

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Дальневосточный федеральный университет»**

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ (ШКОЛА)**

**Департамент программной инженерии и искусственного интеллекта**

РАЗРАБОТКА ТРАНСЛЯТОРА, ПЕРЕВОДЯЩЕГО ПОДМНОЖЕСТВО ЯЗЫКА **RUST** В ЭКВИВАЛЕНТНОЕ ПОДМНОЖЕСТВО ЯЗЫКА **С++**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

по дисциплине «Теория языков программирования и компиляторы»

по образовательной программе подготовки бакалавров по направлению 09.03.04 Программная инженерия

Выполнил студент

гр. Б9120-09.03.04прогин

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Орлов Г. А.

Руководитель:

ассистент департамента ПИиИИ

Глазырин М.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

(подпись)

г. Владивосток

2024 г

**Оглавление**

[Введение 4](#_Toc157440174)

[1 Неформальная постановка задачи 6](#_Toc157440175)

[2 Синтаксис входного языка 7](#_Toc157440176)

[3 Контекстные условия языка Rust 8](#_Toc157440177)

[4 Контекстные условия языка C++ 9](#_Toc157440178)

[5 Таблица соответствия языков 10](#_Toc157440179)

[6 Проект лексического анализатора 11](#_Toc157440180)

[6.1 Модель данных 11](#_Toc157440181)

[6.2 Таблица ключевых слов 12](#_Toc157440182)

[6.3 Таблица зарезервированных имен 13](#_Toc157440183)

[6.4 Типы лексем 14](#_Toc157440184)

[6.5 Типы ошибок 15](#_Toc157440185)

[6.6 Конечный автомат лексического анализатора 16](#_Toc157440186)

[7 Проект семантического анализатора 17](#_Toc157440187)

[7.1 Модель данных 17](#_Toc157440188)

[7.2 Описание алгоритма семантического анализа кода 19](#_Toc157440189)

[7.3 Пример дерева разбора 20](#_Toc157440190)

[7.4 Типы ошибок 22](#_Toc157440191)

[8 Проект семантического анализатора 23](#_Toc157440192)

[8.1 Методы проверки нарушений контекстных условия 23](#_Toc157440193)

[8.2 Типы ошибок 24](#_Toc157440194)

[9 Проект генератора кода 25](#_Toc157440195)

[9.1 Метод кодогенерации 25](#_Toc157440196)

[9.2 Примеры кодогенерации 26](#_Toc157440197)

[10 Тестирование 28](#_Toc157440198)

[Заключение 68](#_Toc157440199)

[Список литературы 69](#_Toc157440200)

# Введение

В данном отчёте рассматривается создание транслятора с языка Rust на C++. Транслятор - это программное средство, которое преобразует исходный код с одного языка программирования (в данном случае Rust) на другой язык программирования (C++).

Язык программирования представляет собой формализованный набор символов и правил, используемых для написания программного кода. В данном проекте входным языком является Rust, мощный системный язык программирования, который обеспечивает безопасность памяти и высокую производительность. Выходным языком является C++, популярный и широко используемый язык программирования, который обладает мощными возможностями и оптимизацией.

Основная цель создания данного транслятора - обеспечить возможность переноса существующего кода из языка Rust в C++. Это может быть полезно в случае необходимости использования существующих библиотек, инструментов и ресурсов, доступных только на языке C++.

В следующих разделах отчёта будут рассмотрены подробности реализации транслятора, используемые технологии, возможности и ограничения данного подхода, а также результаты тестирования и дальнейшие возможности развития проекта.

**Цель курсового проекта:** разработать проект транслятора, который преобразует программу, содержащую подмножество языка Rust, в программу, содержащую подмножество языка C++ генерируя эквивалентный исходный код.

**Задачи:**

1. Описать грамматику подмножества входного языка;
2. Описать контекстные условия входного и выходного языков;
3. Описать соответствие конструкций входного и выходного языков;
4. Разработать проект лексического анализатора;
5. Разработать проект синтаксического анализатора;
6. Разработать проект семантического анализатора;
7. Разработать проект генератора кода выходного языка;
8. Реализовать транслятор входного языка в выходной;
9. Выполнить тестирование разработанного транслятора.

# 1 Неформальная постановка задачи

Требуется разработать транслятор с языка Rust на C++, который способен преобразовывать исходный код на языке Rust в эквивалентный код на языке C++.

Входной язык программирования, Rust, характеризуется следующим подмножеством функциональности:

Типы данных: целочисленные (integer) и другие стандартные типы данных (например, float).

Операторы: арифметические операторы (+, -, \*, /), логические операторы (==, !=, <, >) и прочие операторы (например, and).

Операторы циклов: while-цикл.

Операторы ветвления: if-else условные операторы.

Стандартные функции: например, функция print для вывода на экран.

Транслятор должен обеспечивать корректное преобразование синтаксиса и семантики кода на языке Rust в аналогичный код на языке C++. Результат выполнения программы транслятора должен быть эквивалентен оригинальной программе на языке Rust, с сохранением структуры и логики программы.

Также требуется обеспечить обработку возможных ошибок, которые могут возникнуть при трансляции кода, и предоставить информативные сообщения об ошибках для облегчения их исправления.

# 2 Синтаксис входного языка

### Обозначения

Перенос строки: \n

Табуляция: \t

Блок кода: {...}

Комментарии: // для однострочных комментариев

### Грамматика

<PROGRAM> ::= <ITEM>\*

<ITEM> ::= <FN\_ITEM> | <STRUCT\_ITEM> | <IMPL\_ITEM>

<FN\_ITEM> ::= fn <IDENT> ( <PARAM\_LIST> ) -> <TYPE> <BLOCK>

<STRUCT\_ITEM> ::= struct <IDENT> { <FIELD\_LIST> }

<IMPL\_ITEM> ::= impl <IDENT> { <FN\_ITEM>\* }

<BLOCK> ::= { <STATEMENT>\* }

<STATEMENT> ::= <LET\_STATEMENT> | <EXPR>;

<LET\_STATEMENT> ::= let <IDENT> = <EXPR>;

<EXPR> ::= <IDENT> | <LITERAL> | <EXPR> + <EXPR> | <EXPR> - <EXPR> | <EXPR> \* <EXPR> | <EXPR> / <EXPR> | <BLOCK>

<PARAM\_LIST> ::= <PARAM> | <PARAM> , <PARAM\_LIST>

<FIELD\_LIST> ::= <FIELD> | <FIELD> , <FIELD\_LIST>

<PARAM> ::= <IDENT> : <TYPE>

<FIELD> ::= <IDENT> : <TYPE>

<TYPE> ::= <IDENT> | <IDENT> <ANGLE\_BRACKET\_TYPES>

<ANGLE\_BRACKET\_TYPES> ::= < <TYPE\_LIST> >

<TYPE\_LIST> ::= <TYPE> | <TYPE> , <TYPE\_LIST>

<IDENT> ::= [a-zA-Z\_][a-zA-Z0-9\_]\*

<LITERAL> ::= <INTEGER> | <BOOLEAN>

<INTEGER> ::= [0-9]+

<BOOLEAN> ::= true | false

# 3 Контекстные условия языка Rust

**Ключевые слова:** Rust имеет определенный набор ключевых слов, таких как let, if, else, while, fn и т. д. Ключевые слова не могут быть использованы как идентификаторы.

**Блочная структура:** программы Rust имеют блочную структуру с помощью фигурных скобок {}. Блоки могут быть вложенными и используются для группировки логически связанных операций или инструкций.

**Объявление переменных:** переменные в Rust должны быть объявлены с использованием ключевого слова let, за которым следует имя переменной и тип (необязательно). Переменные в Rust имеют строгую типизацию.

**Типы данных:** в Rust доступны следующие типы данных: целочисленные типы (i32, u32 и т. д.), дробные типы (f32, f64), логический тип (bool), символьный тип (char) и другие композитные типы данных (например, массивы, кортежи, структуры и перечисления).

**Операторы условия:** в Rust используются операторы условия if-else для выполнения различных действий в зависимости от заданного условия.

**Операторы цикла:** в Rust используется оператор цикла while для повторения блока кода до тех пор, пока заданное условие остается истинным.

**Функции:** в Rust функции объявляются с использованием ключевого слова fn, за которым следует имя функции, параметры (необязательно) и тип возвращаемого значения (необязательно).

**Модули:** Rust поддерживает модули, которые используются для логической группировки кода и обеспечения его организации и повторного использования.

**Комментарии:** в Rust можно добавлять комментарии в код с использованием символов // для однострочных комментариев.

# 4 Контекстные условия языка C++

**Ключевые слова:** C++ имеет определенный набор ключевых слов, таких как int, char, float, if, else, while, for, и т. д. Ключевые слова не могут быть использованы как идентификаторы.

**Блочная структура:** программы на C++ имеют блочную структуру с помощью фигурных скобок {}. Блоки могут быть вложенными и используются для группировки логически связанных операций или инструкций.

**Объявление переменных:** переменные в C++ должны быть объявлены с использованием ключевого слова, за которым следует имя переменной и тип (необязательно). Переменные в C++ имеют строгую типизацию.

**Типы данных:** в C++ доступны следующие типы данных: целочисленные типы (int, char, long и т. д.), дробные типы (float, double), логический тип (bool), символьный тип (char) и другие композитные типы данных (например, массивы, структуры и классы).

**Операторы условия:** в C++ используются операторы условия if-else для выполнения различных действий в зависимости от заданного условия.

