%token BOOL CHAR FLOAT INT STRING

%token TRUE FALSE CHAR\_LITERAL INT\_LITERAL FLOAT\_LITERAL STRING\_LITERAL RAW\_STRING\_LITERAL

%token ID

%token FOR LOOP WHILE IN CONTINUE

%token IF ELSE

%token LET MUT CONST

%token FN ENUM STRUCT TRAIT

%token IMPL SELF BIG\_SELF PUB SELF\_REF MUT\_SELF\_REF MUT\_REF MOD SUPER

%token ';' RIGHT\_ARROW

/\* BREAK и RETURN в документации почему-то присуствует в приоритетах операций. Стоит наверное с этим разобраться \*/

%nonassoc BREAK RETURN

%nonassoc '{' '}'

%right ':'

%right '='

%nonassoc RANGE /\* .. \*/

%left OR /\* || \*/

%left AND /\* && \*/

%left '<' '>' EQUAL NOT\_EQUAL LESS\_EQUAL GREATER\_EQUAL // == != <= >=

%left '+' '-'

%left '\*' '/' '%'

%left '!' '&' UMINUS USTAR /\* - \* \*/

%nonassoc '?'

%left '.' '[' DOUBLEDOTS

%nonassoc '(' ')'

%%

/\* ---------------------- PROGRAM --------------------------- \*/

Program: StmtListEmpty /\* Необходимо уточнить, надо ли как-то обозначить, что Stmt все должны быть Item, иначе программа не заработает \*/

/\* ----------------------------- STATEMENT ----------------------------- \*/

StmtListEmpty: /\* empty \*/

| StmtList

;

StmtList: Stmt

| StmtList Stmt

;

Stmt: ';'

| Item

| LetStmt

| ExprStmt

;

ItemListEmpty: /\*empty\*/

| ItemList

;

ItemList: Item

| ItemList Item

;

Item: SimpleItem

| Visibility SimpleItem

;

SimpleItem: FuncStmt

| StructStmt

| EnumStmt

| ImplStmt

| TraitStmt

| ConstStmt

| ModuleStmt

;

/\* ModuleStmt \*/ /// Можно добавить

/\* ---------- Function ------------ \*/

FuncStmt: DecFuncStmt

| ImplFuncStmt

;

DecFuncStmt: FN ID '(' FuncParamListEmpty ')' ';'

| FN ID '(' FuncParamListEmpty ')' RIGHT\_ARROW Type ';'

;

ImplFuncStmt: FN ID '(' FuncParamListEmpty ')' BlockExpr

| FN ID '(' FuncParamListEmpty ')' RIGHT\_ARROW Type BlockExpr

;

FuncParamListEmpty: /\* empty \*/

| FullFuncParamList

;

FullFuncParamList: SelfParam

| SelfParam ',' FuncParamList

| FuncParamList

;

SelfParam: SELF

| SELF\_REF

| MUT\_SELF\_REF

;

FuncParamList: FuncParam

| FuncParamList ',' FuncParam

;

FuncParam: ID ':' Type /\* Возможен конфликт \*/

| MUT ID ':' Type

| ID ':' MUT\_REF Type

| ID ':' '&' Type

/\* ========== Struct =========== \*/

StructStmt: StructStruct

| TupleStruct /\* Надо ли ? \*/

;

StructStruct : STRUCT ID '{' StructFieldListEmpty '}'

;

StructFieldListEmpty: /\* empty \*/

| StructFieldList

| StructFieldList ','

;

StructFieldList: StructField

| StructFieldList ',' StructField

;

StructField: ID ':' Type /\* Возможен конфликт с FunctionParam \*/

| Visibility ID ':' Type

;

/\*--- TupleStruct ----\*/

TupleStruct: STRUCT ID '(' TupleFieldListEmpty ')'

;

TupleFieldListEmpty: /\* empty \*/

| TupleFieldList

| TupleFieldList ','

;

TupleFieldList: Type /\* Возможен конфликт \*/

| Visibility Type

| TupleFieldList ',' Type

| TupleFieldList ',' Visibility Type

;

/\* ============= ENUM ================ \*/

EnumStmt: ENUM ID '{' EnumItemListEmpty '}'

;

EnumItemListEmpty: /\* empty \*/

| ','

| EnumItemList

| EnumItemList ','

;

EnumItemList: EnumItem

| EnumItemList ',' EnumItem

;

EnumItem: ID

| Visibility ID

| ID '=' ExprWithBlock

| ID '=' ExprWithoutBlock

| Visibility ID '=' ExprWithBlock

| Visibility ID '=' ExprWithoutBlock

;

/\* =========== IMPL ================ \*/

ImplStmt: IMPL Type '{'AssociatedItemListEmpty '}'

| IMPL ID FOR Type '{'AssociatedItemListEmpty '}'

;

AssociatedItemListEmpty: /\* empty \*/

| AssociatedItemList

;

AssociatedItemList: AssociatedItem

| AssociatedItemList AssociatedItem

;

/\* Необходима еще проверка для Impl то что FuncStmt является именно реализацией \*/

AssociatedItem: FuncStmt

| ConstStmt

| Visibility FuncStmt

| Visibility ConstStmt

;

