МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)

Факультет энергетический
Кафедра информатики, вычислительной техники и прикладной математики

КУРСОВОЙ ПРОЕКТ

по дисциплине: Теория языков программирования
на тему «Программная реализация транслятора»

Выполнил ст. гр. ИВТ-20,
Борисова Е.О.

Проверил доцент кафедры ИВТ и ПМ
Коган Е.С.

Чита
2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)

Факультет энергетический
Кафедра информатики, вычислительной техники и прикладной математики

ЗАДАНИЕ
на курсовой проект

по дисциплине: Теория языков программирования

Студенту Борисовой Екатерине Олеговне

направления подготовки 09.03.01 Информатика и вычислительная техника

1 Тема курсового проекта: Программная реализация транслятора.

2 Срок подачи студентом законченной работы: 18.12.2023 г.

3 Исходные данные к проекту: литературные источники, источники в сети интернет.

4 Перечень подлежащих разработке в курсовом проекте вопросов:

а) Постановка и анализ задачи;

б) Анализ данных;

в) Программная реализация.

5 Перечень графического материала (если имеется)

Дата выдачи задания 04.09.2023 г.

Руководитель курсового проекта _____ / Е. С. Коган/

Задание принял к исполнению

05.09.2023 г.

Подпись студента _____ / Е.О. Борисова/

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)

Факультет энергетический
Кафедра информатики, вычислительной техники и прикладной математики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по 09.03.01 Информатика и вычислительная техника

на тему «Программная реализация транслятора»

Выполнил студент группы ИВТ-20 Борисова Екатерина Олеговна

Руководитель работы: доцент кафедры ИВТ и ПМ Коган Евгения Семёновна.

РЕФЕРАТ

Пояснительная записка – 18 с, 6 источников

ТРАНСЛЯТОР, C++, PYTHON, ЛЕКСИЧЕСКИЙ АНАЛИЗАТОР, СИНТАКСИЧЕСКИЙ АНАЛИЗАТОР, ГРАММАТИКА, ЛЕКСЕМА

В работе описывается создания транслятора с языка программирования C++ на Python. Разработка транслятора производится на языке Python при помощи библиотеки rply. Интерфейс разработан при помощи библиотеки Tkinter.

СОДЕРЖАНИЕ

Введение 5

1 Постановка и анализ задачи 7

1.1 Описание предметной области 7

1.2 Постановка задачи 11

1.3 Обоснование выбора средств реализации 11

2 Анализ данных 12

2.1 Промежуточные данные 12

3 Программная реализация 14

Заключение 17

Список использованных источников 18

ВВЕДЕНИЕ

В наше время искусственные языки, которые используют текстовые форматы для описания различных предметных областей, находят применение не только в программировании, но и во многих других сферах. С их помощью создают описания структуры различных документов, 3D виртуальных миров, графических интерфейсов и других объектов, используемых в моделях и в реальном мире. Чтобы такие текстовые описания были правильно составлены и затем корректно интерпретированы, применяются специальные программы, называемые трансляторами. Разработка таких трансляторов сегодня является одной из наиболее актуальных областей в сфере разработки программного обеспечения.

1 Постановка и анализ задачи

1.1 Описание предметной области

Транслятор – программа или техническое средство, выполняющее трансляцию программы [1].

Трансляция программы – преобразование программы, представленной на одном из языков программирования, в программу на другом языке. Транслятор обычно выполняет также диагностику ошибок, формирует словари идентификаторов, выдаёт для печати текст программы и т. д.

Язык, на котором представлена входная программа, называется исходным языком, а сама программа – исходным кодом. Выходной язык называется целевым языком.

Одна из важных ролей транслятора состоит в сообщении об ошибках в исходной программе, обнаруженных во время трансляции.

Процесс отображение входной программы в эквивалентную ей выходную программу состоит из двух частей: анализ и синтез.

