**КУРСОВАЯ РАБОТА**

Разработка и реализация виртуальной машины

по дисциплине «Объектно-ориентированное программирование»

|  |  |
| --- | --- |
| Выполнил  студент группы 3530904/80003 | Родченков С. Н. |
| Руководитель | Коптев Д. А. |

«25» декабря 2019г.

# Содержание

[1 Введение 3](#_Toc28186083)

[1.1 Цель 3](#_Toc28186084)

[1.2 Задачи 3](#_Toc28186085)

[1.3 Основная информация 3](#_Toc28186086)

[2 Основная часть 4](#_Toc28186087)

[2.1 Архитектура 4](#_Toc28186088)

[2.1.1 Класс RVM 4](#_Toc28186089)

[2.1.2 Класс CaseInsensetiveString 4](#_Toc28186090)

[2.1.3 Класс RasmLexer 4](#_Toc28186091)

[2.1.4 Класс RasmTranslator 4](#_Toc28186092)

[2.2 Пользовательское взаимодействие с программой 4](#_Toc28186093)

[2.3 Исходный код 5](#_Toc28186094)

[2.4 Пример работы 5](#_Toc28186095)

[3 Заключение 7](#_Toc28186096)

[4 Список литературы 8](#_Toc28186097)

# Введение

## Цель

Разработка объектно-ориентированной программы с графическим интерфейсом.

## Задачи

1. Разработка архитектуры программы.
2. Реализация классов, составляющих архитектуру программы.
3. Отладка и тестирование программы.
4. Написание отчета о проделанной работе.

## Основная информация

Курсовой проект представляет из себя простейшую виртуальную машину, исполняющую байт-код, сгенерированный написанным ассемблером. Машина поддерживает основные команды: логические и арифметические операции, условные переходы, вызов подпрограмм и возврат из них, перемещение данных заданного размера, а также вызов нескольких прерываний.

Обе описанные сущности связаны консольным приложением, при взаимодействии с которым пользователь может как сгенерировать байт-код из исходного кода, так и исполнить уже созданный байт-код.

Для запуска программы в общем требуется поддержка C++17, однако, для использования виртуальной машины в отдельности, достаточно стандарта С++14. Никаких сторонних библиотек не использовано.

# Основная часть

## Архитектура

Исходный код программы состоит из нескольких классов, отвечающих за разные этапы работы приложения.

### Класс RVM

Класс представляет из себя собственно регистровую виртуальную машину, исполняющую байт-код. Содержит вектор стека и массив регистров.

### Класс CaseInsensetiveString

Представляет из себя обертку класса *std::string*, с той разницей, что результаты операций сравнения и хеширования не зависят от наличия прописных букв, что позволяет упростить трансляцию команд ассемблера, написанных с использованием заглавных букв.

### Класс RasmLexer

Позволяет преобразовывать исходный текст программ в последовательность токенов, обрабатываемую самим транслятором.

### Класс RasmTranslator

Транслирует последовательность токенов, полученную от лексера, в исполняемый виртуальной машиной байт-код. Для хранения лейблов переходов используется *std::unordered\_map*, для хранения и последующей обработки лейблов, к которым было произведено обращение до из объявления используется *std::unordered\_multimap.*

## Пользовательское взаимодействие с программой

Для работы с виртуальной машиной необходимо сначала написать программу на языке ассемблера, после транслировать ее в исполняемый код, запустив приложение с флагом «*/a*» и аргументами – путем к файлу с исходным кодом и путем к файлу, который будет сгенерирован в процессе работы. Для запуска исполнения полученного файла необходимо вновь запустить программу с ключом «*/e*» и путем к файлу байт-кода.

## Исходный код

Весь исходный код, а также файлы проекта Visual Studio и демонстрационная программа на языке ассемблера находятся в свободном доступе по адресу «*https://gitlab.com/rodchenkov.sn/spbspu-cxx-term-works*»

## Пример работы

Создадим текстовый файл и напишем в нем программу (рисунок 1), сохраним его. Произведем сборку (рисунок 2), получив двоичный исполняемый файл. Исполним его на виртуальной машине (рисунок 3).

