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 Николай

Прокопенко Рома

Попов Михаил

Лысенко Арсений

Низамов Роман

Новосибирск

2016

**Содержание**

Введение 3

Постановка задачи 4

Выбор технологии, языка и среды программирования 7

Проектирование информационной системы (блок-схемы) ??? 8

Выбор методов и разработка основных алгоритмов программного продукта 16

Описание используемых процедур и функций 34

Выбор стратегии тестирования и разработка тестов 39

Описание процесса отладки программы. Контрольный пример??? 63

Руководство программиста 94

Заключение 111

Литература 112

**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) [язык программировани](https://ru.wikipedia.org/wiki/Язык_программирования)е 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. Это связано с тем, что каждый участник команды выполнял свою часть работы в удобной ему среде.

**Проектирование информационной системы**

Программный продукт имеет структуру:

* все модули и свитчи (switches) величин находятся в отдельных .c-файлах. (можно подробнее расписать что такое свитч и модуль)
* все прототипы функций и структуры, необходимые для работы программного продукта находятся в файле header.h
* функции конвертирования и парсинга?? также находятся в отдельных .c-файлах.
* сборка программного продукта описана в makefile.

**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:

КОЛЬ ЗДЕСЬ НАПИШИ ОСНОВНЫЕ ФУНКЦИИ (PARSING И Т.Д.) ПРОСТО ПЕРЕЧИСЛИ ИХ

**Описание используемых процедур и функций**

**(Коль опиши Parsing и другие подобные функции)**

**Выбор стратегии тестирования и разработка тестов**

**(здесь надо написать какой подход был использован при написании тестов. И перечислить тесты.**

**Можно взять некоторую теорию отсюда** [**http://evgeny-p.github.io/teaching/materials/docs/lectures/pdf/trpo-05-06.pdf**](http://evgeny-p.github.io/teaching/materials/docs/lectures/pdf/trpo-05-06.pdf) **)**

**Описание процесса отладки программы. Контрольный пример**

При разработке программ 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).

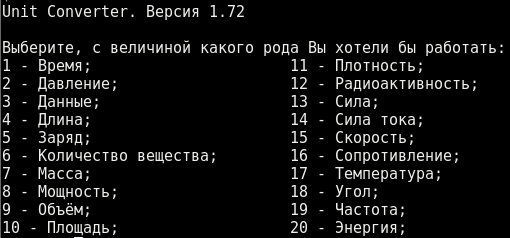
**Синтаксические ошибки** – это ошибки в записи конструкций языка программирования (переменных, функций, выражений, операторов, подпрограмм).

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

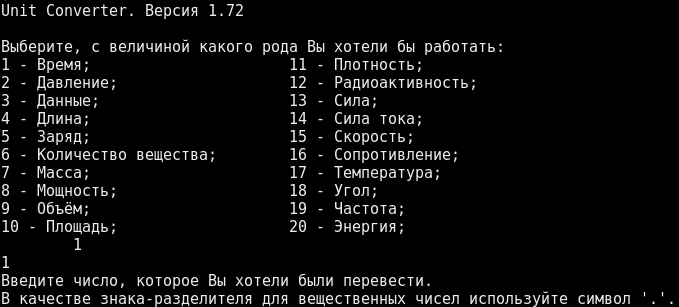
Обнаружение большинства синтаксических ошибок автоматизировано в основных системах программирования. Поиск же семантических ошибок гораздо менее формализован; часть их проявляется при исполнении программы в нарушениях процесса автоматических вычислений и инициируется либо выдачей диагностических сообщений 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.

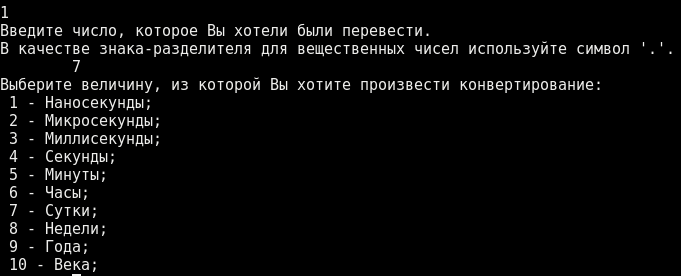
Контрольный пример служит для проверки правильности работы приложения.

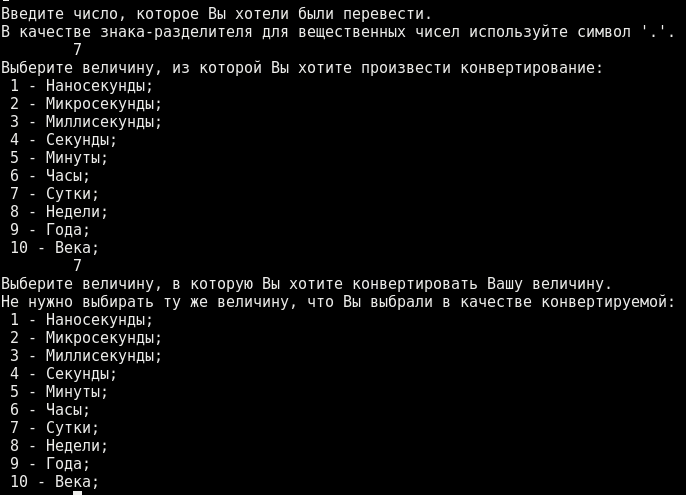
При запуске программного продукта сначала пользователю предлагается выбрать род величины, с которым он хочет работать.