**Операторы цикла:** в C++ используются операторы цикла while, do-while и for для повторения блока кода до тех пор, пока заданное условие остается истинным.

**Функции:** в C++ функции объявляются с использованием ключевого слова, за которым следует имя функции, параметры (необязательно) и тип возвращаемого значения (необязательно).

**Классы и объекты:** в C++ можно определять классы и создавать объекты на их основе. Классы могут содержать переменные-члены и функции-члены.

**Комментарии:** в C++ можно добавлять комментарии в код с использованием символов // для однострочных комментариев и /\* \*/ для многострочных комментариев. Комментарии игнорируются компилятором и служат для описания кода или временного отключения части кода.

# 5 Таблица соответствия языков

Таблица 1 - Таблица соответствия языков

|  |  |  |
| --- | --- | --- |
| **Языковая структура** | **Rust** | **C++** |
| Блочная структура | {} | {} |
| Объявление переменных | let | int, char, float, etc. |
| Типы данных | bool, i32, u32, f32, etc. | bool, int, float, char, etc. |
| Операторы условия | if, else, case | if, else, match |
| Операторы цикла | while, for | while, for |
| Функции | fn | function, void, etc. |
| Классы и объекты | struct | class, object |

# 6 Проект лексического анализатора

Лексер (также известный как лексический анализатор) - это компонент в компиляторе или интерпретаторе, который преобразует последовательность символов в программе (такую как исходный код) в поток лексем. Лексемы представляют собой минимальные синтаксические единицы языка программирования, такие как ключевые слова, операторы, идентификаторы, константы и другие.

Лексический анализ - это процесс разбора исходного кода на отдельные лексемы, определение их типов и последовательности, а также их передачи синтаксическому анализатору для дальнейшей обработки. Таким образом, лексический анализ позволяет компилятору или интерпретатору понять структуру программы на основе ее лексических единиц.

В компиляторе или интерпретаторе лексер обычно работает следующим образом:

Сканирование: лексер последовательно читает исходный код и выделяет лексемы.

Классификация: лексемы классифицируются по типам данных, таким как операторы, идентификаторы, ключевые слова и т.д.

Генерация лексического потока: лексер создает поток лексем, который передается синтаксическому анализатору для дальнейшей обработки.

В результате лексического анализа и сгенерированного потока лексем компилятор или интерпретатор "понимает" структуру программы и может выполнять следующие шаги, такие как синтаксический анализ, анализ семантики и генерация исполняемого кода или выполнение программы.

## 6.1 Модель данных

Какие классы/структуры использовали, почему. Диаграммы классов можно вставить, если есть. Описать как представлены лексемы и токены в памяти.

## 6.2 Таблица ключевых слов

Ключевые слова - это зарезервированные слова в языках программирования, которые имеют особое значение и не могут быть использованы в качестве идентификаторов (имен переменных, функций и т.д.).

Ключевые слова в языке Rust отличаются от остальных лексем (т.е. символов и последовательностей символов) тем, что они имеют заранее определенное значение и назначение в языке. Компилятор Rust распознает эти ключевые слова и использует их для определения структуры и семантики программы.

* as
* break
* const
* continue
* crate
* else
* enum
* extern
* false
* fn
* for
* if
* impl
* in
* let
* loop
* match
* mod
* move
* mut
* pub
* ref
* return
* self
* static
* struct
* super
* trait
* true
* type
* unsafe
* use
* where
* while

## 6.3 Таблица зарезервированных имен

В языке программирования Rust существуют зарезервированные имена, которые в настоящее время не используются, но они зарезервированы для будущего использования. Они имеют те же ограничения, что и строгие ключевые слова. Вот список этих зарезервированных имен:

* abstract
* become
* box
* do
* final
* macro
* override
* priv
* try
* typeof
* unsized
* virtual
* yield

## 6.4 Типы лексем

Таблица 2 - Типы лексем

|  |  |
| --- | --- |
| **Идентификатор** | **Ключевое слово, зарезервированное имя** |
| Оператор | = : ASSIGN  == : IS\_EQUAL  != : INEQUALITY  < : LESS  > : MORE  + : PLUS  - : MINUS  \* : MULTYPLE  / : DIVIDE  && : AND  || : OR |
| Спец символы | ( : LEFT\_BRACKET  ) : RIGHT\_BRACKET  [ : LEFT\_SQUARE\_BRACKET  ] : RIGHT\_SQUARE\_BRACKET  ; : SEMICOLON  , : COMMA |

## 6.5 Типы ошибок

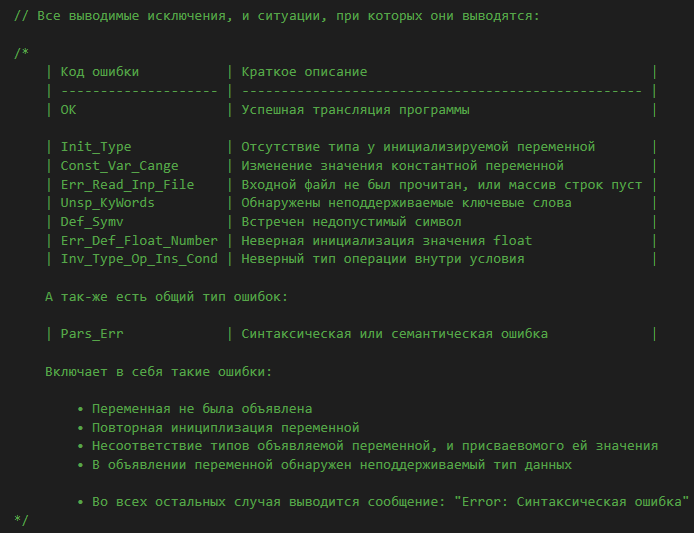


Рисунок 1 – Типы ошибок Лексического анализатора

## 6.6 Конечный автомат лексического анализатора

В конечном автомате кодируется информация о возможных последовательностях символов, которые могут встречаться в токенах2. Каждое состояние в автомате соответствует определенному местоположению в структуре токена, а переходы между состояниями определяются входными символами.

Таким образом, конечный автомат лексического анализатора позволяет эффективно сканировать исходный код и разбивать его на токены, которые затем используются на следующих этапах процесса компиляции.

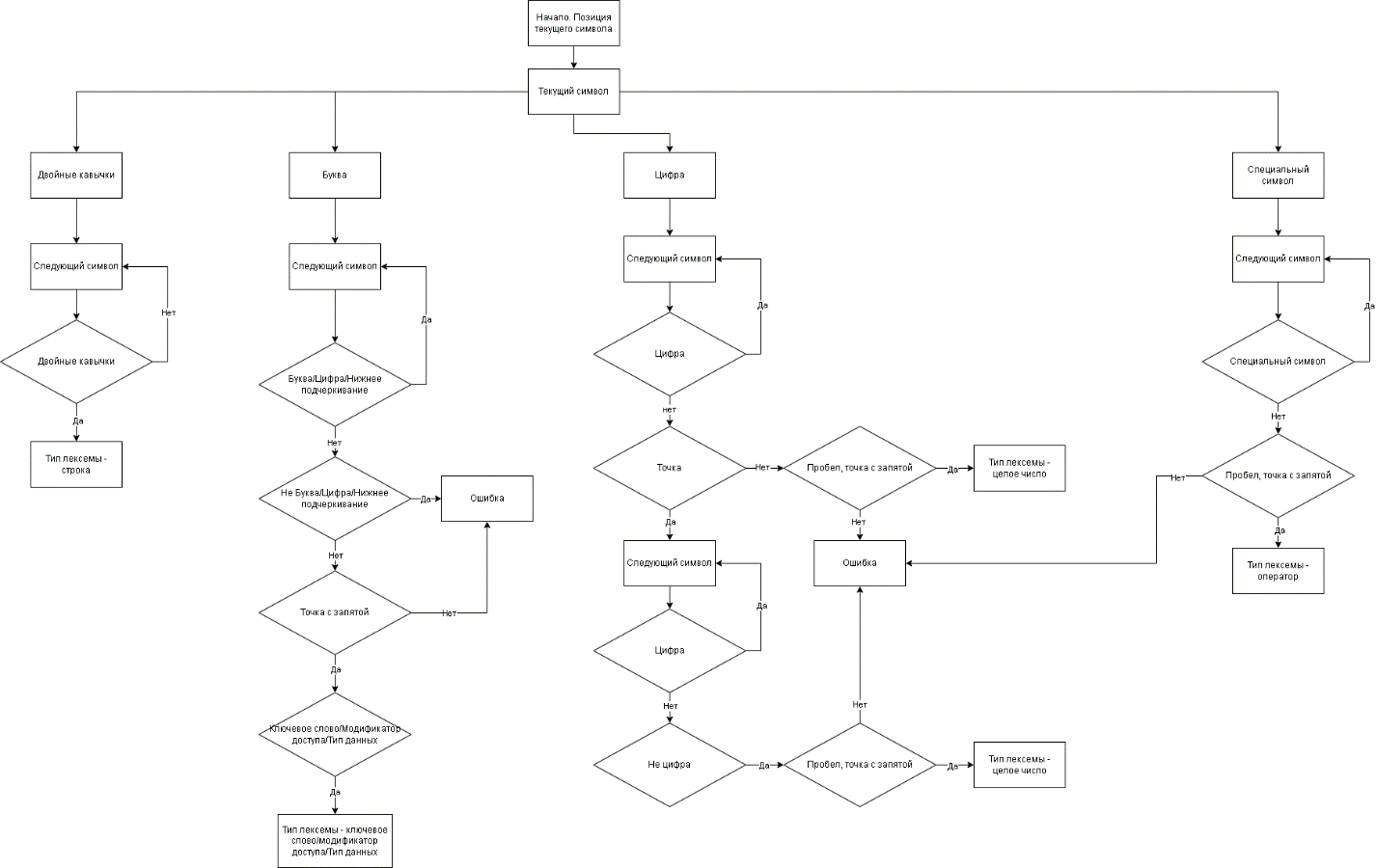


Рисунок 2 – Конечный автомат лексического анализатора

# 7 Проект семантического анализатора

Семантический анализатор (или семантический анализ) - это процесс анализа смысловой структуры текста с целью понимания его семантического значения и выявления связей между различными элементами.

