1. Задания для лексического анализа

Задача 1.1. *С помощью генератора лексических анализаторов Мяука для приводимого ниже языка напишите и протестируйте лексический анализатор.*

программа → *(переменные|прототип|функция){переменные|прототип|функция}*
переменные →**перем** *списид:тип_перем{;списид:тип_перем}*
списид →*ид{,ид}*
тип_перем →**цел**|**вещ**|**лог**|**симв**|**строка**|**ничто**|**массив**[*[выр]{,[выр]}*]*тип_перем*
выр →$выр_0$[?$выр_0$:$выр_0$]
$выр_0$ →$выр_1${(||!||^^|!^^)$выр_1$}
$выр_1$ →$выр_2${(&&|!&&)$выр_2$}
$выр_2$ →{!}$выр_3$
$выр_3$ →$выр_4${(<|>|<=|>=|=|!=)$выр_4$}
$выр_4$ →$выр_5${(|~|^|~^)$выр_5$}
$выр_5$ →$выр_6${(&|~&|≪|≫)$выр_6$}
$выр_6$ →{~}$выр_7$
$выр_7$ →$выр_8${(+|-)$выр_8$}
$выр_8$ →$выр_9${(*|/|%|/.)$выр_9$}
$выр_9$ →$выр_{10}$[(**|**.)$выр_9$]
$выр_{10}$ →[+|-]$выр_{11}$
$выр_{11}$→$выр_{12}$[#$выр_{12}$]
$выр_{12}$ →целое|вещественное|**истина**|**ложь**|строковое|(*выр*)|*имя*
имя →*ид{([списвыр])|[списвыр]}*
списвыр →*выр{,выр}*
прототип →**прото** *заголовок_функ*
заголовок_функ →**функ** *ид*(*списформпарам*):*тип_перем*
списформпарам →ε| *группа_парам{;группа_парам}*
группа_парам →[**перем**|**конст**]*списид:тип_перем*
функция →*заголовок_функ* {*тело_функ*}
тело_функ →{*переменные*|*операторы*}
операторы →[*оператор*]{;[*оператор*]}
оператор →*цикла*|*присваивания*|*перехода*|*ввода*|*вывода*|*условный*
условный →**если** *выр* **то**{*тело_функ*}{**инес** *выр* **то**{*тело_функ*}}[**иначе**{*тело_функ*}]**всё**
цикла →[*ид*::](**для** *ид* **из** *выр*{*тело_функ*}|**для** *ид* := *выр*,*выр*[,*выр*]{*тело_функ*}|
пока *выр*{*тело_функ*}|**повторяй**{*тело_функ*}**покуда** *выр*)
присваивания →*имя* := *выр*
перехода →**выйди** [**из** *ид*]|**продолжи** [*ид*]|**возврат** [*выр*]
ввода →**ввод**(*списимён*)
вывода →**вывод**(*списимён*)
списимён →*имя*{,*имя*}

Здесь целочисленные и вещественные литералы выглядят так (записаны соответствующие регулярные определения):

целое →*десятичное*|*восьмеричное*|*двоичное*|*шестнадцатеричное*
десятичное →*дес_цифра*('?*дес_цифра*)*
восьмеричное →0o*восьм_цифра*('?*восьм_цифра*)*
двоичное →0(b|B)*двоичн_цифра*('?*двоичн_цифра*)*
шестнадцатеричное →0(x|X)*шестн_цифра*('?*шестн_цифра*)*
дес_цифра →0|1|2|3|4|5|6|7|8|9
восьм_цифра →0|1|2|3|4|5|6|7
двоичн_цифра →0|1
шестн_цифра →0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f|A|B|C|D|E|F
вещественное →*целая_часть*(*дробная_часть*)?(E|e)(+|-)?*порядок*
целая_часть →*десятичное*
дробная_часть →.*десятичное*
порядок →*десятичное*

Строковый литерал — это любая (в том числе и пустая) последовательность символов, заключённая в одинарные кавычки. Если в строковом литерале нужно указать саму одинарную кавычку,

то эту кавычку нужно удвоить. Символьный литерал представляет собой либо заключённый в одинарные кавычки символ, либо символ, заданный своим кодом. Последнее представляет собой знак $, после которого идёт целое число, являющееся кодом символа. Число это может быть двоичным, восьмеричным, шестнадцатеричным, или десятичным Внутреннее представление строковых и символьных литералов — в кодироке Unicode, а именно, в её четырёхбайтовом варианте. Текст программы — только в кодировке UTF-8.