/\* ============ TRAIT ================ \*/

TraitStmt: TRAIT ID '{' AssociatedItemListEmpty '}'

;

/\* ============ CONST =============== \*/

ConstStmt: CONST ID ':' Type '=' ExprWithBlock ';'

ConstStmt: CONST ID ':' Type '=' ExprWithoutBlock ';'

| CONST ID ':' Type ';'

;

/\* =========== Module ================= \*/

ModuleStmt: MOD ID ';'

MOD ID '{' ItemListEmpty '}'

/\* ========= LetStmt ============ \*/

LetStmt: LET ID '=' ExprWithBlock ';'

LetStmt: LET ID '=' ExprWithoutBlock ';'

| LET ID ':' Type '=' ExprWithBlock ';'

| LET ID ':' Type '=' ExprWithoutBlock ';'

| LET MUT ID ';'

| LET MUT ID ':' Type ';'

| LET MUT ID '=' ExprWithBlock ';'

| LET MUT ID '=' ExprWithoutBlock ';'

| LET MUT ID ':' Type '=' ExprWithBlock ';'

| LET MUT ID ':' Type '=' ExprWithoutBlock ';'

;

/\* === Expression Statement === \*/

ExprStmt: ExprWithoutBlock ';'

| ExprWithBlock ';'

;

/\* конфликт при "| ExprWithBlock"

/\*----------------------- EXPRESSION ---------------------- \*/

ExprListEmpty: /\*empty\*/

| ExprList ','

| ExprList

;

ExprList: ExprWithBlock

| ExprWithoutBlock

| ExprList ',' ExprWithBlock

| ExprList ',' ExprWithoutBlock

ExprWithoutBlock: CHAR\_LITERAL // Литераллы

| STRING\_LITERAL

| RAW\_STRING\_LITERAL

| INT\_LITERAL

| FLOAT\_LITERAL

| TRUE

| FALSE

| ExprWithoutBlock '+' ExprWithoutBlock // Мат операции

| ExprWithoutBlock '+' ExprWithBlock

| ExprWithBlock '+' ExprWithoutBlock

| ExprWithBlock '+' ExprWithBlock

| ExprWithoutBlock '-' ExprWithoutBlock

| ExprWithoutBlock '-' ExprWithBlock

| ExprWithBlock '-' ExprWithoutBlock

| ExprWithBlock '-' ExprWithBlock

| ExprWithoutBlock '/' ExprWithoutBlock

| ExprWithoutBlock '/' ExprWithBlock

| ExprWithBlock '/' ExprWithoutBlock

| ExprWithBlock '/' ExprWithBlock

| ExprWithoutBlock '\*' ExprWithoutBlock

| ExprWithoutBlock '\*' ExprWithBlock

| ExprWithBlock '\*' ExprWithoutBlock

| ExprWithBlock '\*' ExprWithBlock

| ExprWithoutBlock '%' ExprWithoutBlock

| ExprWithoutBlock '%' ExprWithBlock

| ExprWithBlock '%' ExprWithoutBlock

| ExprWithBlock '%' ExprWithBlock

| ExprWithoutBlock AND ExprWithoutBlock // Логические операции

| ExprWithoutBlock AND ExprWithBlock

| ExprWithBlock AND ExprWithoutBlock

| ExprWithBlock AND ExprWithBlock

| ExprWithoutBlock OR ExprWithoutBlock

| ExprWithoutBlock OR ExprWithBlock

| ExprWithBlock OR ExprWithoutBlock

| ExprWithBlock OR ExprWithBlock

| ExprWithoutBlock '=' ExprWithoutBlock // ПРИСВАИВАНИЕ

| ExprWithoutBlock '=' ExprWithBlock

| ExprWithBlock '=' ExprWithoutBlock

| ExprWithBlock '=' ExprWithBlock

| ExprWithoutBlock EQUAL ExprWithoutBlock // СРАВНЕНИЕ

| ExprWithoutBlock EQUAL ExprWithBlock

| ExprWithBlock EQUAL ExprWithoutBlock

| ExprWithBlock EQUAL ExprWithBlock

| ExprWithoutBlock NOT\_EQUAL ExprWithoutBlock

| ExprWithoutBlock NOT\_EQUAL ExprWithBlock

| ExprWithBlock NOT\_EQUAL ExprWithoutBlock

| ExprWithBlock NOT\_EQUAL ExprWithBlock

| ExprWithoutBlock '>' ExprWithoutBlock

| ExprWithoutBlock '>' ExprWithBlock

| ExprWithBlock '>' ExprWithoutBlock

| ExprWithBlock '>' ExprWithBlock

| ExprWithoutBlock '<' ExprWithoutBlock

| ExprWithoutBlock '<' ExprWithBlock

| ExprWithBlock '<' ExprWithoutBlock

| ExprWithBlock '<' ExprWithBlock

| ExprWithoutBlock GREATER\_EQUAL ExprWithoutBlock

| ExprWithoutBlock GREATER\_EQUAL ExprWithBlock

| ExprWithBlock GREATER\_EQUAL ExprWithoutBlock

| ExprWithBlock GREATER\_EQUAL ExprWithBlock

| ExprWithoutBlock LESS\_EQUAL ExprWithoutBlock

| ExprWithoutBlock LESS\_EQUAL ExprWithBlock

| ExprWithBlock LESS\_EQUAL ExprWithoutBlock

| ExprWithBlock LESS\_EQUAL ExprWithBlock

| '(' ExprWithBlock ')'

| '(' ExprWithoutBlock ')'

| '-' ExprWithoutBlock %prec UMINUS // ОТРИЦАНИЕ

| '-' ExprWithBlock %prec UMINUS

| '!' ExprWithoutBlock

| '!' ExprWithBlock

| ExprWithoutBlock '?' //

| ExprWithBlock '?'

