федеральное государственное бюджетное образовательное учреждение высшего образования

«Вологодский государственный университет»

Институт математики, естественных и компьютерных наук

Кафедра информатики и информационных технологий

КУРСОВАЯ РАБОТА

Дисциплина: «Информационные технологии»

Наименование темы: «Создание калькулятора калорийности проектным методом»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 09.03.02  код направления  подготовки/  специальности | \_\_\_43.09\_\_.  код выпускающей  кафедры | \_\_\_\_\_\_9\_\_\_\_\_\_\_  регистрационный номер по журналу | Очная  код формы  обучения | 2019  год |

|  |  |
| --- | --- |
| Руководитель | \_\_\_\_\_\_\_\_Никифоров Олег Юрьевич\_\_\_\_\_\_\_\_ |
| Выполнил студент | \_\_\_\_\_\_\_\_\_\_\_Иванов Денис Геннадьевич\_\_\_\_\_\_\_\_\_ |
| Группа, курс | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ИТ-21, 2 курс\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата сдачи | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата защиты  Оценка по защите | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Вологда

2019 г.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc28017548)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РЕАЛИЗАЦИИ КАЛЬКУЛЯТОРА КАЛОРИЙ. 8](#_Toc28017549)

[1.1 Описание предметной деятельности. 8](#_Toc28017550)

[1.2 Описание инструментариев. 11](#_Toc28017551)

[1.3 Общее резюме. 16](#_Toc28017552)

[ГЛАВА 2. ПРАКТИЧЕСКИЕ ОСОБЕННОСТИ РАЗРАБОТКИ. 17](#_Toc28017553)

[2.1 Проектирование продукта. 17](#_Toc28017554)

[2.2 Разработка ПО. 22](#_Toc28017555)

[2.3 Заключение. 63](#_Toc28017556)

[ГЛАВА 3. УПРАВЛЕНИЕ ПРОЕКТОМ СОЗДАНИЯ. 64](#_Toc28017557)

[3.1 Описание команды. 64](#_Toc28017558)

[3.2 Задачи проекта. 65](#_Toc28017559)

[3.3 Бюджет. 67](#_Toc28017560)

[ЗАКЛЮЧЕНИЕ 68](#_Toc28017561)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 69](#_Toc28017562)

# ВВЕДЕНИЕ

Питание играет очень важную роль в жизни человека. Да что там говорить, без этого невозможно обойтись. Именно правильное питание является одним из самых важных условием успеха в повседневной жизни человека.

И так, что же необходимо сделать? Нужно создать приложение, которое позволило бы помочь людям в реализации правильного питания. Эта программа должна обладать красивым дизайном, простым интерфейсом, полезным функционалом.

В первую очередь необходимо реализовать центральное поле. В этом центральном поле должны находиться все приемы пищи пользователя данным программным обеспечением. При этом должна отображаться основная информация о каждом приеме пищи, а также о каждом продукте, которые соответственно и входят в приемы пищи. Далее надо реализовать динамическое добавление продуктов питания и приемов пищи. Ведь сразу же после добавления основная информация будет полностью меняться.

После центрального поля мы нуждаемся в конечном подсчете информации, т.е. подведение итогов. Информация считывается с центрального поля, с каждого приема пищи, с каждого продукта. Далее на основе полученных данных мы должны получить общий результат.

Вторым важным аспектом этого программного продукта является база данных продуктов. В ней будет приведена такая важная информация, как название продукта, количество белков, количество жиров, количество углеводов, и количество калорий. Весь расчет ведется на 100 грамм продукта. Эта база данных также должна еще уметь реализовывать ряд важных задач, такие как, добавление нового продукта, его редактирование (при этом после редактирования продукта изменения происходят и в центральном поле), сохранение после внесенных корректировок в базу данных, удаление ненужной информации и выход.

Как это все будет реализовываться?

Реализовываться все это будет с помощью программной среды Delphi. Центральное поле будет представлять собой TreeView. Это очень удобно, т.к. этот компонент обладает необходимыми процедурами, функциями для нашей задачи. Каждый прием пищи будет являться узлом, а каждый продукт будет являться подузлом узла приема пищи. Очень удобно создавать различные функции обработки дерева.

База данных продуктов будет представлять собой StringGrid таблицу. Почему именно таблица? Это тоже очень удобно. Каждая ячейка этой таблицы базы данных будет нести свою полезную информацию. Обрабатывать данные таблицы тоже очень легко. Да и к тому же, само представление информации в виде таблицы позволяет создать ряд преимуществ, например, удобство.

Работа с калькулятором калорий будет осуществляться с помощью кнопок. Эти кнопки будут программироваться на выполнение определенной функции.

Для чего же нужен калькулятор калорий?

Дело в том, что подсчет калорийности своего рациона является самым эффективным способом похудения или же набора мышечной массы или жировой, или же просто поддержания веса в постоянном значении. Много людей не хотят заниматься этим делом. А почему? Дело в том, что они считают, что это очень сложно. Ведь, действительно, узнать сколько белков, жиров, углеводов, калорий содержится в продукте на 100 грамм, а затем перевести это в свою необходимую массу, а затем узнать сколько приходится на каждый прим пищи, и в итого на весь свой рацион достаточно проблематично, особенно если с арифметикой не так уж и хорошо обстоят дела. Для этого и нужно это программное обеспечение. просто введите название продукта, его БЖУ на 100 грамм продукта и внесите необходимую массу в свой рацион питания. Это многое упрощает. Многие люди, которые занимаются здоровым образом жизни, занимаются профессиональным спортом, ведут правильное питание нуждаются в подобных программах. Конечно, можно и самому набросать на листочке все это дело с помощью обычного калькулятора арифметических операций, но это долго. Да, и когда будет нужда внести какие-либо корректировки, возникнет масса очень нехороших проблем.

Теперь стоит описать каждую функцию калькулятора калорий более подробно. Их создано немало в приложении, а именно 19.

И так, после открытия программы пользователю предоставляется основное центральное поле, в котором содержатся основные приемы пищи и основные продукты.

Функция “Добавить прием пищи в конец” позволяет добавлять прием пищи, и добавляется он именно в конец рациона.

Функция “Добавить прием пищи поверх выделенного” реализует создание нового приема пищи над выделенным. Если прием пищи не выделен, либо же выделен продукт, то выводится специальное сообщение, которое указывает пользователю о совершенной ошибке.

Вместе эти две функции с функцией удаления (о ней чуть позже) могут пользователю редактировать и создавать абсолютно любой рацион по их усмотрению, что очень удобно.

Рассмотрим функцию “Добавить продукт в конец”. Она подобно уже рассмотренной выше функции “Добавить прием пищи в конец” создает новый продукт в конце списка приема пищи. При это должен быть выделен необходимый прием пищи.

Далее функция под названием “Добавить продукт поверх выделенного”. Она реализует новый продукт и вставляет его над выделенным. Если продукт не выделен, то всплывающее сообщение укажет об этом.

При создании приема пищи предлагается ввести его название. Реализована защита от ошибок. Т.е. если пользователь, забыл ввести название, то программа укажет об этом.

Аналогично и при выборе продукта. Когда выбирается продукт, подключается база данных, в которой осуществляется выбор необходимого продукта, а также вводится его масса, далее осуществляется подсчет БЖУ на выбранную массу, и продукт уже добавляется непосредственно в рацион. Также реализуется защита от непредвиденных обстоятельств. Если будет введено неверное значение массы, то приложение укажет об этом пользователю, а если он забудет ввести, то всплывающее сообщение напомнить ему об этом.

Если вдруг пользователю будет необходимо переименовать прием пищи, то функция “Переименовать название приема пищи” легко справится с данной задачей. Также создана защита от неправильных данных.

Рацион питания пользователя, представленный на центральном поле ввода, может, так сказать сворачиваться и раскрываться.

Функция “Свернуть” убирает все продукты питания из видимой области.

Функция “Раскрыть” возвращает все продукты питания в видимую область центрального поля ввода.

Функция “База продуктов” переключает пользователя на все его созданные продукты. Включается новая форма с новыми кнопками управления.

После создания своего рациона и после внесения корректировок человека, который пользуется данным обеспечением, необходимо изменения все сохранить для последующего открытия рациона. Этим занимаются функции “Сохранить” и “Сохранить как…”. В первом случае изменения заносятся в уже выбранный файл, а во втором случае подключается диалог выбора необходимого файла.

Функция “Открыть…” подключает диалог выбора нужного составленного меню и открывает его.

Важной функцией является “Удалить”. При этом при выборе приема пищи удаляется прием пищи, и соответственно при выборе продукта удаляется продукт. Если не выделен ни прием пищи, ни продукт, выводится сообщения, как и в выше рассмотренных случаях о необходимости выделения удаляемого объекта.

Последней этого окна является функция “Закрыть”, которая осуществляет процедуры закрытия приложения и выхода из него.

Теперь необходимо перейти к окну базы данных и описать ее функции.

“Добавить продукт” создает новый продукт в базе данных. Открывается диалоговое окно, в котором предлагается ввести количество БЖУ на 100 грамм описываемого продукта. Есть защита от несанкционированных данных. Новый продукт добавляется в конец.

Кнопка “Редактировать” позволяет поменять название, изменить количество БЖУ продукта. При этом после изменения в самой базе данных происходит анализ всего рациона питания пользователя, и корректировка вносятся туда тоже. При воде неверных данных программа сигнализирует об этом всплывающими сообщениями и предлагается осуществить ввод снова, что является важным аспектом любой программы.

После внесения каких-либо изменений в базу данных результат можно сохранить “Сохранить изменения”. Сохранения осуществляется в специальный текстовый файл. Сразу же после включения данного программного обеспечения весь текстовый файл с продуктами прогружается в программу, и там в ней происходит работа с этими данными.