Идентификатор представляет собой последовательность букв, цифр, и знаков подчёркивания, и может начинаться либо с буквы, либо со знака подчёркивания. Под буквой понимается либо русская буква, либо латинская буква.

При реализации сканера для хранения целочисленного значения лексемы следует использовать значение типа `unsigned __int128`*, а для хранения вещественного — значение типа* `__float128`*.*

Задача 1.2. *С помощью генератора лексических анализаторов Мяука для приводимого ниже языка напишите и протестируйте лексический анализатор.*

программа →*(переменные|прототип|функция){переменные|прототип|функция}*
переменные →**var** *списид:тип_перем{;списид:тип_перем}*
списид →*ид{,ид}*
тип_перем →**int**|**float**|**bool**|**char**|**string**|**void**|**array**[*[выр]{,[выр]}*]*тип_перем*
выр →*выр*$_0$[?*выр*$_0$:*выр*$_0$]
выр$_0$ →*выр*$_1${(||!||^^|!^^)*выр*$_1$}
выр$_1$ →*выр*$_2${(&&|!&&)*выр*$_2$}
выр$_2$ →{!}*выр*$_3$
выр$_3$ →*выр*$_4${(<|>|<=|>=|=|!=)*выр*$_4$}
выр$_4$ →*выр*$_5${(||~||^|~^)*выр*$_5$}
выр$_5$ →*выр*$_6${(&|~&|≪|≫)*выр*$_6$}
выр$_6$ →{~}*выр*$_7$
выр$_7$ →*выр*$_8${(+|-)*выр*$_8$}
выр$_8$ →*выр*$_9${(*|/|%|/.)*выр*$_9$}
выр$_9$ →*выр*$_{10}$[(**|**.)*выр*$_9$]
выр$_{10}$ →[+|-]*выр*$_{11}$
выр$_{11}$→*выр*$_{12}$[#*выр*$_{12}$]
выр$_{12}$ →*целое*|*вещественное*|**true**|**false**|*строковое*|(*выр*)|*имя*
имя →*ид*{([*списвыр*])|[*списвыр*]}
списвыр →*выр*{,*выр*}
прототип →**proto** *заголовок_функ*
заголовок_функ →**func** *ид*(*списформпарам*):*тип_перем*
списформпарам →ε| *группа_парам*{;*группа_парам*}
группа_парам →[**var**|**const**]*списид*:*тип_перем*
функция →*заголовок_функ* {*тело_функ*}
тело_функ →{*переменные*|*операторы*}
операторы →[*оператор*]{;[*оператор*]}
оператор →*цикла*|*присваивания*|*перехода*|*ввода*|*вывода*|*условный*
условный →**if** *выр* **then**{*тело_функ*}{**elseif** *выр* **then**{*тело_функ*}}[**else**{*тело_функ*}]**endif**
цикла →[*ид*::](**for** *ид* **in** *выр*{*тело_функ*}|**for** *ид* := *выр*,*выр*[,*выр*]{*тело_функ*}|
while *выр*{*тело_функ*}|**repeat**{*тело_функ*}**until** *выр*)
присваивания →*имя* := *выр*
перехода →**exit** [*ид*]|**continue** [*ид*]|**return** [*выр*]
ввода →**read**(*списимён*)
вывода →**print**(*списимён*)
списимён →*имя*{,*имя*}

Здесь целочисленные и вещественные литералы выглядят так (записаны соответствующие регулярные определения):

целое →*десятичное*|*восьмеричное*|*двоичное*|*шестнадцатеричное*
десятичное →*дес_цифра*('?*дес_цифра*)*
восьмеричное →0o*восьм_цифра*('?*восьм_цифра*)*
двоичное →0(b|B)*двоичн_цифра*('?*двоичн_цифра*)*
шестнадцатеричное →0(x|X)*шестн_цифра*('?*шестн_цифра*)*
дес_цифра →0|1|2|3|4|5|6|7|8|9

восьм_ цифра →*0*|*1*|*2*|*3*|*4*|*5*|*6*|*7*
двоичн_ цифра →*0*|*1*
шестн_ цифра →*0*|*1*|*2*|*3*|*4*|*5*|*6*|*7*|*8*|*9*|a|b|c|d|e|f|A|B|C|D|E|F
вещественное →*целая_ часть*(*дробная_ часть*)?(E|e)(+|-)?*порядок*
целая_ часть →*десятичное*
дробная_ часть →.*десятичное*
порядок →*десятичное*

Строковый литерал — это любая (в том числе и пустая) последовательность символов, заключённая в одинарные кавычки. Если в строковом литерале нужно указать саму одинарную кавычку, то эту кавычку нужно удвоить. Символьный литерал представляет собой либо заключённый в одинарные кавычки символ, либо символ, заданный своим кодом. Последнее представляет собой знак $, после которого идёт целое число, являющееся кодом символа. Число это может быть двоичным, восьмеричным, шестнадцатеричным, или десятичным Внутреннее представление строковых и символьных литералов — в кодироке Unicode, а именно, в её четырёхбайтовом варианте. Текст программы — только в кодировке UTF-8.

