**Министерство образования Тверской области**

**ГБПОУ «Тверской колледж им. А.Н. Коняева»**

Цикловая комиссия специальности 230115

«Программирование в компьютерных системах»

(наименование специальности)

УТВЕРЖДАЮ

Председатель

цикловой комиссии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

**Пояснительная записка**

К курсовому проекту по МДК 01.02 «Прикладное программирование»

На тему: «Органайзер питания»

Студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись, Дата Инициалы, Фамилия

Обозначение курсового проекта \_\_КП\_\_\_\_\_группа\_\_\_\_\_\_\_\_\_\_\_

Руководитель проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись, Дата Инициалы, фамилия

Проект (работа) защищен(а) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка\_\_\_\_\_\_\_

Подпись, дата

Члены комиссии:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись, дата Инициалы, фамилия

Тверь

2017

Содержание

[1 Теоретический раздел 4](#_Toc485569463)

[1.1 Обоснование необходимости создания спортивного приложения 4](#_Toc485569464)

[1.2 Перечень исходных web сайтов, используемых при разработке 4](#_Toc485569465)

[1.3 Технолгии используемые при разработке спортивного приложения 4](#_Toc485569466)

[1.3.1 Delphi 4](#_Toc485569467)

[1.3.2 Целевая платформа 5](#_Toc485569468)

[1.3.3 Отличие от других языков программирования 5](#_Toc485569469)

[1.3.3 Синтаксис языка 7](#_Toc485569470)

[1.3.3.1 Система типов 7](#_Toc485569471)

[1.3.3.2 Операторы 9](#_Toc485569472)

[1.3.3.3 Пример использования оператора in 10](#_Toc485569473)

[1.3.3.4 Объекты 11](#_Toc485569474)

[1.3.3.5 Классы 11](#_Toc485569475)

[1.3.4 Объектно ориентированные особенности языка 11](#_Toc485569476)

[1.3.4.1 Инкапсуляция 11](#_Toc485569477)

[1.3.4.2 Наследование 12](#_Toc485569478)

[1.3.4.3 Полиморфизм 12](#_Toc485569479)

[2. Практический раздел……………………………………………………… .......13](#_Toc485569480)

[2.1. Техническое задание на разработку *Органайзер питания* 13](#_Toc485569481)

[2.1.1 Введение 13](#_Toc485569482)

[2.1.2 Основание для разработки 13](#_Toc485569483)

[2.1.3 Назначение разработки 13](#_Toc485569484)

[2.1.4 Требования к программе или программному изделию 13](#_Toc485569485)

[2.1.5. Требования к программной документации 14](#_Toc485569486)

[2.1.6. Технико-экономические показатели 15](#_Toc485569487)

[2.1.7. Стадии и этапы разработки 15](#_Toc485569488)

[2.1.8 Порядок контроля и приёмки 15](#_Toc485569489)

[2.2 Предметная область 15](#_Toc485569490)

[2.2.1 Основные требования к спортивному питанию 16](#_Toc485569491)

[2.2.2 Белки 16](#_Toc485569492)

[2.2.3 Углеводы 17](#_Toc485569493)

[2.2.4 Жиры 18](#_Toc485569494)

[2.3 Перечень функций, которые должно выполнять приложение. 18](#_Toc485569495)

[2.4 Постановка целей и задач курсового проекта 18](#_Toc485569496)

[2.5 Разработка структурной модели размещения данных в приложении 18](#_Toc485569497)

[2.6 Внешний вид приложения 19](#_Toc485569498)

[2.7 Разработка информационного и программного обеспечения приложения 19](#_Toc485569499)

[2.8 Список используемых источников и литературы 20](#_Toc485569500)

[3 Заключение 21](#_Toc485569501)

[3.1 Тестирование программы 21](#_Toc485569502)

[3.2 Защита информации 24](#_Toc485569503)

# 1 Теоретический раздел

## 1.1 Обоснование необходимости создания спортивного приложения

В современном мире с постоянно растущим количеством дел, люди забыли о правильном режиме дня, питании и качестве пищи. Частыми недосыпами, перееданиями, а то и вовсе недоеданием люди ставят под угрозу своё здоровье. Данная программа напомнит людям о здоровом образе жизни, направит начинающих спортсменов на нужное русло и не забудет о тех кому просто необходимо похудеть. Мой выбор темы пал именно на этот топик не случайно, я часто посещаю тренажёрный зал и в определённый момент сам испытал нехватку подобной программы в сфере спорта.

## 1.2 Перечень исходных web сайтов, используемых при разработке

Ниже приведён перечень веб-ресурсов при помощи которых велась разработка программы:

<http://steelsports.ru/pitanie-dlya-nabora-myshechnoj-massy/> - сайт использованный при работе с предметной областью.