Продукт можно удалить из базы данных “Удалить продукт”. Удаляется абсолютно любой выбранный в таблице продукт, и удаление происходит со смещением всех продуктов, что очень удобно.

Функция “закрыть” закрывают окно базы данных.

# ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РЕАЛИЗАЦИИ КАЛЬКУЛЯТОРА КАЛОРИЙ.

## 1.1 Описание предметной деятельности.

Правильное питание играет очень важную роль в жизни человека. Об этом уже не раз говорилось. Если необходимо создать красивое тело или же улучшить его, то самым эффективным методом будет подсчет БЖУ, калорийности. Следует более серьезно описать эту предметную деятельность. Нужно описать что представляют собой белки, для чего нужны жиры, что дают углеводы, как правильно подсчитать сколько калорий в составленном рационе.

Следует начать с белков. Белок - это очень важный компонент, необходимый организму, при этом абсолютно любому. Все живые организмы строятся на этих “кирпичиках”. Белки задействуются организмом во многих процессах, протекающих в организме. Следует привести пример того же синтеза белка. Синтез белка представляет собой очень сложный процесс, который протекает внутри клетки живых организмов. Происходит считывание информации в ядре клетки на основе именно белковых структур, которые переносятся транспортными рибонуклеиновыми кислотами. Также с помощью белковых структур осуществляется построение костных тканей, мышечных тканей. Именно эти ткани формируют опорно-двигательную систему. Очень много белка содержится в куриных белках, в твороге, в мясе, в молоке, в протеиновых коктейлях, которые очень популярны в теме телостроительства. После того как пища попала в пищеварительную систему человека, там происходит расщепление белков на аминокислоты. Много спортсменов при интенсивных тренировках увеличивают свой объем потребления белков. Но необходимо во всем знать меру, найти золотую середину. Ведь, если употреблять этот важный строительный компонент, то можно серьёзно ударить по здоровью своего организма. Поэтому необходимо просчитывать норму самостоятельно, относительно характеристикам именно своего организма. С этим и помогает справиться калькулятор калорий. Нормально употреблять 1 грамм белка на килограмм собственного веса. Если в рационе будет очень мало белка, то это может привести к слабости мышечной ткани, будут проблемы с кожей, гормональный фон может будет ухудшен, а это, ведь есть не очень хорошо. Также могут возникнуть проблемы с иммунитетом. А что будет, если белка очень много? Последствия тоже не самые приятные. В первую очередь будет дана очень большая нагрузка на печень, что может привести к её разрушению. У женщин может возникнуть гормональный дисбаланс.

Теперь поговорим о углеводах - источниках энергии живых организмов. Они образуются в основном благодаря распаду крахмалов и сахаров. Углеводы бывают простые и сложные. Сложные обитают, так сказать, как правило, в крупах, некоторых фруктах, овощах. Это из-за сложности молекулярных структур. Более сложные элементы при распаде дают большое количество килоджоулей энергии. Теперь о “местонахождении” простых углеводов. Они содержатся в сладостях, изделиях из муки, очень сладких продуктах. Желательно отдавать предпочтение сложным углеводам. Дело в том, что из-за простые вызывают резкий прирост сахара в крови, что в свою очередь не есть хорошо. Также они очень быстро используются организмом, они очень быстро сгорят, чувство голода наступает очень быстро. При сбросе лишнего веса этот фактор играет очень важную роль.

Следует указать ряд функций углеводов. Они обеспечивают большую часть энергии для организма. Также с помощью углеводов осуществляется строительство структур тела. Конечно, в основном эта функция ложится на белки, но без углеводов в этом процессе тоже никак. При недостатке углеводов последствия, как и в первом случае, могут быть не из приятных. В первую очередь будет нехватка энергии на основные процессы жизнедеятельности организма. Начнутся проблемы во внутренних системах тела. Может возникнуть ряд неприятных заболеваний. Мозг является командным центром управления. Ему необходимо очень много энергии. А, ведь, если ее не будет, то, соответственно, начнутся проблемы с интеллектуальной деятельностью. Не следует также и получать углеводов сверх нормы. Могут возникнуть ряд заболеваний, такие как, сахарный диабет, лишний вес, прыщи. Лишний вес является проблемой очень многих людей. Именно умеренное число углеводов способно избежать всего этого. В этом поможет калькулятор калорийности. Углеводы следует брать из круп, макаронов твердых сортов фруктов с низким гликемическим индексом.

Теперь перейдем к жирам. Жиры - это тоже очень важный компонент, необходимый для полноценного существования организма в окружающей среде. Эти органические вещества обладают сложной молекулярной структурой, соответственно при распаде этого вещества образуется в два раза больше энергии нежели при распаде углевода или белка. Жиры называют еще липидами. Они выполняют прежде всего функцию энергообеспечения, так же, как и углеводы. Они также участвуют и в структурообразующей работе. Липиды являются главным компонентом мембраны клеток организма. В теле человека происходит накопление жировой ткани как раз и за счет жиров. Откладывание происходит в жировые клетки. В очень сложный период для организма, к примеру нехватка пищи, эти запасы непременно идут в расход, и за счет этого ущерб организму минимизируется. Жиры бывают двух типов. Первый тип - это животный жир. Источниками таких жиров являются пища животного происхождения. Например, сливочное масло, мясо, рыбий жир. При комнатной температуре, как правило, имеют твердую форму. Второй тип жиров - это растительные жиры, или, как их еще называют, растительные масла. Как правило, имеют жидкую консистенцию. Находятся растительные жиры в продуктах растительного происхождения. К примеру, оливковое масло, пальмовое масло, растительное масло.

Как уже было сказано выше, много людей страдает лишним весом. Многие борются с этой неэстетичной проблемой. Очень часто при попытках сбросить лишний вес совершается очень глупая ошибка - полное исключение жиров из своего рациона. Это категорически неправильно. Дело в том, что жиры являются частью процессов, протекающих внутри организма, и их недостаток может привести к серьёзным проблемам. Вот только некоторые из них: проблемы со психикой (агрессивность), упадок сил, нарушение зрительной системы, дефицит массы тела. Чтобы избежать всех этих проблем, необходимо находить правильный количественный баланс ЖБУ и калорийности. Разработанное приложение отлично справляется с данной задачей. Всего-то вводится значения БЖУ, указанных на упаковке продукта, на 100 грамм. А далее считается все самостоятельно, относительно выбранной массы. Переизбыток жиров в рационе также несет ряд негативных последствий. К ним можно отнести образование тромбов, из-за того, что кровь густеет, эти тромбы являются причиной закупорки сосудистой системы человека, что приводит к таким печальным последствиям, как инфаркт, инсульт. При повышении нормы потребления жиров также увеличивается риск сахарного диабета.

Для того чтобы узнать сколько необходимо употреблять БЖУ следует обратиться к специалистам в области питания, например, к диетологам. Также имеются расчетные калькуляторы нормы БЖУ. 1 грамм белка дает 4 килокалории. 1 грамм углевода дает тоже 4 килокалории. 1 грамм жира образует аж 9 килокалорий.

Отличительной особенностью данного ПО является оригинальный дизайн, понятный интерфейс. В некоторых подобных приложениях имеется ограничение на количество приемов пищи. В этом же решении ограничений не имеется.

## 1.2 Описание инструментариев.

При создании приложения была использована среда создания программных средств Delphi.

Delphi - это язык программирования высоко уровня, основной задачей которого является создание прикладного программного обеспечения. Появился он в 1995 году. Создателем этого языка является Андерс Хейлсберг. Delphi в некоторых отношениях более прост в использовании, по сравнению с C++, C#. Он является идеальным стартом для любого программиста.

Интерфейс Delphi представляет собой окно объектов. В нем располагаются основные объекты программ. Слева находится инспектор объектов, в котором задаются основные свойства объектов, а также определяется какое-либо событие, например, OnClick - нажатие. Чуть выше инспектора объектов находится дерево объектов. Оно отображает какие объекты были использованы при создании программных решений. Центральную часть занимает форма Form, определяющая внешний вид программы. Она содержит использованные компоненты. Также рядом с центральной формой располагается окно программного кода. В этом месте и происходит основная работа программиста. Столько пота было пролито в этом месте.

В данном приложении были использованы следующие компоненты:

1. Form
2. sBitBtn
3. sOpenDialog
4. sPanel
5. sSaveDialog
6. sSkinManager
7. sTreeView
8. Label
9. sLabelFX
10. InternalSkins
11. TsStoredSkin

Form. Форма - это основной и очень важный компонент этой среды. На форме размещаются все остальные компоненты. При запуске программы именно форма отображается первой. Она видимая часть приложения. В программе может размещаться несколько форм, но при этом первая форма главная, а остальные уже будут считаться дополнительными. Приложение с несколькими формами будет считаться уже многооконной программой. Формы бывают двух разновидностей: модальные и немодальные. Модальные формы запрещают обращение к остальным формам программы, т.е. необходимо закрытие для обращения к другим формам. Немодальные же формы наоборот позволяют пользователю взаимодействовать со всеми формами одновременно.

sBitBtn. Это кнопка. Приставка “s” означает, что использовались дополнительные библиотеки, а именно AlphaSkins. При нажатии на кнопку, которая располагается на форме, запускается какая-либо процедура, описанная программистом на окне кода. Имеется еще кнопка Button. BitBtn отличается тем, что на ней можно разместить пиктографическое изображение, а также имеется возможность более разнообразного дизайна. Также имеется свойство Kind, которое определяет стандарт кнопки, например, выход, помощь, ОК. Также есть много других важных свойств.

sOpenDialog. Он представляет собой диалоговое окно с пользователем выбора нужного файла. OpenDialog обладает рядом полезных свойств. Он несет в себе имя файла, выбранного для открытия. Можно задавать размер окна, название окна, менять дизайн.

