START:

<start>🡪 def <function><start> | <Class><start> | <For><start> | <While><start> | <If><start> | <Try><start> | <Del> nl <start> |<initialize>nl <start> | €

INITIALIZATION STATEMENT:

<initialize>🡪ID icg\_lineString += tlist.at(itr).getValuePart(); <ID\_rel\_>|<const><init2>|\*<init3>| accessModifier init4>|<static\_final>ID<ID\_rel> = <init5> | this AcOp ID<ID\_rel><init1> | (<exp>)<OE> | Not icg\_lineString += "Not "; <exp\_F> <OE> Output (icg\_lineString + icg\_tempOprand) | <List2> | <Dictionary2>

<init1>🡪AsOP icg\_tempOprand = "=" + icg\_lineString + op[0]; <exp> Output (icg\_lineString + icg\_tempOprand) |= icg\_lineString += "="; <init5> |<init2>

<init2> 🡪 <OE> Output (icg\_lineString + icg\_tempOprand) <In>

<init3>🡪accessModifier <init4>|<init4>

<init4>🡪<static\_final>ID<ID\_rel>=<init5>|ID<ID\_rel> = <init5>

<init5>🡪<const><OE> Output (icg\_lineString + icg\_tempOprand) |<This>ID icg\_tempOprand += N; icg\_tempOprand += N; <ID\_rel><init6> | <List> Output (icg\_lineString + icg\_tempOprand) | <Dictionary> Output (icg\_lineString + icg\_tempOprand) | <Class\_call> | <lambda>

<init6>🡪= icg\_lineString = icg\_tempOprand + "="; <init5> icg\_lineString = icg\_tempStack.back() + icg\_tempOprand; Output (icg\_lineString) | <init2>

<ID\_rel\_>🡪<ID\_rel1\_>| (<arg\_list\_call>) Output (icg\_lineString) <AcOP>

<ID\_rel1\_>🡪[<exp>] <ID\_rel1\_>| AcOp ID <ID\_rel\_> | <init1>

<AcOP>🡪 AcOp ID <ID\_rel\_>|Є

<AM>🡪 accessModifier | €

<static\_final>🡪 static <s\_final> | final

<s\_final>🡪 final | €

<id\_const>🡪<This> ID icg\_tempOprand += N; <ID\_rel> | <const>

<const>🡪 int\_const icg\_tempOprand = tlist.at(itr).getValuePart(); | float\_const icg\_tempOprand = tlist.at(itr).getValuePart(); | string\_const icg\_tempOprand = tlist.at(itr).getValuePart(); | char\_const icg\_tempOprand = tlist.at(itr).getValuePart(); | bool\_const icg\_tempOprand = tlist.at(itr).getValuePart();

<Global>🡪 global <initialize>

<pointer>🡪 \* | €

EXPRESSION:

<OE>🡪<exp\_MDM><exp\_PM><exp\_RELOP><exp\_AND><exp\_OR>

<exp>🡪<expAND><exp\_OR>

<exp\_OR>🡪 Or icg\_tempOprand += op; <expAND><exp\_OR> | €

<expAND>🡪<expRELOP><exp\_AND>

<exp\_AND>🡪 And icg\_tempOprand += op; <expRELOP><exp\_AND> | €

<expRELOP>🡪<expPM><exp\_RELOP>

<exp\_RELOP>🡪 RelOp icg\_tempOprand += tlist.at(itr).getValuePart(); <expPM><exp\_RELOP> | €

<expPM>🡪<expMDM><exp\_PM>

<exp\_PM>🡪 PM icg\_tempOprand += op; <expMDM><exp\_PM> | €

<expMDM>🡪<exp\_F><exp\_MDM>

<exp\_MDM>🡪 DM icg\_tempOprand += op; <exp\_F><exp\_MDM> | \* icg\_tempOprand += op; <exp\_F><exp\_MDM> |€

<exp\_F>🡪<This>ID <ID\_rel><In> | <const><In> | (<exp>) | Not <exp\_F> | <lambda>

<ID\_rel> 🡪 [exp]<ID\_rel1> | (<arg\_list\_call>)<ID\_rel1> | AcOp ID <ID\_rel> | €

