1. <START> 🡪 ***{ S = CreateScope() }*** <DEFS***(S)***>***{ DeleteScope() }***
2. <DEFS***(S)***> 🡪 <MST***(S)***> <DEFS***(S)***> |

<CLASS> <DEFS***(S)***> |

<FUNCTION-DEC><DEFS***(S)***> | $

1. <MST ***(S, &RT)***> 🡪 € |

<SST ***(S, &RT)***><MST ***(S, &RT)***>

1. <SST ***(S, &RT)>*** 🡪 <SST1 ***(S, &RT)*** >

* 1. <SST1 ***(S, &RT)*** >🡪 <DEC***(S)***> | <WHILE ***(S)*** > | <FOR ***(S)*** > | <DO-WHILE ***(S)*** > | <CONST-DT ***(S)*** > | <INC-DEC-PRE ***(S)*** > | <IF-ELSE ***(S)*** > |<SWITCH-CASE***(S)*** > | <RETURN***(S, &RT)***>

|<GTSWID ***(S)*** >

* 1. <WHILE> 🡪 while (<EXP***(&T)***> ***{if(!Compatibility(&T,bool,”cond”)) {Type Mismatch}}***)

<BODY***(S)***>

1. <DO-WHILE> 🡪 do ***{S = CreateScope() }***

<BODY***(S)***> while ( <EXP***(&T)***> ) ***{ DeleteScope() }***

1. <FUNCTION-DEC>🡪 function <FUNC-DEF-1>
   1. <FUNC-DEF-1>🡪ID ***{N = ID. VP}***

(***{S = CreateScope ()}***

<DEC-PARAMS ***(&PL,S)***>)

{<MST ***(S, &RT)***>} ***{if (! insertCT (N, PL->RT, null, null) {Redeclaration Error} DeleteScope ()}***

1. <CLASS> 🡪 class ID ***{N = ID. VP}*** <CLASS-STRUCT***(N)***>
   1. <CLASS-STRUCT ***(N)***> 🡪 <CLASS-BODY ***(N, null)***> |

: ID ***{PN = ID.VP if(LookupCT(PN)){ PL = ID.VP }}*** <CLASSBODY ***(N, PL)***>

* 1. <CLASSBODY ***(N, PL)***>🡪***{ref = createRef() if(!insertCT(N, ”class”, PL , & ref) Redeclaration Error }***

{<CLASS-MST ***(&ref)***>}

* 1. <CLASS-MST ***(&ref)***> 🡪 € |

<CLASS-CONSTRUCOR ***(&ref)***> <CLASS-ST***(&ref)***> <CLASS-MST ***(&ref)***>

* + 1. <CLASS-CONSTRUCOR ***(&ref)***> 🡪 € | constructor ***{S = CreateScope()}***  (<DEC-PARAMS **(&PL,S)**>) ***{if(!insertCDT(N,PL->void,*** ***public, null, ref)) {Redeclaring Constructor}*** {<CONSTRUCTOR-BODY ***(S, ref)***>} ***{DeleteScope()}***
    2. <CONSTRUCTOR-BODY ***(S, ref)***> 🡪 € |

<C-B-I ***(S, ref)***>

<CONSTRUCTOR-BODY ***(S, ref)***>

* + 1. <C-B-I ***(S, ref)***> 🡪 <THIS-VAR ***(S, ref)*** >|<SST1 ***(S, &RT, ref)***>
    2. <THIS\_VAR ***(S, ref)***> 🡪 this. ID ***{N = ID. VP T=lookupCDT(N, ref)}***

AOR <DEC2 ***(TR, S, ref)***>

***{TA­ =Compatibility (T, TR***, ***AOR)***

***if(!TA ){Type Mismatch }***

* 1. <CLASS-ST***(&ref)***> 🡪 <CLASS-FUNC ***(&ref)*** > |

<DEC ***(&ref)*** > |

<PROTECTED ***(&ref)*** >

* 1. <CLASS-FUNC ***(&ref)***> 🡪 ID ***{N = ID.VP }*** <FUNC-DEF***(&ref, N)***>

* 1. <FUNC-DEF ***(&ref, N)***> 🡪 AOR (***{ S = CreateScope() }***

<DEC-PARAMS***(&PL,S)***> ) => { <MST ***(S, &RT)*** > } ***{if(!insertCDT(N, PL->RT, public, null, ref ) { Redeclaration Error }}*** |