sPanel. Этот компонент представляет собой панельку, у которой имеется возможность содержать некоторые другие компоненты. Она упрощает работу программиста. Например, имеется ряд подобных кнопок, и после размещения их на панели предоставляется возможность менять некоторые важные свойства всех сразу. Не надо менять свойства кнопок по отдельности. Это экономит время. Имеются свойства расположения, выравнивания. Можно задавать ей необходимые размеры. Можно создавать очень интересные дизайнерские решения.

sSaveDialog. Этот компонент подобен sOpenDialog. Только в первом случае происходит поиск файла для открытия, тут же осуществляется поиск файла для сохранения. Точно также открывается красивое окно диалога. На нем, располагается ряд кнопок управления, а также структура файлов персонального компьютера пользователя. Размеры окна можно редактировать, также можно изменять стиль окна. В нашем случае благодаря подключении библиотеки AlphaSkin окно становится более симпатичным. Имеется большой выбор дизайнерских решений. При наведении курсора на кнопку диалогового окна, кнопка загорается красивым цветом, что несомненно должно радовать рядового пользователя.

sSkinManager. Этот компонент принадлежит как раз установленной дополнительной библиотеке. Главной его задачей является создания красивого стиля программного приложения. Он позволяет в считанные секунды менять внешний вид программного обеспечения. Имеется ряд очень важных свойств.

sTreeView. Данный компонент является практически самым важным компонентом данного программного приложения. Практически весь программный код связан именно с объектом дерева. Прием пищи, продукт, вся основная информация содержится в ветвях дерева. Также именно к ветвям дерева присоединялись данные - указатель на список. В этом указатели хранится основная информация БЖУ. Очень разумно использовать данный компонент в этой работе. Дело в том, что появляется возможность легко обходить все данные, самые необходимые функции и процедуры уже готовы. Поле sTreeView занимает практически всю рабочую часть программы. Была использована возможность режима “ReadOnly”. После него узлы дерева не должны быть редактированы. Если потребуется редактировать какой-либо элемент, то настройка происходит в программном окне базы данных продуктов. После того как откорректирован какой-либо продукт, поправка происходит во всем дереве.

Так как, этот компонент важен, следует рассмотреть его некоторый функционал, который использовался при создании счетчика калорийности пищи. Функция “Add” позволяет добавить новый узел. При этом добавление происходит в конец уровня. Функция “AddObject” добавляет данные в узел. После создания ячейки памяти и после того как ей присвоили конкретные значения, вся информация записывается в узел дерева. Также имеется ряд подобных функций вставки, но область, где появится новый узел будет разной во всех случаях. Процедура “Delete” позволяет убрать ненужный узел из дерева. После того как это произошло, происходит считывание дерева снова, и данные обновляются. Для удобства была использована процедура “FullCollapse”. После нажатия на соответствующую кнопку, все продукты, входящие в приемы пищи, сворачиваются. Противоположенная процедура - “FullExpand”. Она позволяет раскрыть узлы продуктов.

Label. Данный компонент представляет собой метку. Этот объект очень полезен. С помощью него происходит отображение конечной информации. Свойство “Caption” как раз и содержало эту информацию. Размер метки можно легко редактировать, соответственно и размер текста тоже. Дизайнерские работы ведутся в свойстве Font. Там можно выбирать тип шрифта, его размер, его цвет и т.д.

sLabelFX. Этот объект является частным случаем label. Он использовался так, как обладает большим количеством настройки дизайна. Метка данного типа идет в комплекте с AiphaSkin.

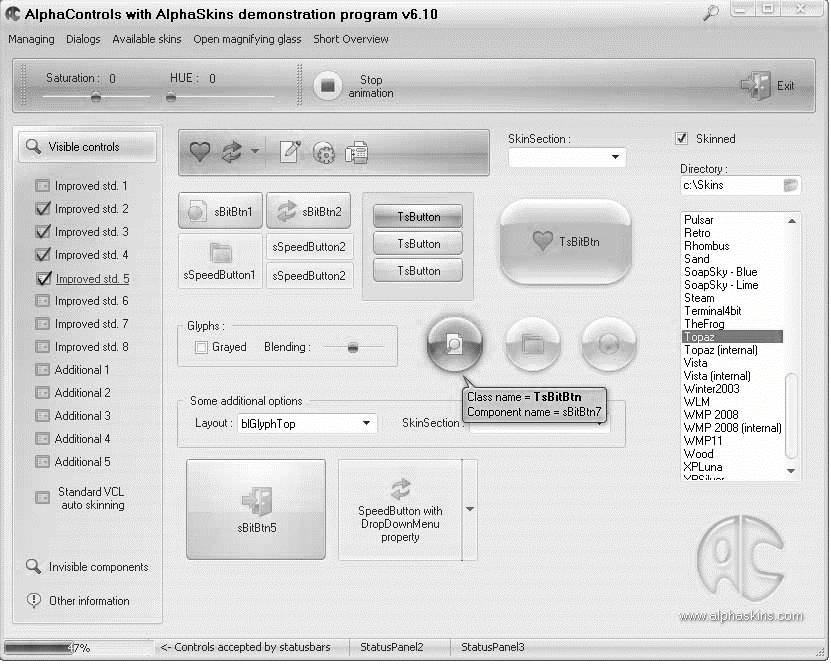
InternalSkins и TsStoredSkin. Эти два компонента связаны с этой библиотекой. Следует уделить пару слов данному нововведению. AlphaControls представляет собой пакет со стандартными и уникальными компонентами. Как говорилось выше, появляется больше возможностей по созданию внешнего вида программного обеспечения. Хороший дизайн программы - залог успеха у потребителей. В отличии от стандартного решения можно применять градиентные настройки, эффекты затуманивания, отражения, даже появляется возможность небольших анимаций, и это далеко не весь набор преимуществ.

Рисунок 1.1-Пример дизайна программы

1.3 Общее резюме.

И так, это приложение должно помочь людям в реализации правильного питания. Теперь не придется все пересчитывать с самого начала из-за глупых ошибок. Очень сложные вычислительные процессы программа берет на себя. Приятный и понятный интерфейс помогут в создании идеального рациона пищи.

Еще раз хотелось бы отметить, что много людей ошибаются, переходя на рацион “и так сойдет”. Недостаток какого-либо элемента или же его переизбыток может негативно сказаться на здоровье человека.

Все действует согласно физике. Закон сохранения энергии не взят с потолка. Если в тело человека поступает больше энергии, то он начнет полнеть. Так заложено биологической программой, дабы увеличить шансы выживания в окружающей среде. Если имеется дефицит энергии, то тело начинает убавлять в весе.

Чтобы найти правильный баланс нужно считать. С этим-то и справляется калькулятор калорийности пищевых продуктов.

База данных, рассчитанная на очень большое число продуктов, центральное поле рациона, которое может содержать также очень много продуктов и приемов пищи, возможность сохранения результатов - всё это поможет в этом нелегком деле.

# ГЛАВА 2. ПРАКТИЧЕСКИЕ ОСОБЕННОСТИ РАЗРАБОТКИ.

## 2.1 Проектирование продукта.

Проектирование проекта очень важная часть создания программного продукта. В данном разделе будут описаны блоки, будет разобрана общая схема работы продукта.

Данное приложение можно подразделить на пять больших блоков. Эти блоки связаны между собой.

Вот эти блоки:

1. Блок центрального поля
2. Блок клавишного управления
3. Блок базы данных
4. Блок управления базой данных
5. Блок выбора продукта

О каждом блоке следует рассказать более подробно.

Блок центрального поля представляет собой основное окно вывода рациона пользователя. Как уже было сказано выше в нем отображается основная информация о приемах пищи и о выбранных продуктах.

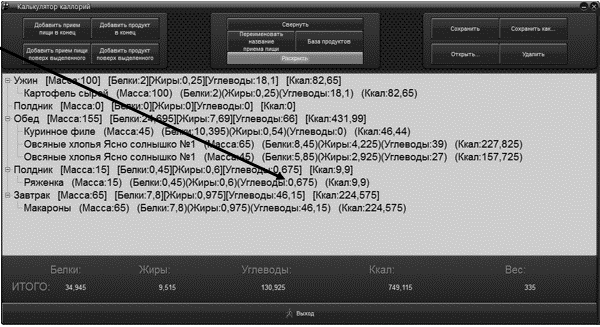
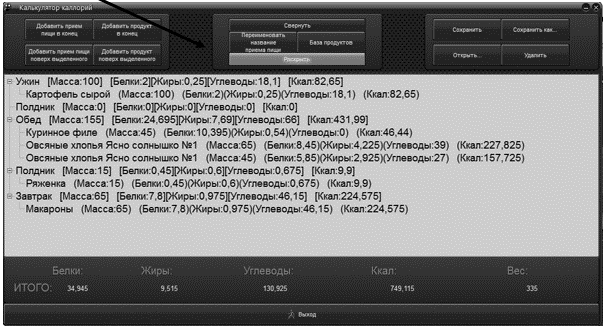
Как мы видим на рисунке 2.1, программа показывает прием пищи, продукты, и суммарное значение всех параметров.

Рисунок 2.1 - Блок центрального поля

Например, возьмем прием пищи под названием “Обед”. Масса всего приема пищи составляет 155 грамм. Суммарное число белков 10,395 грамм. Жиров порядка 0,54 грамма, а углеводов 66 грамм. Посчитав число калорий, которые образуются в результате распада жиров, белков, углеводов, мы получим 431,99, как и показано выше на рисунке. Далее в обед входит 5 продуктов, а именно: куриное филе, масса которой составляет 45 грамм и две порции хлопьев “Ясно солнышко” по 65 и 45 грамм соответственно.