## 7.1 Модель данных

Для реализации синтаксического анализатора в рамках данного проекта был выбран язык программирования C#.

В процессе реализации синтаксического анализатора были использованы различные средства и библиотеки, доступные для C#, включая классы для работы со строками, коллекциями и файлами. Это позволило эффективно реализовать разбор и анализ исходного кода, а также обработку ошибок и вывод результатов.

Диаграмма моделей данных для синтаксического анализатора включает в себя следующие компоненты:

1. **Токены**: Это базовые единицы, на которые разбивается исходный код. Токены могут быть ключевыми словами, идентификаторами, литералами, операторами и т.д.
2. **Продукции грамматики**: Это правила, которые определяют, как токены могут быть объединены для формирования более сложных структур, таких как выражения, операторы, функции и т.д.
3. **Символы грамматики**: Это элементы, которые могут быть как терминальными (токенами), так и нетерминальными (продукциями грамматики).
4. **Дерево разбора**: Это структура данных, которая представляет синтаксическую структуру исходного кода. Узлы дерева представляют символы грамматики, а ребра представляют отношения между ними.



Рисунок 3 – Модель данных

## 7.2 Описание алгоритма семантического анализа кода

1. **Парсинг исходного кода:** Используйте синтаксический анализатор Rust для преобразования исходного кода в абстрактное синтаксическое дерево (AST).
2. **Обход дерева:** Обойдите AST и для каждого узла выполните соответствующую семантическую проверку. Это может включать проверку типов, проверку жизненного цикла, проверку доступности и т.д.
3. **Сбор информации:** Соберите информацию, такую как типы переменных, функций и их области видимости, для использования в последующих проверках.
4. **Проверка ошибок:** Проверьте наличие ошибок, таких как использование необъявленных переменных, несоответствие типов, неправильное использование ключевых слов и т.д.
5. **Генерация отчета:** Сгенерируйте отчет об ошибках или предупреждениях для представления пользователю.

## 7.3 Пример дерева разбора

Дерево разбора парсера:

FUNCTION

NAME : MAIN

ARGUMENT

NO\_ARGUMENTS

BODY

PRINT

ARGUMENTS

STRING : "Hello, world!"

FUNCTION

NAME : 123

ARGUMENT

Дерево разбора парсера:

INITIALIZATION\_VAR

NAME : length

TYPE : INT

VALUE : 2

ASSIGMENT : =

ID : length

NUM\_INT : 4

INITIALIZATION\_VAR

NAME : width

TYPE : INT

VALUE : 8

INITIALIZATION\_VAR

NAME : result

TYPE : INT

VALUE : 0

ASSIGMENT : =

ID : result

OPERATION : \*

ID : length

ID : width

PRINT

ARGUMENTS

STRING : "length {} \* width {} = result {}"

ID : length

ID : width

ID : result

Дерево разбора парсера:

INITIALIZATION\_VAR

NAME : a

TYPE : INT

VALUE : 7

INITIALIZATION\_VAR

NAME : bool1

TYPE : BOOL

VALUE : false

INITIALIZATION\_VAR

NAME : char1

TYPE : CHAR

VALUE : 'a'

INITIALIZATION\_VAR

NAME : bool2

TYPE : BOOL

VALUE : false

INITIALIZATION\_VAR

NAME : char2

TYPE : CHAR

VALUE : 'a'

INITIALIZATION\_VAR

NAME : str1

TYPE : STRING

VALUE : "123"

Дерево разбора парсера:

FUNCTION

NAME : MAIN

ARGUMENT

NO\_ARGUMENTS

BODY

INITIALIZATION\_VAR

NAME : a

TYPE : INT

VALUE : 0

INITIALIZATION\_VAR

NAME : b

Дерево разбора парсера:

INITIALIZATION\_VAR

NAME : a

TYPE : INT

VALUE : 25

INITIALIZATION\_VAR

NAME : b

TYPE : INT

VALUE : 4

ASSIGMENT : =

ID : b

OPERATION : \*

ID : b

ID : b

Дерево разбора парсера:

INITIALIZATION\_VAR

NAME : a

TYPE : INT

VALUE : 25

INITIALIZATION\_VAR

NAME : b

TYPE : INT

VALUE : 4

ASSIGMENT : =

ID : b

OPERATION : \*

ID : b

ID : b

## 7.4 Типы ошибок

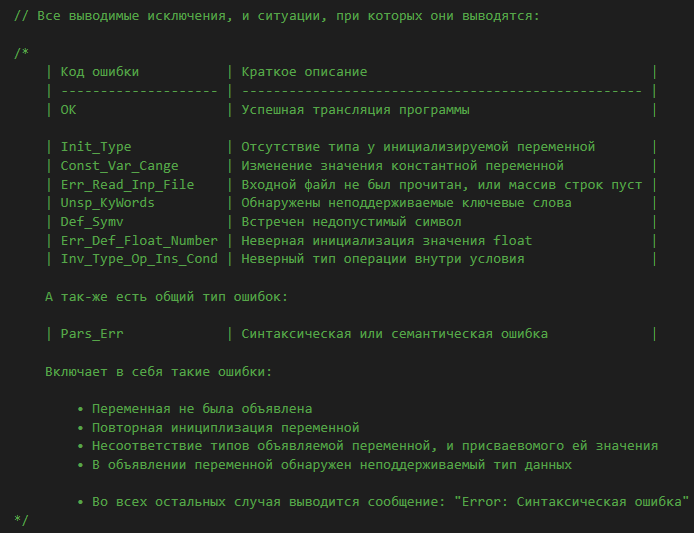


Рисунок 4 – Типы ошибок Парсера

# 8 Проект семантического анализатора

Семантический анализатор - это компонент транслятора, который обрабатывает семантические отношения в исходном коде. Он следует за синтаксическим анализатором и использует дерево разбора, созданное синтаксическим анализатором, чтобы проверить исходный код на семантическую корректность.

Семантический анализатор проверяет код на наличие ошибок, которые не могут быть обнаружены на этапе синтаксического анализа. Например, он проверяет, что переменные и функции объявлены перед использованием, что функции вызываются с правильным количеством аргументов, и что значения присваиваются переменным правильного типа.

## 8.1 Методы проверки нарушений контекстных условия

Контекстные условия проверяются, исходя из типа значения, присваемового инициализируемой переменной. При её инициализации.

## 8.2 Типы ошибок

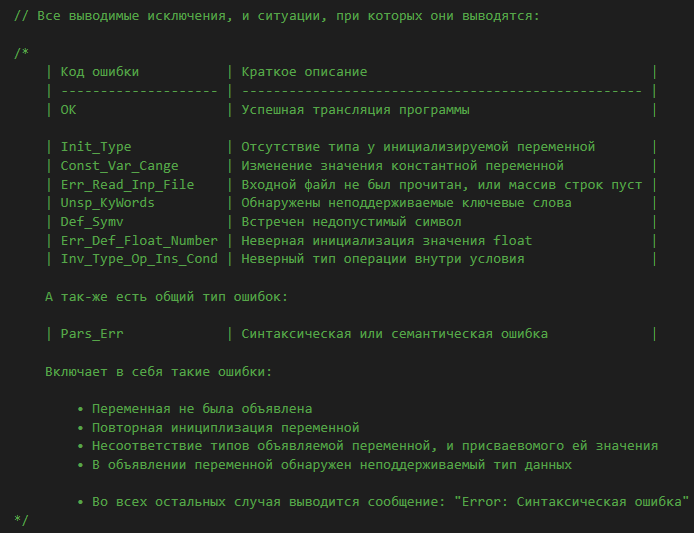


Рисунок 5 – Типы ошибок Семантического анализатора

# 9 Проект генератора кода

Генератор кода в трансляторе является одной из важных компонентов, отвечающих за преобразование входного языка программирования в целевой код, который может быть выполнен на целевой платформе или архитектуре. Он работает в конце процесса трансляции, после анализа и синтеза исходного кода.

Генератор кода получает на вход абстрактное синтаксическое дерево (AST) или промежуточное представление программы (IR) и использует различные оптимизации и техники для преобразования кода в целевой язык или инструкции, понятные целевому исполнительному окружению.

Задачи генератора кода включают выбор правильных инструкций, управление памятью и регистрами, оптимизацию вычислений и преобразование алгоритмов в эффективный и оптимальный код. В зависимости от целевой платформы, генератор кода может выполнять дополнительные задачи, такие как генерация машинного кода или байт-кода.

