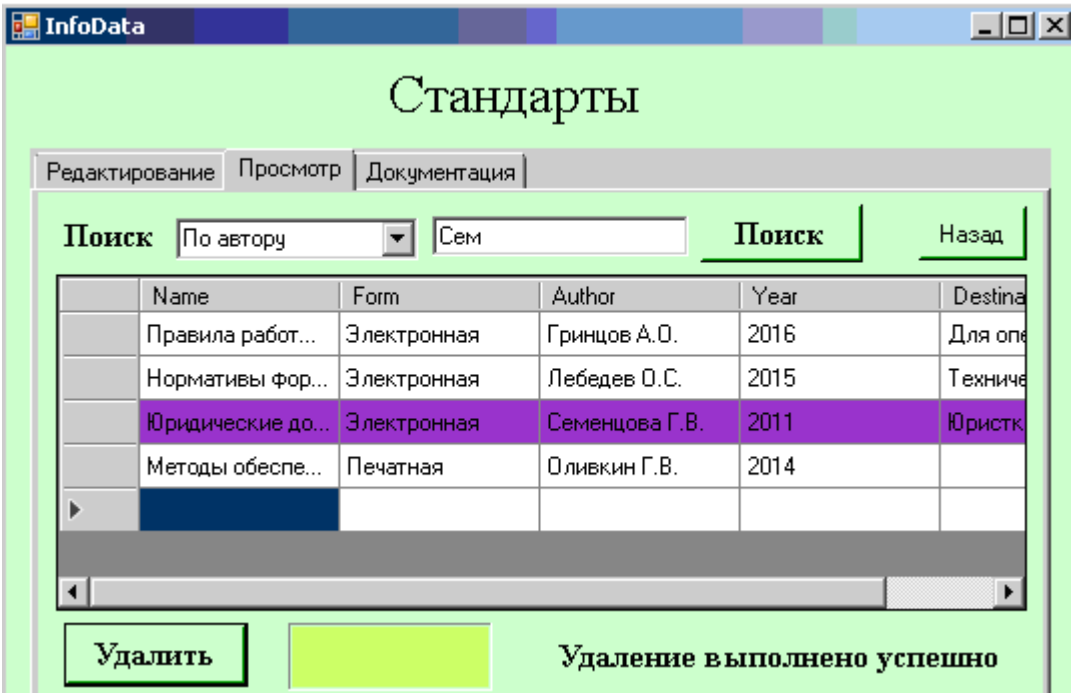

Рисунок 20 - Пример результата удаления записи из таблицы базы данных

Выводы по главе 3

В рамках данной главы приведено описание основных возможностей использования разработанного программного обеспечения с помощью таких средств разработка, как язык программирования C#, интегрированная среда разработки ПО Visual Studio 2010, фреймворка .NET, технологии Windows Forms. Приведены результаты создания пользовательского интерфейса разработанного ПО, отражены основные компоненты, классы, модули и описаны ключевые функциональные возможности по обработке созданной системой данных, хранимых в созданной БД. Обеспечена возможность авторизации в созданной системе. Для исключения возможности ложных срабатываний операций удаления данных из БД предусмотрена валидационная проверка в виде сообщения о подтверждении действий пользователя.

ЗАКЛЮЧЕНИЕ

В рамках первой главы приведены результаты проведенного анализа специфики исторического развития средства разработки программ, описаны предпосылки развития современных средств создания вычислительных программ, проведен обзор этапов создания первых языков программирования.

В рамках второй главы данной работы описаны результаты анализа языков разработки программного кода. Описан ряд существующих классификаций высокоуровневых языков программирования, приведено описание существующих типов трансляторов, а именно, интерпретаторов, компиляторов и ассемблеров.

Кратко обозначены ключевые отличия между декларативными и императивными языками разработки ПО, описаны логические и функциональные подходы к созданию программного кода, приведен рейтинг популярности средств разработки на 2017-й год. Проведен анализ возможностей современных языков программирования C#, Python и C++, кратко обозначены их основные преимущества, недостатки и особенности.

В рамках данной главы приведено описание основных возможностей использования разработанного программного обеспечения с помощью таких средств разработка, как язык программирования C#, интегрированная среда разработки ПО Visual Studio 2010, фреймворка .NET, технологии Windows Forms.

Приведены результаты создания пользовательского интерфейса разработанного ПО, отражены основные компоненты, классы, модули и описаны ключевые функциональные возможности по обработке созданной системой данных, хранимых в созданной БД.

Обеспечена возможность авторизации в созданной системе. Для исключения возможности ложных срабатываний операций удаления данных из БД предусмотрена валидационная проверка в виде сообщения о подтверждении действий пользователя.

В процессе выполнения работы достигнута поставленная цель посредством решения следующих задач:

1. Проведен анализ специфики исторического развития средств разработки программ.

2. Осуществлен анализ языков написания программного кода.

3. Выполнен анализ особенностей классификации высокоуровневых языков программирования.

4. Осуществлена разработка программного обеспечения с помощью языка C#.

5. Проведено описание структуры проекта и интерфейса пользователя разработанного программного продукта.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Галин А.Б. Информатика: учебник. – М.: Феникс, 2012. – 223 с.
2. Голицына О.Л., Партыка Т.Л., Языки программирования. Учебное пособие. – М.: Форум, 2012. – 467 с.
3. Джосаттис Н.М. Стандартная библиотека C++. Справочное руководство. — М.: Вильямс, 2014. – 1136 с.
4. Забудский Е.И. Объектно-ориентированный анализ и программирование на языке C# — М.: Кафедра ОИиППО ГУ-ВШЭ, 2012. — 705 с.
5. Иванова Г.С. Основы программирования Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2012. –303 с.
6. Камаев В.А., Костерин В.В. Технологии программирования Учебник. — М.: Высшая школа, 2016. - 314 с.
7. Колесников Е.А. Перфокарты. Технико-исторические заметки. – М.: Реноме, 2016. — 184 с.
8. Лутц М. Изучаем Python. СПб.: Символ-Плюс Медиа, 2012. – 1280 с.
9. Могилев А.В., Листрова Л.В. Методы программирования. Компьютерные вычисления. – СПб.: БХВ-Петербург, 2013. – 320 с.
10. Нейгел К., Ивьен Б., C# 5.0 и платформа .NET 4.5 для профессионалов. М.: Академия, 2014. – 1440 с.
11. Одинец В.П. Зарисовки по истории компьютерных наук. – Сыктывкар: Коми пединститут, 2013. — 421 с.
12. Орлов С.А. Теория и практика языков программирования. – СПб.: Питер, 2014. – 688 с.
13. Пройдаков Э.М. Страницы истории отечественных ИТ. – М.: Альпина Паблишер, 2016. — 241 с.
14. Рихтер Д. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#. – СПб.: Питер, 2012. – 720 с.
15. Саммерфилд М. Программирование на Python 3. СПб.: Символ-Плюс, 2012. – 608 с.
16. Семакин И.Г. Информатика и информационно-коммуникационные технологии. Базовый курс: - М: БИНОМ, Лаборатория знаний, 2012. – 314 с.
17. Сузи Р.А. Язык программирования Python. М.: Интуит, 2016. – 351 с.

18. Тюгашев А.А.. Основы программирования. Часть 2. – СПб.: Университет ИТМО, 2016. – 160 с.
19. Фет Я.И. Хрестоматия по истории информатики. – Новосибирск: Гео, 2014. — 559 с.
20. Хорев П.Б. Объектно-ориентированное программирование. – Москва: Академия, 2012. – 446 с.