**LL1 Proof**

**s**

**Start(done):**

**Start:**

<start> → <start\_body> main(){<main\_body> }

<start\_body> → <dec><start\_body>|<func\_st><start\_body> | <struct\_dec><start\_body>| <class\_dec><start\_body>|<struct\_obj><start\_body> | <obj\_decl><start\_body> | ε

**Main:**

<main\_body> → <dec><main\_body>|<if\_else><main\_body>| <break><main\_body> | <continue><main\_body> | <OE>;<main\_body>| <try\_catch><main\_body>| <for\_st><main\_body>| <print><main\_body>| <struct\_obj><main\_body> | <obj\_decl><main\_body> |<assign\_st><main\_body> | <struct\_dec><main\_body> |<return><main\_body>|<class\_dec><main\_body>|ε

| <NT> | first | Follow |
| --- | --- | --- |
| <start> | int  const  float  string  bool  char  def  virtual  struct  class  st  obj  ε | $ |
| <start\_body> | int  const  float  string  bool  char  def  virtual  struct  class  st  obj  ε | main |
| <main\_body> | int  const  float  string  bool  char  if  break  continue  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  check  for  print  st  return  obj  assign  struct  class  ε | } |

**Declaration:(LL1 proved)(done)**

<dec>→<DT1> ID <init> ;

<init>→ <SA><init5> | [<init1> | ε

<init1>→ <OE>]<Arr><init6> | ]<Arr><init6>

<init6> → <SA><init7> | ε

<init5>→<OE> | <take>

<init7> → {<OE><init2>}<init2>

<init2>→ <or><OE><init2>| {<OE><init2>}<init2> | ε

<SA>→ =

<or> → ,

<const>→int\_const|float\_const|char\_const|string\_const

<DT>→ int | float | char | string | bool

<DT1>→ const <DT> | <DT>

| <NT> | first | follow |
| --- | --- | --- |
| <dec> | const, int, float, char, string, bool | int  const  float  string  bool  char  def  virtual  ,  )  throw  public  private  protected  if  break  continue  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  return  check  for  print  st  obj  assign  struct  class  }  ;  construct  ~  main  elif  else |
| <init> | =  [  ,  ε | ; |
| <init1> | this  { int\_const, flt\_const, char\_const, bool\_const, string\_const }  Inc\_Dec  (  !  ID  ] | ; |
| <init2> | ,  {  ε | }  ; |
| <init5> | this  { int\_const, flt\_const, char\_const, bool\_const, string\_const }  Inc\_Dec  (  !  ID  take | ; |
| <init6> | =  ε | ; |
| <init7> | { | ; |

**Assignment\_statement:(done)**

<assign\_st>→ assign <D2><X2><list>;

<D2> -> Inc\_Dec <inc\_dec\_op><p> | <p1> | ε

<list> → =<SSA>

<SSA> → <OE> | <take>

<dot22> → .ID<dot22> | ε | (para).ID<dot22> | [<OE>]<dot22>

<X2> → ID<dot22> | (ID<dot22>)

| <NT> | first | follow |
| --- | --- | --- |
| <assign\_st> | assign | int  const  float  string  bool  char  def  virtual  ,  )  throw  public  private  protected  if  break  continue  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  return  check  for  print  st  obj  assign  struct  class  }  ;  construct  ~  main  elif  else |
| <D2> | Inc\_Dec  this  ε | ID  ( |
| <X2> | ID  ( | = |
| <list> | = | ; |
| <SSA> | this  { int\_const, flt\_const, char\_const, bool\_const, string\_const }  Inc\_Dec  (  !  ID  take | ; |
| <dot22> | .  (  [  ε | )  = |

**Print():(done)**

**Print():**

<print> →print<index>;

<index>→(<OE>)

| **Non-terminal** | **first** | **follow** |
| --- | --- | --- |
| <print> | print |  |
| <index> | ( | ; |

**Throw():(done)**

<throw> →throw<index>;

<index>→(<OE>)

| Non\_terminal | first | follow |
| --- | --- | --- |
| <throw> | throw |  |
| <index> | ( | ; |

**If\_else:(done)**

<if\_else>→ if<if2>

<if2>→(<cond>)<if\_body><elif>

<elif>→elif <if2> | else:<if\_body> | ε

< if\_body>→ {< if\_SST>} |;

