|  |  |  |
| --- | --- | --- |
| 1- <start> 🡪 <defs> <VI> main (<NV>) {<MST>}$ | $  <VI> 🡪 void  <NV> 🡪~ | void  <defs>🡪<class><defs> | ID <defs1><defs> |DT ID <defs2><defs> |static <type> (<fn\_dec> | ~  <type> 🡪ID ID | DT ID  <  > 🡪<Ass\_st> | ID <X>  <X> 🡪<obj\_dec> | (<terminal>  <terminal> 🡪<fn\_dec> | <constructor\_dec>  <defs2> 🡪 <DT\_dec> | (<fn\_dec> | \*first<start>={class, ID, DT, Static, void }  \*first<VI>={ void,int }  \*first<NV>= { ~,void }  \*first<defs>={ class,ID,DT,static,~ }  \*first<type>={ID,DT}  \*first<defs1>={ AOP,ID }  \*first<X>={ [,=,;,( , , }  \*first<terminal>= { [,=,,,;,( }  \*first<defs2>= {[, ~, AOP, ID, (} | <start> ->{$}  <vi> ->{main}  <nv> ->{ ) }  <mst> ->{ } }  <class> ->{class,ID,DT,static,void,int}  <defs> --{void,int}  <def1> ->{class,ID,DT,static,void,int}  <def2> -{class,ID,DT,static,void,int}  <type> -- { ( }  <fun\_dec> -{void,int}  <ass\_st>-{class,ID,DT,static,void,int}  <x> -{class,ID,DT,static,void,int}  <obj\_dec>,<terminal>,<fn\_dec>,  <constructor\_dec>,<dt\_dec>-> {class,ID,DT,static,void,int} |
| 2- <MST>🡪~ | <SST><MST> | \*first<MST>={~,if,while,switch,for,return,continue,  break, ID, DT } | <sst> -- {if,while,switch,for,return,continue,  Break,ID,DT,} } |
| 3- <SST>🡪<if\_else> | <while\_st> | <switch> | <for\_st> |<return> | continue;| break; | ID <SST1> | DT ID <SST2>  <SST1> 🡪<array> <L2>| ID <Xxx> //L1 IS REMOVED  <L2>🡪<fn\_call> | inc\_dec;|<Ass\_st>  <Xxx> 🡪<obj\_dec> | ( <constructor\_dec>  <SST2> 🡪 <DT\_dec> | \*first<SST>={if,while,switch,for,return,continue,break, ID, DT}  \*first<SST1>= {[, .,inc\_dec,AOP,ID}  //\*first<L1>= {.,~}  \*first<L2>={., (,inc\_dec , AOP}  \*first<Xxx>= { [, ID, **,** , ; , ( }  \*first<SST2 > = { [, AOP, **,** ; } | <if\_else>,<while>,<switch>,<for>,<sst1>  ,<sst2> -- {if,while,switch,for,return,continue,  Break,ID,DT,} }  <xxx> -- {if,while,switch,for,return,continue,  Break,ID,DT,},class,static,void,int }  <array> --{.,(,inc\_dec,AOP}  <L2> --{if,while,switch,for,return,continue,  Break,ID,DT,} }  <fn\_call> --{if,while,switch,for,return,continue,  Break,ID,DT,} }  <ass\_st> --{if,while,switch,for,return,continue,break,id,dt,class,static,void int}  <obj\_dec> --{if,while,switch,for,return,continue,  Break,ID,DT,},class,static,void,int }  <constructor\_dec> --{if,while,switch,for,return,continue,  Break,ID,DT,},class,static,void,int }  <dt\_dec> --{if,while,switch,for,return,continue,  Break,ID,DT,},class,static,void,int } |
| 4- <return> 🡪 return <OE1> ;  <OE1> 🡪 <OE> | ~ | \*first<return>= {return}  \*first<OE1>= {const, (, !, ID,~} | <OE1> -- { ; } |
| 5- <fn\_call> 🡪 <fn\_call 1>;  <fn\_call1>🡪<checkID> (<arg>)  <checkID>🡪. ID<array> <checkID> | ~  <arg> 🡪 <OE><arg1>| ~  <arg1> 🡪 ,<OE><arg1>| ~ | \*first<fn\_call>={first<fn\_call1>}  \* first<fn\_call1>={ ., ( }  \* first<checkID>={.,~}  \*first<arg>= {const, (, !, ID,~}  \*first<arg1>={, , ~} | <fn\_call> -- --{if,while,switch,for,return,continue,  Break,ID,DT,} ,;}  <checkID> -- { ( }  <arg> -- { ) }  <array> -- {.,(,inc\_dec,AOP}  <oe> -- { **,** , ) }  <arg1> -- { ) } |
| 9- <inc\_dec> 🡪 <inc\_dec1>;  <inc\_dec1> 🡪 inc\_dec | \*first<inc\_dec1>={ inc\_dec } | <inc\_dec> -- { ; } |
| 6- <DT\_dec> 🡪 <new\_arr> | <init>  <new\_arr> 🡪 [<OE>]<array2>  <array2>🡪= {<array3>} ; | ;  <array3> 🡪 <OE><array4>  <array4> 🡪 , <OE> <array4> | ~  <init> -->AOP <init2> | <list>  <init2> --><OE> ;| <init4>  <init3> -->AOP <init4>| ~  <init4> --> ID <init3> <list> | <const> < list>  <list> -->,ID <init3> <list> | ; | \*first<DT\_dec>= {[, AOP, **,** ;}  \*first<new\_arr>= {[}  \*first<array2>= {=,;}  \*first<array3>= {const, (, !, ID}  \*first<array4>= {**,**,~}  \*first<init>= {AOP, **,** , ;}  \*first<init2>= {const, (, !, ID}  \*first<init3>= {AOP, ~}  \*first<init4>= {ID, first<const>}  \*first<list>= {**,** , ;} | <array4> --{ } }  <init3> -- { **,** , ;} |