## 9.1 Метод кодогенерации

Метод кодогенерации сход с методом построения дерева разбора Парсера – модуль кодогенерации, в цикле считывает ключевые слова, из дерева разбора парсера. И если есть нужная конструкция (их последовательность), с нужными аргументами – то он генерирует код, необходимой структуры.

Если конструкция неполная, или построена неправильно – он не выдаёт ошибку, а просто игнорирует её.

## 9.2 Примеры кодогенерации

---------------------------------------------------

Кодогенератор С++

#include <iostream>

using namespace std;

int main()

{

int a = 0;

for (int i = 0; i < 5; i++)

{

std::cout << "Число: " << i << std::endl;

}

return 0;

}

---------------------------------------------------

Кодогенератор С++

#include <iostream>

using namespace std;

int main()

{

int a = 10;

if (a < 5)

{

std::cout << "a меньше 5" << std::endl;

}

else if (a == 5)

{

std::cout << "a равно 5" << std::endl;

}

else

{

std::cout << "a больше 5" << std::endl;

}

return 0;

}

---------------------------------------------------

Кодогенератор С++

#include <iostream>

using namespace std;

int main()

{

int j = 0;

while(j < 5)

{

std::cout << "Число: " << j << std::endl;

j += 1;

}

return 0;

}

---------------------------------------------------

Кодогенератор С++

#include <iostream>

using namespace std;

int main ()

{

another\_function(5);

return 0;

}

void another\_function(int x)

{

std::cout << "The value of x is: " << x << std::endl;

}

---------------------------------------------------

Кодогенератор С++

#include <iostream>

using namespace std;

int main()

{

int a = 5;

int b = 10;

int c = 2;

int result1;

int result2;

result1 = a + b \* a;

std::cout << "Результат без скобок: " << result1 << std::endl;

result2 = a + b \* a;

std::cout << "Результат со скобками: " << result2 << std::endl;

return 0;

}

---------------------------------------------------

Кодогенератор С++

#include <iostream>

using namespace std;

int main()

{

int a = 10;

a += 5;

return 0;

}

---------------------------------------------------

# 10 Тестирование

### Любая простая ошибка:

// Провека ошибки +++

​

fn main() {

   println!("Hello, world!");

   fn 123

}

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// Провека ошибки +++

​

fn main() {

   println!("Hello, world!");

   fn 123

}

​

Все распознанные лексемы:

​

0    fn         -  FUNCTION

1    main       -  MAIN

2   (          -  OPEN\_BRACKET

3   )          -  CLOSED\_BRACKET

4   {          -  START\_VOID

5    println!   -  PRINT

6   (          -  OPEN\_BRACKET

7    "Hello, world!"  -  STRING

8   )          -  CLOSED\_BRACKET

9   ;          -  END\_LINE

10   fn         -  FUNCTION

11   123        -  NUM\_INT

12   }          -  END\_VOID

Дерево разбора парсера:

FUNCTION

 NAME : MAIN

 ARGUMENT

   NO\_ARGUMENTS

 BODY

   PRINT

     ARGUMENTS

       STRING : "Hello, world!"

   FUNCTION

     NAME : 123

     ARGUMENT

​

Семантический анализатор:

​

————————————————————————————————————

В процессе построения дерева разбора

программы произошла ошибка!

​

Подробности ошибки:

                      ▼

( "Hello, world!" ) ; fn 123 }

​

Error: Синтаксическая ошибка.

---------------------------------------------------

​

Время выполнения программы: 0,46 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: Pars\_Err

​

Трансляция программы остановлена с исключением:

Error: Синтаксическая ошибка.

### Неинициализированная переменная:

// Провека ошибки +++

​

fn main() {

   let a = 2;

   let b = 1;

​

   c = a + b;

​

   println!("с = {}", с);

}

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// Провека ошибки +++

​

fn main() {

   let a = 2;

   let b = 1;

​

   c = a + b;

​

   println!("с = {}", с);

}

​

Все распознанные лексемы:

​

0    fn         -  FUNCTION

1    main       -  MAIN

2   (          -  OPEN\_BRACKET

3   )          -  CLOSED\_BRACKET

4   {          -  START\_VOID

5    let        -  LET

6    a          -  ID

7    =          -  ASSIGNMENT\_OPERATION\_\_SET

8    2          -  NUM\_INT

9   ;          -  END\_LINE

10   let        -  LET

11   b          -  ID

12   =          -  ASSIGNMENT\_OPERATION\_\_SET

13   1          -  NUM\_INT

14   ;          -  END\_LINE

15   c          -  ID

16   =          -  ASSIGNMENT\_OPERATION\_\_SET

17   a          -  ID

18   +          -  ARITHMETIC\_OPERATION\_\_ADD

19   b          -  ID

20   ;          -  END\_LINE

21   println!   -  PRINT

22   (          -  OPEN\_BRACKET

23   "с = {}"   -  STRING

24   ,          -  COMMA

25   с          -  ID

26   )          -  CLOSED\_BRACKET

27   ;          -  END\_LINE

28   }          -  END\_VOID

Дерево разбора парсера:

FUNCTION

 NAME : MAIN

 ARGUMENT

   NO\_ARGUMENTS

 BODY

   INITIALIZATION\_VAR

     NAME : a

     TYPE : INT

     VALUE : 2

 INITIALIZATION\_VAR

   NAME : b

   TYPE : INT

   VALUE : 1

! В результате разбора исходной программы возникла ошибка!

Переменная 'c' не была объявлена.

​

Семантический анализатор:

​

————————————————————————————————————

В процессе построения дерева разбора

программы произошла ошибка!

​

Подробности ошибки:

        ▼

b = 1 ; c = a +

​

Семантическая ошибка: Переменная 'c' не была объявлена.

---------------------------------------------------

​

Время выполнения программы: 0,31 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: Pars\_Err

​

Трансляция программы остановлена с исключением:

Семантическая ошибка: Переменная 'c' не была объявлена.

### Неинициализированная переменная:

// Провека ошибки +++

​

a = 5;

let mut a = 25;

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// Провека ошибки +++

​

a = 5;

let mut a = 25;

​

Все распознанные лексемы:

​

0    a          -  ID

1    =          -  ASSIGNMENT\_OPERATION\_\_SET

2    5          -  NUM\_INT

3   ;          -  END\_LINE

4    let        -  LET

5    mut        -  MUT

6    a          -  ID

7    =          -  ASSIGNMENT\_OPERATION\_\_SET

8    25         -  NUM\_INT

9   ;          -  END\_LINE

Дерево разбора парсера:

! В результате разбора исходной программы возникла ошибка!

Переменная 'a' не была объявлена.

​

Семантический анализатор:

​

————————————————————————————————————

В процессе построения дерева разбора

программы произошла ошибка!

​

Подробности ошибки:

​

= 5 ;

​

Семантическая ошибка: Переменная 'a' не была объявлена.

---------------------------------------------------

​

Время выполнения программы: 0,15 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: Pars\_Err

​

Трансляция программы остановлена с исключением:

Семантическая ошибка: Переменная 'a' не была объявлена.

### Неинициализированная переменная:

// Провека ошибки +++

​

let mut num = 2;

num = 4;

nam = 4;

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// Провека ошибки +++

​

let mut num = 2;

num = 4;

nam = 4;

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    num        -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    2          -  NUM\_INT

5   ;          -  END\_LINE

6    num        -  ID

7    =          -  ASSIGNMENT\_OPERATION\_\_SET

8    4          -  NUM\_INT

9   ;          -  END\_LINE

10   nam        -  ID

11   =          -  ASSIGNMENT\_OPERATION\_\_SET

12   4          -  NUM\_INT

13   ;          -  END\_LINE

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : num

 TYPE : INT

 VALUE : 2

ASSIGMENT : =

 ID : num

 NUM\_INT : 4

! В результате разбора исходной программы возникла ошибка!

Переменная 'nam' не была объявлена.

​

Семантический анализатор:

​

————————————————————————————————————

В процессе построения дерева разбора

программы произошла ошибка!

​

Подробности ошибки:

          ▼

num = 4 ; nam = 4 ;

​

Семантическая ошибка: Переменная 'nam' не была объявлена.

---------------------------------------------------

​

Время выполнения программы: 0,19 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: Pars\_Err

​

Трансляция программы остановлена с исключением:

Семантическая ошибка: Переменная 'nam' не была объявлена.

