Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Кафедра вычислительной техники

Отчет по лабораторной работе № 4

по дисциплине «Теория формальных языков и компиляторов»

на тему «Синтаксис языков программирования. Нисходящий синтаксический анализ»

Студент: Смирнов А.М.

Вариант: 14431332

Группа: АВТ-709

Преподаватель: Малявко А.А.

Новосибирск, 2020

# Цель работы

# Изучение основных идей и понятий нисходящих методов синтаксического анализа, выявление свойств формальных грамматик, необходимых для реализации нисходящего восстановления дерева грамматического разбора, приобретение навыков построения процедурной и различных автоматных реализаций нисходящего анализа, исследование поведения нисходящих синтаксических акцепторов.

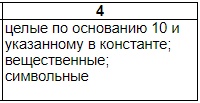
# Постановка задачи

* 1. Используя пакет ВебТрансЛаб:
* изучить и освоить проверку принадлежности грамматики к классу **LL(1)**, используя в качестве проверяемых грамматики, полученные при выполнении работы №4;
* построить конечный автомат со стековой памятью и несколькими состояниями (шаблон …SyntAsMultiFSM…), разобраться в структуре управляющей таблицы автомата, уяснить способы формирования и использования всех полей;
* построить конечный автомат со стековой памятью и одним состоянием, управляемый входным символом и символом, снятым с верхушки стека (шаблон …SyntAsSingleFSM…), разобраться в структуре управляющей таблицы автомата, уяснить способы формирования и использования клеток таблицы;
* построить процедурную реализацию рекурсивного спуска (шаблон …SyntAsRD…), уяснить способы формирования функций этого акцептора;
  1. Выполнить трассировку процессов нисходящего синтаксического акцепта, изучить поведение всех построенных синтаксических акцепторов при разборе как правильных предложений, так и предложений с намеренно внесенными синтаксическими ошибками.
  2. Проанализировать и сравнить между собой все полученные тексты программ и результаты выполнения пункта 3.2. Оценить степень пригодности изученных вариантов реализации нисходящих синтаксических акцепторов для выполнения курсовой работы.
  3. Подготовить, сдать и защитить отчет к лабораторной работе.

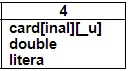
Вариант: 14431332

1. 

*Рис. 1 – Идентификаторы.*

1. 

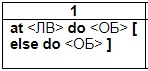
*Рис. 2 – Константы.*

1. 

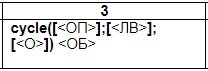
*Рисунок 3 – Типы.*

1. 

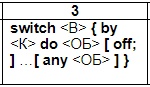
*Рисунок 4 – Оператор присваивания.*

1. 

*Рисунок 5 – Условный оператор.*

1. 

*Рисунок 6 – Оператор цикла.*

1. 

*Рисунок 7 – Оператор переключателя.*

# Ход работы

**Исправления и новые правила:**

В соответствие с замечаниями и комментариями была исправлена грамматика:

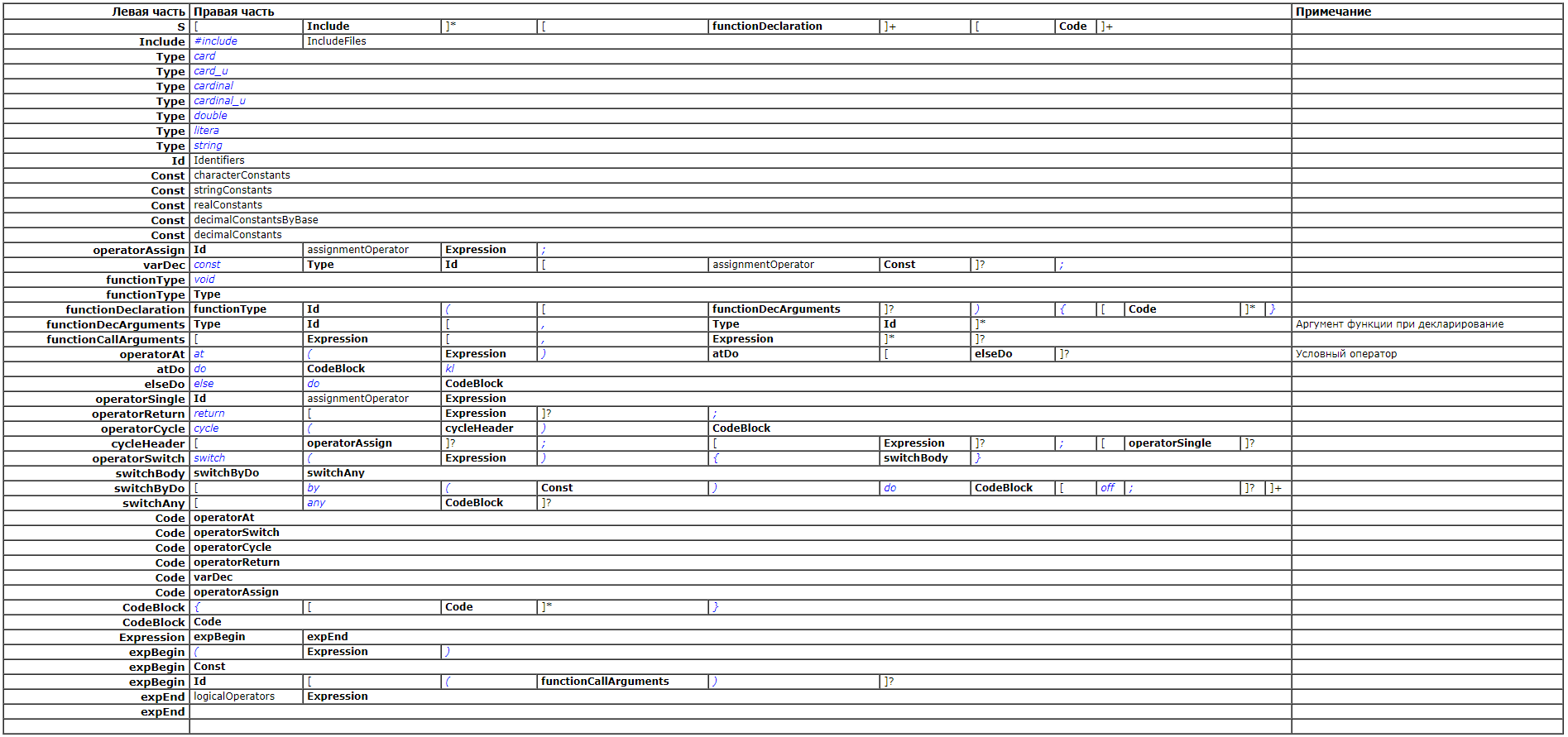
* Доработаны типы данных.
* Разделены оператор присваивания и объявление переменной с инициализацией.
* Переработаны условный оператор, операторы цикла и переключателя.
* Переработан оператор return.
* Добавлено новое правило для блока кода.

Правило общего вида программы было изменено. Теперь код программы обязан содержать хотя бы одну функцию, даже пустую. Помимо этого, программа обязана содержать хотя бы один оператор.

Добавлены новые правила для нетерминала Expression*.* Данный нетерминалможет быть:

* Любым математическим выражением.
* Константой.
* Идентификатором.
* Результатом вызова функции.

Функцию можно вызвать в любом месте. То есть как одиночный оператор, как оператор в блоке кода, в математическом выражении и при присваивании. Обновлённые правила показаны на Рис. 8.



*Рис. 8 - Обновленные правила синтаксиса.*

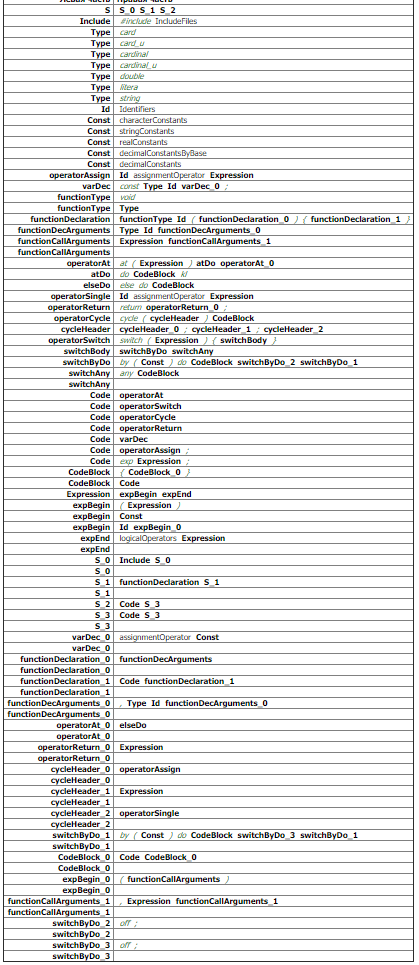
Множества выбора правил для данной грамматики показаны на рисунке 9.



