Грамматика

Программа ::= список_определений

Тип ::= значимый_тип | **void**

значимый_тип ::= **short** | **int** | **long** | **float** | **double**

Определение ::= **void** определение_функции | определение_элементов

определение_элементов ::= значимый_тип определение_функции_или_переменных

определение_функции_или_переменных ::=

определение_функции | список_определений_переменных **;**

определение_переменной ::= определение_простой_переменной | определение_массива

определение_простой_переменной ::= идентификатор_переменной возможная_инициализация

возможная_инициализация ::= **=** выражение

определение_массива ::= идентификатор_массива(Syntax_IdentArr) список_границ

граница ::= **[** целое **]**

определение_функции ::= идентификатор_функции(Syntax_IdentFunc) формальные_параметры тело_функции

тело_функции ::= ; | **{}** | ‘{‘ определения_переменных_операторы ‘}’

формальные_параметры ::= **()** | (список формальных параметров)

формальный параметр ::= значимый_тип определение_параметра

определение_параметра ::= идентификатор_переменной | идентификатор_массива список_границ

определения_переменных_операторы ::= список_определений_переменных_или_операторов

определение_переменных_или_оператор ::=

определение_переменных_и_массивов | оператор | выражение_оператор

выражение_оператор ::= ; | выражение ;

определение_переменных_и_массивов ::= значимый_тип список_определений_переменных **;**

оператор ::= If | Do | While | For | Switch | Return | Continue | Break | Goto | метка

Expression ::= 13 приоритетов операций, выражения через запятую

Отладка

Expression ::= Целое – отлаживается структура программы

S_ ::= Expression - отлаживается выражение

Задание

Дописать грамматику в части операторов

Примечание

В окончательных вариантах грамматик есть отличия от приведенной выше.

При описании оператора **if** вероятно сообщение о дублировании.

Описание грамматики

Есть две грамматики, записанные по правилам итерационных формул

Grammar.bif – Тип определяется сканером как совокупность лексем

- Syntax_DefType – для переменных и массивов (**short**, **int**, **long**, **float**, **double**)
- Syntax_DefFunc – для функций (**void**, **short**, **int**, **long**, **float**, **double**)

GrammarType.bif – Тип определяется в грамматике как совокупность правил

Type ::= { ValueType | void }

ValueType ::= { short | int | long | float | double }

Шаги выполнения

Шаг первый

Grammar.bat или GrammarType.bat разбирают описание грамматики, проверяют ее на некие правила, формируют таблицу грамматики Grammar.tab, для работы на следующих шагах.

Проверяется дублирование правил и отсутствие описаний используемых формул.

В файле Grammar.lexeme указываются лексемы, используемые в грамматике.

В файл Grammar.grammar выводится оптимизированное представление грамматики.

Шаг второй – прямое обращение к сканеру

Rulus.bat – читает грамматику Grammar.tab и проверяет по ней пример test_g.cpp

Одновременно создается абстрактное синтаксическое дерево AST test_g.auto.

Rulus!.bat – включает отладочный режим

Шаг второй – запись лексем в промежуточный файл

WriteLex.bat или WriteTypeLex.bat – читают пример test_g.cpp и преобразуют его в файл лексем test_g.lex

RulusLex.bat – читает грамматику Grammar.tab и проверяет по ней пример test_g.lex