Вторым блоком является блок клавишного управления. Он располагается чуть выше первого блока и представляет собой панель с клавишами для управления создания рациона питания.

Рисунок 2.2 - Блок клавишного управления

Как мы видим на рисунке 2.2, данный блок содержит 12 кнопок управления. 13 кнопка находится снизу.

Названия кнопок управления:

1. Добавить прием пищи в конец
2. Добавить прием пищи поверх выделенного
3. Добавить продукт в конец
4. Добавить продукт поверх выделенного
5. Свернуть
6. Переименовать название приема пищи
7. База продуктов
8. Развернуть
9. Сохранить
10. Сохранить как…
11. Открыть…
12. Удалить
13. Выход

Все эти кнопки были рассмотрены в предыдущей главе и поэтому описывать их снова не имеет смысла.

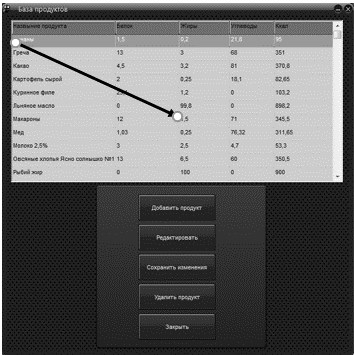
Третий блок - это блок базы данных. В базе данных находятся все продукты, которые внесены пользователем приложения. Более того отображаются параметры на 100 грамм массы. Об этом тоже уже было сказано. Блок обозначен на рисунке 2.3.

Рисунок 2.3 - Блок базы данных

Сейчас у нас выбран продукт под названием “бананы”. В 100 грамм бананов содержится 1,5 грамм белка, 0,2 грамма жира, 21,8 грамм углеводов. В сумме это все даёт 95 килокалорий. Довольно неплохой продукт для набора мышечной массы человека.

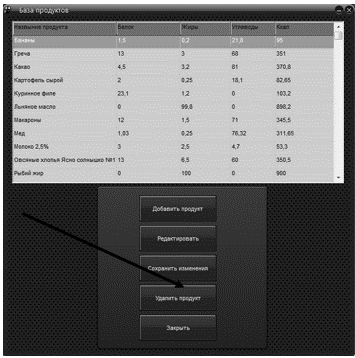
Блок управления базой данных содержит в себе кнопки управления базой продуктов. Он предоставлен на рисунке 2.4.

Рисунок 2.4 - Блок управления базой данных

Блок управления содержит следующие кнопки:

1. Добавить продукт
2. Редактировать
3. Сохранить изменения
4. Удалить продукт
5. Закрыть

Описание всех этих элементов уже имеется.

Перейдём к последнему модулю. Он представляет собой блок выбора продуктов. Описание процесса добавление продукта уже было рассмотрено. После нажатия необходимых кнопок идет процесс открытия этого блока и осуществляется выбор нужного продукта.

Он содержит кнопку “Выбрать продукт”. Также размещено несколько полей вывода информации, а именно:

1. Наименование
2. Белки
3. Жиры
4. Углеводы
5. Ккал

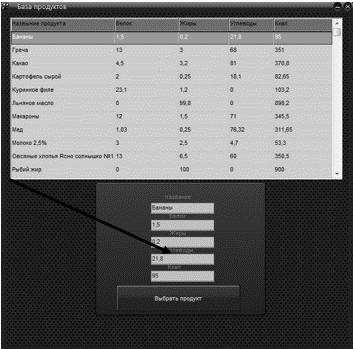
Вот рисунок данного интерфейса:

Рисунок 2.5 - Блок выбора продуктов

Следует обсудить общую схему работы продукта. И так, после включения программы с рабочего стола windows открывается блок центрального поля, на котором пользователь получает основную информацию своего рациона питания. Также взору предоставляется блок клавишного управления, благодаря которому осуществляется построение рациона. При нажатии на кнопку “База данных” открывается блок базы данных. В нем пользователь получает информацию о задействованных продуктах, также предоставляется возможность изменения составляющих этой базы.

Далее после закрытия блока базы данных, пользователь производит добавления продукта нажатием соответствующей клавиши на панели. Открывается блок выбора продукта. Там происходит отображение КБЖУ выбранного продукта. Перед тем как выбрать продукт, программой предлагается ввести его значение массы.

## 2.2 Разработка ПО.

Теперь следует обсудить очень важную часть работы, а именно создание программного обеспечения. Код написан на языке паскаль. Данный язык программирования очень популярен. Его изучают в старших классах школы. Также не пренебрегают им и на первых курсах университета. Основой среды Delphi является именно язык Pascal.

Ниже будет представлен код самой программы, а также будут даны некоторые пояснения реализации алгоритмов работы программного обеспечения.

unit Unit1kur;

Данная строка представляет собой заголовок. После того, как пользователь изменит название, оно автоматически поменяется и в разделе, представленном выше.

interface

Этот строка подключает модулю из других проектов.

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ComCtrls, StdCtrls, Buttons, Unit1kur1pr, sSkinManager,

sTreeView, sBitBtn, sLabel, ExtCtrls, sPanel, Grids, sButton, sDialogs;

Это раздел подключенных модулей. Их тут 22.

type

Harr=^BZHU;

BZHU=record

Massa:real;

Bel:real;

Zhir:real;

Ugl:real;

Kkal:real;

end;

TForm1 = class(TForm)

Label3: TLabel;

Label8: TLabel;

Label9: TLabel;

Label10: TLabel;

Label11: TLabel;

sSkinManager1: TsSkinManager;

sBitBtn1: TsBitBtn;

sBitBtn2: TsBitBtn;

TreeView1: TsTreeView;

sLabelFX1: TsLabelFX;

sBitBtn3: TsBitBtn;

sBitBtn4: TsBitBtn;

sBitBtn5: TsBitBtn;

sBitBtn6: TsBitBtn;

sBitBtn7: TsBitBtn;

sLabelFX2: TsLabelFX;

sLabelFX3: TsLabelFX;

sLabelFX4: TsLabelFX;

sLabelFX5: TsLabelFX;

sLabelFX6: TsLabelFX;

sBitBtn8: TsBitBtn;

sPanel1: TsPanel;

sPanel2: TsPanel;

sBitBtn9: TsBitBtn;

sBitBtn10: TsBitBtn;

sPanel3: TsPanel;

sPanel4: TsPanel;

sBitBtn11: TsBitBtn;

sBitBtn12: TsBitBtn;

sBitBtn13: TsBitBtn;

sSaveDialog1: TsSaveDialog;

sOpenDialog1: TsOpenDialog;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure vnecenie(var strGlav:string; var Har:Harr);

procedure podschitivanie;

procedure Itogo;

procedure BitBtn5Click(Sender: TObject);

procedure BitBtn6Click(Sender: TObject);

procedure BitBtn7Click(Sender: TObject);

procedure sBitBtn8Click(Sender: TObject);

procedure sBitBtn11Click(Sender: TObject);

procedure sBitBtn10Click(Sender: TObject);

procedure sButton1Click(Sender: TObject);

procedure sBitBtn12Click(Sender: TObject);

procedure sBitBtn13Click(Sender: TObject);

В разделе, представленном выше дается названия процедурам, которые буду использоваться приложением, также объявляются входные и выходные параметры.

private

{ Private declarations }

public

{ Public declarations }

end;

В данном случае идет описание частных и общих объявлений. Заканчивается всё зарезервированным словом end.

var

Form1: TForm1;

Har:Harr;

strGlav:string;

fail:string;

Это раздел описания данных. Эти данные или переменные будут использоваться алгоритмами программы. Они являются глобальными, то есть известны абсолютно всем процедурам и функциям.

implementation

{$R \*.dfm}

Это основная описательная часть процедур. Здесь будут написаны процедуры и функции программного продукта.

procedure TForm1.Button1Click(Sender: TObject); {прием пищи}

var

str2:string;

i1:integer;

label 3;

begin

If TreeView1.Selected=nil then

Begin

ShowMessage('Узел не выбран');

exit;

End;

3:

If TreeView1.Selected.Level=0 then

Begin

str2:=InputBox('Прием пищи', 'Пожалуйста, введите новое название приема пищи', '');

If str2='' then

Begin

ShowMessage('Название не введено');

Exit;

End;

i1:=0;

Repeat

inc(i1);

If (str2[i1]='[') or (str2[i1]=']') then

Begin

ShowMessage('Название не должно содержать: [ ]');

GoTo 3

End;

Until i1=length(str2);

str2:=str2+' [Масса:0] [Белки:0][Жиры:0][Углеводы:0] [Ккал:0]';

Treeview1.Items.Insert(Treeview1.Selected,str2);

End;

If TreeView1.Selected.Level<>0 then

Begin

str2:=InputBox('Прием пищи', 'Пожалуйста, введите новое название приема пищи', '');

If str2='' then

Begin

ShowMessage('Название не введено');

Exit;

End;

i1:=0;

Repeat

inc(i1);

If (str2[i1]='[') or (str2[i1]=']') then

Begin

ShowMessage('Название не должно содержать: [ ]');

GoTo 3

End;

Until i1=length(str2);

str2:=str2+' [Масса:0] [Белки:0][Жиры:0][Углеводы:0] [Ккал:0]';

Treeview1.Items.AddFirst(Treeview1.Items.Item[0],str2);

End;

podschitivanie;

end;

Здесь представлена реализация кнопки “Добавить прием пищи поверх выделенного”. Сначала объявляется защита от несанкционированных данных. То есть, если узел не будет выбран или в названии будут использованы неправильные символы. Далее, после того, как имя введено верно, оно присваивается узлу дерева. После изменения, произведенных в дереве, применяется процедура podschitivanie, которая считает суммарные значения КБЖУ.