*Рис. 9 – Множество выбора.*

Можно заметить, что для каждого нетерминала множества выбора его правил попарно не пересекаются. Значит грамматика принадлежит классу LL(1)-грамматик.

Правила грамматики показаны на рисунке 10.



*Рис. 10 – Правила грамматики.*

Управляющая таблица нисходящего автомата с несколькими состояниями показана в Приложении 1.

Для тестирование обновлённых правил синтаксиса воспользуемся программой, написанной в прошлой лабораторной работе, и немного доработаем её.

#include <math.h>

#include <iostream>

cardinal S\_omenumber(cardinal G\_){

return G\_ + 5;

}

card S\_tarter(){

const cardinal N\_ := 5;

const cardinal I\_;

N\_ := S\_omenumber(N\_);

cycle(I\_ := 0; I\_ < N\_; ){

switch(I\_){

by(2) do {

I\_ := I\_ + 1;

}

off;

any {

I\_ := I\_ + 2;

}

}

}

at(N\_ < 10) do {

N\_:=15;

} kl

else do {

N\_:=5;

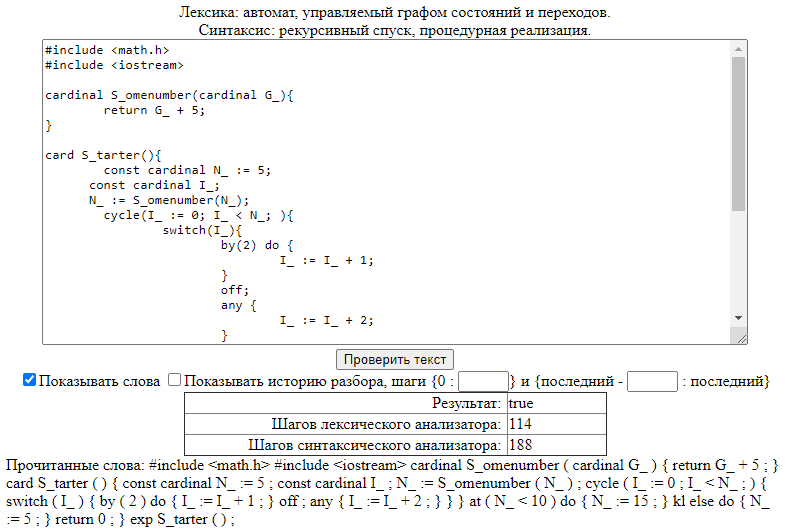
}

return 0;

}

exp S\_tarter();

Результат обновленных правил синтаксиса показан на рисунке 11.



*Рис. 11 - Проверка обновленных правил синтаксиса*

# Теоретическая часть

LL(1)-грамматикой называется такая контекстно-свободная грамматика, у которой множества выбора правил с одинаковой нетерминалом в левой част и попарно не пересекаются.

Принято считать, что символы в названии класса LL(1)-грамматик обозначают следующее.

Первая буква L (сокращение слова left - левый) - чтение слов анализируемого предложения производится слева направо.

Вторая буква L (сокращение слова leftmost - самый левый) - на каждом шаге принимается решение для замены самого левого нетерминала из текущего уровня восстанавливаемого дерева.

Цифра 1 в скобках обозначает количество символов из начала остатка предложения, необходимых для принятия решения о выборе правила на каждом шаге детерминированного нисходящего восстановления дерева грамматического разбора.