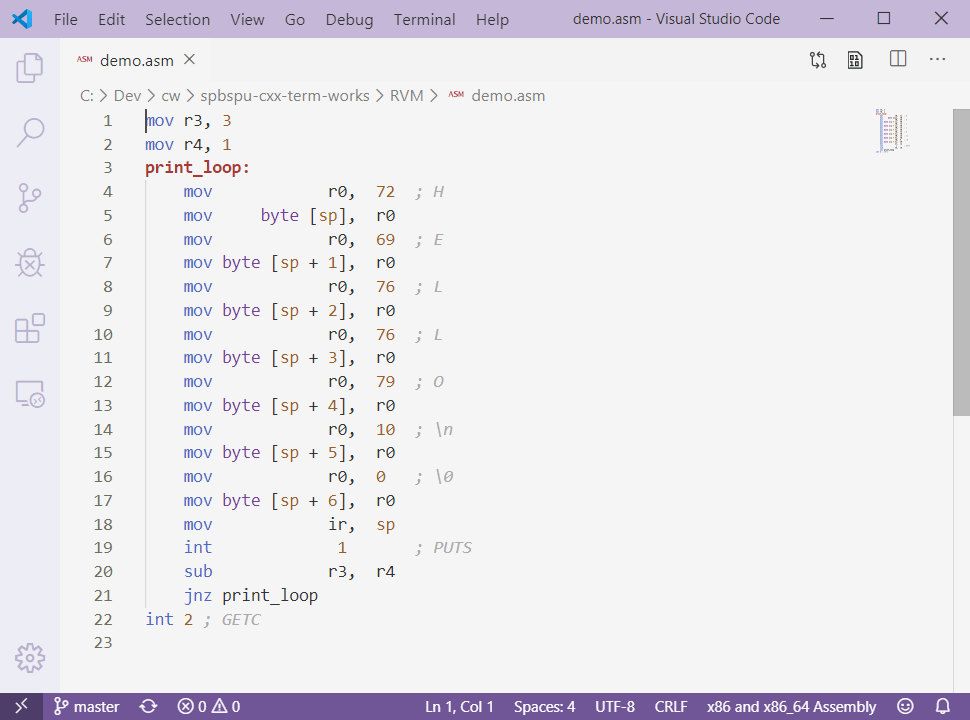


Рисунок 1 – исходный код демонстрационной программы

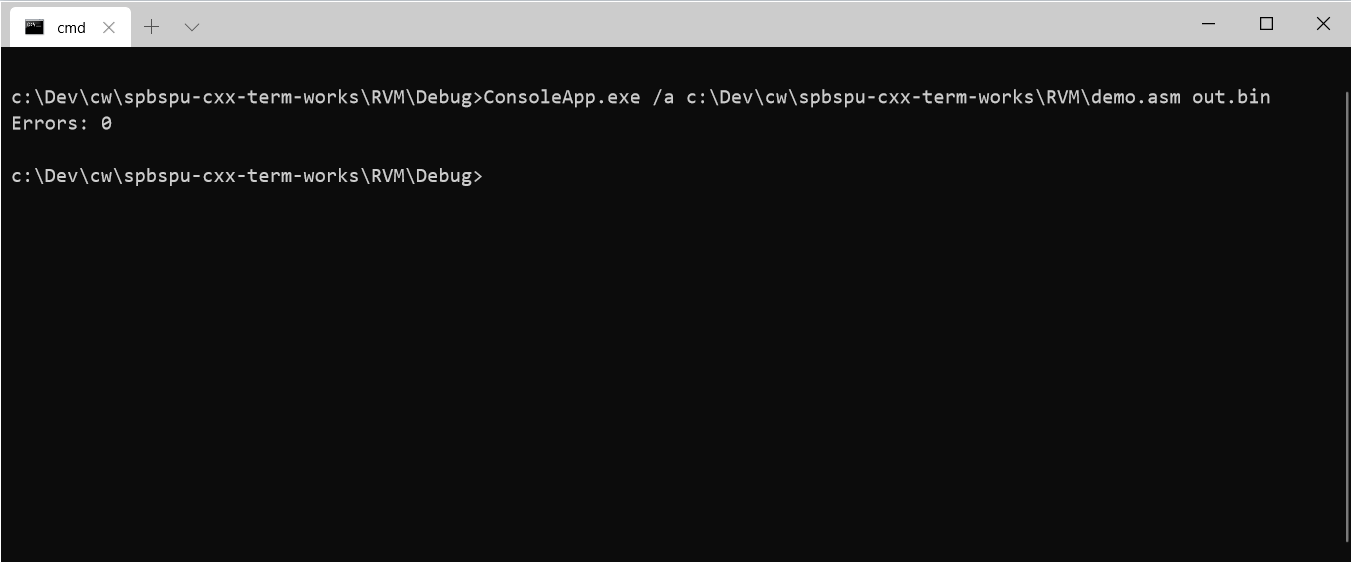


Рисунок 2 – процесс успешной трансляции демонстрационной программы

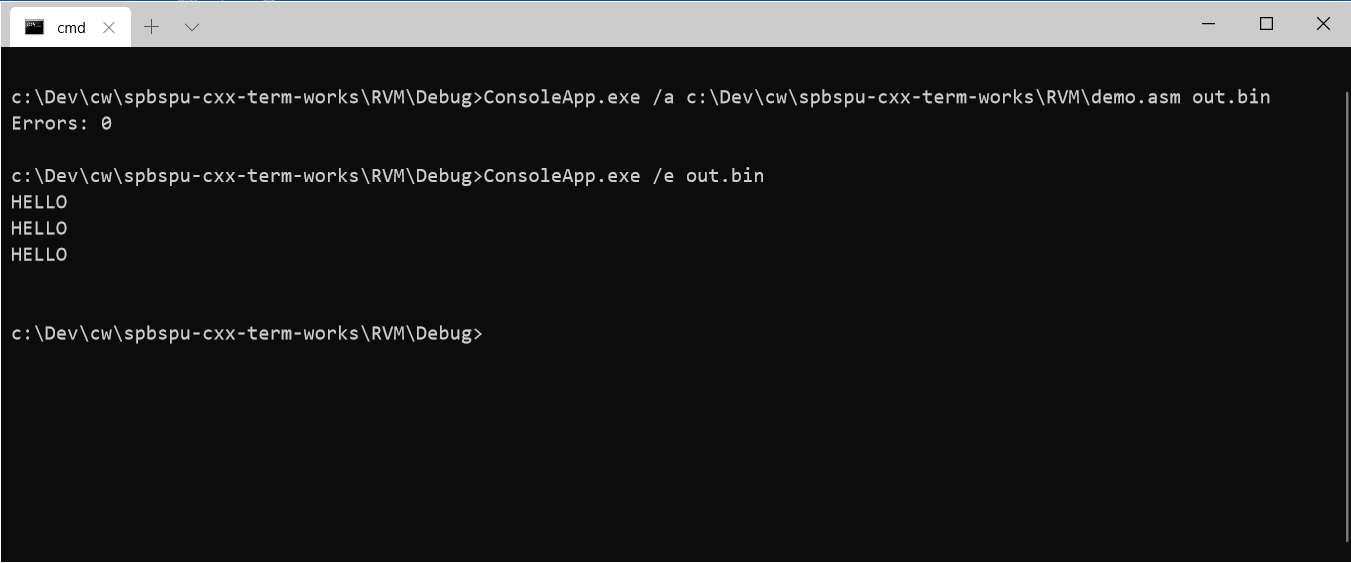


Рисунок 3 – результат исполнения демонстрационной программы

# Заключение

В результате выполнения данной работы была разработана объектно-ориентированная программа с консольным интерфейсом, разработана архитектура классов программы, проведена отладка и тестирование. Были приобретены навыки разработки программного продукта с использованием стандартной библиотеки шаблонов.

# Список литературы

**C++17 STL. Стандартная библиотека шаблонов** [Книга] / авт. Я. Галовиц. - СПБ : Питер, 2018. - стр. 432.

**cppreference** [В Интернете] // cppreference. - 2000 г.. - https://en.cppreference.com/w/.

**Архитектура компьютера** [Книга] / авт. Таненбаум Э. Остин Т.. - СПБ : Питер, 2018. - 6-е : стр. 816.