Laba Grammar

1.

decs

bool x;

sced

x 8= true;

2.

decs

int x;

int y;

sced

x 8= x+y;

3.

decs

bool x;

sced

x 8= true;

if8 x ==8 true

~

x 8= false;

~;

4.

decs

int countel;

sced

countel 8= 0;

while8 countel < 2

~

countel 8= countel +1;

~

5.

decs

arr(5,int) x;

sced

x 8= [1,2,3,4,5];

plint(x[2]);

6.

decs

int countel;

def8 plintAddOne(int var)

~

plint(var+1);

~;

sced

countel 8= 2;

call8 plintAddOne(countel);

|  |  |
| --- | --- |
| Program | Declarations OneStatement\* |
| Declarations | **decs** OneDeclaration\* **sced**  **e** |
| OneDeclaration | **TypeIdentifier** Identifier**;**  **def8** Identifier**(**(HeaderList|**e**)**)**  **~**  OneStatement\*  **~;** |
| HeaderList | Header (, Header)\* |
| Header | **TypeIdentifier** Identifier |
| OneStatement | **if8** LogicalExpression **~** OneStatement\* **~**  (**e**|**else8** ~OneStatement\*~)**;**  **while8** LogicalExpression **~** OneStatement\* **~;**  Identifier **8=** (Expression | ValueList | lead())**;**  **plint(**Expression**);**  **call8** Identifier **(** ( ExpressionList | **e** ) **)**; |
| ExpressionList | Expression **( ,** Expression **)** \* |
| Expression | primary-Expression (Operator primary-Expression)\* |
| primary-Expression | **Identifier** | **Value**  ( Expression )  Identifier[IntegerLiteral] |
| LogicalExpression | primary-Expression LogicalOperator primary-Expression |
| Operator | **8=** (Assignment Operator)  **+**  **-**  **\***  **/** |
| LogicalOperator | **==8** (Equivalence Operator)  **<**  **>** |
| TypeIdentifier | **int**  **bool**  **double**  **arr(**IntegerLiteral, int|bool|double**)** |
| ValueList | **Value**  **[Value,**ValueList**]** |
| Value | **IntegerLiteral**  **DoubleLiteral**  **BoolLiteral**  **Identifier** |