<https://www.freedieting.com/> - сайт использованный при работе с предметной областью.

<http://fitbreak.ru/diet/104-pravilnoe-pitanie-dlya-nabora-mishechnoi-massi> - сайт использованный при работе с предметной областью.

### 1.3 Технолгии используемые при разработке спортивного приложения

### 1.3.1 Delphi

**Delphi**—[императивный](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%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), [структурированный](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%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), [объектно-ориентированный](https://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) [язык программирования](https://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) со [строгой](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B8_%D1%81%D0%BB%D0%B0%D0%B1%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [статической типизацией](https://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) переменных. Основная область использования — написание прикладного программного обеспечения. Первоначально носил название [Object Pascal](https://ru.wikipedia.org/wiki/Object_Pascal) и исторически восходит к одноимённому диалекту языка, разработанному в фирме [Apple](https://ru.wikipedia.org/wiki/Apple) в 1986 году группой [Ларри Теслера](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%80%D1%80%D0%B8_%D0%A2%D0%B5%D1%81%D0%BB%D0%B5%D1%80)[[2]](https://ru.wikipedia.org/wiki/Delphi_(%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)#cite_note-2). Однако в настоящее время термин [Object Pascal](https://ru.wikipedia.org/wiki/Object_Pascal) чаще всего употребляется в значении языка [среды программирования Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8)). Начиная с Delphi 7[[3]](https://ru.wikipedia.org/wiki/Delphi_(%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)#cite_note-3), в официальных документах [Borland](https://ru.wikipedia.org/wiki/Borland) стала использовать название Delphi для обозначения языка [Object Pascal](https://ru.wikipedia.org/wiki/Object_Pascal).

### 1.3.2 Целевая платформа

Изначально среда разработки Delphi была предназначена исключительно для разработки приложений [Windows](https://ru.wikipedia.org/wiki/Windows), затем был реализован вариант для платформ [Linux](https://ru.wikipedia.org/wiki/Linux) (как [Kylix](https://ru.wikipedia.org/wiki/Kylix)), однако после выпуска в 2002 году Kylix 3 его разработка была прекращена, и вскоре было объявлено о поддержке [Microsoft .NET](https://ru.wikipedia.org/wiki/.NET_Framework), которая, в свою очередь, была прекращена с выходом Delphi 2007.

Наряду с поддержкой разработки 32 и 64-разрядных программ для Windows, реализована возможность создавать приложения для Apple [Mac OS X](https://ru.wikipedia.org/wiki/Mac_OS_X_10.0) (начиная с Embarcadero Delphi XE2), [iOS](https://ru.wikipedia.org/wiki/IOS) (включая симулятор, начиная с XE4 посредством собственного компилятора), а также, в Delphi XE5, для Google Android (непосредственно исполняемые на [ARM](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0))-процессоре).

### 1.3.3 Отличие от других языков программирования

При создании языка (и здесь качественное отличие от языка C) не ставилась задача обеспечить максимальную производительность исполняемого кода или лаконичность исходного кода для экономии оперативной памяти. Изначально, язык ставил во главу угла стройность и высокую читаемость, поскольку был предназначен для обучения дисциплине программирования. Эта изначальная стройность, в дальнейшем, как по мере роста аппаратных мощностей, так и в результате появления новых парадигм, упростила расширение языка новыми конструкциями.

Так, сложность объектного C++, по сравнению с C, выросла весьма существенно и затруднила его изучение в качестве первого языка программирования, чего нельзя сказать об Object Pascal относительно Pascal.

Ниже перечислены некоторые отличия синтаксических конструкций Delphi от семейства C-подобных языков (C/C++/Java/C#):

В Delphi формальное начало любой программы четко отличается от других участков кода и должно располагаться в определенном, единственном в рамках проекта, исходном файле с расширением dpr (тогда как другие файлы исходных текстов программы имеют расширение pas) В С-подобных языках программирования в качестве входа обычно используется глобальная функция или статический метод с именем main и определенным списком параметров, причём такая функция может быть расположена в любом из файлов исходного текста проекта.

* В Delphi идентификаторы типов, переменных, а равно и ключевые слова читаются *независимо* от регистра: например идентификатор SomeVar полностью эквивалентен somevar. Регистро-зависимые идентификаторы в начале компьютерной эпохи ускоряли процесс компиляции, и кроме того, позволяли использовать очень короткие имена, порой отличающиеся лишь регистром.