( ***{ S = CreateScope() }*** <DEC-PARAMS ***(&PL,S)*** >)

{<MST ***(S, &RT)*** >} ***{if(!insertCDT(N, PL->RT, public, null, ref ) { Redeclaration Error }}***

* 1. <PROTECTED ***(&ref)*** >🡪 protected: <PRO-NEXT ***(&ref)***>
  2. <PRO-NEXT ***(&ref)***> 🡪 <CLASS-ST ***(&ref)***> | {<PRO-BODY ***(&ref)***>}
  3. <PRO-BODY ***(&ref)***> 🡪 <CLASS-MST ***(&ref)***>

1. <DEC\_PARAMS ***(&PL, S)***> 🡪 DT ***{T=DT.VP}***

ID ***{N = ID. VP}***

<DEC1 ***(T, S)***> ***{ if (!insertST(N ,T ,S){Redeclaration***

***Error} else {PL = T}}*** <NEXT\_PARAM ***(&PL, S)*** > |

ID ***{N = ID. VP, T = ’const’}***

<CONST\_DEC\_PARAM ***(T, S)*** > ***{if (!insertST (N ,T ,S){ Redeclaration Error} else {PL = T}}***

<NEXT\_PARAM ***(&PL, S)***> |

€ ***{PL=null}***

* 1. <CONST\_DEC\_PARAM ***(T, S)***> 🡪 € | AOR <CONST\_DT1>
  2. <NEXT\_PARAM ***(&PL, S)***> 🡪 , <NEXT\_DEC\_PARAM ***(&PL, S)***> | €
  3. <NEXT\_DEC\_PARAM ***(&PL, S)***> 🡪 DT ***{T=DT.VP}***

ID ***{N = ID.VP}***

<DEC1 ***(T, S)***> ***{ if (!insertST(N ,T ,S){Redeclaration***

***Error} else {PL += T}}***<NEXT\_PARAM ***(&PL, S)*** > |

ID ***{N = ID. VP, T = ’const’}***

<CONST\_DEC\_PARAM ***(T, S)***> ***{if (!insertST(N ,T, S){ Redeclaration Error} else {PL += T}}***

<NEXT\_PARAM ***(&PL, S)***>

1. <DEC ***(S)***> 🡪 *DT* ***{T = DT. VP}***

ID ***{N = ID. VP}***

***<***DEC1***(T, S)***> ***{if (! inserts (N, T, S)) {Redeclaration}}***

<NEXTDEC ***(T, S)***>

* 1. <DEC1***(T, S)***> 🡪 AOR

<DEC2***(&T****R****, S)****>* ***{TA­ =Compatibility (T, TR***, ***AOR) if(!TA ){Type Mismatch }***| €

* 1. <DEC2***(&TR, S)***> 🡪 ID ***{N = ID. VP, CN=null}***

<DEC3***(N, T, S, CN)***> |

<EXP***(&T)***>

* 1. <DEC3***(N,T,S, CN)***> 🡪 <MERGED\_INIT ***(N,&TL, S, CN)***>

<EXP1***(TL, &T, S)***> |

<EXP1***(TL, &T, S)***> |

€ ***{T= LookupST (N, S) if (!TL){ Undeclared Variable } }***

* 1. <MERGED\_INIT ***(&T, N, CN, S)*** > 🡪 <MOV ***(&T, N, CN, S)*** > |

[<EXP***(T1)***> ***{IF (! Compatibility (T1, INT) {Type Mismatch}***]

<M\_N\_ARR ***(&T, N, S)***>

* 1. <M\_N\_ARR ***(&T, N, S)***> 🡪 <MOV ***(&T, N, S)***> |

[<EXP>***(T1)***> ***{IF (! Compatibility (T1, INT) {Type Mismatch}***]

<M\_N\_ARR2***(&T, N, S)*** > |

€ ***{T=LookupST (N, S) if (!T){ Undeclared***

***Variable }***

* 1. <M\_N\_ARR2 ***(&T, N, S)***> 🡪 <MOV ***(&T, N, S)***> |

€ ***{T=LookupST (N, S) if (! T) {Undeclared Variable}***

* 1. <MOV ***(&T, N, CN, S)***> 🡪 . ***{if(CN==null){CN = LookupST (N, S)}***

***else{ A = LookupCDT(N, CN) if(A==DT){T=A}else{CN=A} }***

***//check if for Undeclared***

ID ***{N= ID. VP}***

<MERGED ***(&T, N, CN, S)***> |

(<CALLING\_PARAMS***(&PL)***>)***{CN=LookupFT(N,PL,CN)}***

<MERGED ***(&T, null, CN, S)***>

* 1. <MERGED ***(&T, N, CN, S)***> 🡪 ***{if ( CN != null ) T1 = LookupCDT (N, CN)}***