Анализ разбивает входную программу на составные части и накладывает на них грамматическую структуру. Затем использует эту структуру для создания промежуточного представления входной программы. Если анализ обнаруживает, что входная программа неверно составлена синтаксически или семантически, он должен выдать информативные сообщения об этом, чтобы пользователь мог исправить обнаруженные ошибки. Анализ также собирает информацию об исходной программе и сохраняет её в структуре данных, называемой таблицей идентификаторов (символов), которая передаётся вместе с промежуточным представлением синтезу.

Синтез строит требуемую целевую программу на основе промежуточного представления и информации из таблицы идентификаторов.

Анализ часто называют начальной стадией, а синтез – заключительной.

Если посмотреть на процесс трансляции более детально, можно увидеть, что он представляет собой последовательность фаз, каждая из которых преобразует одно из представлений входной программы в другое. На практике некоторые фазы могут быть объединены, а межфазное промежуточное представление может не строиться явно. Таблица идентификаторов, в которой хранится информация обо всей входной программе, используется всеми фазами транслятора [2].

C++ – это компилируемый, статически типизированный язык программирования общего назначения. Он поддерживает несколько парадигм программирования, включая процедурное программирование, объектно-ориентированное программирование и обобщенное программирование.

Язык предлагает обширную стандартную библиотеку, в которой есть популярные контейнеры и алгоритмы, средства ввода-вывода, поддержка регулярных выражений, многопоточность и многое другое. C++ сочетает в себе особенности как высокоуровневых, так и низкоуровневых языков программирования. По сравнению с предшествующим языком C, в C++ больше внимания уделяется поддержке объектно-ориентированного и обобщенного программирования.

Синтаксис C++ наследует свои корни от языка C, и одним из первоначальных принципов разработки было обеспечение совместимости с C [3].

Идентификатор в C++ – это имя, используемое для обозначения переменных, функций, классов, модулей или других пользовательских элементов. Идентификаторы начинаются с буквы (от A до Z или от a до z) или символа подчеркивания (_), за которым могут идти буквы, цифры (от 0 до 9) или дополнительные символы подчеркивания.

C++ чувствителен к регистру, поэтому, например, идентификаторы _d и Id считаются разными.

Блоки кода в C++ отделяются фигурными скобками, что позволяет структурировать код и даже размещать его в одной строке. Команды в языке завершаются точкой с запятой.

В C++ присутствуют ключевые слова, определяющие типы переменных:

– int – для целочисленных переменных;

– float – для переменных с плавающей запятой;

– string – для строк;

– char – для отдельных символов;

– bool – для логических переменных, которые могут быть либо true, либо false.

Условные операторы в C++ обозначаются ключевым словом if, за которым следует блок кода в фигурных скобках. Для описания циклов используются следующие конструкции:

– while – для циклов с предварительным условием;

– do...while – для циклов с постусловием;

– for – для циклов с заданным числом повторений.

Для логических операций C++ использует:

– && для логического "И";

– || для логического "ИЛИ";

– ! для логического "НЕ".

Для ввода и вывода данных на экран используются cin>> для ввода и cout<< для вывода.

Python – это интерпретируемый, динамически типизированный язык программирования общего назначения. Он поддерживает множество парадигм программирования, включая процедурное программирование, объектно-ориентированное программирование, функциональное программирование и другие.

Python предлагает богатую стандартную библиотеку, которая включает широкий спектр модулей и пакетов для работы с различными задачами, такими как обработка строк, работа с файлами, регулярные выражения, сети,

многопоточность и многое другое. Python ориентирован на высокоуровневое программирование, уделяя особое внимание простоте, читаемости кода и быстрой разработке [4].

Python предоставляет встроенные структуры данных, такие как списки, кортежи, словари и множества, что делает его гибким и универсальным. Благодаря поддержке интерпретации кода, Python позволяет разработчикам экспериментировать и вносить изменения в реальном времени без необходимости компиляции.

Синтаксис Python значительно отличается от языка C, делая акцент на простоте и читабельности. Одним из принципов разработки Python было обеспечение высокой читаемости кода с минимальным использованием специальных символов.