И хотя к настоящему времени обе эти практики − использование нескольких идентификаторов, различающихся лишь регистром, равно как и чрезмерная их лаконичность, осуждены и не рекомендованы к применению, практически все унаследованные от С языки − C++, Java, C# − являются *регистро-зависимыми*, что, с одной стороны, требует достаточно большой внимательности к объявлению и использованию идентификаторов, а с другой — принуждает писать более строгий код, когда каждая переменная имеет чётко определённое имя (вариации регистра могут вызвать путаницу и ошибки).

* В Delphi в исходных файла pas (которые, как правило, и содержат основное тело программы) на уровне языковых средств введено строгое разделение на интерфейсный раздел и раздел реализации. В интерфейсной части содержатся лишь объявления типов и методов, тогда как код реализации в интерфейсной части не допускается на уровне компиляции. Подобное разделение свойственно также языкам C/C++, где в рамках культуры и парадигмы программирования вводится разделение на заголовочные и собственно файлы реализации, но подобное разделение не обеспечивается на уровне языка или компилятора.

В C# и Java такое разделение утрачено вовсе − реализация метода, как правило, следует сразу же после его объявления. Инкапсуляция обеспечивается лишь принадлежностью метода к той или иной области видимости. Для просмотра одной только интерфейсной части модуля исходного кода используются специальные средства.

* В Delphi метод или функция четко определяются зарезервированными для этого ключевыми словами procedure или function, тогда как в C-подобных языках различие обуславливается ключевым словом, определяющим тип возвращаемого значения.

### 1.3.3 Синтаксис языка

### 1.3.3.1 Система типов

[Система типов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2) в Delphi [строгая](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%B3%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [статическая](https://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). Поддерживаются следующие [типы данных](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85):

* целочисленные, знаковые, и без знаковые: Byte, Shortint, Word, Smallint, Cardinal,Integer, UInt64, Int64
* типы-перечисления, задаваемые пользователем
* вещественные типы Single, Double, Extended (только x86-32), унаследованный тип Real, работающий в режиме целочисленной эмуляции. Тип Currency вещественное фиксированной точности.
* строки. Тип string — автоматический распределяемый в памяти, с подсчётом ссылок и парадигмой Copy-On-Write. В поздних версиях Delphi символы двухбайтные, Unicode-совместимые. AnsiString — аналогичная реализация для строк с шириной символа в один байт. Такие строки содержат в служебном поле информацию о кодировке. В Windows компиляторах с ранних версий существует тип WideString, полностью совместимый типу BSTR в Component Object Model. Допускается использование примитивных строковых типов, в стиле языка C: PChar и PWideChar
* массивы. Одномерные, многомерные фиксированной длины, а также подобные им динамические, с подсчётом ссылок.
* множества, состоящие из элементов типа-перечисления. Максимальный размер такого перечисления — 256 элементов.
* [Записи](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BF%D0%B8%D1%81%D1%8C_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)). Структурный (value) тип без поддержки наследования. Начиная с Delphi 2006 добавлена поддержка инкапсуляции, методов, свойств. Перегрузка операторов.
* Классы и обобщённые классы (generics). Неявно-ссылочный тип. Поддержка инкапсуляции, наследования, полиморфизма, в том числе виртуальных конструкторов, атрибутов, обобщённых параметров для класса и отдельных методов, а также диспетчеризации методов по индексу. Класс может реализовать один или несколько интерфейсов, в том числе опосредованно, делегируя реализацию интерфейса свойству или полю. Множественное наследование не поддерживается. В Delphi класс является неявным ссылочным типом.
* Указатели на функции и методы, а также указатели на анонимные функции.
* Типы-[метаклассы](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%BA%D0%BB%D0%B0%D1%81%D1%81), содержащие указатель на тип объекта (но не сам объект). В основном введены для реализации виртуальных конструкторов и автоматической сериализации.
* интерфейсы. COM-совметимые (в Windows-компиляторе), унаследованные от одного предка. Множественное наследование не поддерживается.
* Диспинтерфейсы, для работы с интерфейсами IDispatch в режиме позднего связывания.
* [Вариантные типы](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Variant_type) Variant и OleVariant
* Старые объекты, поддерживаемые для совместимости с Turbo Pascal. В отличие от экземпляра класса, объект может быть размещен в стеке, или статически.

### 1.3.3.2 Операторы

Арифметические: + — \* / div mod Сложение, вычитание, умножение, деление(дающее вещественный результат), целочисленное деление, выделение остатка.

По типу возвращаемого значения *различаются* операторы целочисленного деления (div и mod) и оператор /. Последний, применяемый как к целочисленным, так и к вещественным операндам, всегда в результате дает вещественный тип. Оператор сложения + используется также для конкатенации строк (когда используются встроенные строковые типы).

