



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

ОТЧЕТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Анализ синтаксиса и семантики стековых
языков программирования

Студент

ИУ6-73Б

(Группа)

(Подпись, дата)

В.К. Залыгин

(И.О. Фамилия)

Руководитель

(Подпись, дата)

Б.И. Бычков

(И.О. Фамилия)

Оценка _____

2025 г.

по теме **Анализ синтаксиса и семантики стековых языков программирования**

Залыгин Вячеслав Константинович
(Фамилия, имя, отчество)

Источник тематики (кафедра, предприятие, НИР) кафедра

Техническое задание: — выполнить анализ способов представления данных в распределенных системах, осуществить выбор способов для хранения и обработки данных об успеваемости студентов в электронном университете

- 1) Отчет на 25-30 листах формата А4.
- 2) Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
Необходимый иллюстративный графический материал включить в качестве рисунков в расчетно-пояснительную записку
- 3) Приложение А. Техническое задание на ВКРБ на 5-8 листах формата А4.

Руководитель

 (Подпись, дата)

 В.К. Залыгин
(И.О. Фамилия)

Студент

 (Подпись, дата)

 Б.И. Бычков
(И.О. Фамилия)

РЕФЕРАТ

РПЗ 14 с., 0 рис., 0 табл., 0 источн., 0 прил.

СТЕК, КОМПИЛЯТОР, СТЕКОВЫЙ ЯЗЫК, ЯЗЫК ПРОГРАММИРОВАНИЯ, ОБРАТНАЯ ПОЛЬСКАЯ ЗАПИСЬ

Объектом анализа являются стековые языки программирования.

Цель работы – проанализировать существующие подходы к построению стековых языков программирования, сделать анализ синтаксиса и семантики языков программирования, выявить идеи, которые лежат в основе построения компиляторов для данных языков.

В результате работы выполнен аналитический обзор таких аспектов стековых языков как: область применения, модель исполнения, используемые синтаксические конструкции и их семантика, типизация, статический и динамический анализ программ, возможные оптимизации, работа с памятью и подходы к построению стандартной библиотеки.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	6
1 Определение исполняющей машины	8
2 Обзор существующих решений	10
2.1 Конкатенативные языки	10
2.2 Другие языки	10
2.3 Выводы	10
3 Анализ синтаксиса	11
4 Анализ семантики	12
4.1 Типизация	12
4.2 Статический и динамический анализ	12
4.3 Оптимизации	12
4.4 Управление памятью	12
4.5 Стандартная библиотека	12
4.6 Выводы	12
ЗАКЛЮЧЕНИЕ	13
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	14

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

компилятор

ВВЕДЕНИЕ

Стековые (или стек-ориентированные) языки программирования характеризуются применением стека данных в качестве основного механизма передачи информации и хранения результатов вычислений. Стек-ориентированность позволяет программам на таких языках выглядеть компактно и эффективно исполняться. Исторически первым стековым языком стал Forth, разработанный Чарльзом Муром в начале 1970-х годов. Язык Forth изначально создавался для системного и низкоуровневого программирования, но в целом позволяет писать достаточно выразительный и понятный высокоуровневый код. Forth наиболее часто используется именно при разработке встроенных устройств. Например, этот язык применялся в ряде космических миссий NASA (включая проекты Voyager и Deep Impact), где программные системы работали на специализированных процессорах со стековой архитектурой (Harris RTX2000/2010).

В начале 2000-х годов возрос интерес исследователей к более высокоуровневым стековым языкам. Был предложен термин «конкатенативность» для обозначения семейства стековых языков, в которых программа воспринимается как функция, преобразующая последовательность аргументов в последовательность результатов, а конкатенация функций в тексте эквивалентна их композиции во время выполнения (отсюда и название — конкатенативные языки). Язык Joy, разработанный Манфредом фон Туном и выпущенный в 2001 году, полностью избегает переменных, предлагает фиксированный набор комбинаторов для работы со стеком. Joy позволил заложить в теорию стековых языков строгие формальные основы. Дальнейшим развитием идей Joy стал язык Factor, созданный Славой Пестовым и впервые опубликованный в 2003 году. В отличие от Forth и ранних стековых языков, Factor позиционируется как современный высокоуровневый язык общего назначения: он динамически типизирован, поддерживает объектно-ориентированные конструкции и автоматическое управление памятью, что делает его пригодным для создания как скриптов, так и крупных приложений. Несмотря на относительно узость области применения по

сравнению с наиболее популярными языками, стековые языки продолжают эволюционировать. Их принципы легли в основу ряда виртуальных машин. Так, современный байткод WebAssembly представляет собой стековую виртуальную машину, предназначенную для эффективной работы в Web-среде. Виртуальные машины языков Java (JVM) и C# (.NET) также используют стековые языки для близкого к машине представления программ.

В целом, благодаря экономичности и простоте реализации (а значит, и простоте портирования на разнообразные архитектуры) стековые языки нашли своё применение в низкоуровневых системах (системы реального времени, встраиваемые системы), различных виртуальных машинах (WebAssembly, JVM, .NET), графических процессорах и других специализированных областях.

1 Определение исполняющей машины

Типичная исполняющая модель стекового языка – это виртуальная стековая машина, имеющая по крайней мере один стек данных и набор инструкций для работы со стеком. Например, классическая реализация Forth оперирует двумя стеками: стек данных для хранения аргументов и результатов и отдельный стек возвратов для адресов возврата при вызове подпрограмм и хранения управляющей информации (например, параметров циклов). Исходный код напрямую отражает работу этой виртуальной машины и записывается в форме обратной польской нотации (Reverse Polish Notation, RPN) – сначала указываются операнды, затем операции над ними. Каждая элементарная операция или функция в стековом языке называется словом. Программа представляет собой последовательность слов, которые могут быть либо встроенными примитивами (например, арифметические операции, операции над стеком), либо определенными пользователем субпрограммами. В процессе выполнения поддерживается указатель инструкций (program counter), указывающий на текущую исполняемую команду, который последовательно продвигается по программе, если не изменён особыми словами управления потоком. Большинство инструкций в такой машине берёт необходимые операнды с вершины стека, выполняет вычисление и кладёт полученный результат обратно на стек. Например, исполнение последовательности из трёх инструкций – «push 3», «push 4», «add» – приведёт к тому, что в конце на вершине стека останется число 7. Когда выполнение программы завершено, результат обычно считается находящимся на вершине стека, откуда его можно извлечь для дальнейшего использования.

Таким образом, стековый язык можно рассматривать как язык для абстрактной машины со стеком. Основные компоненты модели исполнения – это стек как структура данных и набор команд, которые либо модифицируют содержимое стека (например, арифметические операции, берущие операнды из стека), либо управляют порядком исполнения (ветвления, вызовы и возвраты). Каждое слово программы транслируется в одну или несколько команд стековой машины. Такая абстракция лежит в основе многих систем: от высокоуровневых

языков (Forth, Factor и др.) до байткод-машин (тот же WebAssembly или Java Virtual Machine, JVM). В случае WebAssembly, например, стандарт прямо указывает, что Wasm представляет собой компактный двоичный формат инструкций для стековой виртуальной машины. Это подчёркивает универсальность стековой модели исполнения в современном программном обеспечении.

2 Обзор существующих решений

2.1 Конкатенативные языки

2.2 Другие языки

2.3 Выводы

3 Анализ синтаксиса

4 Анализ семантики

4.1 Типизация

4.2 Статический и динамический анализ

4.3 Оптимизации

4.4 Управление памятью

4.5 Стандартная библиотека

4.6 Выводы

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