<cond>→ <OE>

<if\_SST> -> <dec><if\_SST> | <if\_else><if\_SST> | <try\_catch><if\_SST> | <OE>;<if\_SST> | <struct\_obj><if\_SST> | <obj\_decl><if\_SST>| <for\_st><if\_SST> | <print><if\_SST>| <return><if\_SST> | <break><if\_SST> | <throw><if\_SST> |<continue><if\_SST> |<assign\_st><if\_SST> | ε

| **<NT>** | **first** | **follow** |
| --- | --- | --- |
| **<if\_else>** | **If** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **throw**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main**  **elif**  **else** |
| **<if2>** | **(** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **throw**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main**  **elif**  **else** |
| **<cond>** | **this**  **{int\_const, flt\_const, string\_const, char\_const}**  **Inc\_Dec**  **(**  **!**  **ID** | **)** |
| **<elif>** | **elif**  **else**  **ε** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **throw**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main**  **elif**  **else** |
| **<if\_body>** | **{**  **;** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **throw**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main**  **elif**  **else** |
| **<if\_SST>** | **const, int, float, bool, char, string**  **if**  **break**  **continue**  **this**  **{int\_const, flt\_const, string\_const, char\_const}**  **Inc\_Dec**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **throw**  **ε** | **}** |

**Take:(done)**

<take> → take <DT> ID <AO> take <index>

<index>→ (<index1>);

<index1>→<OE>

| non\_terminal | first | follow |
| --- | --- | --- |
| <take> | take | ; |
| <indexA> | ( | ; |

**Break:done**

<break> → break

| non\_terminal | first | follow |
| --- | --- | --- |
| <break> | break |  |

**Continue:done**

<continue> → continue

| **Non\_terminal** | **first** | **follow** |
| --- | --- | --- |
| **<continue>** | **continue** |  |

**For:**

<for\_st>→for(<init11>;<cond11>;<inc\_dec11>)<for\_body>

<init11>→ <dec> | <assign\_st>| ε

<cond11>→ <OE> |ε

<LO>→ <&> | <|>

<inc\_dec11>→ <Inc\_Dec\_obj\_call>| ε

<inc\_dec\_op>→ ++ | --

< for\_body>→ {< funct\_SST>} | ;

<for\_SST> → <dec> <for\_SST> | <if\_else> <for\_SST>| <break> <for\_SST> | <continue><for\_SST> | <return><for\_SST>| <OE>; <for\_SST> | <try\_catch><for\_SST>| <for\_st><for\_SST>| <print><for\_SST>| <struct\_obj><for\_SST> | <obj\_decl><for\_SST> | <assign\_st><for\_SST> |ε

| **<NT>** | **first** | **follow** |
| --- | --- | --- |
| **<for\_st>** | **for** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **throw**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main**  **elif**  **else** |
| **<init11>** | **const, int, float, char, string, bool**  **assign**  **ε** | **;** |
| **<cond11>** | **this**  **{int\_const, flt\_const, string\_const, char\_const}**  **Inc\_Dec**  **(**  **!**  **ID**  **ε** | **;** |
| **<inc\_dec11>** | **Inc\_Dec**  **ε** | **)** |
| **<for\_body>** | **{**  **;** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **throw**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main**  **elif**  **else** |
| **<for\_SST>** | **const, int, float, char, string, bool**  **if**  **break**  **continue**  **return**  **this**  **{int\_const, flt\_const, string\_const, char\_const}**  **Inc\_Dec**  **(**  **!**  **ID**  **check**  **for**  **print**  **st**  **obj**  **assign**  **ε** | **}** |

**Function:**

<func\_st> → <key> def <DT2> ID <func\_I>

<func\_I> →(<param>) < funct\_body>

<param> →<dec><param1> | <struct\_obj><param1> | <obj\_decl><param1> | ε

<param1> →,<param2> <param1>| ε

<param2> → <dec> | <struct\_obj> | <obj\_decl>

<key> →virtual | ε

< funct\_body>→ {< funct\_SST>} |;

<funct\_SST>→<dec><funct\_SST>|<if\_else><funct\_SST>| <break><funct\_SST> |<continue><funct\_SST> | <OE>;<funct\_SST>| <return><funct\_SST>|<try\_catch><funct\_SST>| <for\_st><funct\_SST>|<print><funct\_SST>|<struct\_obj> <funct\_SST> |<obj\_decl><funct\_SST>|<assign\_st><funct\_SST>| <struct\_dec><funct\_SST>|<class\_dec><funct\_SST>|ε