[Бинарные](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82%D0%BE%D0%B2%D1%8B%D0%B5_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8)/логические: not and or xor Инверсия(отрицание), «И», «ИЛИ», Исключающее «ИЛИ». Тип операции (бинарная или логическая) зависит от типа первого операнда.

К битовым операторам целочисленных типов относятся также shl, shr — операторы сдвига, соответствующие по смыслу одноименным командам процессоров Intel x86.

Порядковые операторы(операторы сравнения)= <> > < >= <= — равенства, неравенства(соответствует оператору != в C-подобных языках), больше, меньше, не меньше, не больше — применяются ко всем порядковым и вещественным типам и возвращают значение типа boolean.

К операторам множества относятся + - \* in — сложение, вычитание, пересечение множеств, а также оператор теста вхождения, которые используются для манипуляций со встроенным типом множества. Первые три возвращают тип множества, последний — булевый тип.

### 1.3.3.3 Пример использования оператора in

Операторы приведения типов — () as is — безусловное приведение, безопасное приведение объектных и интерфейсных типов, а также оператор теста принадлежности к типу(возвращает булевое значение). Безусловное(небезопасное) приведение используется в функциональном стиле(слева пишется идентификатор типа, справа в скобках приводимое к нему выражение) и применяется к порядковым, вещественным, структурным, ссылочным, строковым типам типам. При этом для ссылочных(включая неявно-ссылочные) типов **не** происходит действительного приведения, а лишь новая интерпретация тех же данных.

Операторы as и is применяются к типам, допускающим полиморфное поведение — экземплярам класса и интерфейсам. Первый приводит безопасное (в смысле невозможности неверной интерпретации) приведение типа, а второй тестирует поддержку экземпляром класса или интерфейсом некоторого класса или интерфейса. Нужно помнить, что в отличие от языка C# неудавшееся приведение типа оператором as возбуждает исключение.

Ссылочные операторы ^ @ — используются для работы с указателями.

Оператор ^ разыменовывает указатель. Оператор @ выполняет обратное действие, возвращая адрес переменной. Поддерживается простые операции сложения и вычитания над типизированными указателями, с учётом размера типов, на которые они указывают (*умная* арифметика указателей).

Оператор присваивания :=. В Delphi оператор присваивания, образует не выражение, а операцию, таким образом, не допускается «нанизывание» присваиваний.

### 1.3.3.4 Объекты

[Объекты](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%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)) как экземпляры класса объявляются в программе в разделе var как обычные переменные. Например:

**var**

CCircle1: TColorCircle;

CircleA: TCircle;

Как и в случае записей, для обращения к конкретному элементу объекта (полю, свойству или методу) указывается имя объекта и имя элемента, разделённые точкой, то есть имя элемента является составным.

### 1.3.3.5 Классы

В языке Object Pascal классы — это специальные типы данных, используемые для описания объектов. Соответственно объект, имеющий тип какого-либо класса, является экземпляром (instance) этого класса или переменной этого типа.

Класс представляет собой особый тип, имеющий в своём составе такие элементы, как поля, свойства и методы. Поля класса аналогичны полям записи и служат для хранения информации об объекте. *Методами* называются процедуры и функции, предназначенные как правило для обработки полей. *Свойства* занимают промежуточное положение между полями и методами.

### 1.3.4 Объектно-ориентированные особенности языка

### 1.3.4.1 Инкапсуляция

Объединение и скрытие объектных данных, а также обрабатывающих их методов, внутри конкретного класса от пользователя называется инкапсуляцией.

### 1.3.4.2 Наследование

При создании новых объектов способность получить все свойства и методы от своих предков называют наследованием. Такие объекты наследуют после своего создания все поля, свойства, события, методы и прочее от своих предков. Наследование часто избавляет разработчиков от рутинной работы и позволяет не мешкая приступить к разработке чего-то нового.

### 1.3.4.3 Полиморфизм

Delphi реализует классическую модель полиморфизма, принятую в *прикладных* языках программирования, когда методы базового класса, а также ссылочные переменные типа базового класса, — способны манипулировать экземплярами классов-потомков на основе контракта, заданного в базовом классе. Контрактом в этом случае выступает объявление абстрактных методов в базовом классе.

# 2. Практический раздел

## 2.1. Техническое задание на разработку *Органайзер питания*

### 2.1.1 Введение

Наименование приложения - *Органайзер питания*.

Приложения поможет привести в порядок своё питание начинающим спортсменам.

### 2.1.2 Основание для разработки

Основанием для разработки является задание на курсовую работу по дисциплине “М.Д.К. 01.02. Прикладное программирование”.

### 2.1.3 Назначение разработки

Функциональное назначение приложения: Органайзер питания предназначен для улучшения качества жизни нуждающимся пользователям.