| '\*' ExprWithoutBlock %prec USTAR // ГАДОСТЬ

| '\*' ExprWithBlock %prec USTAR

| '&' ExprWithoutBlock

| '&' ExprWithBlock

| '[' ExprListEmpty ']' // ЗАДАЕМ МАССИВ

| '[' ExprWithoutBlock ';' ExprWithoutBlock ']'

| '[' ExprWithoutBlock ';' ExprWithBlock ']'

| '[' ExprWithBlock ';' ExprWithoutBlock ']'

| '[' ExprWithBlock ';' ExprWithBlock ']'

| ExprWithoutBlock '[' ExprWithoutBlock ']' // Обращаемся к элементу массива

| ExprWithoutBlock '[' ExprWithBlock ']'

| ExprWithBlock '[' ExprWithoutBlock ']'

| ExprWithBlock '[' ExprWithBlock ']'

| ExprWithoutBlock '.' INT\_LITERAL // Обращаемся к элементу tuple

| ExprWithBlock '.' INT\_LITERAL

| CONTINUE

| RANGE

| BREAK ExprWithoutBlock

| BREAK ExprWithBlock

| BREAK

| RANGE ExprWithoutBlock // диапозон

| RANGE ExprWithBlock

| ExprWithoutBlock RANGE

| ExprWithBlock RANGE

| ExprWithoutBlock RANGE ExprWithoutBlock

| ExprWithoutBlock RANGE ExprWithBlock

| ExprWithBlock RANGE ExprWithoutBlock

| ExprWithBlock RANGE ExprWithBlock

| RETURN

| RETURN ExprWithoutBlock

| RETURN ExprWithBlock

| '(' ExprListEmpty ')'

| ExprWithoutBlock '(' ExprListEmpty ')'

| ExprWithBlock '(' ExprListEmpty ')'

| ExprWithoutBlock '.' ID

| ExprWithBlock '.' ID

| ExprWithoutBlock '.' ID '(' ExprListEmpty ')' // Конфликт вроде как решается сдвигом т.е. действием по умолчанию

| ExprWithBlock '.' ID '(' ExprListEmpty ')'

| SUPER

| PathCallExpr

| PathCallExpr '(' ExprListEmpty ')' //Вызов функции по пути // Cпросить можно ли сделать более простую реализацию

| PathCallExpr '{' StructExprFieldList '}' // тут че-то все поплыло

;

/\* CRATE DOLLAR\_CRATE отправляются на свалку Struct Tuple тоже) \*/

PathCallExpr: ID DOUBLEDOTS ID

| SUPER DOUBLEDOTS ID

| SELF DOUBLEDOTS ID

| PathCallExpr DOUBLEDOTS ID

StructExprFieldList: StructExprField

StructExprField ',' StructExprField

StructExprField: ID

| ID ':' ExprWithoutBlock

| ID ':' ExprWithBlock // Мб сделать проще и юзать тут тип?

ExprWithBlock: BlockExpr

| LoopExpr

| IfExpr

;

BlockExpr: '{' StmtListEmpty '}'

;

LoopExpr: InfiniteLoopExpr

| PredicateLoopExpr

| IteratorLoopExpr

;

InfiniteLoopExpr: LOOP BlockExpr

;

PredicateLoopExpr: WHILE ExprWithBlock BlockExpr

| WHILE ExprWithoutBlock BlockExpr

;

IteratorLoopExpr: FOR ID IN ExprWithBlock BlockExpr

| FOR ID IN ExprWithoutBlock BlockExpr

;

IfExpr: IF ExprWithoutBlock BlockExpr

| IF ExprWithBlock BlockExpr

| IF ExprWithoutBlock BlockExpr ELSE BlockExpr

| IF ExprWithBlock BlockExpr ELSE BlockExpr

| IF ExprWithoutBlock BlockExpr ELSE IfExpr

| IF ExprWithBlock BlockExpr ELSE IfExpr

;

/\*-------------------------TYPE -------------------------- \*/

Type: BOOL

| CHAR

| FLOAT

| INT

| STRING

| ID

| '[' Type ';' ExprWithBlock ']'

| '[' Type ';' ExprWithoutBlock ']'

;

/\* Не доделан. Можно добавить TupleType \*/

/\*---------------------- VISIBILITY ------------------------- \*/

Visibility: PUB

| PUB '(' SUPER ')'

| PUB '(' SELF ')'

;