<DT2> → void | int | float | char | string | bool

| <NT> | first | follow |
| --- | --- | --- |
| <func\_st> | virtual  def | int  const  float  string  bool  char  def  virtual  public  private  protected  if  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  check  for  print  st  obj  assign  struct  class  }  construct  ~  main |
| <func\_I> | ( | int  const  float  string  bool  char  def  virtual  public  private  protected  if  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  check  for  print  st  obj  assign  struct  class  }  construct  ~  main |
| <DT2> | void, int, float, bool, char, string | ID |
| <param> | const, int, float, bool, char, string  st  obj  ε | ) |
| <param1> | ,  ε | ) |
| <param2> | const, int, float, bool, char, string  st  obj | ,  ) |
| <funct\_body> | {  ; | int  const  float  string  bool  char  def  virtual  public  private  protected  if  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  check  for  print  st  obj  assign  struct  class  }  construct  ~  main |
| <funct\_SST> | const, int, float, bool, char, string  if  break  continue  this  {int\_const, flt\_const, string\_const, char\_const}  Inc\_Dec  (  !  ID  return  check  for  print  st  obj  assign  struct  class  ε | } |

**Function\_Call:(done)**

<func\_call1> → <p1> ID (<para>);

<para>→ <OE><para1>| ε

<para1>→ ,<OE><para1> | ε

<p1> → this.

| non\_terminal | first | follow |
| --- | --- | --- |
| <func\_call1>  satisfied(c1) | this |  |
| <para>  satisfied(c1) | first(OE)  ε | ) |
| <para1>  satisfied(c1) | ,  ε | ) |

**Return:(done)**

<return> → return<index1>;

<index1> → <OE> | ε

| non\_terminal | first | follow |
| --- | --- | --- |
| <return> | return | int  const  float  string  bool  char  def  virtual  throw  public  private  protected  if  break  continue  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  return  check  for  print  st  obj  assign  struct  class  }  construct  ~  main  elif  else |
| <index> | this  {int\_const, flt\_const, string\_const, char\_const}  Inc\_Dec  (  !  ID  ε | ; |

**Increment and decrement:(done)**

<Inc\_dec>→ Inc\_Dec <p> ID <Arr><inc\_dec\_op>; | <Inc\_dec\_op> ID<Arr> ;

<inc\_dec\_op>→ ++ | --

<Arr>→ [<OE> ]<Arr> | ε

| Non\_terminal | first | follow |
| --- | --- | --- |
| <inc\_dec> | Inc\_dec  ++  – |  |
| <inc\_dec\_op> | ++  – | ;  id |
| <arr> | [  ε | ++  – |

***CLASS\_DEC:*(done)**

<class\_dec> → class ID <class\_dec1>;

<class\_dec1>→<inherit> <class\_body>

<inherit> →:<access\_Mod> ID <Multiple\_inherit> | ε

<Multiple\_inherit> → ,<access\_Mod>ID<Multiple\_inherit>| ε

< class\_body> → {< class\_SST>} | ε

<access\_Mod> → public | private | protected | ε

<class\_SST>→<access\_Mod>:<class\_SST>|<dec><class\_SST>| <func\_st><class\_SST>|<constructor><class\_SST>| <destructor> <class\_SST>|<if\_else><class\_SST>|<try\_catch><class\_SST>| <for\_st><class\_SST>|<print><class\_SST>| <class\_dec> <class\_SST> | <struct\_decl> <class\_SST> |<OE>;<class\_SST> |<struct\_obj> <class\_SST> | <obj\_decl><class\_SST> |<assign\_st> <class\_SST> | ε

| <NT> | first | Follow |
| --- | --- | --- |
| <class\_dec> | class | int  const  float  string  bool  char  def  virtual  public  private  protected  if  break  continue  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  return  check  for  print  st  obj  assign  struct  class  }  construct  ~  main |
| <class\_dec1> | :  ε | ; |
| <inherit> | :  ε | {  ; |
| <class\_body> | {  ε | ; |
| <access\_Mod> | public  private  protected  ε | ID  : |
| <Multiple\_inherit> | ,  ε | {  ; |
| <class\_SST> | public  private  protected  const, int, float, char, string, bool  def  virtual  construct  ~  if  check  for  print  class  struct  this  {int\_const, flt\_const, string\_const, char\_const}  Inc\_Dec  (  !  ID  st  obj  assign  ε | } |

**Object declaration:(done)**

<obj\_decl>→ obj ID ID <N1> ;