### Простой пример программы:

// +++

​

let mut num = 2;

num = 4;

println!("num = {}", num);

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

let mut num = 2;

num = 4;

println!("num = {}", num);

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    num        -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    2          -  NUM\_INT

5   ;          -  END\_LINE

6    num        -  ID

7    =          -  ASSIGNMENT\_OPERATION\_\_SET

8    4          -  NUM\_INT

9   ;          -  END\_LINE

10   println!   -  PRINT

11   (          -  OPEN\_BRACKET

12   "num = {}"  -  STRING

13   ,          -  COMMA

14   num        -  ID

15   )          -  CLOSED\_BRACKET

16   ;          -  END\_LINE

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : num

 TYPE : INT

 VALUE : 2

ASSIGMENT : =

 ID : num

 NUM\_INT : 4

PRINT

 ARGUMENTS

   STRING : "num = {}"

   ID : num

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main()

{

 int num = 2;

 num = 4;

 std::cout << "num = " << num << std::endl;

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,35 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Пример программы, со сложным выводом:

// +++

​

let mut length = 2;

length = 4;

let mut width = 8;

let mut result = 0;

result = length\*width;

​

​

println!("length {} \* width {} = result {}", length, width, result);

​

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

let mut length = 2;

length = 4;

let mut width = 8;

let mut result = 0;

result = length\*width;

​

​

println!("length {} \* width {} = result {}", length, width, result);

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    length     -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    2          -  NUM\_INT

5   ;          -  END\_LINE

6    length     -  ID

7    =          -  ASSIGNMENT\_OPERATION\_\_SET

8    4          -  NUM\_INT

9   ;          -  END\_LINE

10   let        -  LET

11   mut        -  MUT

12   width      -  ID

13   =          -  ASSIGNMENT\_OPERATION\_\_SET

14   8          -  NUM\_INT

15   ;          -  END\_LINE

16   let        -  LET

17   mut        -  MUT

18   result     -  ID

19   =          -  ASSIGNMENT\_OPERATION\_\_SET

20   0          -  NUM\_INT

21   ;          -  END\_LINE

22   result     -  ID

23   =          -  ASSIGNMENT\_OPERATION\_\_SET

24   length     -  ID

25   \*          -  ARITHMETIC\_OPERATION\_\_MULT

26   width      -  ID

27   ;          -  END\_LINE

28   println!   -  PRINT

29   (          -  OPEN\_BRACKET

30   "length {} \* width {} = result {}"  -  STRING

31   ,          -  COMMA

32   length     -  ID

33   ,          -  COMMA

34   width      -  ID

35   ,          -  COMMA

36   result     -  ID

37   )          -  CLOSED\_BRACKET

38   ;          -  END\_LINE

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : length

 TYPE : INT

 VALUE : 2

ASSIGMENT : =

 ID : length

 NUM\_INT : 4

INITIALIZATION\_VAR

 NAME : width

 TYPE : INT

 VALUE : 8

INITIALIZATION\_VAR

 NAME : result

 TYPE : INT

 VALUE : 0

ASSIGMENT : =

 ID : result

 OPERATION : \*

   ID : length

   ID : width

PRINT

 ARGUMENTS

   STRING : "length {} \* width {} = result {}"

   ID : length

   ID : width

   ID : result

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main()

{

 int length = 2;

 length = 4;

 int width = 8;

 int result = 0;

 result = length \* width;

 std::cout << "length " << length << " \* width " << width << " = result " << result << std::endl;

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,23 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Пример инициализации переменных различных поддерживаемых типов:

// +++

​

fn main() {

   let a: i8 = 0;

   let b: u8 = 0;

   let c: i16 = 0;

   let d: u16 = 0;

   let e: i32 = 0;

   let f: u32 = 0;

   let g: i64 = 0;

   let h: u64 = 0;

   let i: f32 = 0.0;

   let j: f64 = 0.0;

}

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

fn main() {

   let a: i8 = 0;

   let b: u8 = 0;

   let c: i16 = 0;

   let d: u16 = 0;

   let e: i32 = 0;

   let f: u32 = 0;

   let g: i64 = 0;

   let h: u64 = 0;

   let i: f32 = 0.0;

   let j: f64 = 0.0;

}

​

Все распознанные лексемы:

​

0    fn         -  FUNCTION

1    main       -  MAIN

2   (          -  OPEN\_BRACKET

3   )          -  CLOSED\_BRACKET

4   {          -  START\_VOID

5    let        -  LET

6    a          -  ID

7   :          -  COLON

8    i8         -  int

9    =          -  ASSIGNMENT\_OPERATION\_\_SET

10   0          -  NUM\_INT

11   ;          -  END\_LINE

12   let        -  LET

13   b          -  ID

14   :          -  COLON

15   u8         -  int

16   =          -  ASSIGNMENT\_OPERATION\_\_SET

17   0          -  NUM\_INT

18   ;          -  END\_LINE

19   let        -  LET

20   c          -  ID

21   :          -  COLON

22   i16        -  int

23   =          -  ASSIGNMENT\_OPERATION\_\_SET

24   0          -  NUM\_INT

25   ;          -  END\_LINE

26   let        -  LET

27   d          -  ID

28   :          -  COLON

29   u16        -  int

30   =          -  ASSIGNMENT\_OPERATION\_\_SET

31   0          -  NUM\_INT

32   ;          -  END\_LINE

33   let        -  LET

34   e          -  ID

35   :          -  COLON

36   i32        -  int

37   =          -  ASSIGNMENT\_OPERATION\_\_SET

38   0          -  NUM\_INT

39   ;          -  END\_LINE

40   let        -  LET

41   f          -  ID

42   :          -  COLON

43   u32        -  int

44   =          -  ASSIGNMENT\_OPERATION\_\_SET

45   0          -  NUM\_INT

46   ;          -  END\_LINE

47   let        -  LET

48   g          -  ID

49   :          -  COLON

50   i64        -  long long int

51   =          -  ASSIGNMENT\_OPERATION\_\_SET

52   0          -  NUM\_INT

53   ;          -  END\_LINE

54   let        -  LET

55   h          -  ID

56   :          -  COLON

57   u64        -  long long int

58   =          -  ASSIGNMENT\_OPERATION\_\_SET

59   0          -  NUM\_INT

60   ;          -  END\_LINE

61   let        -  LET

62   i          -  ID

63   :          -  COLON

64   f32        -  float

65   =          -  ASSIGNMENT\_OPERATION\_\_SET

66   0.0        -  NUM\_FLOAT

67   ;          -  END\_LINE

68   let        -  LET

69   j          -  ID

70   :          -  COLON

71   f64        -  double

72   =          -  ASSIGNMENT\_OPERATION\_\_SET

73   0.0        -  NUM\_FLOAT

74   ;          -  END\_LINE

75   }          -  END\_VOID

Дерево разбора парсера:

FUNCTION

 NAME : MAIN

 ARGUMENT

   NO\_ARGUMENTS

 BODY

   INITIALIZATION\_VAR

     NAME : a

     TYPE : INT

     VALUE : 0

 INITIALIZATION\_VAR

   NAME : b

   TYPE : INT

   VALUE : 0

INITIALIZATION\_VAR

 NAME : c

 TYPE : INT

 VALUE : 0

INITIALIZATION\_VAR

 NAME : d

 TYPE : INT

 VALUE : 0

INITIALIZATION\_VAR

 NAME : e

 TYPE : INT

 VALUE : 0

INITIALIZATION\_VAR

 NAME : f

 TYPE : INT

 VALUE : 0

INITIALIZATION\_VAR

 NAME : g

 TYPE : LLINT

 VALUE : 0

INITIALIZATION\_VAR

 NAME : h

 TYPE : LLINT

 VALUE : 0

INITIALIZATION\_VAR

 NAME : i

 TYPE : FLOAT

currType = FLOAT

 VALUE : 0.0

INITIALIZATION\_VAR

 NAME : j

 TYPE : DOUBLE

currType = DOUBLE

 VALUE : 0.0

END\_FUNC

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main ()

{

 int a = 0;

 int b = 0;

 int c = 0;

 int d = 0;

 int e = 0;

 int f = 0;

 long long int g = 0;

 long long int h = 0;

 float i = 0.0;

 double j = 0.0;

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,26 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Пример инициализации строковых и символьных переменных:

// +++

​

let mut a = 7;

let bool1: bool = false;

let char1: char = 'a';

let bool2 = false;

let char2 = 'a';

let str1 = "123";

// let mut str1 = "123";

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

let mut a = 7;

let bool1: bool = false;

let char1: char = 'a';

let bool2 = false;

let char2 = 'a';

let str1 = "123";

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    a          -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    7          -  NUM\_INT

5   ;          -  END\_LINE

6    let        -  LET

7    bool1      -  ID

8   :          -  COLON

9    bool       -  bool

10   =          -  ASSIGNMENT\_OPERATION\_\_SET

11   false      -  FALSE

12   ;          -  END\_LINE

13   let        -  LET

14   char1      -  ID

15   :          -  COLON

16   char       -  char

17   =          -  ASSIGNMENT\_OPERATION\_\_SET

18   '         - ONCE\_QUOTAT

19   a          -  ID

20   '         - ONCE\_QUOTAT

21   ;          -  END\_LINE

22   let        -  LET

23   bool2      -  ID

24   =          -  ASSIGNMENT\_OPERATION\_\_SET

25   false      -  FALSE

26   ;          -  END\_LINE

27   let        -  LET

28   char2      -  ID

29   =          -  ASSIGNMENT\_OPERATION\_\_SET

30   '         - ONCE\_QUOTAT

31   a          -  ID

32   '         - ONCE\_QUOTAT

33   ;          -  END\_LINE

34   let        -  LET

35   str1       -  ID

36   =          -  ASSIGNMENT\_OPERATION\_\_SET

37   "123"      -  STRING

38   ;          -  END\_LINE

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : a

 TYPE : INT

 VALUE : 7

INITIALIZATION\_VAR

 NAME : bool1

 TYPE : BOOL

 VALUE : false

INITIALIZATION\_VAR

 NAME : char1

 TYPE : CHAR

 VALUE : 'a'

INITIALIZATION\_VAR

 NAME : bool2

 TYPE : BOOL

 VALUE : false

INITIALIZATION\_VAR

 NAME : char2

 TYPE : CHAR

 VALUE : 'a'

INITIALIZATION\_VAR

 NAME : str1

 TYPE : STRING

 VALUE : "123"

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

#include <string>

using namespace std;

​

int main()

