OpenQASM 3 (legible) grammar

```
      <program>  ::=  <header> <globalstmt|stmt>*

   <globalstmt>  ::=  <subroutinedef>
                   |  <externdef>
                   |  <gatedef>
                   |  <calibration>
                   |  <qdecl>
                   |  #pragma { <stmt>* }
<subroutinedef>  ::=  def <id>(<arg...>?)  { <stmt>* <returnstmt>?  }
                   |  def <id>(<arg...>?)  -> <ctype> { <stmt>* <returnstmt>?  }
    <externdef>  ::=  extern <id>(<ctype...>?);
                   |  extern <id>(<ctype...>?)  -> <ctype>;
      <gatedef>  ::=  gate <id> <id...> <qblock>
                   |  gate <id>(<id...>?)  <id...> <qblock>
  <calibration>  ::=  defcalgrammar "openpulse";
                   |  defcalgrammar <string>;
                   |  defcal <id> <id...> { .* }
                   |  defcal <id>(<carg...>?)  <id...> { .* }
                   |  defcal <id>(<exp...>?)  <id...> { .* }
                   |  defcal <id> <id...> -> <ctype> { .* }
                   |  defcal <id>(<carg...>?)  <id...> -> <ctype> { .* }
                   |  defcal <id>(<exp...>?)  <id...> -> <ctype> { .* }
        <qdecl>  ::=  qreg <id>; | qreg <id>[<exp>];
                   |  qubit <id>; | qubit[<exp>] <id>;

         <stmt>  ::=  <exp>;
                   |  <indexid> <assignop> <exp>;
                   |  measure <indexid> -> <indexid>;
                   |  <indexid> = measure <indexid>;
                   |  <classicdecl>
                   |  if (<exp>) <prgmblock>
                   |  if (<exp>) <prgmblock> else <prgmblock>
                   |  for <id> in <setdecl> <prgmblock>;
                   |  while (<exp>) <prgmblock>;
                   |  end;
                   |  let <id> = <exp>;
                   |  let <id> = <exp> ++ <exp> ++ ...  ++ <exp>;
                   |  <qstmt>

        <qstmt>  ::=  <gatemodifier>* <id> <indexid...>;
                   |  <gatemodifier>* <id>(<exp...>) <indexid...>;
                   |  <gatemodifier>* gphase(<exp>) <indexid...>?;
                   |  measure <indexid>;
                   |  reset <indexid>;
                   |  barrier <indexid...>?;
                   |  <timestmt>
                   |  <qloop>
```

```
<timestmt>  ::=  delay[<exp>] <indexid...>;
               | delay(<exp...>?)[<exp>] <indexid...>;
               | rotary[<exp>] <indexid...>;
               | rotary(<exp...>?)[<exp>] <indexid...>;
               | box <qblock>
               | box[<exp>] <qblock>

<prgmblock>  ::=  <stmt>
               | <control>
               | { <stmt|control>* }
  <control>  ::=  break; | continue; | <returnstmt>
<returnstmt>  ::=  return;
               | return <exp>;
               | return measure <indexid>;

    <qblock>  ::=  { <qstmt>* }
     <qloop>  ::=  for <id> in <setdecl> <qstmt>
               | for <id> in <setdecl> <qblock>
               | while (<exp>) <qstmt> | while (<exp>) <qblock>

      <arg>  ::=  <carg> | <qarg>
     <carg>  ::=  <singledesignatortype> <id>
               | <nodesignatortype> <id>
               | creg <id> | creg <id>[<exp>] | bit <id> | bit[<exp>] <id>
               | complex[<singledesignatortype>] <id>
               | const <arrayreftype> <id> | mutable <arrayreftype> <id>
<arrayreftype>  ::=  array[<nonarraytype>, <exp...>]
               | array[<nonarraytype>, #dim = <exp>]