<EXP1***(&T, T1)***> |

<MERGED\_INIT ***(&T, N, CN, S)***>

<DEC1>

1. <EXP ***(&T, S)***> 🡪 <VAL ***(&TL, S)***> <EXP1 ***(&TL, &T, S)***>
   1. <EXP1 ***(&TL, &T, S)***> 🡪 € ***{T = TL}*** |

<Q\_DASH ***(T­L ,&T ,S)***>

<R\_DASH ***(T­L ,&T ,S)***>

<S\_DASH ***(T­L ,&T ,S)***>

<T\_DASH ***(T­L ,&T ,S)***>

<E\_DASH ***(T­L ,&T ,S)***>

* 1. <E\_DASH ***(T­L ,&T ,S)***> 🡪 || ***{O = ||.VP}***

<T ***(&TR )***> ***{TA = Compatibility(TL, TR , O)}***

<E\_DASH ***(T­A ,&T ,S)***> |

€ ***{T = TL}***

* 1. <T ***(&T, S)***> 🡪 <S ***(&TL , S)***> <T\_DASH ***(T­L ,&T ,S)***>
  2. <T\_DASH ***(T­L ,&T ,S)***> 🡪 && ***{O = &&.VP}***

<S ***(&TR )***> ***{TA = Compatibility(TL, TR , O)}***

<T\_DASH ***(T­A ,&T ,S)***> | € ***{T = TL}***

* 1. <S ***(&T, S)***> 🡪 <R ***(&TL,S)***> <S\_DASH ***(T­L ,&T ,S)***>
  2. <S\_DASH ***(T­L ,&T ,S)***> 🡪 ROP ***{O = ROP.VP}***

<R ***(&TR )***> ***{TA = Compatibility(TL, TR , O)}*** <S\_DASH ***(T­A ,&T ,S)***> | € ***{T = TL}***

* 1. <R ***(&T, S)***> 🡪 <Q ***(&TL , S)***> <R\_DASH ***(T­L ,&T ,S)***>
  2. <R\_DASH ***(T­L ,&T ,S)***> 🡪 PM ***{O = PM.VP}***

<Q ***(&TR )***> ***{TA = Compatibility(TL, TR , O)}***   
<R\_DASH ***(T­A ,&T ,S)***> | € ***{T = TL}***

* 1. <Q ***(&T)***> 🡪 <VAL ***(&TL )***> <Q\_DASH ***(T­L ,&T ,S)***>
  2. <Q\_DASH ***(T­L ,&T ,S)***>🡪 MDM ***{O = MDM.VP}***

<VAL ***(&TR )***> ***{TA = Compatibility(TL, TR , O)}***

<Q\_DASH ***(T­A ,&T ,S)***> | € ***{T = TL}***

* 1. <VAL ***(&T, S)***> 🡪 <F ***(&T, S)***> | <E ***(&T, S)***>
  2. <F ***(&T, S)***> 🡪 ID ***{N = ID.VP}***

<NEW\_ASGN ***(N, &T, S)***>

* 1. <E ***(&T, S)***> 🡪 (<EXP ***(&T, S)***>) |

!<VAL ***(&T, S)***> |

-- <F ***(&T, S)***> |

++ <F ***(&T, S)***> |

<CONST ***(&T)***> |

this . <F ***(&T, S)***>

1. <START> 🡪<DEFS>
2. <DEFS> 🡪 <MST> <DEFS> | <CLASS> <DEFS> | <FUNCTION-DEC><DEFS>
3. <MST> 🡪 € |<SST><MST>
4. <SST ***(S, &RT)>*** 🡪 <SST1 ***(S, &RT)*** >