<N1> → [OE]<Arr><N2> | (parameter) <N5> | <SA> <OE> | ε

<N2> → <SA><N3> | ε

<N3> → {<OE><N4>}<N4>

<N4> → <or><OE><N4>| {<OE><N4>}<N4>| ε

<N5> → <SA><OE> | ε

<parameter> → <OE><parameter1>

<parameter1> → ,<OE><parameter1> | ε

| **<NT>** | **first** | **Follow** |
| --- | --- | --- |
| **<obj\_decl>** | **obj** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **throw**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main**  **elif**  **else** |
| **<N1>** | **[**  **(**  **=**  **ε** | **;** |
| **<N2>** | **=**  **ε** | **;** |
| **<N3>** | **{** | **;** |
| **<N4>** | **,**  **{**  **ε** | **}**  **;** |
| **<N5>** | **=**  **ε** | **;** |
| **<parameter>** | **this**  **{int\_const, flt\_const, string\_const, char\_const}**  **Inc\_Dec**  **(**  **!**  **ID** | **)** |
| **<parameter1>** | **,**  **ε** | **)** |

**Object Calling:ll1 proved**

**(ID wise):**

<obj\_call> → <B’>

<B’> → ID <dot> | <p1> ID <dot>

<para> →<OE><para1>| ε

<para1> →,<OE> <para1> | ε

<dot> → .ID<dot> | ε | [<OE>]<dot> | (<para>)<dot>

| <NT> | first | Follow |
| --- | --- | --- |
| <obj\_call> | ID  this | )  ;  ]  ,  {  }  MDM  PM  >=, <=, >, <, ==, !=  &  | |
| <B’> | ID  this | |  &  PM  MDM  >=, <=, <, >, ==, !=  )  ;  ]  ,  {  } |
| <dot> | .  [  (  ε | )  >=, <=, <, >, ==, !=  &  |  PM  MDM  ++  –  ;  ]  ,  {  } |
| <para> | this  {int\_const, flt\_const, string\_const, char\_const}  Inc\_Dec  (  !  ID  ε | ) |
| <para1> | ,  ε | ) |

**Exceptional Handeling:**

<try\_catch >→check{<try\_SST>}<catch><catch1>

<try\_SST>→<dec>< try\_SST>| <If\_else>< try\_SST>|<try\_catch>< try\_SST>| <for\_loop>< try\_SST> | <print>< try\_SST> | <throw>< try\_SST> | <OE>; <try\_SST>| <assign\_st>< try\_SST> |<return>< try\_SST>|<struct\_obj>< try\_SST> | <obj\_decl> < try\_SST>| ε

<catch>→ catch(<catch\_para>){<catch\_SST>}

<catch\_para>→ <obj\_decl> | …

<catch\_SST>→<dec><catch\_SST>|<If\_else><catch\_SST>|<OE>;<catch\_SST>|<assign\_st><catch\_SST> |<return><catch\_SST> | <for\_st><catch\_SST>|<print><catch\_SST>| <throw><catch\_SST> |<struct\_obj><catch\_SST>| <obj\_decl> <catch\_SST> | ε

<catch1>→ <catch><catch1> | ε

| **<NT>** | **first** | **Follow** |
| --- | --- | --- |
| **<try\_catch>** | **check** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **throw**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main**  **elif**  **else** |
| **<try\_SST>** | **const, int, float, char, string, bool**  **if**  **check**  **for**  **print**  **throw**  **this**  **{int\_const, flt\_const, string\_const, char\_const}**  **Inc\_Dec**  **(**  **!**  **ID**  **assign**  **return**  **st**  **obj**  **ε** | **}** |
| **<catch>** | **catch** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **throw**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main**  **elif**  **else**  **catch** |
| **<catch\_para>** | **obj**  **…** | **)** |
| **<catch\_SST>** | **const, int, float, char, string, bool**  **if**  **for**  **print**  **throw**  **this**  **{int\_const, flt\_const, string\_const, char\_const}**  **Inc\_Dec**  **(**  **!**  **ID**  **assign**  **return**  **st**  **obj**  **ε** | **}** |

| **catch1** | **catch**  **ε** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **throw**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main**  **elif**  **else** |
| --- | --- | --- |