<ID\_rel1> 🡪 AcOp ID <ID\_rel> | <ID\_rel2>

<ID\_rel2> 🡪[exp]<ID\_rel1> | €

<This>🡪 this AcOp | €

<In>🡪 in icg\_tempOprand += " in "; <In\_>| €

<In\_>🡪 ID icg\_tempOprand += N; | <List>

BODY:

<body>🡪<S\_St> |nl IndentInit <M\_St> IndentOut 1

<M\_St>🡪<S\_St> icg\_lineString = icg\_tempOprand = ""; <M\_St\_>

<M\_St\_>🡪<M\_St> | €

<S\_St>🡪 <S\_St1> nl| <S\_St2>

<S\_St1>🡪 break icg\_writer << "JMP " << icg\_tempLabelStack.back() << endl; | continue icg\_writer << "JMP " << icg\_tempLabelStack.back() << endl;| return <sst1> icg\_writer << "RET " << icg\_tempLabelStack.back() << endl; | <Del>| <Global> |<initialize> | pass icg\_writer << "halt" << endl;

<S\_St2>🡪<For> | <While> | <If> | <Try>

<sst1> 🡪 <exp> | <List2> | Dictionary2>  
<NL>🡪 nl | €

CLASS:

<Class>🡪 icg\_lineString = icg\_tempOprand = ""; class ID (<inherit>) : nl IndentInit <class\_body><NL> IndentOut

<Inherit>🡪 ID <inherit\_> | €

<inherit\_>🡪 , ID | €

<class\_body>🡪<class\_body1> | pass

<class\_body1>🡪 icg\_lineString = currentClass + "\_"; <initialize> nl <class\_body3> | icg\_lineString = currentClass + "\_";def <class\_body2><class\_body3>

<class\_body2>🡪<constructor> | <function>

<class\_body3>🡪< class\_body1> | €

CLASS CALL:

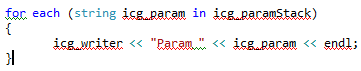
<Class\_call>🡪 new ID ( <arg\_list\_call> )

CONSTRUCTOR:

<constructor>🡪<AM> ID icg\_lineString += N; (<arg\_list>) icg\_lineString += ", " + to\_string(icg\_argCount) + " proc"; icg\_writer << icg\_lineString << endl; : <body> icg\_writer << "endp" << endl << endl;

<arg\_list>🡪<data\_type><pointer> ID <arg\_list1> | €

<arg\_list1>🡪 , <data\_type><pointer> ID <arg\_list1> | €

<arg\_list\_call>🡪deque<string> icg\_paramStack; icg\_lineString = icg\_tempOprand;<exp> icg\_lineString += "\_" + split(T, "\_").front(); <arg\_list\_call1(icg\_paramStack)>  icg\_tempOprand = "Call " + currentClass + "\_" + icg\_lineString + ", " + to\_string(icg\_argCount) + ""; icg\_lineString = icg\_tempStack.back(); | €

<arg\_list\_call1(icg\_paramStack)>🡪 , <exp><arg\_list\_call1> | €

FUNCTION:

<function>🡪 icg\_lineString = currentClass + "\_"; <data\_type><AM><function\_>

<function\_> 🡪 <static\_final> ID icg\_lineString += N; (<arg\_list>) icg\_lineString += ", " + to\_string(icg\_argCount) + " Proc"; icg\_writer << icg\_lineString << endl; : <body> icg\_writer << "endp" << endl; | ID icg\_lineString += N; (<arg\_list>) icg\_lineString += ", " + to\_string(icg\_argCount) + " Proc"; icg\_writer << icg\_lineString << endl; : <body> icg\_writer << "endp" << endl;

<data\_type>🡪 DT icg\_lineString += "\_" + this.at(itr).getValuePart(); | string icg\_lineString += "\_" + this.at(itr).getValuePart(); | ID icg\_lineString += "\_" + this.at(itr).getValuePart();

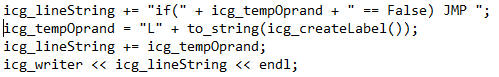
LOOPS:

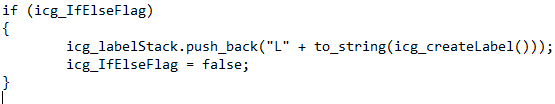
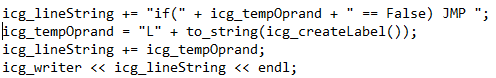
<For>🡪 for icg\_writer << endl << "L" + to\_string(icg\_createLabel()<< ":" << endl; ID  in icg\_tempOprand += " in "; <For\_> icg\_writer << icg\_lineString + icg\_tempOprand << endl; icg\_writer << "if(" << icg\_lineString << " == False) JMP " << "L" + to\_string(icg\_createLabel()<< endl;:<body> 

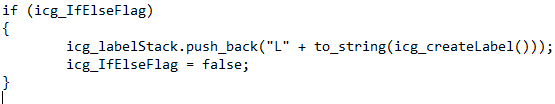
<For\_>🡪 ID icg\_tempOprand += tlist.at(itr).getValuePart(); <ID\_rel> | <List2> | <Dictionary2>

<While>🡪 while icg\_writer << endl << "L" + to\_string(icg\_createLabel()<< ":" << endl; (<exp> icg\_writer << "if(" + icg\_tempOprand + " == False) JMP " << "L" + to\_string(icg\_createLabel()<< endl; ) : <body> 

IF\_ELIF\_ELSE:

<If>🡪 if (<exp>) :  string temp = icg\_tempOprand; <body> icg\_tempOprand = temp; <Elif (icg\_labelStack, icg\_IfElseFlag)>

<Elif (icg\_labelStack, icg\_IfElseFlag)>🡪 elif   (<exp>) :  temp = icg\_tempOprand; <body> icg\_tempOprand = temp; <Elif (icg\_labelStack, icg\_IfElseFlag)> | <Else(icg\_labelStack, icg\_IfElseFlag)>

<Else(icg\_labelStack, icg\_IfElseFlag)>🡪 else   icg\_writer << icg\_tempOprand << ":" << endl; : <body> icg\_writer << icg\_labelStack.back() << ":" << endl; | €

LIST/ARRAY:

<List>🡪<List2> | List icg\_tempOprand += "List"; (icg\_tempOprand += "("; <List1> ) icg\_tempOprand += ")";

<List1>🡪<List2> | ID icg\_tempOprand += tlist.at(itr).getClassPart();

<List2>🡪 [icg\_tempOprand += "["; <List3> ] icg\_tempOprand += "]";

<List3>🡪<id\_const><List4> | <List2><List4> | <Dictionary2> <List4>|Є

<List4>🡪 , icg\_tempOprand += ","; <List5> | €

<List5> 🡪 <id\_const><List4> | <List2><List4> | <Dictionary2> <List4>

DICTIONARY:

<Dictionary>🡪<Dictionary2> | dict icg\_tempOprand += "dict"; (icg\_tempOprand += "("; <Dictionary1>) icg\_tempOprand += ")";

<Dictionary1>🡪 IDicg\_tempOprand += tlist.at(itr).getValuePart(); | <Dictionary2> | €

<Dictionary2>🡪 { icg\_tempOprand += "{"; <Dictionary3> } icg\_tempOprand += "}";

<Dictionary3>🡪<id\_const> : icg\_tempOprand += ":"; <Dictionary4>| €

<Dictionary4>🡪 <id\_const> <Dictionary6> | <Dictionary2> <Dictionary6> | <List2> <Dictionary6>

<Dictionary6>🡪 , icg\_tempOprand += ",";<Dictionary7> | €

<Dictionary7> 🡪<id\_const> : icg\_tempOprand += ":"; <Dictionary4>

DEL:

<Del>🡪 del icg\_tempOprand += "del"; ID icg\_tempOprand += N; <Del\_> icg\_writer << icg\_lineString + icg\_tempOprand << endl;

<Del\_>🡪 [icg\_tempOprand += " + ";<exp>] icg\_writer << icg\_lineString + icg\_tempOprand << endl; | €

TRY\_EXCEPT\_FINALLY:

<Try>🡪 try : <body><Except>

<Except>🡪<Finally> | except (<Exception>) : <body><Except>

<Finally>🡪 finally : <body>

<Exception>🡪 exception ID

LAMBDA:

<Lambda>🡪 lambda ID : <exp>