procedure TForm1.Button2Click(Sender: TObject); {добавить блюдо}

begin

If TreeView1.Selected=nil then

Begin

ShowMessage('Узел не выбран');

exit;

End;

If TreeView1.Selected.Level=0 then

Begin

new(Har);

strGlav:='';

vnecenie(strGlav,Har);

If strGlav='' then

Begin

Dispose(Har);

Exit

End;

Treeview1.Items.AddChildObject(Treeview1.Selected,strGlav,Har);

End;

If (TreeView1.Selected.Level<>0) then

Begin

new(Har);

strGlav:='';

vnecenie(strGlav,Har);

If strGlav='' then

Begin

Dispose(Har);

Exit

End;

Treeview1.Items.InsertObject(Treeview1.Selected,strGlav,Har);

End;

podschitivanie;

end;

В коде представленном выше, происходит программирование кнопки “Добавить продукт поверх выделенного”. Вначале создается защита от неправильных данных, далее создается новый узел дерева с помощью процедуры vnecenie. Она создает новый указатель и формирует строку-название. Далее идет процедура подсчета итого.

procedure TForm1.Button3Click(Sender: TObject); {переименовать прием пищи}

var i1:integer;

str2:string;

Label 3;

begin

If TreeView1.Selected=nil then

Begin

ShowMessage('Узел не выбран');

exit;

End;

3:

If TreeView1.Selected.Level=0 then

Begin

str2:=InputBox('Прием пищи', 'Пожалуйста, введите новое название приема пищи', '');

If str2='' then

Begin

ShowMessage('Название не введено');

Exit;

End;

i1:=0;

Repeat

inc(i1);

If (str2[i1]='[') or (str2[i1]=']') then

Begin

ShowMessage('Название не должно содержать: [ ]');

GoTo 3;

End;

Until i1=length(str2);

str2:=str2+' [Масса:0] [Белки:0][Жиры:0][Углеводы:0] [Ккал:0]';

Treeview1.Selected.Text:=str2;

End

Else

ShowMessage('Выберете нужный прием пищи');

podschitivanie;

end;

Таким образом создана кнопка “Переименовать прием пищи”. Если у пользователя возникнет желание дать другое название, то он воспользуется данной кнопкой. Реализована защита от неверных данных, также от неверно выбранного узла дерева.

procedure TForm1.FormCreate(Sender: TObject);{если приемов пищи нет}

begin

Treeview1.Selected:=Treeview1.Items.Item[0];

If Treeview1.Items.Count=0 then

Treeview1.Items.Insert(nil,InputBox('Прием пищи', 'Пожалуйста, введите новое название приема пищи', ''));

Itogo;

end;

При создании формы необходимы начальные условия. Их реализация написана выше.

procedure TForm1.BitBtn1Click(Sender: TObject); {включение новой формы}

begin

Form2.sPanel1.Visible:=true;

Form2.sPanel2.Visible:=false;

Form2.ShowModal;

end;

Этот код описывает кнопку “База данных”. Подключается новая форма с таблицей продуктов и отключается видимость панели кнопки выбора.

procedure TForm1.vnecenie(var strGlav: string; var Har: Harr); {процедура создания блюда}

var

strBel,strZhir,strUgl,strKkal,strMassa,kk:string;

begin

ShowMessage('Выберите любое блюдо из вашего списка');

Form2.sPanel1.Visible:=false;

Form2.sPanel2.Visible:=true;

Form2.Edit1.Text:=Form2.StringGrid1.Cells[0,Form2.StringGrid1.Row];

Form2.Edit2.Text:=Form2.StringGrid1.Cells[1,Form2.StringGrid1.Row];

Form2.Edit3.Text:=Form2.StringGrid1.Cells[2,Form2.StringGrid1.Row];

Form2.Edit4.Text:=Form2.StringGrid1.Cells[3,Form2.StringGrid1.Row];

Form2.Edit5.Text:=Form2.StringGrid1.Cells[4,Form2.StringGrid1.Row];

Form2.StringGrid1.Row:=1;

Form2.ShowModal;

Form2.proverka('Масса',kk);

If kk='' then

Begin

Exit;

End;

Har^.Massa:=StrToFloat(kk);

Har^.Bel:=Har^.Massa\*VibrBel/100;

Har^.Zhir:=Har^.Massa\*VibrZhir/100;

Har^.Ugl:=Har^.Massa\*VibrUgl/100;

Har^.kkal:=4\*Har^.Bel+9\*Har^.Zhir+4\*Har^.Ugl;

strMassa:='';

strBel:='';

strZhir:='';

strUgl:='';

strKkal:='';

strGlav:='';

strMassa:=FloatToStr(Har^.Massa);

strBel:=FloatToStr(Har^.Bel);

strZhir:=FloatToStr(Har^.Zhir);

strUgl:=FloatToStr(Har^.Ugl);

strKkal:=FloatToStr(Har^.kkal);

strGlav:=VibrNaz;

strGlav:=strGlav+' (Масса:'+strMassa+')'+' (Белки:'+strBel+')'+'(Жиры:'+strZhir+')'+'(Углеводы:'+strUgl+')'+' (Ккал:'+strKkal+')';

end;

Данная процедура очень важна. Она используется в некоторых кнопках приложения.

procedure TForm1.BitBtn5Click(Sender: TObject);

begin

If TreeView1.Selected=nil then

Begin

ShowMessage('Узел не выбран');

exit;

End;

If TreeView1.Selected.Level=0 then

Begin

new(Har);

strGlav:='';

vnecenie(strGlav,Har);

If strGlav='' then

Begin

Dispose(Har);

Exit

End;

Treeview1.Items.AddChildObject(Treeview1.Selected,strGlav,Har);

End;

If (TreeView1.Selected.Level<>0) then

Begin

new(Har);

strGlav:='';

vnecenie(strGlav,Har);

If strGlav='' then

Begin

Dispose(Har);

Exit

End;

Treeview1.Items.AddObject(Treeview1.Selected,strGlav,Har);

End;

podschitivanie;

end;

Это кнопка “Добавить продукт в конец”. Подобная кнопка уже описана выше.

procedure TForm1.BitBtn6Click(Sender: TObject);

var i1:integer;

str2:string;

Label 3;

begin

If TreeView1.Selected=nil then

Begin

ShowMessage('Узел не выбран');

exit;

End;

3:

If TreeView1.Selected.Level=0 then

Begin

str2:=InputBox('Прием пищи', 'Пожалуйста, введите новое название приема пищи', '');

If str2='' then

Begin

ShowMessage('Название не введено');

Exit;

End;

i1:=0;

Repeat

inc(i1);

If (str2[i1]='[') or (str2[i1]=']') then

Begin

ShowMessage('Название не должно содержать: [ ]');

GoTo 3

End;

Until i1=length(str2);

str2:=str2+' [Масса:0] [Белки:0][Жиры:0][Углеводы:0] [Ккал:0]';

Treeview1.Items.Add(Treeview1.Selected,str2);

End;

If TreeView1.Selected.Level<>0 then

Begin

str2:=InputBox('Прием пищи', 'Пожалуйста, введите новое название приема пищи', '');

If str2='' then

Begin

ShowMessage('Название не введено');

Exit;

End;

i1:=0;

Repeat

inc(i1);

If (str2[i1]='[') or (str2[i1]=']') then

Begin

ShowMessage('Название не должно содержать: [ ]');

GoTo 3

End;

Until i1=length(str2);

str2:=str2+' [Масса:0] [Белки:0][Жиры:0][Углеводы:0] [Ккал:0]';

Treeview1.Items.Add(Treeview1.Items.Item[0],str2)

End;

podschitivanie;

end;

Так происходит добавление какого-либо приема пищи в конец. Кнопка - “Добавить прием пищи в конец”.

procedure TForm1.podschitivanie;

var

uzel,uzel1:TTreeNode;

str1:string;

i1:integer;

belok,zhiri,uglevod,kkalorii,ves:real;

ukazatelTek:Harr;

label 4;

begin

uzel:=nil;

Repeat

If uzel=nil then

uzel:=TreeView1.Items.Item[0]

Else

uzel:=uzel.GetNext;

4:

If (uzel.Count=0) and (uzel.Level=0) then

Begin

str1:=uzel.Text;

i1:=0;

Repeat

inc(i1);

Until (uzel.Text[i1+1]=' ') and (uzel.Text[i1+2]=' ') and (uzel.Text[i1+3]=' ') and (uzel.Text[i1+4]='[') and (uzel.Text[i1+5]='М');

inc(i1);

Repeat

Delete(str1,i1,1);

Until i1-1=length(str1);

uzel.Text:=str1+' [Масса:0]'+' [Белки:0]'+'[Жиры:0]'+'[Углеводы:0]'+' [Ккал:0]';

End;

If (uzel.Count<>0) and (uzel.Level=0) then

Begin

uzel1:=nil;

belok:=0;

zhiri:=0;

uglevod:=0;

kkalorii:=0;

ves:=0;

Repeat

If uzel1=nil then

uzel1:=uzel.GetNext

Else

uzel1:=uzel1.GetNext;

ukazatelTek:=uzel1.Data;

belok:=belok+ukazatelTek^.Bel;

zhiri:=zhiri+ukazatelTek^.Zhir;

uglevod:=uglevod+ukazatelTek^.Ugl;

ves:=ves+ukazatelTek^.Massa;

Until (uzel1.GetNext=nil) or (uzel1.GetNext.Level=0);

kkalorii:=belok\*4+uglevod\*4+zhiri\*9;

str1:=uzel.Text;

i1:=0;

Repeat

inc(i1);

Until (uzel.Text[i1+1]=' ') and (uzel.Text[i1+2]=' ') and (uzel.Text[i1+3]=' ') and (uzel.Text[i1+4]='[') and (uzel.Text[i1+5]='М');

inc(i1);