**INC\_DEC\_OBJECT :-ll1 proofed**

<Inc\_Dec\_obj\_call> → Inc\_Dec <B1>

<B1> → ID <dot><inc\_dec\_op> | <inc\_dec\_op> <X11>|<B2> <inc\_dec\_op>

<B2>→ (ID <dot>) | <p1>ID <dot>

<X11> -> (ID <dot>)| ID<dot>

<para> →<OE><para1> | ε

<para1> →,<OE> <para1> | ε

<dot> → .ID<dot> | ε | [<OE>]<dot> | (<para>)<dot>

| <NT> | first | Follow |
| --- | --- | --- |
| <Inc\_Dec\_obj\_call> | Inc\_Dec | )  ;  ]  ,  {  }  PM  MDM  >=, <=, <, >, ==, !=  &  | |
| <B1> | ID  ++  --  (  this | )  ;  ]  ,  {  }  MDM  PM  &  |  >=, <=, <, >, ==, != |
| <B2> | (  this | ++  -- |
| <dot> | .  [  (  ε | )  ++  PM  MDM  >=, <=, <, >, ==, !=  &  |  –  ;  ]  ,  {  } |
| <X11> | (  ID | )  ;  ]  ,  {  }  PM  MDM  &  |  >=, <=, <, >, ==, != |
| <para> | this  {int\_const, flt\_const, string\_const, char\_const}  Inc\_Dec  (  !  ID  ε | ) |
| <para1> | ,  ε | ) |

**Constructor:(done)**

<constructor> → construct ID<indexC>;

<indexC> → (<param>)<constructor\_Body>

<param>→<dec><param1> | <struct\_obj><param1> | <obj\_decl> <param1> | ε

<param1> →,<param2> <param1>| ε

<param2> → <dec> | <struct\_obj> | <obj\_decl>

< constructor\_Body>→{< constructor\_SST>}| ε

<constructor\_SST>→<dec><constructor\_SST>|<if\_else> <constructor\_SST>|<try\_catch><constructor\_SST>|<for\_st> <constructor\_SST> | <print><constructor\_SST>|<struct\_dec> <constructor\_SST>|<return><constructor\_SST>| <OE>; <constructor\_SST>|<struct\_obj><constructor\_SST>|<obj\_decl> <constructor\_SST> |<assign\_st> <constructor\_SST> | ε

| <NT> | first | Follow |
| --- | --- | --- |
| <constructor> | construct | int  const  float  string  bool  char  def  virtual  public  private  protected  if  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  check  for  print  st  obj  assign  struct  class  }  construct  ~ |
| <indexC> | ( | ; |
| <param> | const, int, float, char, string, bool  st  obj  ε | ) |
| <param1> | ,  ε | ) |
| <param2> | const, int, float, char, string, bool  st  obj | ,  ) |
| <constructor\_Body> | {  ε | ; |
| <constructor\_SST> | const, int, float, char, string, bool  if  check  for  print  struct  return  this  {int\_const, flt\_const, string\_const, char\_const}  Inc\_Dec  (  !  ID  st  obj  assign  ε | } |

**Destructor: (done)**

<destructor> → ~ ID<indexD>

<indexD> → () {<destructor\_SST>}

<destructor\_SST>→<dec><destructor\_SST>|<if\_else><destructor\_SST> | <OE>;<destructor\_SST>| <try\_catch><destructor\_SST>| <for\_st><destructor\_SST>|<print><destructor\_SST>|<struct\_obj> <destructor\_SST> |<obj\_decl><destructor\_SST>| <assign\_st> <destructor\_SST>| <struct\_dec><destructor\_SST>|<class\_dec> <destructor\_SST> | ε

| <NT> | first | Follow |
| --- | --- | --- |
| <destructor> | ~ | int  const  float  string  bool  char  def  virtual  public  private  protected  if  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  check  for  print  st  obj  assign  struct  class  }  construct  ~ |
| <indexD> | ( | int  const  float  string  bool  char  def  virtual  public  private  protected  if  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  check  for  print  st  obj  assign  struct  class  }  construct  ~ |
| <destructorSST> | const, int, float, char, string, bool  if  this  {int\_const, flt\_const, string\_const, char\_const}  Inc\_Dec  (  !  ID  check  for  print  st  obj  assign  struct  class  ε | } |