Идентификатор в Python – это имя, используемое для обозначения переменных, функций, классов, модулей или других пользовательских элементов. Идентификаторы начинаются с буквы (от A до Z или от a до z) или символа подчеркивания (_), за которым могут идти буквы, цифры или дополнительные символы подчеркивания. Python, как и C++, чувствителен к регистру, поэтому, например, идентификаторы varName и VarName считаются разными.

В отличие от C++, блоки кода в Python определяются отступами, а не фигурными скобками. Это делает код более структурированным и понятным на первый взгляд. Команды в Python не требуют точки с запятой в конце каждой строки, что упрощает синтаксис.

Python имеет множество встроенных типов данных, таких как:

– int – для целых чисел;
– float – для чисел с плавающей запятой;
– str – для строк;
– bool – для логических переменных, представляющих True или False;
– list, tuple, dict, и set – для коллекций, данных.

Для условных операторов в Python используется ключевое слово if, за которым следует блок кода с отступом. Для циклов применяются ключевые слова for и while. Цикл for обычно используется для перебора элементов коллекций или диапазонов, а while – для циклов с предварительным условием.

Для логических операций Python использует:

– and для логического "И";
– or для логического "ИЛИ";
– not для логического "НЕ".

Для ввода и вывода данных в Python используются встроенные функции, такие как input() для ввода и print() для вывода на экран.

1.2 Постановка задачи

Разрабатываемый транслятор будет иметь возможность выполнять трансляцию с языка программирования Python на C++.

При трансляции транслятор будет выполнять фазы лексического анализа, синтаксического анализа, а также генерации целевого исходного кода. Важно отметить, что на каждой из фаз транслятор должен осуществлять жёсткий контроль ошибок в анализируемом им исходном коде и по мере их нахождения уведомлять об этом пользователя.

1.3 Обоснование выбора средств реализации

Реализация выполнена на языке программирования Python с использованием библиотеки rply. Интерфейс написан с использованием библиотеки Tkinter.

2 Анализ данных

Данные, с которыми работает транслятор можно разделить на несколько групп:

– входные данные – текст программы на C++;

– промежуточные данные – это данные, используемые транслятором во время работы;

– выходные данные – текст программы на Python или C++.

2.1 Промежуточные данные

Перевод одного языка программирования в другой происходит поэтапно. Сначала исходный код поступает в лексический анализатор [5], который на выходе создает таблицы идентификаторов и лексем. Эти таблицы затем передаются синтаксическому анализатору [6], который формирует дерево разбора.

Лексема – это минимальная структурная единица языка, состоящая из базовых символов, без включения других структурных элементов. Лексемы в языке программирования включают идентификаторы, константы, ключевые слова, операторы и другие элементы.

Таблица лексем содержит весь текст исходной программы, обработанный лексическим анализатором. В ней могут присутствовать все типы лексем, и любая лексема может встречаться неограниченное количество раз.

Таблица идентификаторов включает только идентификаторы и константы. Она не содержит ключевых слов языка, знаков операций или разделителей. В таблице идентификаторов каждая лексема появляется лишь однажды, независимо от количества ее вхождений в исходный код.

Результат синтаксического анализа представляет собой дерево разбора. Оно отличается от абстрактного синтаксического дерева тем, что содержит

узлы для синтаксических правил, которые не влияют на семантику программы.

3 Программная реализация

Первая фаза работы транслятора – это лексический анализатор. Для этого сначала создаётся лексический анализатор на основе списка токенов tkIDsList и регулярных выражений CRegExList. tkIDsList содержит в себе идентификаторы токенов, которые представляют различные части синтаксиса языка C++, такие как ключевые слова, операторы, идентификаторы переменных, типы данных и так далее. CRegExList содержит регулярные выражения, которые соответствуют этим токенам, указывая, как определять их в тексте.

После построения лексического анализатора в него передаются входные данные (например, код на C++), где они разбираются на отдельные лексемы. Лексический анализатор последовательно сканирует текст, используя регулярные выражения, чтобы идентифицировать токены. При этом он игнорирует пробелы, комментарии и другие несущественные символы, которые не влияют на синтаксис.