* 1. <SST1 ***(S, &RT)*** >🡪 <DEC***(S)***> | <WHILE ***(S)*** > | <FOR ***(S)*** > | <DO-WHILE ***(S)*** > | <CONST-DT ***(S)*** > | <INC-DEC-PRE ***(S)*** > | <IF-ELSE ***(S)*** > |<SWITCH-CASE***(S)*** > | <RETURN***(S, &RT)***>

|<GTSWID ***(S)*** >

1. <FUNCTION-DEC>🡪 function <FUNC-DEF-1>
   1. <FUNC-DEF-1>🡪ID ( <DEC-PARAMS> ) { <MST> }
2. <CLASS> 🡪 class ID <CLASS-STRUCT>
   1. <CLASS-STRUCT> 🡪 <CLASS-BODY> | : ID <CLASSBODY>
   2. <CLASSBODY>🡪 { <CLASS-MST> }
   3. <CLASS-MST> 🡪 € |

<CLASS\_CONSTRUCOR> <CLASS-ST***(&ref)***> <CLASS- MST***(&ref)***>

* + - 1. <CLASS\_CONSTRUCOR> 🡪 € | constructor (<DEC\_PARAMS>) {<CONSTRUCTOR\_BODY>}
      2. <CONSTRUCTOR\_BODY> 🡪 € | <C\_B\_I> <CONSTRUCTOR\_BODY>
      3. <C\_B\_I> 🡪 <THIS\_VAR>|<SST1>
  1. <CLASS-ST> 🡪 <CLASS-FUNC> | <DEC> | <PROTECTED>
  2. <CLASS-FUNC> 🡪 ID <FUNC-DEF>
  3. <FUNC-DEF> 🡪 AOR ( <DEC-PARAMS> ) => { <MST> } |

(<DEC-PARAMS>){<MST>}

* 1. <PROTECTED>🡪 protected: <PRO-NEXT>
  2. <PRO-NEXT> 🡪 <CLASS-ST> | { <PRO-BODY>}
  3. <PRO-BODY> 🡪 <CLASS-MST>

1. <DEC\_PARAMS > 🡪 DT ID <DEC1><NEXT\_PARAM> |

ID <CONST\_DEC\_PARAM> <NEXT\_PARAM> |

€

* 1. <CONST\_DEC\_PARAM> 🡪 € | AOR <CONST\_DT1>
  2. <NEXT\_PARAM> 🡪 , <NEXT\_DEC\_PARAM> | €
  3. <NEXT\_DEC\_PARAM> 🡪 DT ID <DEC1><NEXT\_PARAM> |

ID <CONST\_DEC\_PARAM> <NEXT\_PARAM> |

1. <Dec> 🡪 DT ID <DEC1> <NEXTDEC>
   1. <DEC1> 🡪 AOR <DEC2> | €
   2. <DEC2> 🡪 ID <DEC3> | <E><EXP1>
   3. <DEC3> 🡪 <MERGED\_INIT> <EXP1> | <EXP1> | €
   4. <MERGED\_INIT> 🡪 <MOV> | [<EXP>] <M\_N\_ARR>
   5. <M\_N\_ARR> 🡪 <MOV> | [<EXP>] <M\_N\_ARR2> | €
   6. <M\_N\_ARR2> 🡪 <MOV> | €
   7. <MOV> 🡪 .ID<MERGED> | (<CALLING\_PARAMS>) <MERGED>
   8. <MERGED> 🡪 <EXP1> | <MERGED\_INIT> <DEC1>
2. <EXP> 🡪 <VAL> <EXP1>
   1. <EXP1> 🡪 <Q\_DASH> <R\_DASH> <S\_DASH><T\_DASH> <E\_DASH>
   2. <E\_DASH> 🡪 ||<T><E\_DASH> | €
   3. <T> 🡪 <S> <T\_DASH>
   4. <T\_DASH> 🡪 && <S><T\_DASH> | €
   5. <S> 🡪 <R> <S\_DASH>
   6. <S\_DASH> 🡪 ROP <R> <S\_DASH> | €
   7. <R> 🡪 <Q> <R\_DASH>
   8. <R\_DASH> 🡪 PM <Q> <R\_DASH> | €
   9. <Q> 🡪 <VAL> <Q\_DASH>
   10. <Q\_DASH>🡪 MDM <VAL> <Q\_DASH> | €
   11. <VAL> 🡪 <F> | <E>
   12. <F> 🡪 ID <NEW\_ASGN>
   13. <E> 🡪 (<EXP >) | !<VAL > | -- <F > | ++ <F > | <CONST > |this . <F>