OpenQASM 3 (legible) grammar

```
          <program>  ::=  <version>?  <stmt>*
          <version>  ::=  OPENQASM <int>; | OPENQASM <int>.<int>;

             <stmt>  ::=  <pragma>
                        | <annotation>* <unannotated-stmt>
           <pragma>  ::=  pragma ... | #pragma ...
       <annotation>  ::=  @<id> ...

<unannotated-stmt>  ::=  let <id> = <exp> ++ <exp> ++ ...  ++ <exp>;
                        | <indexid> <assignop> <exp|measureexp>;
                        | barrier <gateoperand...>?;
                        | box <scope> | box[<exp>] <scope>
                        | break;
                        | cal { CALIBR_BLOCK? }
                        | defcalgrammar <string>;
                        | <scalartype|arraytype> <id>;
                        | <scalartype|arraytype> <id> = <declexp>;
                        | const <scalartype> <id> = <declexp>;
                        | continue;
                        | <defstmt>
                        | <defcalstmt>
                        | delay[<exp>] <gateoperand...>?;
                        | end;
                        | <exp>;
                        | <externstmt>
                        | for <scalartype> <id> in <setdecl> <stmt|scope>
                        | <gatecall>
                        | <gatedef>
                        | if (<exp>) <stmt|scope>
                        | if (<exp>) <stmt|scope> else <stmt|scope>
                        | include <string>;
                        | input <scalartype|arraytype> <id>;
                        | output <scalartype|arraytype> <id>;
                        | <measureexp>; | <measureexp> -> <indexid>;
                        | creg <id>; | creg <id>[<exp>];
                        | qreg <id>; | qreg <id>[<exp>];
                        | <qubittype> <id>;
                        | reset <gateoperand>;
                        | return <exp|measureexp>?;
                        | while (<exp>) <stmt|scope>

            <scope>  ::=  { <stmt>* }

      <gateoperand>  ::=  <indexid> | <hardwarequbit>
    <hardwarequbit>  ::=  $[0-9]+
        <qubittype>  ::=  qubit | qubit[<exp>]
```

```
      <argdef>  ::=  <scalartype> <id> | <qubittype> <id> | <arrayreftype> <id>
                |  creg <id> | creg <id>[<exp>] | qreg <id> | qreg <id>[<exp>]
     <defstmt>  ::=  def <id>(<argdef...>?)  <scope>
                |  def <id>(<argdef...>?)  -> <scalartype> <scope>
   <externarg>  ::=  <scalartype> | <arrayreftype> | creg | creg[<exp>]
  <externstmt>  ::=  extern <id>(<externarg...>?);
                |  extern <id>(<externarg...>?)  -> <scalartype>;

<gatemodifier>  ::=  inv @ | pow(<exp>) @
                |  ctrl @ | ctrl(<exp>) @
                |  negctrl @ | negctrl(<exp>) @
    <gatecall>  ::=  <gatemodifier>* <id> <gateoperand...>;
                |  <gatemodifier>* <id>(<exp...>?)  <gateoperand...>;
                |  <gatemodifier>* gphase(<exp>) <gateoperand...>?;
     <gatedef>  ::=  gate <id> <id...> <scope>
                |  gate <id>(<id...>?)  <id...> <scope>

         <tgt>  ::=  measure | reset | delay | <id>
         <opd>  ::=  <hardwarequbit> | <id>
  <defcalstmt>  ::=  defcal <tgt> <opd...> { CALIBR_BLOCK? }
                |  defcal <tgt> <opd...> -> <scalartype> { CALIBR_BLOCK? }
                |  defcal <tgt>(<exp|argdef...>?)  <opd...> { CALIBR_BLOCK? }
                |  defcal <tgt>(<exp|argdef...>?)  <opd...> -> <scalartype>
                   { CALIBR_BLOCK? }

<arrayliteral>  ::=  {<exp|arrayliteral>}
                |  {<exp|arrayliteral>, <exp|arrayliteral>, ...}
  <measureexp>  ::=  measure <gateoperand>
     <declexp>  ::=  <arrayliteral|exp|measureexp>

     <setdecl>  ::=  { <exp...> }
                |  [<exp>?:<exp>?] | [<exp>?:<exp>?:<exp>]
                |  <id>

   <scalartype>  ::=  bit | bit[<exp>] | int | int[<exp>] | uint | uint[<exp>]
                |  float | float[<exp>] | angle | angle[<exp>] | bool
                |  duration | stretch | complex | complex[<scalartype>]
    <arraytype>  ::=  array[<scalartype>, <exp...>]
 <arrayreftype>  ::=  const <arrayrefspec> | mutable <arrayrefspec>
 <arrayrefspec>  ::=  array[<scalartype>, <exp...>]
                |  array[<scalartype>, #dim = <exp>]

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

          <exp>  ::=  (<exp>)
                |  <exp> <indexoperator>
                |  <exp> <b-op> <exp>
                |  <u-op> <exp>
                |  <scalartype>(<exp>) | <arraytype>(<exp>)
                |  durationof(<scope>)
                |  <id>(<exp...>?)
```

```
                  |  <id> | <int> | <nondecimalint> | <real> | <imag>
                  |  <bool> | <bitstring> | <time> | <hardwarequbit>

        <b-op>  ::=  || | && | | | ^ | & | == | != | > | < | >= | <=
                  |  << | >> | + | - | * | / | % | **
        <u-op>  ::=  ~ | ! | -

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

         <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]*'

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

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

<builtin-call>  ::=  arccos | arcsin | arctan | ceiling | cos | exp | floor | log
                  |  mod | popcount | pow | rotl | rotr | sin | sqrt | tan
                  |  sizeof | real | imag
    <constant>  ::=  pi | π | tau | τ | euler | ε
```