{

 int a = 7;

 bool bool1 = false;

 char char1 = 'a';

 bool bool2 = false;

 char char2 = 'a';

 std::string str1 = "123";

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,26 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Попытка инициализации неподдерживаемых типов переменных:

// Провека ошибки +++

​

fn main() {

   let a: u16 = 0;

   let b: i128 = 0;

   let c: u128 = 0;

   let d: isize = 0;

   let e: usize = 0;

}

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

​

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// Провека ошибки +++

​

fn main() {

   let a: u16 = 0;

​

   let b: i128 = 0;

   let c: u128 = 0;

   let d: isize = 0;

   let e: usize = 0;

}

​

Все распознанные лексемы:

​

0    fn         -  FUNCTION

1    main       -  MAIN

2   (          -  OPEN\_BRACKET

3   )          -  CLOSED\_BRACKET

4   {          -  START\_VOID

5    let        -  LET

6    a          -  ID

7   :          -  COLON

8    u16        -  int

9    =          -  ASSIGNMENT\_OPERATION\_\_SET

10   0          -  NUM\_INT

11   ;          -  END\_LINE

12   let        -  LET

13   b          -  ID

14   :          -  COLON

15   i128       -  none

16   =          -  ASSIGNMENT\_OPERATION\_\_SET

17   0          -  NUM\_INT

18   ;          -  END\_LINE

19   let        -  LET

20   c          -  ID

21   :          -  COLON

22   u128       -  none

23   =          -  ASSIGNMENT\_OPERATION\_\_SET

24   0          -  NUM\_INT

25   ;          -  END\_LINE

26   let        -  LET

27   d          -  ID

28   :          -  COLON

29   isize      -  none

30   =          -  ASSIGNMENT\_OPERATION\_\_SET

31   0          -  NUM\_INT

32   ;          -  END\_LINE

33   let        -  LET

34   e          -  ID

35   :          -  COLON

36   usize      -  none

37   =          -  ASSIGNMENT\_OPERATION\_\_SET

38   0          -  NUM\_INT

39   ;          -  END\_LINE

40   }          -  END\_VOID

​

Дерево разбора парсера:

​

FUNCTION

 NAME : MAIN

 ARGUMENT

   NO\_ARGUMENTS

 BODY

   INITIALIZATION\_VAR

     NAME : a

     TYPE : INT

     VALUE : 0

 INITIALIZATION\_VAR

   NAME : b

​

Ошибка! В объявлении переменной b обнаружен неподдерживаемый тип данных: i128

​

---------------------------------------------------

Трансляция программы остановлена с исключением: Переменная была инициализирована неподдерживаемым типом данных

​

Семантический анализатор:

​

------------------------------------

В процессе построения дерева разбора

программы произошла ошибка!

​

Подробности ошибки:

          ▼

0 ; let b : i128 = 0

​

Ошибка! В объявлении переменной b обнаружен неподдерживаемый тип данных: i128

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: Pars\_Err

​

Трансляция программы остановлена с исключением:

Ошибка! В объявлении переменной b обнаружен неподдерживаемый тип данных: i128

### Попытка использовать встроенные библиотеки и модули:

// Провека ошибки +++

​

io::stdin().read\_line(&mut input).expect("Ошибка чтения строки"); // Читаем строку из входного потока

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

​

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// Провека ошибки +++

​

io::stdin().read\_line(&mut input).expect("Ошибка чтения строки"); // Читаем строку из входного потока

​

Все распознанные лексемы:

​

0    o          -  ID

1   ::         -  INSIDE\_LINK

2    stdin      -  ID

3   (          -  OPEN\_BRACKET

4   )          -  CLOSED\_BRACKET

5   .          -  DOT

6    read\_line  -  ID

7   (          -  OPEN\_BRACKET

8    &          -  BITWISE\_OPERATION\_\_AND

9    mut        -  MUT

10   input      -  ID

11   )          -  CLOSED\_BRACKET

12   .          -  DOT

13   expect     -  ID

14   (          -  OPEN\_BRACKET

15   "Ошибка чтения строки"  -  STRING

16   )          -  CLOSED\_BRACKET

17   ;          -  END\_LINE

​

! Во входной программе были распознаны неподдерживаемые ключевые слова, которые не могу быть обработаны данным транслятором.

К сожалению, этот транслятор - учебный проект, и реализовать функционал переноса всех ключевых слов, с их возможностями - была бы непосильная задача

​

Конкретно эти лексемы в вашем коде мы не можем обработать:

INSIDE\_LINK

​

Пожалуйста, введите программу, без таких ключевых слов.

​

---------------------------------------------------

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: Unsp\_KyWords

​

Трансляция программы остановлена с исключением:

Обнаружены неподдерживаемые ключевые слова.

### Простой пример программы:

// +++

​

let mut a = 25;

let mut b = 4;

b = b \* b;

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

let mut a = 25;

let mut b = 4;

b = b \* b;

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    a          -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    25         -  NUM\_INT

5   ;          -  END\_LINE

6    let        -  LET

7    mut        -  MUT

8    b          -  ID

9    =          -  ASSIGNMENT\_OPERATION\_\_SET

10   4          -  NUM\_INT

11   ;          -  END\_LINE

12   b          -  ID

13   =          -  ASSIGNMENT\_OPERATION\_\_SET

14   b          -  ID

15   \*          -  ARITHMETIC\_OPERATION\_\_MULT

16   b          -  ID

17   ;          -  END\_LINE

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : a

 TYPE : INT

 VALUE : 25

INITIALIZATION\_VAR

 NAME : b

 TYPE : INT

 VALUE : 4

ASSIGMENT : =

 ID : b

 OPERATION : \*

   ID : b

   ID : b

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main()

{

 int a = 25;

 int b = 4;

 b = b \* b;

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,22 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Изменение значения константной переменной:

// Провека ошибки +++

​

let mut a = 25;

let b = 4;

b = b \* b;

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

​

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// Провека ошибки +++

​

let mut a = 25;

let b = 4;

b = b \* b;

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    a          -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    25         -  NUM\_INT

5   ;          -  END\_LINE

6    let        -  LET

7    b          -  ID

8    =          -  ASSIGNMENT\_OPERATION\_\_SET

9    4          -  NUM\_INT

10   ;          -  END\_LINE

11   b          -  ID

12   =          -  ASSIGNMENT\_OPERATION\_\_SET

13   b          -  ID

14   \*          -  ARITHMETIC\_OPERATION\_\_MULT

15   b          -  ID

16   ;          -  END\_LINE

​

Дерево разбора парсера:

​

INITIALIZATION\_VAR

 NAME : a

 TYPE : INT

 VALUE : 25

INITIALIZATION\_VAR

 NAME : b

 TYPE : INT

 VALUE : 4

ASSIGMENT : =

 ID : b

 OPERATION : \*

   ID : b

   ID : b

​

---------------------------------------------------

Семантический анализатор:

​

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main()

{

 int a = 25;

 int b = 4;

​

! Обнаружена ошибка: Переменная b была инициализирована как константная,

но далее в коде ей присваивается новое значение

​

​

 return 0;

}

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: Const\_Var\_Cange

​

Трансляция программы остановлена с исключением:

Изменение значения константной переменной

### Повторное объявление переменной:

// Провека ошибки +++

​

let mut b = 4;

b = b \* b;

let mut b = 4;

v5.0 Final

Автор: Gogik Ortey

​

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// Провека ошибки +++

​

let mut b = 4;

b = b \* b;

let mut b = 4;

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    b          -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    4          -  NUM\_INT

5   ;          -  END\_LINE

6    b          -  ID

7    =          -  ASSIGNMENT\_OPERATION\_\_SET

8    b          -  ID

9    \*          -  ARITHMETIC\_OPERATION\_\_MULT

10   b          -  ID

11   ;          -  END\_LINE

12   let        -  LET

13   mut        -  MUT

14   b          -  ID

15   =          -  ASSIGNMENT\_OPERATION\_\_SET

16   4          -  NUM\_INT

17   ;          -  END\_LINE

​

Дерево разбора парсера:

​

INITIALIZATION\_VAR

 NAME : b

 TYPE : INT

 VALUE : 4

ASSIGMENT : =

 ID : b

 OPERATION : \*

   ID : b

   ID : b

INITIALIZATION\_VAR

 NAME : b

---------------------------------------------------

​

Ошибка! Переменная b была инициализирована второй раз

​

Семантический анализатор:

​

------------------------------------

В процессе построения дерева разбора

программы произошла ошибка!

​

Подробности ошибки:

        ▼

b \* b ; let mut b =

​

Повторная инициплизация переменной

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: Pars\_Err

​

Трансляция программы остановлена с исключением:

Повторная инициплизация переменной

### Задание строк:

// +++

​

let mut str1 = "123";

let e: String = "Hello, Rust!";

str1 = e;

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

let mut str1 = "123";

let e: String = "Hello, Rust!";

str1 = e;

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    str1       -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    "123"      -  STRING

5   ;          -  END\_LINE

6    let        -  LET

7    e          -  ID

8   :          -  COLON

9    String     -  ID

10   =          -  ASSIGNMENT\_OPERATION\_\_SET

11   "Hello, Rust!"  -  STRING

12   ;          -  END\_LINE

13   str1       -  ID

14   =          -  ASSIGNMENT\_OPERATION\_\_SET

15   e          -  ID

16   ;          -  END\_LINE

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : str1

 TYPE : STRING

 VALUE : "123"

INITIALIZATION\_VAR

 NAME : e

 TYPE : STRING

 VALUE : "Hello, Rust!"