Эксплуатационное назначение приложения: Органайзер питания будет применяться для получения наставлений по приёмам пищи.

### 2.1.4 Требования к программе или программному изделию

**2.1.4.1 Требования к функциональным характеристикам**

Разрабатываемое приложение должен обладать следующими функциями:

* Достоверность данных;
* Описывать информацию об ошибках;
* Отвечать на действия пользователя;
* Ввод входных данных с клавиатуры;
* Контроль вводимых данных;
* Быстрота расчётов алгоритмов;
* Получение исчерпывающей информации;

**2.1.4.2 Требования к надежности**

Приложение должно работать без сбоев, а также гарантировать правильность выполнения вышеперечисленных операций, предоставлять доступный и простой интерфейс, контроль вводимых данных и понятно описывать причины ошибок.

**2.1.4.3 Условия эксплуатации**

Пользователь должен знать простейшие принципы работы с компьютером, различать целые и вещественные данные, и вводить корректную информацию.

**2.1.4.4 Требования к составу и параметрам технических средств**

Для удовлетворяющей работы программы необходимы следующие требования:

* Процессор Intel Pentium и выше;
* Оперативная память не менее 512 Mb;
* Свободного места на диске 100 Mb;
* Видеокарта от 128 Mb и выше;

**2.1.4.5 Требования к информационной и программной совместимости**

Для пользования приложением необходима операционная система Windows XP или выше, для внесения изменений в программу, рекомендовано использовать Embarcadero Rad Studio 2010 года.

**2.1.4.6 Требования к маркировке и упаковке**

Не предъявляются.

**2.1.4.7 Требования к транспортировке и хранению**

Программа транспортируется на дисковом носителе, а для её хранения предложено использовать жёсткий диск.

### 2.1.5. Требования к программной документации

Не предъявляются.

### 2.1.6. Технико-экономические показатели

Технико-экономические показатели не рассчитывались. Программа сокращает работу высчитывания формул, пользователю не придётся делать это вручную, в добавок пропадает вероятность допущения ошибки в расчётах.

### 2.1.7. Стадии и этапы разработки

В данном пункте приведена таблица стадий и этапов разработки программы (Таблица 1 - Стадии и этапы разработки). В ней наглядно описаны даты начал и завершений каждого из модулей.

Таблица 2 - Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Этап разработки | Сроки исполнения | | Документ |
| Начало | Завершение |
| Анализ и планирование требований | 26.02.17 | 03.03.17 | Техническое задание |
| Проектирование | 03.03.17 | 07.03.17 | Отчет |
| Реализация | 07.03.17 | 10.06.17 | Код приложения |
| Тестирование и отладка | 10.06.17 | 12.06.17 | Отчет |
| Эксплуатация и сопровождение | от 12.06.17 |  | Отчет |

### 2.1.8 Порядок контроля и приёмки

Контроль и приёмка Программы проводится в виде контрольного примера.

### 2.2 Предметная область

Предметной областью разработки данного приложения, является спорт, а именно его пищевая составляющая.

Питание спортсменов отличается от питания обычных людей по нескольким причинам. Во-первых, активные занятия спортом требуют гораздо большего количества энергозатрат, чем работа в офисе или даже на производстве. Во-вторых, тяжёлые нагрузки и довольно специфические требования к функциональности организма спортсмена требуют особого подхода к составу рациона. Правильное питание для спортсменов должно быть калорийным, здоровым и разнообразным.

## 2.2.1 Основные требования к спортивному питанию

Чтобы добиться высоких спортивных результатов, важно правильно дозировать нагрузки и обеспечивать восстановление после них. Восстановление невозможно без полноценного питания — это то, что поставляет организму энергию и материал для воспроизводства новых клеток.

Рацион спортсмена должен разрабатываться с учётом выполнения следующих задач:

* Обеспечение организма необходимым количеством калорий, микроэлементов и витаминов (показатели калорийности зависят от конкретных спортивных задач);
* Активация и нормализация обменных процессов (этот пункт обеспечивают биологические активные вещества и различные натуральные добавки);
* Регуляция веса (на разных этапах спортсменам необходимо увеличивать, уменьшать или поддерживать массу в неизменном состоянии);
* Изменение морфологических показателей (увеличение мышечной массы и уменьшение жировых отложений).

### 2.2.2 Белки

Это наиважнейший компонент в спортивном питании. Значение белков в организме многогранно:

* Это строительный материал, из которого организм производит все биологические структуры (из протеина состоит мышечная ткань, связки, сухожилия, органическая часть костной ткани);
* Белки выступают в качестве катализаторов, ускоряющих в миллионы раз биохимические процессы;
* Все энергетические и метаболические реакции, протекающие в клетках, зависят от активности белков-ферментов;
* Белки входят в состав гормонов и являются факторами роста;
* Протеины выполняют транспортную функцию, обеспечивая ткани кислородом и питательными веществами;
* Участвуют в иммунной защите (антитела по сути есть специфические белковые молекулы).