| 7- <fn\_dec> --> <para>) <inherit> {<MST>}  <word1> --> virtual | static | const  <para> --> <def> <E> | void | ~  <def> --> ID ID|DT ID  <E> --> ,<def><E> | ~  <inherit> -->: ID <fn\_call1> | \*first<fn\_dec>= {ID, DT, void, )}  \*first<word1>={ virtual,static,const}  \*first<para>={ID,DT,void,~}  \*first<def>={ID,DT}  \*first<E>={, , ~}  \*first<inherit> ={:} | <E> -- { ) } |
| 8- <Ass\_st> --> <Ass\_st1><Xx> ;  <Ass\_st1> --> AOP <OE>  <Xx> --> <Ass\_st1><Xx> | ~  <array> --> [<OE>]| ~ | \*first<Ass\_st1>={AOP}  \*first<Xx >={AOP,~}  \*first<array>{[,~} | <Xx> -- { ; } |
| 10- <OE> --> <AE> <OE'>  <OE'> --> ||<AE><OE'> | ~  <AE> --> <RE> <AE'>  <AE'> --> &&<RE><AE'> | ~  <RE> --> <PE><RE'>  <RE'> -->ROP <PE><RE'>|~  <PE> --> <ME><PE'>  <PE'> --> PM <ME><PE'>|~  <ME> --> <F><ME'>  <ME'> --> MDM <F><ME'>|~  <F> --> <CONST> |(<OE>) |!<PE> |ID <XOE1>  <XOE1> 🡪<fn\_call> |<array><nt2> inc\_dec|~  <nt2> 🡪 .ID<array><nt2>|~ | \*first<OE>=first<AE>=first<RE>=first<PE>=  first<ME>=first<F>  \*first<F>= {const, (, !, ID}  \*first<OE’>= {||, ~}  \*first<AE’>= {&&, ~}  \*first<RE’>= {ROP, ~}  \*first<PE’>= {PM, ~}  \*first<ME’>= {MDM, ~}  \*first<XOE1> = {[,.,(,~}  \*first<nt2>={.,~} | <OE`>=<AE`>=<RE`>=<PE`>=<ME`>= { , , ) }  <XOE1>--{const,(,!,ID}  <nt2> -- {inc\_dec}  he |
| 11- <obj\_dec> --><array><new\_init> <list2>  <new\_init> --> = ID < new\_init > | ~  <list2> --> ,ID <array><init><list2> | ; | \*first<obj\_dec>={ [, =, **,**, ;}  \*first< new\_init >{=, ID ,~}  \*first<list2>={ **,** , ;} | <init>--{, , ; } |
| 12-<constructor\_dec> --> <arg>); | \*first<constructor\_dec>= {const, (, !, ID,)} |  |
| 12- <if\_else> --> if(<OE>){<MST>} <o\_else>  <o\_else> -->else {<MST>}|~ | \*first<if\_else>={if}  \*first<o\_else>={else,~} | <o\_else>--{if,while,switch,for,return,continue,  Break,ID,DT,} } |
| 13- <while\_st> --> while (<OE>){<MST>} | \*first<while\_st>={while} |  |
| 14- <switch> --> switch(<OE>){<case><default>}  <case> --> case <OE>: {<MST>} <case>| ~  <default> --> default:{<MST>} | ~ | \*first<switch>= {switch}  \*first<case>= {case, ~}  \*first<default>= {default, ~} | <case> -- {default, } }  <default> -- { } } |
| 15- <class> --> class ID <chk\_inhrt> {<class\_body>};  <chk\_inhrt> -->: AM ID | ~  <class\_body> -->ID <X1> <class\_body> | DT ID <X2><class\_body> | AM: <class\_body> | <word><type> (<fn\_dec> | ~  <word> --> virtual | const | static  <X1> --> <constructor\_fn> | ID <X3>  <X2> --> ( <fn\_dec> | <DT\_dec>  <X3> --> ( <fn\_dec> | <obj\_dec> | \*first<class>={class}  \*first<chk\_inhrt>= {:, ~}  \*first<class\_body>= {ID, DT,AM,virtual, const, static,~}  \*first<word>= {virtual, const, static}  \*first<X1>= {(, ID}  \*first<X2>= {[ ,(, AOP, **,** ;}  \*first<X3>={(,[, ID, , , ;} | <chk\_inhrt> --{ { }  <class body> -- { } } |
| 16- <for\_st> -->for(<C1><C2>;<C3>) { <MST>}  <C1> --> DT ID <DT\_dec>|ID <Ass\_st>|;  <C2> --> <OE> | ~  <C3> --> ID <X11> | inc\_dec | ~  <X11> --> <Ass\_st1> | inc\_dec | \*first<for\_st>={for}  \*first<C1>= {DT, ID, ;}  \*first<C2>= {const, (, !, ID,~}  \*first<C3>= {ID, inc\_dec, ~}  \*first<X11>= {AOP, inc\_dec} | <c2> { ; }  <c3> { ) } |
| 17)<const> --> int\_const | Float\_const | string\_const | bool\_const |char\_const | \*first<const>= {int, float, string, bool, char} |  |
| 18) <constructor\_fn> --> (<para>) {<MST>}; | \*first< constructor\_fn>={(} |  |