ASSIGMENT : =

 ID : str1

 ID : e

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

#include <string>

using namespace std;

​

int main()

{

 std::string str1 = "123";

 std::string e = "Hello, Rust!";

 str1 = e;

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,2 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Большой пример с объявлением различных переменных:

// +++

​

fn main() {

   let mut a: i32 = 5; // int

   let b: f32 = 4.5; // float

   let c: bool = true; // bool

   let d: char = 'R'; // char

   let e: String = "Hello, Rust!"; // string

​

   // Неиспользуемая переменная

   let f: i32 = 10;

}

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

fn main() {

   let mut a: i32 = 5; // int

   let b: f32 = 4.5; // float

   let c: bool = true; // bool

   let d: char = 'R'; // char

   let e: String = "Hello, Rust!"; // string

​

   // Неиспользуемая переменная

   let f: i32 = 10;

}

​

Все распознанные лексемы:

​

0    fn         -  FUNCTION

1    main       -  MAIN

2   (          -  OPEN\_BRACKET

3   )          -  CLOSED\_BRACKET

4   {          -  START\_VOID

5    let        -  LET

6    mut        -  MUT

7    a          -  ID

8   :          -  COLON

9    i32        -  int

10   =          -  ASSIGNMENT\_OPERATION\_\_SET

11   5          -  NUM\_INT

12   ;          -  END\_LINE

13   let        -  LET

14   b          -  ID

15   :          -  COLON

16   f32        -  float

17   =          -  ASSIGNMENT\_OPERATION\_\_SET

18   4.5        -  NUM\_FLOAT

19   ;          -  END\_LINE

20   let        -  LET

21   c          -  ID

22   :          -  COLON

23   bool       -  bool

24   =          -  ASSIGNMENT\_OPERATION\_\_SET

25   true       -  TRUE

26   ;          -  END\_LINE

27   let        -  LET

28   d          -  ID

29   :          -  COLON

30   char       -  char

31   =          -  ASSIGNMENT\_OPERATION\_\_SET

32   '         - ONCE\_QUOTAT

33   R          -  ID

34   '         - ONCE\_QUOTAT

35   ;          -  END\_LINE

36   let        -  LET

37   e          -  ID

38   :          -  COLON

39   String     -  ID

40   =          -  ASSIGNMENT\_OPERATION\_\_SET

41   "Hello, Rust!"  -  STRING

42   ;          -  END\_LINE

43   let        -  LET

44   f          -  ID

45   :          -  COLON

46   i32        -  int

47   =          -  ASSIGNMENT\_OPERATION\_\_SET

48   10         -  NUM\_INT

49   ;          -  END\_LINE

50   }          -  END\_VOID

Дерево разбора парсера:

FUNCTION

 NAME : MAIN

 ARGUMENT

   NO\_ARGUMENTS

 BODY

   INITIALIZATION\_VAR

     NAME : a

     TYPE : INT

     VALUE : 5

 INITIALIZATION\_VAR

   NAME : b

   TYPE : FLOAT

currType = FLOAT

   VALUE : 4.5

INITIALIZATION\_VAR

 NAME : c

 TYPE : BOOL

 VALUE : true

INITIALIZATION\_VAR

 NAME : d

 TYPE : CHAR

 VALUE : 'R'

INITIALIZATION\_VAR

 NAME : e

 TYPE : STRING

 VALUE : "Hello, Rust!"

INITIALIZATION\_VAR

 NAME : f

 TYPE : INT

 VALUE : 10

END\_FUNC

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main ()

{

 int a = 5;

 float b = 4.5;

 bool c = true;

 char d = 'R';

 std::string e = "Hello, Rust!";

 int f = 10;

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,2 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Проверка двойной операции:

// +++

​

let mut a = 10;

a += 5;

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

let mut a = 10;

a += 5;

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    a          -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    10         -  NUM\_INT

5   ;          -  END\_LINE

6    a          -  ID

7    +          -  ARITHMETIC\_OPERATION\_\_ADD

8    =          -  ASSIGNMENT\_OPERATION\_\_SET

9    5          -  NUM\_INT

10   ;          -  END\_LINE

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : a

 TYPE : INT

 VALUE : 10

ASSIGMENT : +=

 ID : a

 NUM\_INT : 5

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main()

{

 int a = 10;

 a += 5;

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,21 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Операции со скобками:

// +++

​

let a = 5;

let b = 10;

let c = 2;

​

let mut result1: u16;

let mut result2: u16;

​

// Без скобок, умножение выполняется первым

result1 = a + b \* c;

println!("Результат без скобок: {}", result1);

​

// Со скобками, сложение выполняется первым

result2 = (a + b) \* c;

println!("Результат со скобками: {}" result2);

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

let a = 5;

let b = 10;

let c = 2;

​

let mut result1: u16;

let mut result2: u16;

​

// Без скобок, умножение выполняется первым

result1 = a + b \* c;

println!("Результат без скобок: {}", result1);

​

// Со скобками, сложение выполняется первым

result2 = (a + b) \* c;

println!("Результат со скобками: {}" result2);

​

Все распознанные лексемы:

​

0    let        -  LET

1    a          -  ID

2    =          -  ASSIGNMENT\_OPERATION\_\_SET

3    5          -  NUM\_INT

4   ;          -  END\_LINE

5    let        -  LET

6    b          -  ID

7    =          -  ASSIGNMENT\_OPERATION\_\_SET

8    10         -  NUM\_INT

9   ;          -  END\_LINE

10   let        -  LET

11   c          -  ID

12   =          -  ASSIGNMENT\_OPERATION\_\_SET

13   2          -  NUM\_INT

14   ;          -  END\_LINE

15   let        -  LET

16   mut        -  MUT

17   result1    -  ID

18   :          -  COLON

19   u16        -  int

20   ;          -  END\_LINE

21   let        -  LET

22   mut        -  MUT

23   result2    -  ID

24   :          -  COLON

25   u16        -  int

26   ;          -  END\_LINE

27   result1    -  ID

28   =          -  ASSIGNMENT\_OPERATION\_\_SET

29   a          -  ID

30   +          -  ARITHMETIC\_OPERATION\_\_ADD

31   b          -  ID

32   \*          -  ARITHMETIC\_OPERATION\_\_MULT

33   c          -  ID

34   ;          -  END\_LINE

35   println!   -  PRINT

36   (          -  OPEN\_BRACKET

37   "Результат без скобок: {}"  -  STRING

38   ,          -  COMMA

39   result1    -  ID

40   )          -  CLOSED\_BRACKET

41   ;          -  END\_LINE

42   result2    -  ID

43   =          -  ASSIGNMENT\_OPERATION\_\_SET

44   (          -  OPEN\_BRACKET

45   a          -  ID

46   +          -  ARITHMETIC\_OPERATION\_\_ADD

47   b          -  ID

48   )          -  CLOSED\_BRACKET

49   \*          -  ARITHMETIC\_OPERATION\_\_MULT

50   c          -  ID

51   ;          -  END\_LINE

52   println!   -  PRINT

53   (          -  OPEN\_BRACKET

54   "Результат со скобками: {}"  -  STRING

55   result2    -  ID

56   )          -  CLOSED\_BRACKET

57   ;          -  END\_LINE

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : a

 TYPE : INT

 VALUE : 5

INITIALIZATION\_VAR

 NAME : b

 TYPE : INT

 VALUE : 10

INITIALIZATION\_VAR

 NAME : c

 TYPE : INT

 VALUE : 2

INITIALIZATION\_VAR

 NAME : result1

 TYPE : INT

INITIALIZATION\_VAR

 NAME : result2

 TYPE : INT

ASSIGMENT : =

 ID : result1

 OPERATION : \*

   OPERATION : +

     ID : a

     ID : b

   ID : c

PRINT

 ARGUMENTS

   STRING : "Результат без скобок: {}"

   ID : result1

ASSIGMENT : =

 ID : result2

 OPERATION : \*

   OPERATION : +

     ID : a

   ID : b

   ID : c

PRINT

 ARGUMENTS

STRING : "Результат со скобками: {}"

ID : result2

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main()

{

 int a = 5;

 int b = 10;

 int c = 2;

 int result1;

 int result2;

 result1 = a + b \* a;

 std::cout << "Результат без скобок: " << result1 << std::endl;

 result2 = a + b \* a;

 std::cout << "Результат со скобками: " << result2 << std::endl;

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,22 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Определение и вызов функции:

// +++

​

fn main() {

   another\_function(5);

}

​

fn another\_function(x: i32) {

   println!("The value of x is: {}", x);

}

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

fn main() {

   another\_function(5);

}

​

fn another\_function(x: i32) {

   println!("The value of x is: {}", x);

}

​

Добавили имя функции в объявленную область: another\_function

​

Все распознанные лексемы:

​

0    fn         -  FUNCTION

1    main       -  MAIN

2   (          -  OPEN\_BRACKET

3   )          -  CLOSED\_BRACKET

4   {          -  START\_VOID

5    another\_function  -  ID

6   (          -  OPEN\_BRACKET

7    5          -  NUM\_INT

8   )          -  CLOSED\_BRACKET

9   ;          -  END\_LINE

10   }          -  END\_VOID

11   fn         -  FUNCTION

12   another\_function  -  ID

13   (          -  OPEN\_BRACKET

14   x          -  ID

15   :          -  COLON

16   i32        -  int

17   )          -  CLOSED\_BRACKET

18   {          -  START\_VOID

19   println!   -  PRINT

20   (          -  OPEN\_BRACKET

21   "The value of x is: {}"  -  STRING

22   ,          -  COMMA

23   x          -  ID

24   )          -  CLOSED\_BRACKET

25   ;          -  END\_LINE

26   }          -  END\_VOID

Дерево разбора парсера:

FUNCTION

 NAME : MAIN

 ARGUMENT

   NO\_ARGUMENTS

 BODY

   CALL\_FUNCTION

     NAME : another\_function

     ARGUMENT

       NAME : 5

END\_FUNC

FUNCTION

 NAME : another\_function

 ARGUMENT

   NAME : x

   TYPE : INT

 BODY

   PRINT

     ARGUMENTS

       STRING : "The value of x is: {}"

       ID : x

END\_FUNC

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main ()

{

 another\_function(5);

​

 return 0;

}

​

void another\_function(int x)

{

 std::cout << "The value of x is: " << x << std::endl;

}

---------------------------------------------------

​

Время выполнения программы: 0,14 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Цикл while:

// +++

​

// Цикл while

let mut j = 0;

while (j < 5) {

   println!("Число: {}", j);

   j += 1;

}

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

// Цикл while

let mut j = 0;

while (j < 5) {

   println!("Число: {}", j);

   j += 1;

}

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    j          -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    0          -  NUM\_INT

5   ;          -  END\_LINE

6    while      -  WHILE

7   (          -  OPEN\_BRACKET

8    j          -  ID

9    <          -  COMPARISON\_OPERATION\_\_LESS

10   5          -  NUM\_INT

11   )          -  CLOSED\_BRACKET

12   {          -  START\_VOID

13   println!   -  PRINT

14   (          -  OPEN\_BRACKET

15   "Число: {}"  -  STRING

16   ,          -  COMMA

17   j          -  ID

18   )          -  CLOSED\_BRACKET

19   ;          -  END\_LINE

20   j          -  ID

21   +          -  ARITHMETIC\_OPERATION\_\_ADD

22   =          -  ASSIGNMENT\_OPERATION\_\_SET

23   1          -  NUM\_INT

24   ;          -  END\_LINE

25   }          -  END\_VOID

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : j

 TYPE : INT

 VALUE : 0

WHILE

 CONDITION

   OPERATION : <

     ID : j

     NUM\_INT : 5

 BODY

   PRINT

     ARGUMENTS

       STRING : "Число: {}"

       ID : j

   ASSIGMENT : +=

   ID : j

   NUM\_INT : 1

END\_FUNC

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main()

{

 int j = 0;

 while(j < 5)

{

   std::cout << "Число: " << j << std::endl;

   j += 1;

}

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,3 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Условие if, else if и else:

// +++

​

// if else и else if

​

let a = 10;

​

if (a < 5) {

   println!("a меньше 5");

} else if (a == 5) {

   println!("a равно 5");

} else {

   println!("a больше 5");

}

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

// if else и else if

​

let a = 10;

​

if (a < 5) {

   println!("a меньше 5");

} else if (a == 5) {

   println!("a равно 5");

} else {

   println!("a больше 5");

}

​

Все распознанные лексемы:

​

0    let        -  LET

1    a          -  ID

2    =          -  ASSIGNMENT\_OPERATION\_\_SET

3    10         -  NUM\_INT

4   ;          -  END\_LINE

5    if         -  IF

6   (          -  OPEN\_BRACKET

7    a          -  ID

8    <          -  COMPARISON\_OPERATION\_\_LESS

9    5          -  NUM\_INT

10   )          -  CLOSED\_BRACKET

11   {          -  START\_VOID

12   println!   -  PRINT

13   (          -  OPEN\_BRACKET

14   "a меньше 5"  -  STRING

15   )          -  CLOSED\_BRACKET

16   ;          -  END\_LINE

17   }          -  END\_VOID

18   elif       -  ELIF

19   (          -  OPEN\_BRACKET

20   a          -  ID

21   ==         -  COMPARISON\_OPERATION\_\_EQUAL

22   5          -  NUM\_INT

23   )          -  CLOSED\_BRACKET

24   {          -  START\_VOID

25   println!   -  PRINT

26   (          -  OPEN\_BRACKET

27   "a равно 5"  -  STRING

28   )          -  CLOSED\_BRACKET

29   ;          -  END\_LINE

30   }          -  END\_VOID

31   else       -  ELSE

32   {          -  START\_VOID

33   println!   -  PRINT

34   (          -  OPEN\_BRACKET

35   "a больше 5"  -  STRING

36   )          -  CLOSED\_BRACKET

37   ;          -  END\_LINE

38   }          -  END\_VOID

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : a

 TYPE : INT

 VALUE : 10

IF

 CONDITION

   OPERATION : <

     ID : a

     NUM\_INT : 5

 BODY

   PRINT

     ARGUMENTS

       STRING : "a меньше 5"

END\_FUNC

ELIF

 CONDITION

   OPERATION : ==

     ID : a

     NUM\_INT : 5

 BODY

   PRINT

     ARGUMENTS

       STRING : "a равно 5"

END\_FUNC

ELSE

 CONDITION

BODY

 PRINT

   ARGUMENTS

     STRING : "a больше 5"

END\_FUNC

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main()

{

 int a = 10;

 if (a < 5)

{

   std::cout << "a меньше 5" << std::endl;

}

 else if (a == 5)

{

   std::cout << "a равно 5" << std::endl;

}

 else

{

   std::cout << "a больше 5" << std::endl;

}

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,33 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

### Цикл for:

// +++

​

let mut a = 0;

// Цикл for

for i in 0..5 { // от 0 до 4

   println!("Число: {}", i);

}

Программа: Транслятор с языка Rust на язык C++

v5.0 Final

Автор: Gogik Ortey

Основные настройки:

​

> Оборачиваем ли мы на выходе наш код в функцию main : Yes

> Транслятор удаляет неиспользуемые переменные? : No

​

​

Входной файл:

// +++

​

let mut a = 0;

// Цикл for

for i in 0..5 { // от 0 до 4

   println!("Число: {}", i);

}

​

Все распознанные лексемы:

​

0    let        -  LET

1    mut        -  MUT

2    a          -  ID

3    =          -  ASSIGNMENT\_OPERATION\_\_SET

4    0          -  NUM\_INT

5   ;          -  END\_LINE

6    for        -  FOR

7    i          -  ID

8    in         -  IN

9    0          -  NUM\_INT

10   .          -  DOT

11   .          -  DOT

12   5          -  NUM\_INT

13   {          -  START\_VOID

14   println!   -  PRINT

15   (          -  OPEN\_BRACKET

16   "Число: {}"  -  STRING

17   ,          -  COMMA

18   i          -  ID

19   )          -  CLOSED\_BRACKET

20   ;          -  END\_LINE

21   }          -  END\_VOID

Дерево разбора парсера:

INITIALIZATION\_VAR

 NAME : a

 TYPE : INT

 VALUE : 0

FOR

 CONDITION

   VAR : i

   FROM : 0

   TO : 5

 BODY

 START\_VOID

 PRINT

   ARGUMENTS

     STRING : "Число: {}"

     ID : i

END\_FUNC

---------------------------------------------------

Семантический анализатор:

Проблем не обнаружено.

​

---------------------------------------------------

​

Кодогенератор С++

​

#include <iostream>

using namespace std;

​

int main()

{

 int a = 0;

 for (int i = 0; i < 5; i++)

{

   std::cout << "Число: " << i << std::endl;

}

​

 return 0;

}

---------------------------------------------------

​

Время выполнения программы: 0,19 секунд

​

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

​

Программа завершилась с кодом: OK

​

Успешная трансляция программы

# Заключение

В рамках курсовой работы был разработан транслятор, который преобразует программу, содержащую подмножество языка Rust, в программу, содержащую подмножество языка C++, генерируя эквивалентный исходный код. Для этого были выполнены следующие задачи:

1. Описать грамматику подмножества входного языка;
2. Описать контекстные условия входного и выходного языков;
3. Описать соответствие конструкций входного и выходного языков;
4. Разработать проект лексического анализатора;
5. Разработать проект синтаксического анализатора;
6. Разработать проект семантического анализатора;
7. Разработать проект генератора кода выходного языка;
8. Реализовать транслятор входного языка в выходной;
9. Выполнить тестирование разработанного транслятора.

# Список литературы

1. Язык программирования Rust. Официальная документация [Электронный ресурс] // Режим доступа: <https://www.rust-lang.org/ru>
2. Rust — Мультипарадигмальный компилируемый язык [Электронный ресурс] // Режим доступа: <https://proglib.io/p/rust-eto-novyy-c-ili-ocherednaya-ekzotika-2021-08-20>
3. Приложения CLI - Язык программирования Rust [Электронный ресурс] // Режим доступа: <https://proglib.io/p/rust-eto-novyy-c-ili-ocherednaya-ekzotika-2021-08-20>
4. Работа с языками низкого уровня на Rust: взаимодействие с Assembly и C [Электронный ресурс] // Режим доступа: <https://proglib.io/p/rust-eto-novyy-c-ili-ocherednaya-ekzotika-2021-08-20>
5. Начало работы - Язык программирования Rust [Электронный ресурс] // Режим доступа: <https://proglib.io/p/rust-eto-novyy-c-ili-ocherednaya-ekzotika-2021-08-20>