Идентификатор представляет собой последовательность букв, цифр, и знаков подчёркивания, и может начинаться либо с буквы, либо со знака подчёркивания. Под буквой понимается либо русская буква, либо латинская буква.

При реализации сканера для хранения целочисленного значения лексемы следует использовать значение типа `unsigned __int128`*, а для хранения вещественного — значение типа* `__float128`*.*

Задача 1.3. *С помощью генератора лексических анализаторов Мяука для приводимого ниже языка напишите и протестируйте лексический анализатор.*

программа → (*переменные*|*прототип*|*функция*){*переменные*|*прототип*|*функция*}
переменные →**перем** *списид*:*тип_ перем*{;*списид*:*тип_ перем*}
списид →*ид*{,*ид*}
тип_ перем →**цел**|**вещ**|**лог**|**симв**|**строка**|**ничто**|**массив**[[*выр*]{,[*выр*]}]*тип_ перем*
выр →$выр_0$[?$выр_0$:$выр_0$]
$выр_0$ →$выр_1${(|||!|||^^|!^^)$выр_1$}
$выр_1$ →$выр_2${(&&|!&&)$выр_2$}
$выр_2$ →{!}$выр_3$
$выр_3$ →$выр_4${(<|>|<=|>=|=|!=)$выр_4$}
$выр_4$ →$выр_5${(|~||^|~^)$выр_5$}
$выр_5$ →$выр_6${(&|~&|≪|≫)$выр_6$}
$выр_6$ →{~}$выр_7$
$выр_7$ →$выр_8${(+|-)$выр_8$}
$выр_8$ →$выр_9${(*|/|%|/.)$выр_9$}
$выр_9$ →$выр_{10}$[(**|**.)$выр_9$]
$выр_{10}$ →[+|-]$выр_{11}$
$выр_{11}$→$выр_{12}$[#$выр_{12}$]
$выр_{12}$ →*целое*|*вещественное*|**истина**|**ложь**|*строковое*|(*выр*)|*имя*
имя →*ид*{([*списвыр*])|[*списвыр*]}
списвыр →*выр*{,*выр*}
прототип →**прото** *заголовок_ функ*
заголовок_ функ →**функ** *ид*(*списформпарам*):*тип_ перем*
списформпарам →ε| *группа_ парам*{;*группа_ парам*}
группа_ парам →[**перем**|**конст**]*списид*:*тип_ перем*
функция →*заголовок_ функ* {*тело_ функ*}
тело_ функ →{*переменные*|*операторы*}
операторы →[*оператор*]{;[*оператор*]}
оператор →*цикла*|*присваивания*|*перехода*|*ввода*|*вывода*|*условный*
условный →**если** *выр* **то**{*тело_ функ*}{**инес** *выр* **то**{*тело_ функ*}}[**иначе**{*тело_ функ*}]**всё**
цикла →[*ид*::](**для** *ид* **из** *выр*{*тело_ функ*}[**завершение**{*тело_ функ*}]|
для *ид* := *выр*,*выр*[,*выр*]{*тело_ функ*}[**завершение**{*тело_ функ*}]|
пока *выр*{*тело_ функ*}|**повторяй**{*тело_ функ*}**покуда** *выр*)
присваивания →*имя* := *выр*
перехода →**выйди** [**из** *ид*]|**продолжи** [*ид*]|**возврат** [*выр*]
ввода →**ввод**(*списимён*)

вывода →**вывод**(*списимён*)
списимён →*имя*{,*имя*}

Здесь целочисленные и вещественные литералы выглядят так (записаны соответствующие регулярные определения):

