

« \_\_\_\_ » \_\_\_\_\_ 2026 г. А.В. Пролетарский

2026 г.

# 1 ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку программы «Компилятор стекового языка программирования» [КСЯП], используемой для трансляции в байткод виртуальной машины WebAssembly (Wasm), а также верификации и оптимизации программ, написанных на конкатенативном Cat-подобном языке программирования.

Актуальность разработки обусловлена новизной выбора цели трансляции. С точки зрения языка, прямая трансляция в WebAssembly позволит избежать дополнительных накладных расходов, связанных с тяжеловесной машиной .NET CLR, увеличить скорость работы и снизить потребление памяти Cat-программ. С точки зрения виртуальной машины, текстовый формат программ которой основан на использовании S-выражений, WebAssembly получит инструмент описания программ при помощи конкатенативного языка, что является более распространенным для стековых языков.

## 2 ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

Компилятор для стекового языка программирования разрабатывается в соответствии с тематикой кафедры «Компьютерные системы и сети».

## 3 НАЗНАЧЕНИЕ РАЗРАБОТКИ

Основное назначение компилятора — преобразование (трансляция) программ, написанных на исходном языке в байткод, описанный спецификацией виртуальной машины WebAssembly. Также компилятор может осуществлять верификацию и оптимизацию программы при помощи статического анализа.

## 4 ИСХОДНЫЕ ДАННЫЕ, ЦЕЛИ И ЗАДАЧИ

### 4.1 Исходные данные

4.1.1 Исходными данными для разработки являются следующие материалы:

4.1.1.1 Simple Type Inference for HigherOrder StackOriented Languages [Электронный ресурс]. URL: <https://dcreager.net/remarkable/Diggins2008a.pdf> (дата обращения: 13.10.2025).

4.1.1.2 Cat language repository [Электронный ресурс]. URL: <https://github.com/cdiggins/catlanguage> (дата обращения: 13.10.2025).

4.1.1.3 WebAssembly Specification 1.0 [Электронный ресурс]. URL: <https://webassembly.github.io/spec/versions/core/WebAssemblyU1.0.pdf> (дата обращения: 11.11.2025).

4.1.1.4 Understanding WebAssembly text format [Электронный ресурс]. URL: [https://developer.mozilla.org/en-US/docs/WebAssembly/Guides/Understanding\\_the\\_text\\_format](https://developer.mozilla.org/en-US/docs/WebAssembly/Guides/Understanding_the_text_format) (дата обращения: 11.11.2025).

### 4.2 Цель работы

Целью работы является компилятор стекового языка программирования.

### 4.3 Решаемые задачи

4.3.1 Анализ синтаксиса и семантики различных стековых языков программирования, выделение и сравнение их особенностей.

4.3.2 Анализ средств статического анализа языка программирования Cat.

4.3.3 Анализ устройства байткода и модели исполнения WebAssembly.

4.3.4 Проектирование программных компонентов: парсера, транслятора, статического анализатора.

4.3.5 Выбор технологии и средств разработки для программного обеспечения (далее ПО).

4.3.6 Разработка структуры программного обеспечения и его компонентов.

4.3.7 Разработка алгоритмов трансляции и статического анализа.

4.3.8 Реализация компонентов ПО с использованием выбранных средств разработки.

4.3.9 Проведение сборки и комплексного тестирования ПО.

4.3.10 Разработка технологии многопроходной оптимизации\генерации webassembly байткода\в процессе реализации станет более ясно.

## 5 ТРЕБОВАНИЯ К ПРОГРАММНОМУ ИЗДЕЛИЮ

### 5.1 Требования к функциональным характеристикам

#### 5.1.1 Выполняемые функции:

- генерация байткода WebAssembly для переданной программы на исходном языке;
- верификация и анализ переданной программы на исходном языке с точки зрения корректности использования стека данных;
- оптимизация переданной программы на исходном языке.

#### 5.1.2 Исходные данные

Исходные данные должны быть представлены в виде текста программы на исходном языке и управляющими флагами, указывающими на тип проводимого действия (генерация байткода и/или верификация и/или оптимизация).

Исходный язык должен быть подмножеством языка программирования Cat.