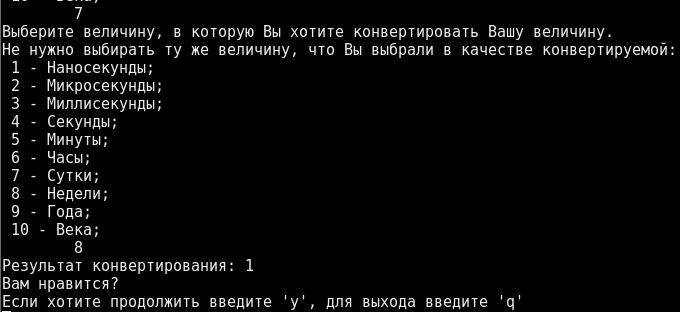


Например, выбираем род величины Время. Для этого вводим порядковый номер рода величины. В нашем случае это 1. Введя порядковый номер пользователю предлагается ввести число, которое он хочет перевести.

Пользователь вводит число, которое он хочет перевести. В нашем случае это 7. Затем пользователю необходимо выбрать величину, которую он хочет конвертировать.

Например, этой величиной будут сутки (порядковый номер 7). И далее пользователю необходимо выбрать величину, в которую он хочет сконвертировать исходную.

В данном случае это недели (порядковый номер 8).

Программа конвертирует 7 дней в недели и на выходе получается ожидаемый результат 1 неделя.

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

Для работы с другими величинами, алгоритм действий аналогичный:

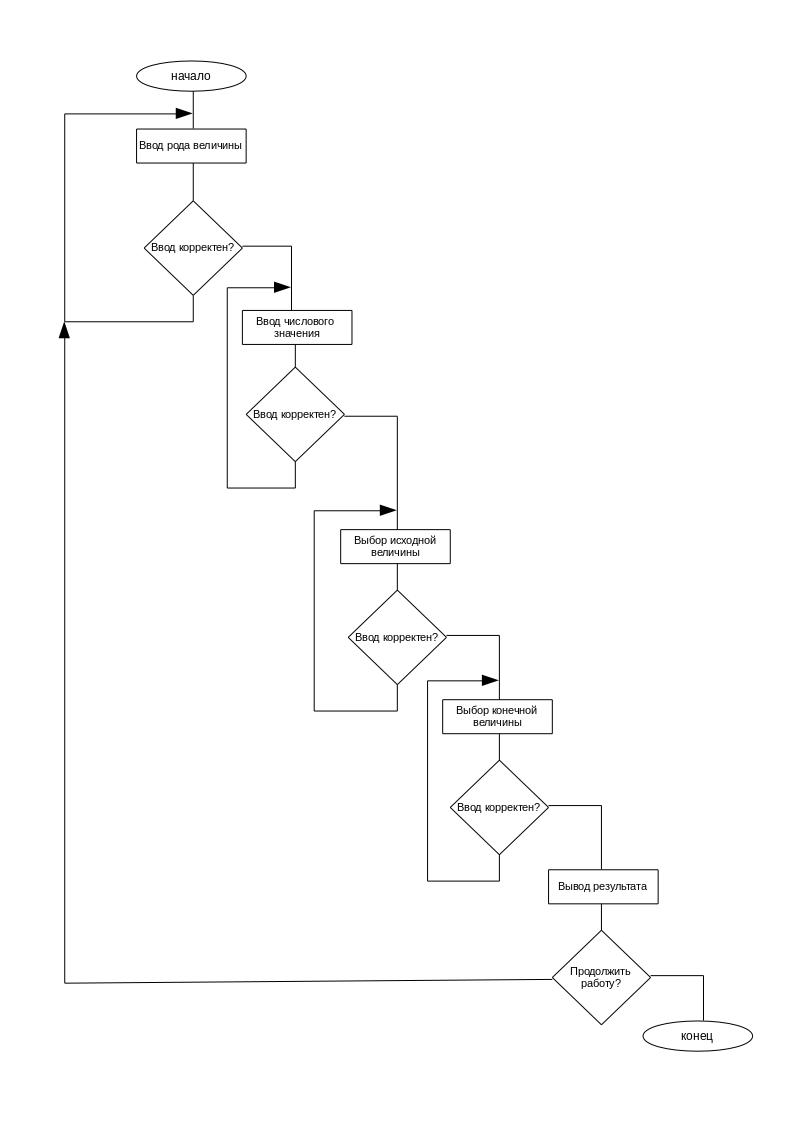
Выбор рода величины ---> ввод числового значения ---> выбор величины, из которой производится конвертирование ---> выбор величины, в которую производится конвертирование ---> получение результата конвертирования. Ниже приведена блок-схема принципа работы программы. 

Рисунок - Блок-схема, демонстрирующая принцип работы программы

**Руководство программиста**

Программный продукт собирается с помощью makefile. В makefile описываются все рецепты сборки объектных файлов, сборки тестов и полная сборка программного продукта. (Коля или Мишаня опишите это дело подробнее)

**Заключение**

В ходе выполнения курсового проекта были выполнены все поставленные задачи.

Четкая структуризация кода и грамотно подобранные имена переменных и функций способствуют удобочитаемости программы.

Законченный программный продукт – это отличный конвертер величин.

**Литература**