Федеральное агентство Российской Федерации по связи и информатизации

Сибирский государственный университет телекоммуникации и информатики

**Курсовой проект по дисциплине «Технологии разработки ПО»**

UnitConverter

**Выполнили:**

Студенты группы ИВ-521

Семенов Н.А.

Прокопенко Р.П.

Попов М.М.

Лысенко А.О.

Низамов Р.И.

Новосибирск

2016

Оглавление

[Назначение разработки заключается в предоставлении заказчику данного программного обеспечения возможности точного и быстрого преобразования физических, математических и компьютерных значений. 4](#_Toc452749118)

**Введение**

Конвертер величин для современного человека - вещь необходимая. Все дело в том, что ежедневно каждый из нас сталкивается с большим потоком информации, которая не всегда бывает до конца понятной.

Для учеников и студентов незаменимую помощь окажет конвертер величин. С его помощью можно без труда определить, сколько метров в футе, что представляет собой ярд или морская миля и многое другое.

**Постановка задачи**

Целью данной работы является разработка программного продукта, который предназначен для конвертирования различных физических, математических и компьютерных величин.

Назначение разработки заключается в предоставлении заказчику данного программного обеспечения возможности точного и быстрого преобразования физических, математических и компьютерных значений.

**Выбор технологии, языка и среды программирования**

При написания программного обеспечения использован язык C.

C (Си) — [компилируемый](https://ru.wikipedia.org/wiki/Компилятор) [статически типизированный](https://ru.wikipedia.org/wiki/Статическая_типизация) процедурный [язык программировани](https://ru.wikipedia.org/wiki/Язык_программирования)я общего назначения.

Написание кода выполнено структурным подходом к программированию, т.е. для работы с различными физическими величинами написаны отдельные модули.

Программный продукт разработан в средах программирования, таких как Dev-C++, CodeBlocks и Geany.

**Проектирование информационной системы**

Программный продукт имеет структуру:

* все методы и интерфейсы находятся в отдельных \*.c-файлах;
* все прототипы методов и структуры, необходимые для работы программного продукта находятся в файле total.h;
* методы защиты также находятся в отдельных \*.c-файлах;
* сборка программного продукта описана в \*.bat-файле для Windows-версии makefile для UNIX-версии.

**Выбор методов и разработка основных алгоритмов программного продукта**

При разработке программного продукта было решено применять максимально возможную декомпозицию всех модулей, структур и данных.

Программный продукт содержит такие основные модули как:

- методы конвертирования;

- методы управления;

- методы защиты программы;

**Описание используемых методов**

Программный продукт содержит следующие методы:

- методы конвертирования:

Методы, описывающие алгоритмы конвертирования величин из одной в иную;

- методы управления:

Методы, отвечающие за ввод данных, а также взаимодействие программы и пользователя;

- методы защиты программы:

Позволяют пользователю пользоваться лишь корректными данными.

**Выбор стратегии тестирования и разработка тестов**

1. Классификация тестирований по доступу к коду:

- метод белого ящика – у тестировщика есть доступ к внутренней структуре и коду приложения;

- метод черного ящика – у тестировщика либо нет доступа, либо недостаточно знаний для понимания, либо так задумано. Другими словами, тестирование на основе документации;

- метод серого ящика – комбинация методов белого и черного ящиков;

2. Классификация тестирований по доступу к коду:

- модульное тестирование – тестирование каждого из модулей отдельно от остальных;

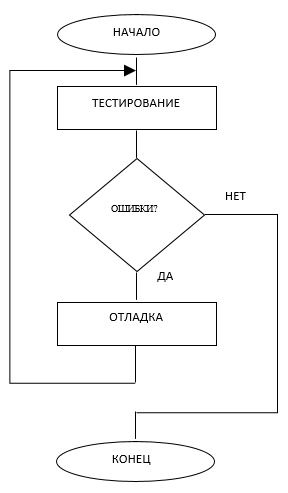
- интеграционное тестирование – тестирование взаимодействия частей программного продукта;

- системное тестирование – проверка приложения как единого целого.

После реализации программного продукта было выполнено следующее:

- модульное тестирование программного продукта;

- Тесты проводились методом серого ящика.



**Описание процесса отладки программы. Контрольный пример**

При разработке программы наиболее трудоемким является этап отладки и тестирования программы. Цель тестирования, т.е. испытания программы, заключается в выявлении имеющихся в программе ошибок. Цель отладки состоит в выявлении и устранении причин ошибок.