Каждая найденная лексема преобразуется в токен, который имеет уникальный идентификатор (определённый в tkIDsList) и значение, соответствующее распознанной части кода. Эти токены сохраняются в последовательности, которая затем передаётся на дальнейшую обработку — в следующую фазу транслятора, в синтаксический анализатор.

Лексический анализатор также может генерировать ошибки, если в тексте обнаруживается что-то, что не соответствует ожидаемым паттернам. Например, это может быть неверный синтаксис, неизвестный символ или другое отклонение от правил языка C++. В случае ошибки лексер может указать точное место проблемы, что помогает синтаксическому анализатору и последующим этапам трансляции корректно реагировать на ошибки и предоставлять полезные сообщения пользователям.

После завершения лексического анализа, обработанные токены передаются в синтаксический анализатор для дальнейшей интерпретации и разбора.

После лексического анализа начинается вторая фаза трансляции – синтаксический анализ. Синтаксический анализатор, или парсер, берет поток токенов, созданных лексическим анализатором, и строит синтаксическое дерево, отражающее структуру кода на языке C++.

Класс CParser, парсер определяется с использованием библиотеки rply, которая предоставляет инструменты для построения грамматических правил и парсинга. CParser принимает список токенов и создает объект ParserGenerator, в котором устанавливаются грамматические правила с помощью декоратора @self.pg.production. Эти правила определяют, как синтаксический анализатор должен интерпретировать различные конструкции кода.

Синтаксический анализ состоит из:

– структуры правил: парсер CParser содержит правила для разбора различных частей кода C++, таких как условные операторы (if, else, elif), циклы (for, while), операции присвоения, определения функций и другие. Эти правила указывают, как токены объединяются для образования синтаксически правильных выражений;

– процесс парсинга: после определения всех грамматических правил парсер строится с помощью parser.pg.build(). Затем поток токенов, полученный из лексического анализатора, передается в парсер для обработки. Парсер использует грамматические правила, чтобы построить синтаксическое дерево, представляющее структуру кода;

– интерпретация кода: Во время парсинга парсер создает структуру, отражающую семантику кода. Например, он может интерпретировать условные выражения, циклы, присвоения и другие компоненты кода C++, преобразовывая их в представление на языке Python или в другую целевую форму.

Обработка ошибок**:**

– парсер может столкнуться с ошибками, если поток токенов не соответствует ожидаемым грамматическим правилам. Это может произойти из-за синтаксических ошибок в коде или из-за несоответствия между токенами и грамматическими правилами;

– для обработки ошибок используется специальный декоратор @self.pg.error, который определяет, что делать в случае ошибки парсинга. Обработчик ошибок вызывает исключение ValueError с информацией о токене, который вызвал ошибку.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были получены навыки по проектированию программ, их отладке и документированию.

В результате работы была написана программа – транслятор с языка программирования C++ на язык программирования Python. Этот процесс включает несколько этапов: лексический анализ, синтаксический анализ и возможную обработку ошибок.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Транслятор. – Текст: электронный // Wikipedia. – 2017. – URL: https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80 (дата обращения: 06.11.2023).

2. Языки программирования и методы трансляции: учеб. Пособие» / О.В. Молдованова. – Новосибирск: ФГОБУ ВПО «СибГУТИ», 2012. – 39 с. (дата обращения: 06.11.2023).

3. Синтаксис C++. – Текст: электронный // Заметки по алгоритмическому программированию. – 2022. – URL: https://notes.algoprog.ru/cpp/syntax.html (дата обращения: 06.11.2023).

4. Синтаксис Python для начинающих. – Текст: электронный // Tochmah. – 2023. – URL: https://tochmah.ru/sintaksis-python-dlya-nachinayuschih/ (дата обращения: 06.11.2023).

5. Лексический анализатор. – Текст: электронный // Wikipedia. – 2022. – URL: https://ru.wikipedia.org/wiki/Лексический_анализ (дата обращения: 06.11.2023).

6. Синтаксический анализатор – Текст: электронный // Tochmah. – 2020. – URL: https://ru.wikipedia.org/wiki/Синтаксический_анализатор (дата обращения: 06.11.2023).