     <qarg>  ::=  qreg <id> | qreg <id>[<exp>]
               | qubit <id> | qubit[<exp>] <id>

    <ctype>  ::=  <nonarraytype> | <arraytype>
<nonarraytype>  ::=  <singledesignatortype>
               | <nodesignatortype>
               | bit | bit[<exp>] | creg | creg[<exp>]
               | complex[<singledesignatortype>]
  <arraytype>  ::=  array[<nonarraytype>, <exp...>]

<singledesignatortype>  ::=  int | int[<exp>] | uint | uint[<exp>] | float | float[<exp>]
               | angle | angle[<exp>]
<nodesignatortype>  ::=  bool | duration | stretch

 <indexentity>  ::=  <exp> | <exp>?:<exp>? | <exp>?:<exp>?:<exp>
<indexoperator>  ::=  [{ <exp...> }]
               | [<indexentity...>]
    <indexid>  ::=  <id> <indexoperator>*
    <setdecl>  ::=  { <exp...> }
               | [<exp>?:<exp>?] | [<exp>?:<exp>?:<exp>]
               | <id>

  <equalsexp>  ::=  = <exp>
  <arrayinit>  ::=  {<exp|arrayinit>}
               | {<exp|arrayinit>, <exp|arrayinit>, ...}
```

```
<classicdecl>  ::=  <singledesignatortype> <id> <equalsexp>?;
                 |  <nodesignatortype> <id> <equalsexp>?;
                 |  creg <id> <equalsexp>?; | creg <id>[<exp>] <equalsexp>?;
                 |  bit <id> <equalsexp>?; | bit[<exp>] <id> <equalsexp>?;
                 |  complex[<singledesignatortype>] <id> <equalsexp>?;
                 |  <arraytype> <id>; | <arraytype> <id> = <arrayinit|exp>;
                 |  const <ctype> <id> <equalsexp>;

        <exp>  ::=  <constant> | <int> | <nondecimalint> | <real> | <imag>
                 |  <bool> | <bitstring> | <id> | <time>
                 |  <math>(<exp...>)
                 |  <ctype>(<exp...>)
                 |  sizeof(<exp...>)
                 |  <id>(<exp...>?)
                 |  durationof(<id>) | durationof(<qblock>)
                 |  (<exp>)
                 |  <exp> <indexoperator>
                 |  <exp> <b-op> <exp>
                 |  <u-op> <exp>

       <math>  ::=  arcsin | sin | arccos | cos | arctan | tan
                 |  exp | ln | sqrt | rotl | rotr | popcount
       <b-op>  ::=  || | && | | | ^ | & | == | != | > | < | >= | <=
                 |  << | >> | + | - | * | / | % | **
       <u-op>  ::=  ~ | ! | -

   <assignop>  ::=  = | += | -= | *= | /= | &= | |= | ~= | ^= | <<= | >>= | %= | **=

<gatemodifier>  ::=  inv @ | pow(<exp>) @
                 |  ctrl @ | ctrl(<exp>) @
                 |  negctrl @ | negctrl(<exp>) @

        <uni>  ::=  [\p{Lu}\p{Ll}\p{Lt}\p{Lm}\p{Lo}\p{Nl}]
         <id>  ::=  (_|$|<uni>|[A-Za-z])(_|$|<uni>|[A-Za-z]|[0-9])*
        <int>  ::=  ([0-9][_]?)*[0-9]
       <real>  ::=  <int>[eE][+-]?<int> | (<int>?.<int>?)([eE][+-]?<int>)?
       <imag>  ::=  (<int>|<real>)im
       <bool>  ::=  true|false
  <bitstring>  ::=  "([01][_]?)*[01]"
       <time>  ::=  (<int>|<real>)(dt|ns|us|µs|ms|s)
     <string>  ::=  "[^"\r\t\n]*" | '[^'\r\t\n]*'
   <constant>  ::=  pi | π | tau | τ | euler | ε

<nondecimalint>  ::=  (0b|0B)([01][_]?)*[01]
                 |  0o([0-7][_]?)*[0-7]
                 |  (0x|0X)([0-9a-fA-F][_]?)*[0-9a-fA-F]

    <comment>  ::=  // ...
                 |  /* ...  */

     <header>  ::=  <version>?  <include>* <io>*
    <version>  ::=  OPENQASM <int|real>;
    <include>  ::=  include <string>;
         <io>  ::=  input <ctype> <id>; | output <ctype> <id>;
```