Repeat

Delete(str1,i1,1);

Until i1-1=length(str1);

uzel.Text:=str1+' [Масса:'+FloatToStr(ves)+']'+' [Белки:'+FloatToStr(belok)+']'+'[Жиры:'+FloatToStr(zhiri)+']'+'[Углеводы:'+FloatToStr(uglevod)+']'+' [Ккал:'+FloatToStr(kkalorii)+']';

If (uzel1.GetNext<>nil) and (uzel1.GetNext.Level=0) then

Begin

uzel:=uzel1.GetNext;

GoTo 4;

End;

End;

Until uzel.GetNext=nil;

Itogo;

end;

Это процедура, которая считывает данные указатели узла и на их основе формирует названия узлов дерева. В конце используется процедура Itogo. Она обходит все дерево и сообщает суммарный результат.

procedure TForm1.BitBtn7Click(Sender: TObject);

var

vet,vet2:TTreeNode;

pr,kol:integer;

begin

If TreeView1.Selected=nil then

Begin

ShowMessage('Узел не выбран');

exit;

End;

vet2:=TreeView1.Selected;

vet:=nil;

pr:=1;

kol:=0;

Repeat

If vet=nil then

vet:=TreeView1.Items.Item[0]

Else

vet:=vet.GetNext;

If vet.Level=0 then

inc(kol);

Until vet.GetNext=nil;

If (kol<=1) and (vet2.Level=0) then

pr:=0;

If (TreeView1.Items.Count>1) and (pr=1) then

Begin

TreeView1.Selected.Delete;

podschitivanie;

End

Else

ShowMessage(‘Должен оставаться хотя бы один прием пищи');

end;

Здесь описывается процедура удаления узла.

procedure TForm1.Itogo;

var

uz:TTreeNode;

Ibel,Izhir,Iug,Ikkal,Ivec:real;

x:Harr;

begin

uz:=nil;

Ibel:=0;

Izhir:=0;

Iug:=0;

Ikkal:=0;

Ivec:=0;

Repeat

If uz=nil then

uz:=TreeView1.Items.Item[0]

Else

uz:=uz.GetNext;

If (uz.Level<>0) and (uz.Count=0) then

Begin

x:=uz.Data;

Ibel:=Ibel+x^.Bel;

Izhir:=Izhir+x^.Zhir;

Iug:=Iug+x^.Ugl;

Ivec:=Ivec+x^.Massa;

End;

Until uz.GetNext=nil;

Ikkal:=Ibel\*4+Iug\*4+Izhir\*9;

Label3.Caption:=FloatToStr(Ibel);

Label8.Caption:=FloatToStr(Izhir);

Label9.Caption:=FloatToStr(Iug);

Label10.Caption:=FloatToStr(Ikkal);

Label11.Caption:=FloatToStr(Ivec);

end;

Это процедура обхода всего дерева. О ней уже говорилось выше. Формируются названия меток.

procedure TForm1.sBitBtn8Click(Sender: TObject);

begin

Close;

end;

Кнопка выхода из программы.

procedure TForm1.sBitBtn11Click(Sender: TObject);

begin

If fail<>'' then

TreeView1.SaveToFile(fail)

Else

If sSaveDialog1.Execute then

Begin

TreeView1.SaveToFile(sSaveDialog1.FileName);

fail:=sSaveDialog1.FileName;

End;

end;

Это диалог сохранения всех изменений, которые были внесены в дерево. Сохраняется всё в отдельный файл.

procedure TForm1.sBitBtn10Click(Sender: TObject);

var ch1:TTreeNode;

nazvanie1,substr:string;

qq:integer;

r:Harr;

begin

If sOpenDialog1.Execute then

Begin

TreeView1.LoadFromFile(sOpenDialog1.FileName);

End;

ch1:=nil;

Repeat

If ch1=nil then

ch1:=TreeView1.Items.Item[0]

Else

ch1:=ch1.GetNext;

If (ch1.Level<>0) then

Begin

nazvanie1:=ch1.Text;

substr:='';

qq:=pos('(Масса:',nazvanie1);

qq:=qq+7;

Repeat

substr:=substr+nazvanie1[qq];

inc(qq);

Until nazvanie1[qq]=')';

new(r);

r^.Massa:=StrToFloat(substr);

substr:='';

qq:=pos('(Белки:',nazvanie1);

qq:=qq+7;

Repeat

substr:=substr+nazvanie1[qq];

inc(qq);

Until nazvanie1[qq]=')';

r^.Bel:=StrToFloat(substr);

substr:='';

qq:=pos('(Жиры:',nazvanie1);

qq:=qq+6;

Repeat

substr:=substr+nazvanie1[qq];

inc(qq);

Until nazvanie1[qq]=')';

r^.Zhir:=StrToFloat(substr);

substr:='';

qq:=pos('(Углеводы:',nazvanie1);

qq:=qq+10;

Repeat

substr:=substr+nazvanie1[qq];

inc(qq);

Until nazvanie1[qq]=')';

r^.Ugl:=StrToFloat(substr);

r^.Kkal:=r^.Bel\*4+r^.Zhir\*9+r^.Ugl\*4;

ch1.Data:=r;

End;

Until ch1.GetNext=nil;

podschitivanie;

end;

Здесь описывается алгоритм открытия сохраненного дерева. Происходит формирование дерева с нуля.

procedure TForm1.sButton1Click(Sender: TObject);

begin

TreeView1.FullExpand;

end;

procedure TForm1.sBitBtn12Click(Sender: TObject);

begin

TreeView1.FullCollapse;

end;

Эти две процедуры раскрывают узлы дерева и закрывают их. Нетрудно догадаться названий соответствующих кнопок.

procedure TForm1.sBitBtn13Click(Sender: TObject);

begin

If sSaveDialog1.Execute then

Begin

TreeView1.SaveToFile(sSaveDialog1.FileName);

fail:=sSaveDialog1.FileName;

End;

end;

Это диалог сохранения файла.

Весь описанный код выше относится к главной форме приложения. Еще имеется вторая форма базы данных продуктов. Код будет расписан ниже.

unit Unit1kur1pr;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Grids, StdCtrls, Buttons, sBitBtn, sEdit, ExtCtrls, sPanel,

sLabel, sButton, ComCtrls;

type

TMyGrid=class(TCustomGrid);

TForm2 = class(TForm)

StringGrid1: TStringGrid;

Memo1: TMemo;

Edit1: TsEdit;

Edit2: TsEdit;

Edit3: TsEdit;

Edit4: TsEdit;

Edit5: TsEdit;

BitBtn1: TsBitBtn;

BitBtn2: TsBitBtn;

BitBtn3: TsBitBtn;

BitBtn4: TsBitBtn;

sPanel1: TsPanel;

sLabelFX2: TsLabelFX;

sLabelFX3: TsLabelFX;

sLabelFX4: TsLabelFX;

sLabelFX5: TsLabelFX;

sPanel2: TsPanel;

sLabelFX1: TsLabelFX;

sBitBtn1: TsBitBtn;

sBitBtn2: TsBitBtn;

procedure FormCreate(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

procedure proverka(q:string; var dannie:string);

procedure BitBtn4Click(Sender: TObject);

procedure DeleteARow(Grid: TStringGrid; ARow: Integer);

procedure StringGrid1SelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);

procedure BitBtn6Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure sBitBtn1Click(Sender: TObject);

procedure sBitBtn2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2: TForm2;

ss,VibrNaz:string;

VibrBel,VibrZhir,VibrUgl:real;

implementation

uses Unit1kur;

{$R \*.dfm}

Данный фрагмент кода уже рассмотрен. Комментировать его снова нет смысла.

procedure TForm2.FormCreate(Sender: TObject); {названия столбцов базы продуктов}

var

i:integer;

begin

Memo1.Lines.LoadFromFile('таблица.txt');

StringGrid1.Cells[0,0]:='Назвыние продукта';

StringGrid1.Cells[1,0]:='Белок';

StringGrid1.Cells[2,0]:='Жиры';

StringGrid1.Cells[3,0]:='Углеводы';

StringGrid1.Cells[4,0]:='Ккал';

For i:=0 to Memo1.Lines.Count-1 do

try

If i>StringGrid1.RowCount-1 then

StringGrid1.RowCount:=i+1;

StringGrid1.Cells[0,i+1]:=Copy(Memo1.Lines.Strings[i],1,pos('|',Memo1.Lines.Strings[i])-1);

StringGrid1.Cells[1,i+1]:=Copy(Memo1.Lines.Strings[i],pos('|',Memo1.Lines.Strings[i])+1,pos('&',Memo1.Lines.Strings[i])-pos('|',Memo1.Lines.Strings[i])-1);

StringGrid1.Cells[2,i+1]:=Copy(Memo1.Lines.Strings[i],pos('&',Memo1.Lines.Strings[i])+1,pos('^',Memo1.Lines.Strings[i])-pos('&',Memo1.Lines.Strings[i])-1);

StringGrid1.Cells[3,i+1]:=Copy(Memo1.Lines.Strings[i],pos('^',Memo1.Lines.Strings[i])+1,pos('#',Memo1.Lines.Strings[i])-pos('^',Memo1.Lines.Strings[i])-1);

StringGrid1.Cells[4,i+1]:=Copy(Memo1.Lines.Strings[i],pos('#',Memo1.Lines.Strings[i])+1,Length(Memo1.Lines.Strings[i])-pos('#',Memo1.Lines.Strings[i]));

except;

end;

Edit1.Text:=StringGrid1.Cells[0,1];

Edit2.Text:=StringGrid1.Cells[1,1];

Edit3.Text:=StringGrid1.Cells[2,1];

Edit4.Text:=StringGrid1.Cells[3,1];

Edit5.Text:=StringGrid1.Cells[4,1];