Главные источники белков для спортсменов это:

* Рыба (в рыбе помимо белков, содержатся также жирные кислоты и витамины);
* Мясо (для спортсменов лучше употреблять нежирные мясные сорта — птицу, кролика, нежные сорта телятины);
* Яйца (в яйце содержатся важнейшие для организма аминокислоты, а также жиры и витамины);
* Молочные продукты (в молочном белке содержится много метионина — незаменимой аминокислоты, которая не производится организмом человека).

### 2.2.3 Углеводы

Углеводы — основа для метаболических и энергетических процессов. Во время нагрузок углеводные соединения расходуются особенно интенсивно. Для спортсменов полезней употреблять «сложные» углеводы, которые содержатся в коричневом рисе, пшенице, чёрном хлебе, фруктах и овощах, а не «простые» — те, что в избытке находятся сахаре, сладостях и лимонаде. Сахар спортсменам лучше заменить на мед, в котором помимо углеводов содержится невероятное количество витаминов и микроэлементов.

### 2.2.4 Жиры

Ещё один источник энергии и строительного материала, который входит в состав всех клеточных мембран. Основной расход жиров у спортсменов приходится на выполнение трудоёмких и медленных упражнений, на поддержание постоянной температуры тела и стабильную работу внутренних органов во время нагрузок.

### 2.3 Перечень функций, которых должно выполнять приложение.

Приложение должно выполнять следующие функции:

1. Облегчить порядок расчётов (не нужно считать вручную);
2. Давать актуальную информацию о питании;
3. Распределить график по приёму пищи;
4. Составлять программу для похудения и для набора массы.

## 2.4 Постановка целей и задач курсового проекта

Основной целью и задачей разработки приложения Органайзер питания, является помощь начинающим спортсменам с организацией питания, с этими установками им будет проще подобрать под себя индивидуальный рацион.

## 2.5 Разработка структурной модели размещения данных в приложении

В данном пункте приведён рисунок модели данных в приложении (Рисунок 1 - Структурная модель приложения).

В проекте существует одна форма на которой по умолчанию расположены 3 кнопки, каждая из которой вызывает свою характерную панель. На этих панелях расположены элементы для работы с определёнными алгоритмами.

Форма1

Кнопка(Расчёт частоты питания)

Кнопка(Расчёт пищевой ценности)

Кнопка(Состав приёма пищи)

Панель3 с алгоритмом

Панель2 с алгоритмом

Панель1 с алгоритмом

Рисунок 2 - Структурная модель приложения.

## 2.6 Внешний вид приложения

На данном рисунке (Рисунок 2 - Внешний вид приложения.) представлено оформление приложения, а именно вид второй панели.

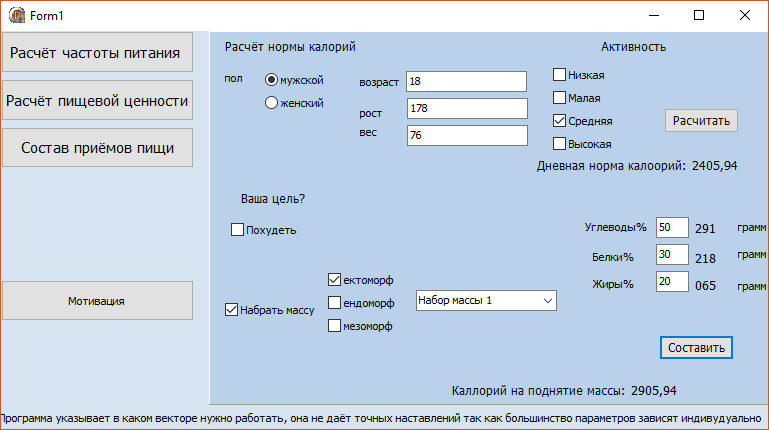


Рисунок 2 - Внешний вид приложения.

### 2.7 Разработка информационного и программного обеспечения приложения

Текст программы не требуется.

## 2.8 Список используемых источников и литературы

1. http://www.delphi-manual.ru/ – справочник Delphi
2. http://www.delphisources.ru – справочник Delphi;
3. http://delphi-box.ru/ – справочник Delphi;
4. ГОСТ 19.201-78 – Стандарт курсового проекта.

### 3 Заключение

### 3.1 Тестирование программы

В текущем разделе описано взаимодействие с элементами приложения.

На данном рисунке (Рисунок 3-Запуск приложения.) изображено стартовое окно приложения.