Отладку программы начинают с составления плана тестирования. Такой план должен представлять себе любой программист. Составление плана опирается на понятие об источниках и характере ошибок. Основными источниками ошибок являются недостаточно глубокая проработка математической модели или алгоритма решения задачи, нарушение соответствия между схемой алгоритма или записью его на алгоритмическом языке и программой, записанной на языке программирования. Невнимательность при наборе программы и исходных данных на клавиатуре устройства ввода.

Нарушение соответствия между детально разработанной записью алгоритма в процессе кодирования программы относится к ошибкам, проходящим вследствие невнимательности программиста. Отключение внимания приводит и ко всем остальным ошибкам, возникающим в процессе подготовки исходных данных и ввода программы в ЭВМ. Ошибки, возникающие вследствие невнимательности, могут иметь непредсказуемые последствия, так как наряду с ошибками описаний массивов, нарушениями баланса скобок возможны и такие ошибки, как потеря операторов, замена букв в обозначениях переменных, отсутствие определений начальных значений переменных, нарушение адресации в массивах.

Учитывая разнообразие источников ошибок, при составлении плана тестирования классифицируют ошибки на два типа: 1 – синтаксические; 2 – семантические (смысловые).

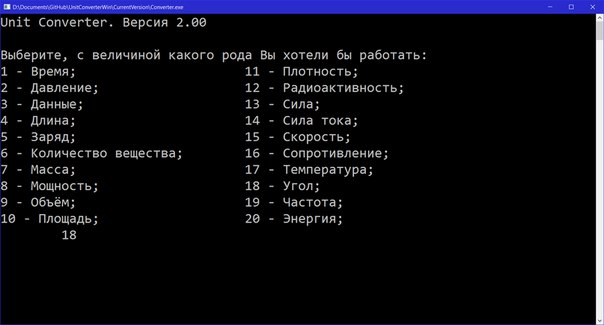
**Синтаксические ошибки** – это ошибки в записи конструкций языка программирования (переменных, функций, выражений, операторов, подпрограмм).

**Семантические ошибки** – это ошибки, связанные с неправильным содержанием действий и использованием недопустимых значений величин.

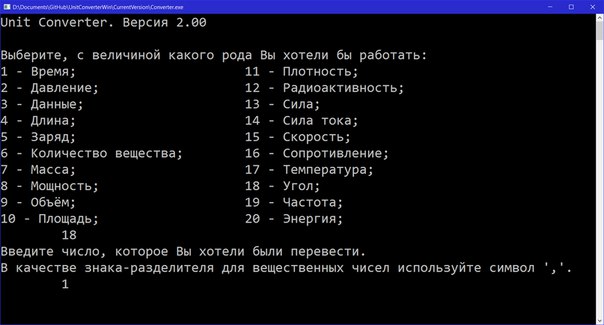
Обнаружение большинства синтаксических ошибок автоматизировано в основных системах программирования. Поиск же семантических ошибок гораздо менее формализован; часть их проявляется при исполнении программы в нарушениях процесса автоматических вычислений и инициируется либо выдачей диагностических сообщений рабочей программы, либо отсутствием печати результатов из-за бесконечного повторения одной и той же части программы (зацикливания), либо появлением непредусмотренной формы или содержания печати результатов.

Контрольный пример служит для проверки правильности работы приложения.

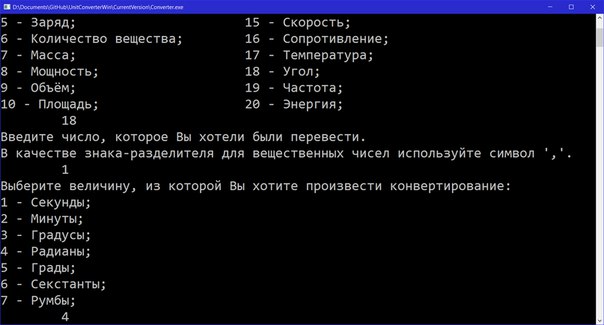
При запуске программного продукта сначала пользователю предлагается выбрать род величины, с которым он хочет работать. Например, выбираем род величины Угол. Для этого вводим порядковый номер рода величины. В нашем случае это 18.



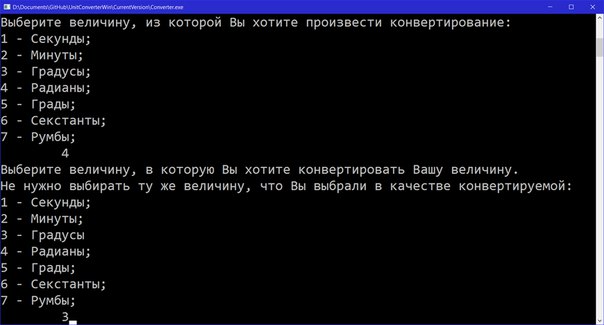
Введя порядковый номер пользователю предлагается ввести число, которое он хочет перевести.