#### 5.1.2 Результаты:

- в случае успешной операции – файл с байткодом WebAssembly или сообщение, подтверждающее корректность программы;
- в случае неуспешной операции – сообщение с описанием ошибки.

### 5.2 Требования к надежности

5.2.1 Предусмотреть контроль консистентности флагов, передаваемых при вызове компилятора.

5.2.2 Предусмотреть контроль синтаксической корректности текста на исходном языке.

5.2.3 Предусмотреть контроль корректности работы программ с стеком данных.

5.3 Условия эксплуатации

5.3.1 Условия эксплуатации в соответствии с СанПиН 2.2.2/2.4.1340-03.

5.4 Требования к составу и параметрам технических средств

Минимальная конфигурация технических средств, на которых может быть запущен компилятор:

– количество ядер процессора – 1 шт;

– объем ОЗУ – 1 Гб.

5.5 Требования к информационной и программной совместимости

ПО должно функционировать на IBM-совместимых персональных компьютерах и иметь интерфейс командной строки.

5.6 Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

5.7 Требования к транспортированию и хранению

Требования к транспортировке и хранению не предъявляются.

## 6 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

6.1 Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

6.2 В состав сопровождающей документации должны входить:

6.2.1 Расчетно-пояснительная записка на 55-60 листах формата А4 (без приложений).

6.2.2 Техническое задание (Приложение А).

6.2.3 Руководство программиста (Приложение Б).

6.3.4 Исходный текст программного компонента статического анализа (Приложение В).

6.4 Графическая часть должна быть выполнена на 6 листах формата А1 (копии формата А3/А4 включить в качестве приложений к расчетно-пояснительной записке):

6.3.1 Схема структурная ПО.

6.3.2 Схемы алгоритмов разбора текста передаваемой программы и статического анализа.

6.3.3 Функциональная диаграмма ПО.

6.3.4 Результаты сравнительного анализа стековых языков.

6.3.5 Диаграмма вариантов использования.

6.3.6 Синтаксическая диаграмма консольного интерфейса ПО в виде РБНФ.

## 7 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Требования не предъявляются.

## 8 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Этапы разработки курсовой работы указаны в таблице 1.

Таблица 1 — Этапы разработки

№	Название этапа	Срок даты, %	Отчетность
1	Разработка технического задания	9.02.2026 – 28.02.2026, 5%	Утвержденное техническое задание и задание на выпуск-ную квалификацион-ную работу
2	Анализ требований и уточнение спецификаций (эскизный проект)	28.02.2026 – 14.03.2026, 20%	Спецификации ПО.
3	Проектирование структуры программной системы, проектирование компонентов (технический проект)	14.03.2026 – 01.04.2026, 25%	Схема структурная системы. Проектная документация: диаграмма вариантов использования, диаграмма вариантов использования, функциональная диаграмма, схемы алгоритмов разбора текста передаваемой программы и статического анализа, синтаксическая диаграмма консольного интерфейса ПО в виде РБНФ.
4	Реализация компонентов и автономное тестирование компонентов. Сборка и комплексное тестирование.	01.04.2026 – 14.05.2026, 25%	Тексты программных компонентов. Тесты, результаты тестирования.
5	Разработка документации.	14.05.2026 – 25.05.2026, 15%	Расчетно-пояснительная записка.
6	Прохождение нормоконтроля, проверка на	25.05.2026 – 6.06.2026, 5%	Иллюстративный материал, доклад, рецензия,

Продолжение таблицы 1

	антиплагиат, получение рецензии, подготовка доклада и предзащита.		справки о нормо-контроле и проценте плагиата.
7	Защита выпускной квалификационной работы.	1.06.2026 – 04.07.2026, 5%	—

## 9 ПОРЯДОК КОНТРОЛЯ И ПРИЕМА

### 9.1 Порядок контроля

Контроль выполнения осуществляется руководителем еженедельно.

### 9.2 Порядок защиты

Защита осуществляется перед государственной экзаменационной комиссией (ГЭК).

### 9.3 Срок защиты

Срок защиты определяется в соответствии с планом заседаний ГЭК.

## 10 ПРИМЕЧАНИЕ

В процессе выполнения работы возможно уточнение отдельных требований технического задания по взаимному согласованию руководителя и исполнителя.