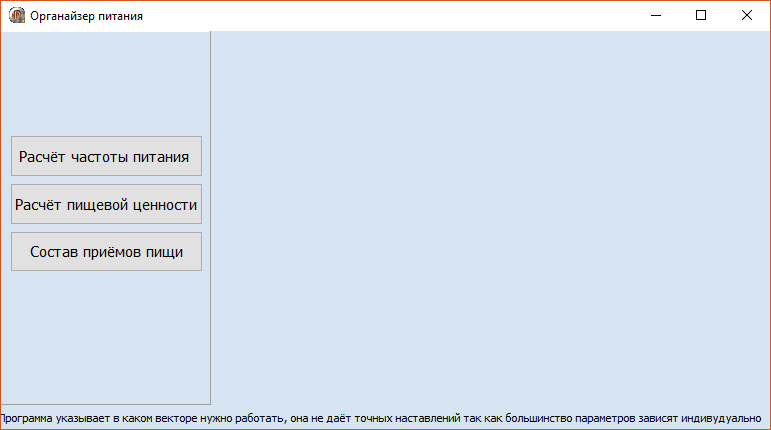


Рисунок 3 -Запуск приложения.

На этом рисунке (Рисунок 4 –Выбран первый раздел приложения (“Расчёт частоты питания”).) иллюстрируются элементы первой панели.

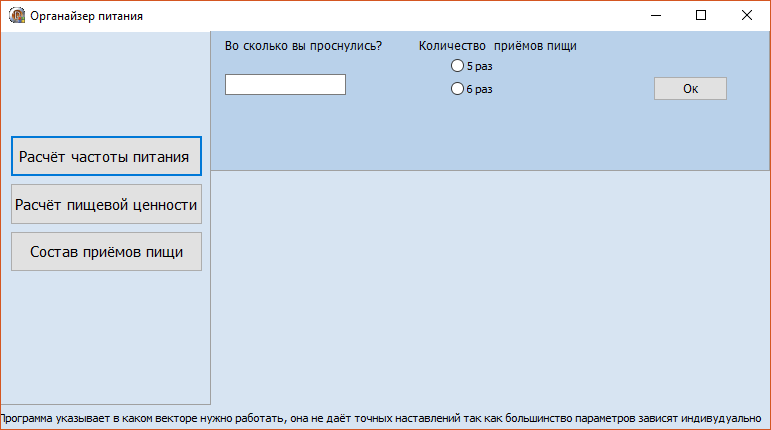


Рисунок 4 –Выбран первый раздел приложения (“Расчёт частоты питания”).

На следующем рисунке (Рисунок 5 –Рассчитанный алгоритм первого раздела.) приведены выходные данный при выполнении алгоритма.

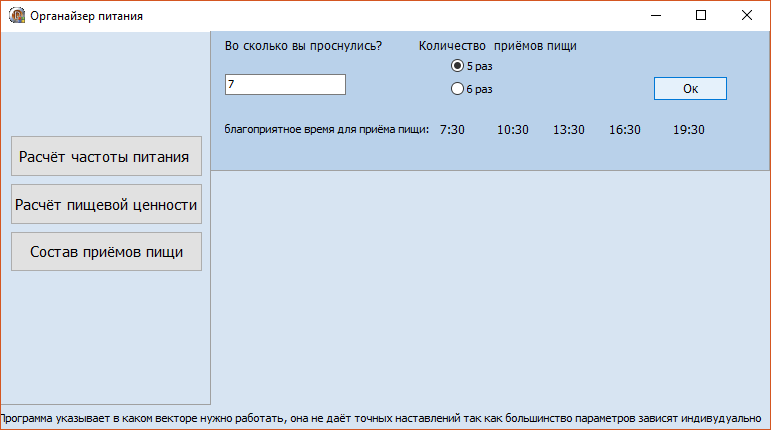


Рисунок 5 –Рассчитанный алгоритм первого раздела.

Следующим рисунком (Рисунок 6- Окно второго раздела.) описан состав второй формы.

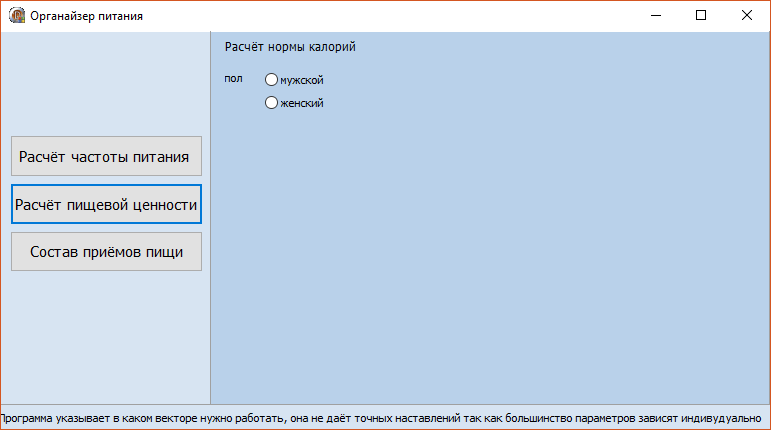


Рисунок 6 - Окно второго раздела.