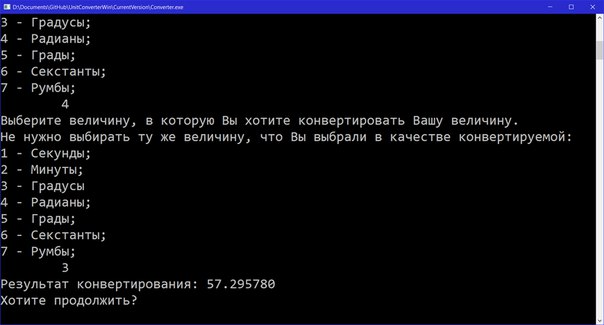
В нашем случае это 1. Затем пользователю необходимо выбрать величину, которую он хочет конвертировать.



Например, этой величиной будут радианы (порядковый номер 4). И далее пользователю необходимо выбрать величину, в которую он хочет сконвертировать исходную.



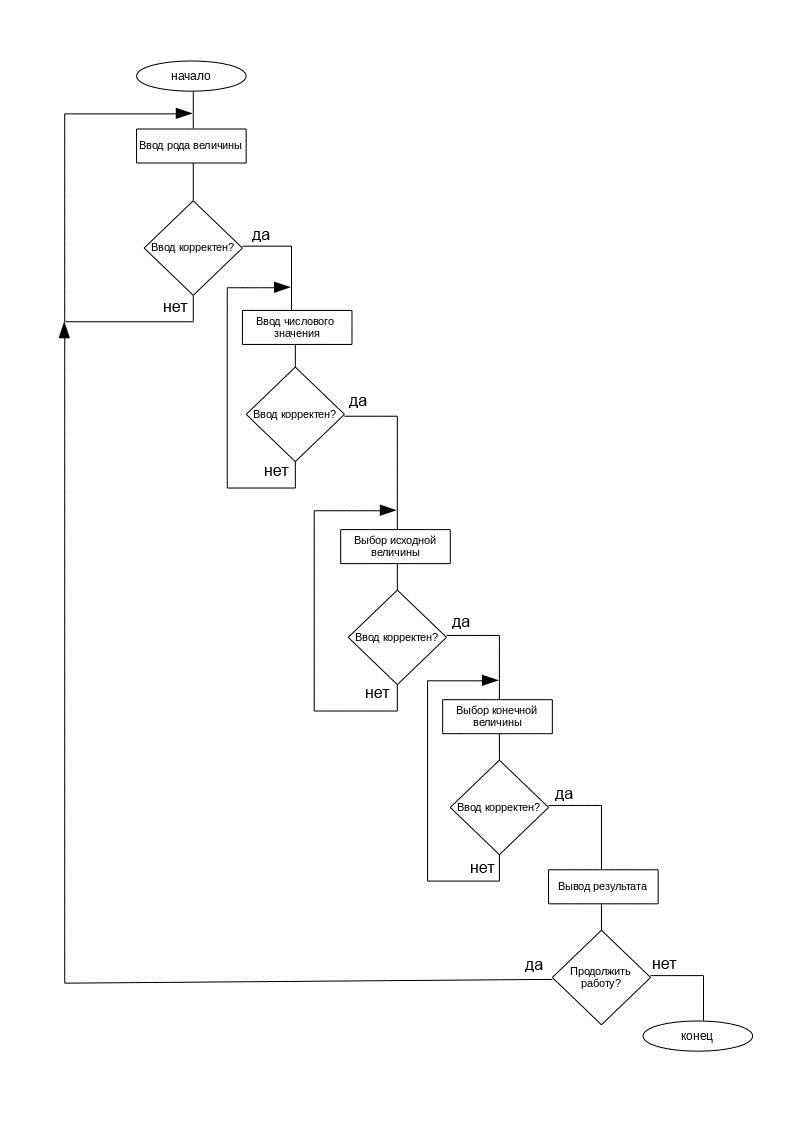
В данном случае это градусы (порядковый номер 3). Программа конвертирует 1 радиан в градусы и выводит результат.



Если пользователь захочет продолжить работу в программе, ему следует нажать клавишу «y», для выхода из программы – любую другую клавишу.