Функционирование конечного автомата со стековой памятью и несколькими состояниями определяется управляющей таблицей. Предполагается, что автомат при запуске оказывается в особом начальном состоянии, на каждом такте по входному символу и текущему состоянию определяет и выполняет операции над входным потоком символов, стековой памятью и собственным состоянием.

Каждому символу каждого правила грамматики должно быть поставлено в соответствии в точности одно состояние автомата. С каждым состоянием должно быть связано множество выбора и два адреса перехода. Под адресом перехода понимается номер состояния.

При соблюдении определенных правил нумерации состояний и введении операции управления остановом по ошибке можно обойтись только одним адресом перехода.

С каждым состоянием должны быть также связаны операции управления стековой памятью и чтением следующего входного символа. Все операции управления могут задаваться булевскими значениями true / false, которые называются флажками. Обозначения для флажков управления операциями:

* + флаг a управляет чтением следующего входного символа;
  + флаг s управляет занесением адреса точки возврата в стек;
  + флаг r обеспечивает переключение автомата в состояние, номер которого снимается с верхушки стека возвратов;
  + флаг e запрещает останов по ошибке, когда состояние соответствует нетерминалу из левой части и есть еще хотя бы одно правило для такого нетерминала.

LL(1)-грамматику легко можно преобразовать в конечный автомат с единственным состоянием и стековой памятью, управляемый:

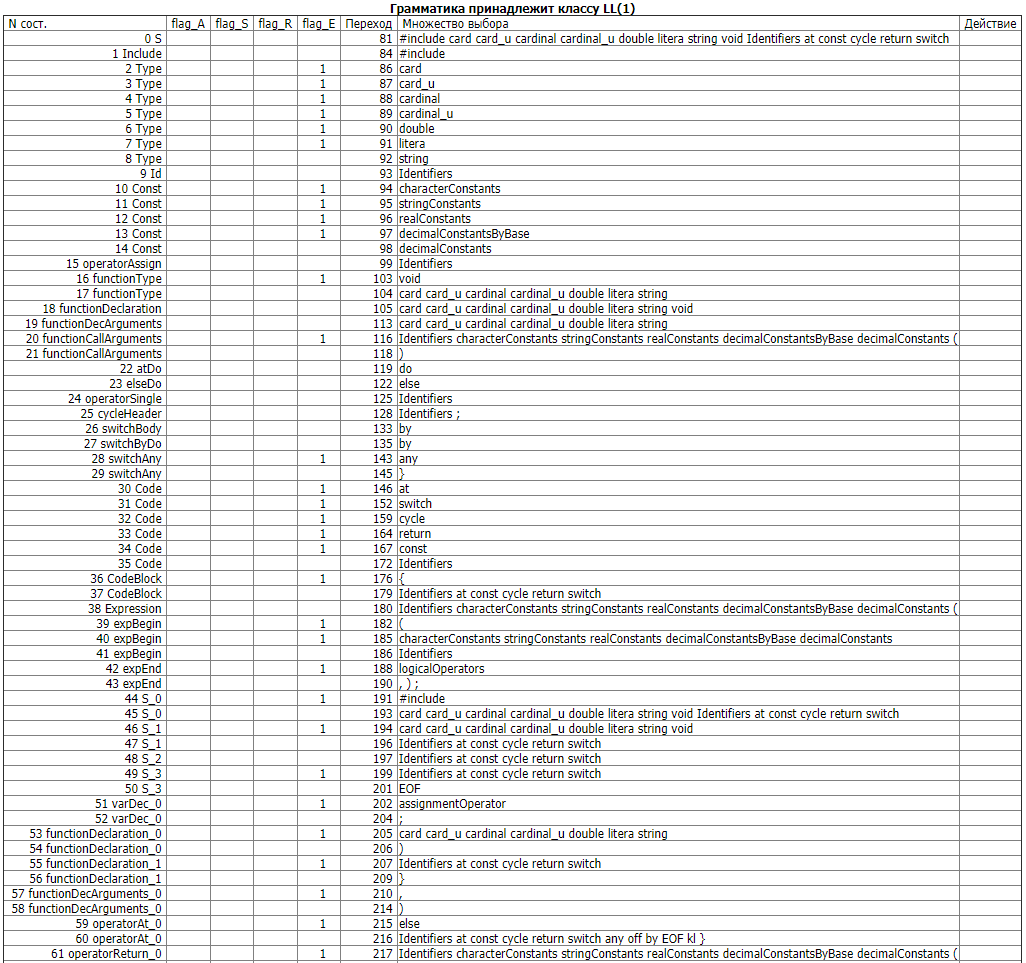
* + текущим входным символом;
  + символом, находящимся на верхушке стека.

