TR-language
5001 строка исходного кода, 134Кб исходник и 151Кб исполняемый файл, 11 классов, 3 недели времени, 1 человек и 1 новый язык программирования. Вопрос дня — зачем всё это? К чему столько работы, ведь языков программирования — пруд пруди, а учить ещё один — лишний труд. В чём его изюминка?
История его создания и причины, по которым он был создан, изложены ниже.
И так, изначальная цель и требования были следующие:

1) Необходимо средство, позволившее бы переводить программы с одного языка на другой
2) Средство должно быть универсальным, т. е. должно переводить любые грамматики

На основе этих требований я пришёл к выводу, что таким средством должен стать некоторый другой язык. Смирившись с мыслью, что мне придётся его писать самому, был составлен ещё один ряд требований:

1) Синтаксис языка должен быть ассемблероподобным, для простоты его проектирования
2) Должны присутствовать основные типы данных
3) Работа с файлами и шаблонами для считывания данных должна быть максимально упрощена

В качестве основных типов были выбраны: числа(длинная 900 разрядная арифметика), строки, файл и вектор с возможностью добавления в начало, в конец, удалением из конца, начала, а также произвольным доступом по индексу. Также реализована структура данных, хранящая строки, и присваивающая им порядковые номера. Для удобной работы с шаблонами предусмотрены множество — упорядоченный набор строк, объекты, они же шаблоны(читай аналог регулярных выражений), и фреймы — считанный шаблон.
По аналогии с ассемблером в процессе разработки были добавлены флаги(массив из ста целых 32 разрядных чисел), и стек, чтобы была возможность осуществлять вызовы функций.

Синтаксис команд с их описанием выглядит следующим образом:
см. приложение 1.

После создания синтаксиса начался процесс написания кода. Весь проект разбит на файлы, каждый из которых описывает свой класс и функции, связанные с ним. Результирующая программа состоит из двух частей: первая переводит текстовый код в байт код, вторая часть является виртуальной машиной. Это ускоряет работу написанной в целом, и сильно упрощает виртуальную машину, так как предобработка имен переменных, функций и меток произведена заранее компилятором.

Описание байткода:
см. приложение2

Что я получил в итоге, или вот она, вундервафля!
Результатом трудов, стал программный продукт из двух исполняемых файлов: компилятора, производящего сборку в байт код, и виртуальной машины, исполняющей байткод. Для компиляции, а в последствии и исполнения анализируется файл settings.txt, где в единственной строке содержится имя файла исходного кода на языке TR-language без расширения(как несложно догадаться, его расширение ”.tr”) находящегося в той же директории, что и файл settings, что и компилятор с виртуальной машиной. Во время компиляции и исполнения будет создан файл filename.trobj и все файлы, создаваемые вашим кодом на языке TR-language.
Во время компиляции и исполнении могут возникнуть ошибки. Так как ресурсов, человекочасов, у меня мало, то отлавливаются далеко не все из них. Но о тех, что таки попались, вы можете узнать, прочитав содержимое файлов в директории tmp. Если таковых

не будет, то отчёт об ошибках так же не будет создан. О том, что что-то пошло не так, вы можете узнать по отрицательному коду возвращённым компилятором или виртуальной машиной. Содержимое файлов папки tmp интуитивно понятно по их имени.
Для удобства работы с новоиспечённым языком была создана подсветка синтаксиса tr-language для notepad++. В процессе создания подсветки синтаксиса была добавлена фича: создание блоков в коде, если после комментариев писать reg или endreg.

Фэйл.
Есть один фэйл, о котором хочется сказать отдельно. Связан он с дублированием байткода программы на языке tr внутри виртуальной машины, при каждом вызове функции. Так если производится копирование всех команд тела функции. Это не есть хорошо, но эта логическая ошибка была поймана на позднем этапе разработки, вследствие чего цена её исправления это отсутствующие человеко-часы в моём распоряжении. Её исправление отложено до лучших времён.

О том, чего хотел, сказано, что получилось в итоге тоже сказано. Осталось продемонстрировать возможности, транслятор с си на паскаль в студию!
См. приложение3