StringGrid1.RowCount:=Memo1.Lines.Count+1;

end;

Когда форма подключается формируется новая таблица, в которой находятся основные продукты пользователя.

procedure TForm2.BitBtn2Click(Sender: TObject); {присваивание ячейкам}

var

nom,i1,ppp:integer;

Label 2;

begin

ppp:=0;

StringGrid1.RowCount:=StringGrid1.RowCount+1;

StringGrid1.Cells[0,StringGrid1.RowCount-1]:='';

StringGrid1.Cells[1,StringGrid1.RowCount-1]:='';

StringGrid1.Cells[2,StringGrid1.RowCount-1]:='';

StringGrid1.Cells[3,StringGrid1.RowCount-1]:='';

StringGrid1.Cells[4,StringGrid1.RowCount-1]:='';

nom:=StringGrid1.RowCount-1;

2:

StringGrid1.Cells[0,nom]:=InputBox('Название продукта', 'Пожалуйста, введите название продукта', '');

For i1:=1 to Length(StringGrid1.Cells[0,nom]) do

If (StringGrid1.Cells[0,nom][i1] ='|') or (StringGrid1.Cells[0,nom][i1] ='&')

or (StringGrid1.Cells[0,nom][i1] ='^')or (StringGrid1.Cells[0,nom][i1] ='#') then

Begin

StringGrid1.Cells[0,nom]:='';

ShowMessage('В названии продукта не должно быть символов: & ^ # | ');

GoTo 2;

End;

If StringGrid1.Cells[0,nom]='' then

Begin

ShowMessage('Ничего не введено');

DeleteARow(StringGrid1,StringGrid1.RowCount);

Exit;

End;

For i1:=1 to nom-1 do

Begin

If StringGrid1.Cells[0,i1]=StringGrid1.Cells[0,nom] then

Begin

StringGrid1.Cells[0,nom]:='';

ShowMessage('Уже есть такой продукт');

GoTo 2;

End;

End;

ss:='';

proverka('Белок',ss);

If ss='' then

Begin

ppp:=1;

DeleteARow(StringGrid1,StringGrid1.RowCount);

Exit;

End;

StringGrid1.Cells[1,nom]:=ss;

ss:='';

proverka('Жиры',ss);

If ss='' then

Begin

ppp:=1;

DeleteARow(StringGrid1,StringGrid1.RowCount);

Exit;

End;

StringGrid1.Cells[2,nom]:=ss;

ss:='';

proverka('Углеводы',ss);

If ss='' then

Begin

ppp:=1;

DeleteARow(StringGrid1,StringGrid1.RowCount);

Exit;

End;

StringGrid1.Cells[3,nom]:=ss;

StringGrid1.Cells[4,nom]:=FloatToStr(StrToFloat(StringGrid1.Cells[1,nom])\*4+StrToFloat(StringGrid1.Cells[2,nom])\*9+StrToFloat(StringGrid1.Cells[3,nom])\*4);

Memo1.Lines.Add(StringGrid1.Cells[0,nom]+'|'+StringGrid1.Cells[1,nom]+'&'+StringGrid1.Cells[2,nom]+'^'+StringGrid1.Cells[3,nom]+'#'+StringGrid1.Cells[4,nom]);

end;

Данный фрагмент кода представляет собой добавление продукта в таблицу данных.

procedure TForm2.proverka(q: string; var dannie:string);{ввод данных в таблицу}

var

i,pr,kol:integer;

s:string;

Label 1;

begin

1:

kol:=0;

pr:=1;

If q='Белок' then

s:=InputBox(q, 'Пожалуйста, укажите кол-во белка на 100 грамм продукта', '');

If q='Жиры' then

s:=InputBox(q, 'Пожалуйста, укажите кол-во жиров на 100 грамм продукта', '');

If q='Углеводы' then

s:=InputBox(q, 'Пожалуйста, укажите кол-во углеводов на 100 грамм продукта', '');

If q='Масса' then

s:=InputBox(q, 'Пожалуйста, укажите массу продукта', '');

If s='' then

Begin

ShowMessage('Ничего не введено');

Exit;

End;

For i:=1 to length(s) do

Begin

If (s[i]<>'0') and (s[i]<>'1') and (s[i]<>'2') and

(s[i]<>'3') and (s[i]<>'4') and (s[i]<>'5') and

(s[i]<>'6') and (s[i]<>'7') and (s[i]<>'8') and

(s[i]<>'9') and (s[i]<>',') then

pr:=0;

End;

For i:=1 to length(s) do

If s[i]=',' then

kol:=kol+1;

If kol>=2 then

pr:=0;

If s[Length(s)]=',' then

pr:=0;

If s[1]=',' then

pr:=0;

If s='' then

pr:=0;

If pr=0 then

Begin

ShowMessage('Ввод неверных данных');

GoTo 1;

End;

dannie:=s;

end;

Этим фрагмент кода используется для защиты от неккоректоного ввода данных.

procedure TForm2.BitBtn4Click(Sender: TObject);

var list1,coplist1:TTreeNode;

staroenazdereva1,staroenaz1:string;

i112:integer;

begin

staroenaz1:=StringGrid1.Cells[0,StringGrid1.Row];

If StringGrid1.RowCount>2 then

Begin

DeleteARow(StringGrid1,StringGrid1.Row);

with StringGrid1 do

begin

Row := 1;

Col := 1;

end;

End

Else

ShowMessage('В таблице должно быть хотя бы 1 продукт');

list1:=nil;

Repeat

If list1=nil then

list1:=Form1.TreeView1.Items.Item[0]

Else

list1:=list1.GetNext;

If list1.Level<>0 then

Begin

staroenazdereva1:=list1.Text;

i112:=0;

Repeat

inc(i112);

Until (staroenazdereva1[i112+1]=' ') and (staroenazdereva1[i112+2]=' ') and (staroenazdereva1[i112+3]=' ') and (staroenazdereva1[i112+4]='(') and (staroenazdereva1[i112+5]='М');

inc(i112);

Repeat

Delete(staroenazdereva1,i112,1);

Until i112-1>=length(staroenazdereva1);

If (staroenazdereva1=staroenaz1) then

Begin

copList1:=List1.GetPrev;

List1.Delete;

List1:=copList1;

End;

End;

Until list1.GetNext=nil;

Form1.podschitivanie;

end;

Здесь описывается процедура удаления ячейки продукта.

procedure TForm2.DeleteARow(Grid: TStringGrid; ARow: Integer);

begin

Memo1.Lines.Delete(StringGrid1.Row-1);

TMyGrid(Grid).DeleteRow(ARow);

end;

procedure TForm2.StringGrid1SelectCell(Sender: TObject; ACol,

ARow: Integer; var CanSelect: Boolean);

begin

Edit1.Text:=StringGrid1.Cells[0,ARow];

Edit2.Text:=StringGrid1.Cells[1,ARow];

Edit3.Text:=StringGrid1.Cells[2,ARow];

Edit4.Text:=StringGrid1.Cells[3,ARow];

Edit5.Text:=StringGrid1.Cells[4,ARow];

end;

Процедура удаления кода обращается к фрагменту кода представленному выше.

procedure TForm2.BitBtn6Click(Sender: TObject);

var

i:integer;

begin

Memo1.Lines.SaveToFile('таблица.txt');

for i:=0 to 100000 do

StringGrid1.Rows[i].Clear;

for i:=0 to 100000 do

StringGrid1.Cols[i].Clear;

FormCreate(Sender);

end;

Здесь описывается сохранения после внесенных изменений.

procedure TForm2.BitBtn1Click(Sender: TObject);

begin

VibrNaz:=Edit1.Text;

VibrBel:=StrToFloat(Edit2.Text);

VibrZhir:=StrToFloat(Edit3.Text);

VibrUgl:=StrToFloat(Edit4.Text);

Form2.Close;

end;

Это процедура выбора продукта, который будет вносится пользователем в рацион питания.

procedure TForm2.sBitBtn1Click(Sender: TObject);

begin

Form2.Close;

end;

Здесь описывается закрытие формы.

procedure TForm2.sBitBtn2Click(Sender: TObject);

var i1:integer;

list:TTreeNode;

staroenaz,staroenazdereva:string;

starbel,starzhir,starugl,starkkal:real;

ukazatel1:Harr;

Label 11;

begin

staroenaz:=StringGrid1.Cells[0,StringGrid1.Row];

11:

StringGrid1.Cells[0,StringGrid1.Row]:=InputBox('Название продукта', 'Пожалуйста, введите название продукта', '');

For i1:=1 to Length(StringGrid1.Cells[0,StringGrid1.Row]) do

If (StringGrid1.Cells[0,StringGrid1.Row][i1] ='|') or (StringGrid1.Cells[0,StringGrid1.Row][i1] ='&')

or (StringGrid1.Cells[0,StringGrid1.Row][i1] ='^')or (StringGrid1.Cells[0,StringGrid1.Row][i1] ='#') then

Begin

StringGrid1.Cells[0,StringGrid1.Row]:='';

ShowMessage('В названии продукта не должно быть символов: & ^ # | ');

GoTo 11;

End;

ss:='';

proverka('Белок',ss);

StringGrid1.Cells[1,StringGrid1.Row]:=ss;

ss:='';

proverka('Жиры',ss);

StringGrid1.Cells[2,StringGrid1.Row]:=ss;

ss:='';

proverka('Углеводы',ss);

StringGrid1.Cells[3,StringGrid1.Row]:=ss;

StringGrid1.Cells[4,StringGrid1.Row]:=FloatToStr(StrToFloat(StringGrid1.Cells[1,StringGrid1.Row])\*4+StrToFloat(StringGrid1.Cells[2,StringGrid1.Row])\*9+StrToFloat(StringGrid1.Cells[3,StringGrid1.Row])\*4);