Для работы с другими родами и видами величин, алгоритм действий аналогичный:

Выбор рода величины ---> ввод числового значения ---> выбор величины, из которой производится конвертирование ---> выбор величины, в которую производится конвертирование ---> получение результата конвертирования. Ниже приведена блок-схема принципа работы программы.

Рисунок - Блок-схема, демонстрирующая принцип работы программы

**Руководство программиста**

Программный продукт собирается с помощью \*.bat-файла для Windows-версии или makefile для UNIX-версии. В \*.bat-файле и makefile описываются все рецепты сборки объектных файлов, сборки тестов и полная сборка программного продукта. Привести листинг файлов (.bat и makefile);

**Заключение**

В ходе выполнения курсового проекта была достигнута поставленная заказчиком цель, решены все стоящие перед командой разработчиков задачи.

Четкая структуризация кода и грамотно подобранные имена переменных, структур и методов способствуют удобочитаемости кода программы, удобной отладке, а также возможности без труда дополнять программный продукт новыми возможностями и алгоритмами конвертирования.

Законченный программный продукт – это отличный конвертер величин.

**Литература**

- Дэвид Гриффитс, Дон Гриффитс - Изучаем программирование на C (Мировой компьютерный бестселлер) – 2013;

- Г. С. Иванова - Технология программирования;

- Курносов - Введение в структуры и алгоритмы обработки данных;

**Приложение 1**

Личный вклад каждого из участников:

|  |  |  |
| --- | --- | --- |
| **ФИО** | **Обязанности** | **Разработанные модули** |
| Семенов Николай Александрович | 1. Team-leader (распределение обязанностей между участниками команды; контроль за добросовестным выполнением каждым из участников своей части проекта);  2. Разработчик программных модулей (написание кода);  3. Разработчик документации (составление технического задания, плана работ);  4. Тестировщик (тестирование работы всего программного продукта в целом). | - методы модуля <Данные>;  - интерфейсная часть модуля <Данные>;  - методы модуля <Сопротивление>;  - интерфейсная часть модуля <Сопротивление>;  - методы модуля <Радиоактивность>;  - интерфейсная часть модуля <Радиоактивность>;  - методы модуля <Заряд>;  - интерфейсная часть модуля <Заряд>;  - метод защиты <Parcing>;  - обработка критических ситуаций модуля <Температура>. |
| Лысенко Арсений  Олегович | 1. Разработчик программных модулей (написание кода);  2. Разработчик документации (составление технического задания, плана работ, отчета и руководства пользователя). | - методы модуля <Длина>;  - интерфейсная часть модуля <Длина>;  - методы модуля <Масса>;  - интерфейсная часть модуля <Масса>;  - методы модуля <Объем>;  - интерфейсная часть модуля <Объем>;  - методы модуля <Температура>;  - интерфейсная часть модуля <Температура>. |
| Попов  Михаил  Михайлович | 1. Разработчик unit-тестов (написание тестов для каждого модуля программного продукта);  2. Разработчик программных модулей (написание кода). | - методы модуля <Скорость>;  - интерфейсная часть модуля <Скорость>;  - методы модуля <Время>;  - интерфейсная часть модуля <Время>;  - методы модуля <Энергия>;  - интерфейсная часть модуля <Энергия>;  - методы модуля <Давление>;  - интерфейсная часть модуля <Давление>;  - методы защиты <Parcing\_id> и <Convert>. |
| Низамов  Роман  Ильдарович | 1. Разработчик программных модулей (написание кода);  2. Тестировщик (тестирование каждого модуля в отдельности). | - методы модуля <Плотность>;  - интерфейсная часть модуля <Плотность>;  - методы модуля <Сила тока>;  - интерфейсная часть модуля <Сила тока>;  - методы модуля <Сила>;  - интерфейсная часть модуля <Сила>;  - методы модуля <Площадь>;  - интерфейсная часть модуля <Площадь>; |
| Прокопенко Роман Павлович | 1. Разработчик программных модулей (написание кода);  2. Тестировщик (тестирование работы всего программного продукта в целом);  3. Разработчик документации (составление технического задания, плана работ, отчета и руководства пользователя). | - методы модуля <Угол>;  - интерфейсная часть модуля <Угол>;  - методы модуля <Мощность>;  - интерфейсная часть модуля <Мощность>;  - методы модуля <Количество вещества>;  - интерфейсная часть модуля <Количество вещества>;  - методы модуля <Частота>;  - интерфейсная часть модуля <Частота>;  - обработка критических ситуаций модуля <Температура>. |