Далее, изображена полученная информация в ходе заполнения формы (Рисунок 7 –Заполненная форма (“Расчёт частоты питания”).).

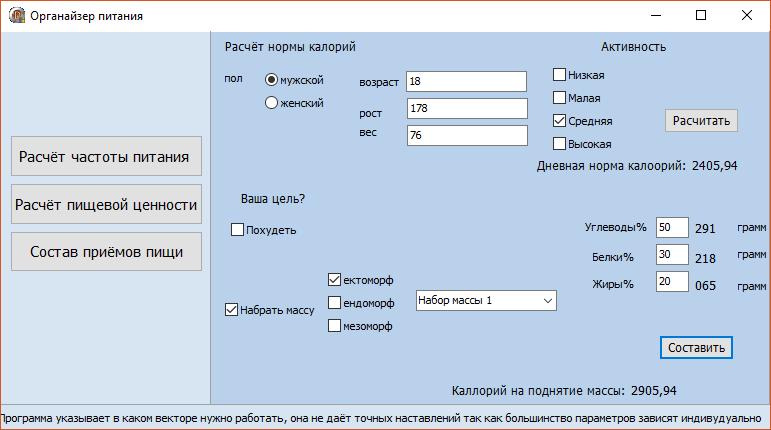


Рисунок 7 –Заполненная форма (“Расчёт частоты питания”).

Следующим рисунком (Рисунок 8 –Внешний вид третьего раздела.) показан состав третьей формы.

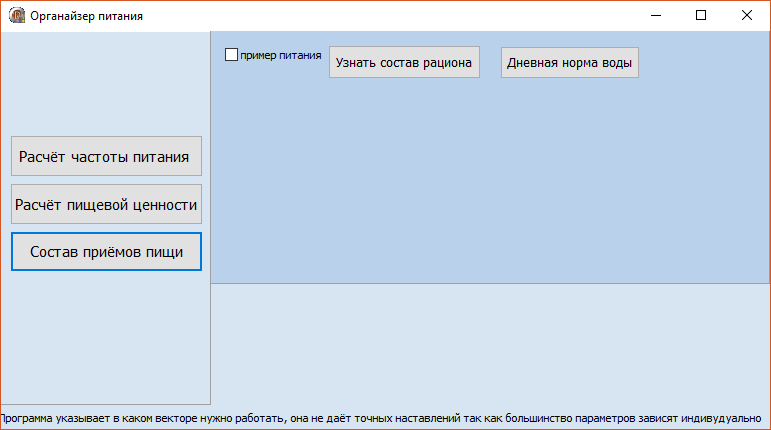


Рисунок 8 –Внешний вид третьего раздела.

Этим рисунком (Рисунок 9 – Полученный результат от раздела (“Состав приёмов пищи”).) описан результат выполнения алгоритмов на третьей форме.

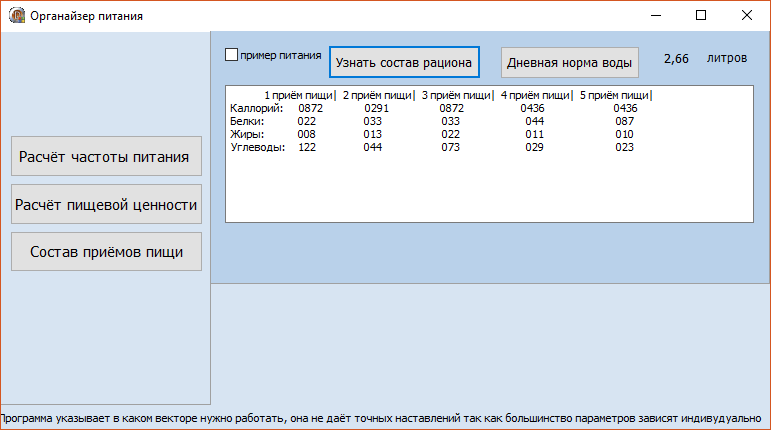


Рисунок 9 – Полученный результат от раздела (“Состав приёмов пищи”).

### 3.2 Защита информации

Программа находится в свободном доступе, каждый может модифицировать её под себя. Единственным выдвигаемым требованием является запрет на представление программы за свою и на стороннюю продажу.