Memo1.Lines[StringGrid1.Row-1]:=StringGrid1.Cells[0,StringGrid1.Row]+'|'+StringGrid1.Cells[1,StringGrid1.Row]+'&'+StringGrid1.Cells[2,StringGrid1.Row]+'^'+StringGrid1.Cells[3,StringGrid1.Row]+'#'+StringGrid1.Cells[4,StringGrid1.Row];

list:=nil;

Repeat

If list=nil then

list:=Form1.TreeView1.Items.Item[0]

Else

list:=list.GetNext;

If list.Level<>0 then

Begin

staroenazdereva:=list.Text;

ukazatel1:=list.Data;

i1:=0;

Repeat

inc(i1);

Until (staroenazdereva[i1+1]=' ') and (staroenazdereva[i1+2]=' ') and (staroenazdereva[i1+3]=' ') and (staroenazdereva[i1+4]='(') and (staroenazdereva[i1+5]='М');

inc(i1);

Repeat

Delete(staroenazdereva,i1,1);

Until i1-1>=length(staroenazdereva);

If (staroenazdereva=staroenaz) then

Begin

ukazatel1^.Bel:= ukazatel1^.Massa\*StrToFloat(StringGrid1.Cells[1,StringGrid1.Row])/100;

ukazatel1^.Zhir:=ukazatel1^.Massa\*StrToFloat(StringGrid1.Cells[2,StringGrid1.Row])/100;

ukazatel1^.Ugl:=ukazatel1^.Massa\*StrToFloat(StringGrid1.Cells[3,StringGrid1.Row])/100;

ukazatel1^.kkal:=4\*ukazatel1^.Bel+9\*ukazatel1^.Zhir+4\*ukazatel1^.Ugl;

list.Text:=StringGrid1.Cells[0,StringGrid1.Row]+' (Масса:'+FloatToStr(ukazatel1^.Massa)+')'+' (Белки:'+FloatToStr(ukazatel1^.Bel)+')'+'(Жиры:'+FloatToStr(ukazatel1^.Zhir)+')'+'(Углеводы:'+FloatToStr(ukazatel1^.Ugl)+')'+' (Ккал:'+FloatToStr(ukazatel1^.Kkal)+')';

list.Data:=ukazatel1;

End;

End;

Until list.GetNext=nil;

Form1.podschitivanie;

end;

В коде, представленном выше задается алгоритм изменения продукта. Меня его название, КБЖУ. Также при этом изменение происходит и в самом дереве-рациона.

## 2.3 Заключение.

Таковы были особенности разработки программного обеспечения. В заключении стоит отметить, что каждый фрагмент кода является очень важной, неотъемлемой частью приложения. Все фрагменты взаимодействуют между собой. В результате тестирования были исключены некоторые недостатки и недоработки. Таким образом формируется работа программы.

# ГЛАВА 3. УПРАВЛЕНИЕ ПРОЕКТОМ СОЗДАНИЯ.

## 3.1 Описание команды.

Над реализацией данного ПО работало немалое количество сотрудников. Необходимо описать задачи каждого. Также нужно очень подробно расписать бюджет и основные этапы создания.

Вот список задействованных профессий:

1. Программист
2. Системный программист
3. Системный администратор
4. Web-дизайнер
5. Верстальщик
6. Тестировщик ПО
7. Специалист по информационной безопасности
8. Оператор ЭВМ

Основная часть работы легла на плечи именно программистов. Они занимались проектированием, производством, и эксплуатацией этого приложения. Был произведен анализ математической модели поведения программы. Алгоритмы, описанные в предыдущей главе, были созданы на основании их знаний.

Системный программист занимался нюансами системы, на основе которой работает калькулятор. Был произведен основательный анализ проблем совместимости с различными видами электронно-вычислительных машин.

Поскольку над созданием работал целый штат сотрудников, было задействовано некоторое количество ЭВМ. Возникла необходимость в их эффективной совместной работе. Для увеличения производительности работы была создана виртуальная компьютерная сеть, в которой происходил обмен данными участников проекта.

Калькулятору калорий необходима очень мощная рекламная база. Было принято решение создания сайта. Сайт осуществлял функции рекламы, раздачи программы. Всеми этими вопросами занимались Web-дизайнер и верстальщик. Дизайн сайта, а также основной интерфейс получились очень дружелюбными и понятными рядовым пользователям компьютерных средств. Это в свою очередь позволило увеличить процент продаж. Эффективная реклама — залог хорошей прибыли.

После создания приложения было осуществлено полное тестирование всех основных функций. Выявленные ошибки были исправлено. Тестированием соответственно занимался программист-тестировщик программного обеспечения.

Сайт в сети интернет, доступ к машинам, на которых происходила разработка, сеть должны быть защищены от несанкционированного доступа. Этим занимался специалист по информационной безопасности. Для полного доступа к приложению необходимо обладать правами, которые приобретаются за реальные деньги. Специалист по информационной безопасности обеспечил защиту от взлома на полную версию.

Оператор ЭВМ занимался вопросами исправности компьютеров. Во время работы над проектом возник ряд неисправностей, который впоследствии был успешно устранен.

Таким образом, в результате слаженной работы сотрудников удалось успешно создать программный продукт, который направлен на решение проблем питания. Как известно, данная область очень востребована для людей.

## 3.2 Задачи проекта.

Далее следует описать основные этапы и задачи разработки:

1. Анализ требований
2. Проектирование
3. Кодирование
4. Тестирование и отладка
5. Внедрение

Первый этап, а именно анализ требований очень важен. Без выявления спроса не будет потребляемого продукта. После того, как потребность была выявлена необходимо знать, какой должна быть программа, какие нужны функции, какой интерфейс применить лучше. Этот этап был тщательно продуман, и как результат продукт получился весьма востребован на рынке программных продуктов.

На стадии проектирования были рассмотрены теоретические основы будущего продукта. На этих основах построено всё приложение. Также была проведена оценка первого этапа. Выявлялись критические места программы. Была сформирована окончательная архитектура результирующей системы программного обеспечения. В результате совещания участников проекта было выбрана среда разработки Delphi, был создан внешний интерфейс приложения. Все описанные выше компоненты очень важны. Без них не обойтись.

Следующий шаг — это непосредственное программирование. Как уже было сказано выше, этим занимался программист. Код достаточно хорошо описан выше. Возникало немало непростых проблем, которые впоследствии усердной работы сотрудников успешно решались.

После программирования приложения следует этап тестирования. На этом этапе должны быть выявлены все ошибки при работе программы и впоследствии устранены. Эту работу на себя берёт тестировщик ПО. В результате, программный продукт получился вполне работоспособным. Тестирование прошло успешно.

После того как продукт был протестирован его необходимо выводить на рынок. Этот процесс несет название “Внедрение”. Необходимо воспроизвести настройки приложения под определенные условия пользования, также имеется необходимость в обучении пользователей умения работы с калькулятором. Было разработано руководство по эксплуатации программы, в котором отражены основные аспекты взаимодействия со всеми компонентами программы. Также есть необходимость в реализации серверных станций поддержки. Там происходит обработка отчетов ошибок, рассматриваются основные вопросы и пожелания пользователей, реализуется контактная связь. Группа разработчиков просто обязана держать обратную связь. Без этого никуда.

## 3.3 Бюджет.

Разработка любого программного продукта требует определённый бюджет. Его следует более подробно описать.

Основная часть бюджета была затрачена на выплату заработной платы всем сотрудникам проекта, а также на основные этапы реализации. Бюджет проекта составлял 2 000 000 рублей.

Анализ проблем потребовал не так уж много средств на реализацию. Этот этап не так уж и сложен по отношению к другим. На его реализацию выделено 200 000 рублей.

Проектирование проекта также не потребовало много средств на реализацию. Все обошлось 100 000 рублей.

Процесс разработки программного кода поглотил немалую часть бюджета. Было затрачено 800 000 рублей. Но эти затраты стоят того. Ведь это практически основная часть работы.

На тестирование и отладку программного обеспечения ушло порядка 300 000 рублей.

Заключительный этап внедрения потребовал оставшуюся часть суммы денежного довольствия. Было выделено 600 000 рублей.

# ЗАКЛЮЧЕНИЕ

Таким образом было разработано программное, призванное облегчить труд человека. Приложение решает немалое количество важных вопросов в области правильного питания.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. *А. Н. Вальвачев, К. А. Сурков, Д. А. Сурков, Ю. М. Четырько.* [Программирование на языке Delphi. Учебное пособие](http://www.rsdn.ru/?article/Delphi/Delphi_7_00.xml). — 2005.
2. *Хавьер Пашеку.* Программирование в Borland Delphi 2006 для профессионалов = Delphi for .NET Developer’s Guide. — М.: [Вильямс](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2006. — 944 с. — [ISBN 0-672-32443-X](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/067232443X).
3. *Нил Дж. Рубенкинг.* Язык программирования Delphi для «чайников». Введение в Borland Delphi 2006 = Delphi for Dummies. — М.: [Диалектика](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B8%D0%B0%D0%BB%D0%B5%D0%BA%D1%82%D0%B8%D0%BA%D0%B0_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2007. — 336 с. — [ISBN 0-7645-0179-8](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/0764501798).
4. Книга о вкусной и здоровой пище / Под ред. акад. [Опарин А. И.](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B0%D1%80%D0%B8%D0%BD_%D0%90._%D0%98.) — М.: Пищевая промышленность, 1970. — С. 11.
5. *С. Н. Удинцев, Ю. Ю. Гичев* «Легендарная диета Аткинса: правда и вымысел», Новосибирск, 2009
6. *Джейн Кирби* Диета для «чайников» — 2006. [ISBN 0-7645-4149-8](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/0764541498)