# 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