**Expression:**

<OE> → <AE><OE’>

<OE’> → | <AE> <AE’> | ε

<AE> → <RE><AE’>

<AE’> → & <RE> <AE’> | ε

<RE> → <E> <RE’>

<RE’> → <RO> <E> <RE’> | ε

<E> → <T> <E’>

<E’> → PM <T> <E’> | ε

<T> → <F> <T’>

<T’> → MDM <F> <T’> | ε

<F>→<const> | <Inc\_Dec\_obj\_call>| (<OE>) |!<F> |<obj\_call>

| <NT> | first | follow |
| --- | --- | --- |
| <OE> | this  {int\_const, flt\_const, string\_const, char\_const}  (  !  Inc\_Dec  ID | )  ;  ]  ,  {  } |
| <AE> | this  {int\_const, flt\_const, string\_const, char\_const}  (  !  Inc\_Dec  ID | |  )  &  ;  ]  ,  {  } |
| <OE’> | |  ε | )  ;  ,  {  }  ] |
| <RE> | this  {int\_const, flt\_const, string\_const, char\_const}  (  !  Inc\_Dec  ID | &  |  )  ;  ,  ]  {  } |
| <AE’> | &  ε | )  |  &  {  }  ]  ;  , |
| <E> | this  {int\_const, flt\_const, string\_const, char\_const}  (  !  Inc\_Dec  ID | <=,>=,>,<,!=,==  &  |  )  {  }  ]  ;  , |
| <RE’> | <=,>=,>,<,!=,==  ε | &  |  )  {  }  ]  ;  , |
| <T> | this  {int\_const, flt\_const, string\_const, char\_const}  (  !  Inc\_Dec  ID | PM  >=,<=,<,>,!=,==  &  |  )  {  }  ;  ,  ] |
| <E’> | PM  ε | >=,<=,<,>,!=,==  &  |  )  {  }  ]  ;  , |
| <F> | this  {int\_const, flt\_const, string\_const, char\_const}  (  !  Inc\_Dec  ID | MDM  PM  >=,<=,<,>,!=,==  &  |  )  {  }  ]  ;  , |
| <T’> | MDM  ε | PM  >=,<=,<,>,!=,==  &  |  )  {  }  ]  ;  , |

**Struct:**

<struct\_dec> → struct ID < struct \_Body>

< struct\_Body>→{< struct\_SST>}| ;

<struct\_SST>→<dec><struct\_SST> | <assign\_st><struct\_SST> | <struct\_obj><struct\_SST> | <obj\_decl><struct\_SST>| <func\_st><struct\_SST>| <print><struct\_SST> | ε

| **<NT>** | **first** | **follow** |
| --- | --- | --- |
| **<struct\_dec>** | **struct** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main** |
| **<struct\_Body>** | **{**  **;** | **int**  **const**  **float**  **string**  **bool**  **char**  **def**  **virtual**  **public**  **private**  **protected**  **if**  **break**  **continue**  **this**  **Inc\_Dec**  **int\_const, flt\_const, string\_const, char\_const, bool\_const**  **(**  **!**  **ID**  **return**  **check**  **for**  **print**  **st**  **obj**  **assign**  **struct**  **class**  **}**  **construct**  **~**  **main** |
| **<struct\_SST>** | **const, int, float, char, string, bool**  **assign**  **st**  **obj**  **def**  **virtual**  **print**  **ε** | **}** |

**Struct Object:**

<struct\_obj>→st ID ID<listA><initA>;

<listA>→,ID<listA> | [<OE>]<listA> |ε

<initA>→ ={<OE><listB>} | ε

<listB>→,<OE><listB> | ε

| <NT> | first() | follow() |
| --- | --- | --- |
| <struct\_obj> | st | int  const  float  string  bool  char  def  virtual  throw  public  private  protected  if  break  continue  this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  return  check  for  print  st  obj  assign  struct  class  }  construct  ~  main  elif  else |
| <listA> | ,  [  ε | =  ; |
| <initA> | =  ε | ; |
| <listB> | ,  ε | } |

**Arr:**

<Arr> -> [<OE>]<Arr> | ε

| <NT> | first | Follow |
| --- | --- | --- |
| <Arr> | [  ε | =  ;  ++  –  )  , |

<dec11>→<DT1> ID <init22>

<init22>→ <SA><OE> | [<init33> |ε

<init33>→ <OE>]<Arr>| ]<Arr>

<SA>→ =

<const>→int\_const|float\_const|char\_const|string\_const

<DT>→ int | float | char | string | bool

<DT1>→ const <DT> | <DT>

| <NT> | first | Follow |
| --- | --- | --- |
| dec11 | const, int, float, char, string, bool | ,  ) |
| init33 | this  Inc\_Dec  int\_const, flt\_const, string\_const, char\_const, bool\_const  (  !  ID  ] | ,  ) |
| init22 | =  [  ε | ,  ) |
| DT1 | const, int, float, char, string, bool | ID |
| DT | int, float, char, string, bool | ID |