Поведение такого автомата определяется управляющей таблицей, столбцы которой соответствуют входным символам, строки - символам, которые могут находиться в стеке, а в клетках указана некоторая последовательность операций над стеком, входным потоком и состоянием автомата.

# Вывод

В ходе выполнения данной лабораторной работы были изучены основные идеи и понятия нисходящих методов синтаксического анализа. Была доработана грамматика языка в соответствие с замечаниями и приведена к классу LL(1)*.*

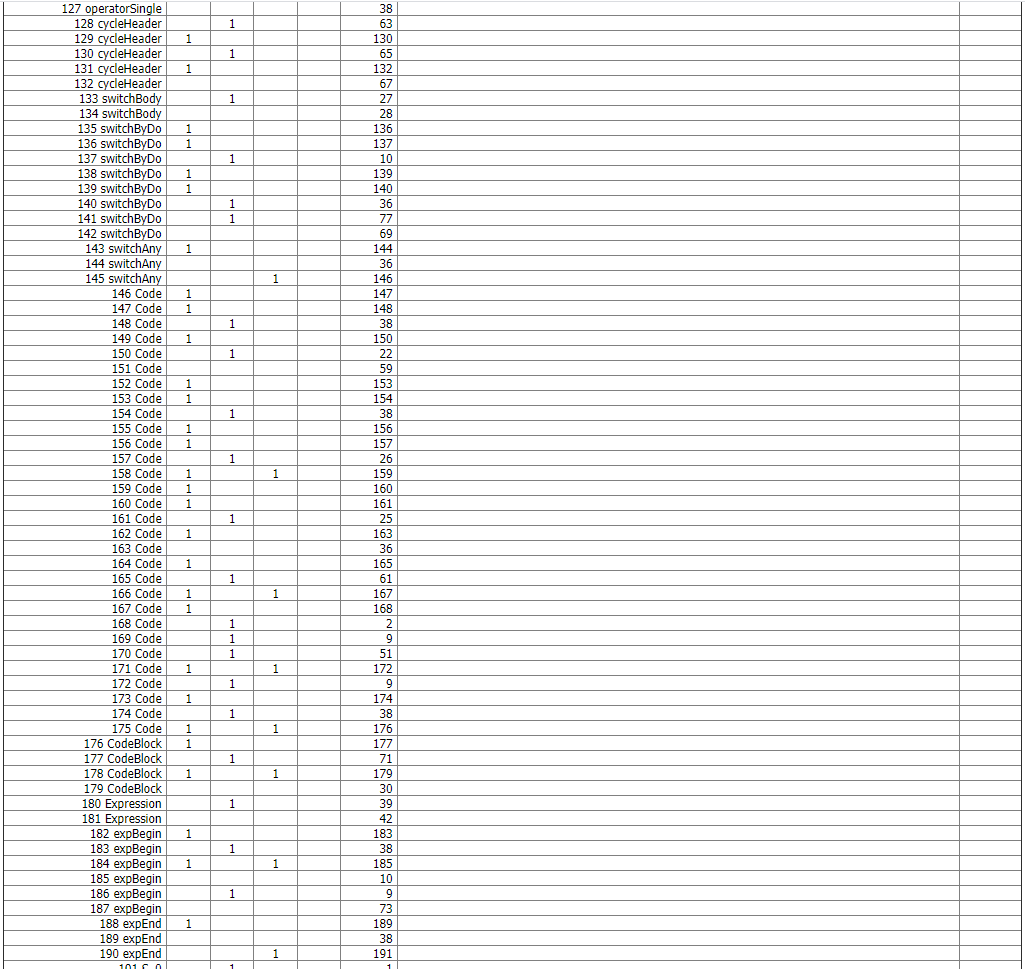
# Приложение 1



# Приложение 1



# Приложение 1



# Приложение 1