целое →*десятичное*|*восьмеричное*|*двоичное*|*шестнадцатеричное*
десятичное →*дес_цифра*('?*дес_цифра*)*
восьмеричное →0o*восьм_цифра*('?*восьм_цифра*)*
двоичное →0(b|B)*двоичн_цифра*('?*двоичн_цифра*)*
шестнадцатеричное →0(x|X)*шестн_цифра*('?*шестн_цифра*)*
дес_цифра →0|1|2|3|4|5|6|7|8|9
восьм_цифра →0|1|2|3|4|5|6|7
двоичн_цифра →0|1
шестн_цифра →0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f|A|B|C|D|E|F
вещественное →*целая_часть*(*дробная_часть*)?(E|e)(+|-)?*порядок*
целая_часть →*десятичное*
дробная_часть →.*десятичное*
порядок →*десятичное*

Строковый литерал — это любая (в том числе и пустая) последовательность символов, заключённая в одинарные кавычки. Если в строковом литерале нужно указать саму одинарную кавычку, то эту кавычку нужно удвоить. Символьный литерал представляет собой либо заключённый в одинарные кавычки символ, либо символ, заданный своим кодом. Последнее представляет собой знак $, после которого идёт целое число, являющееся кодом символа. Число это может быть двоичным, восьмеричным, шестнадцатеричным, или десятичным Внутреннее представление строковых и символьных литералов — в кодироке Unicode, а именно, в её четырёхбайтовом варианте. Текст программы — только в кодировке UTF-8.

Идентификатор представляет собой последовательность букв, цифр, и знаков подчёркивания, и может начинаться либо с буквы, либо со знака подчёркивания. Под буквой понимается либо русская буква, либо латинская буква.

При реализации сканера для хранения целочисленного значения лексемы следует использовать значение типа `unsigned __int128`*, а для хранения вещественного — значение типа* `__float128`*.*

Задача 1.4. *С помощью генератора лексических анализаторов Мяука для приводимого ниже языка напишите и протестируйте лексический анализатор.*

программа → (*переменные*|*прототип*|*функция*){*переменные*|*прототип*|*функция*}
переменные →**var** *списид*:*тип_перем*{;*списид*:*тип_перем*}
списид →*ид*{,*ид*}
тип_перем →**int**|**float**|**bool**|**char**|**string**|**void**|**array**[[*выр*]{,[*выр*]}]*тип_перем*
выр →*выр*$_0$[?*выр*$_0$:*выр*$_0$]
выр$_0$ →*выр*$_1${(||!||^^|!^^)*выр*$_1$}
выр$_1$ →*выр*$_2${(&&|!&&)*выр*$_2$}
выр$_2$ →{!}*выр*$_3$
выр$_3$ →*выр*$_4${(<|>|<=|>=|=|!=)*выр*$_4$}
выр$_4$ →*выр*$_5${(|~|^|~^)*выр*$_5$}
выр$_5$ →*выр*$_6${(&|~&|≪|≫)*выр*$_6$}
выр$_6$ →{~}*выр*$_7$
выр$_7$ →*выр*$_8${(+|-)*выр*$_8$}
выр$_8$ →*выр*$_9${(*|/|%|/.)*выр*$_9$}
выр$_9$ →*выр*$_{10}$[(**|**.)*выр*$_9$]
выр$_{10}$ →[+|-]*выр*$_{11}$
выр$_{11}$→*выр*$_{12}$[#*выр*$_{12}$]
выр$_{12}$ →*целое*|*вещественное*|**true**|**false**|*строковое*|(*выр*)|*имя*
имя →*ид*{([*списвыр*])|[*списвыр*]}
списвыр →*выр*{,*выр*}
прототип →**proto** *заголовок_функ*
заголовок_функ →**func** *ид*(*списформпарам*):*тип_перем*
списформпарам →ε| *группа_парам*{;*группа_парам*}
группа_парам →[**var**|**const**]*списид*:*тип_перем*
функция →*заголовок_функ* {*тело_функ*}

тело_функ →{*переменные*|*операторы*}
операторы →[*оператор*]{;[*оператор*]}
оператор →*цикла*|*присваивания*|*перехода*|*ввода*|*вывода*|*условный*
условный →**if**(*выр*){*тело_функ*}{**elif**(*выр*){*тело_функ*}}[**else**{*тело_функ*}]**endif**
цикла →[*ид*::](**for** *ид* **in** *выр*{*тело_функ*}[**else**{*тело_функ*}]|
for *ид* := *выр*,*выр*[,*выр*]{*тело_функ*}[**else**{*тело_функ*}]|
while *выр*{*тело_функ*}|**repeat**{*тело_функ*}**until** *выр*)
присваивания →*имя* := *выр*
перехода →**exit** [*ид*]|**continue** [*ид*]|**return** [*выр*]
ввода →**read**(*списимён*)
вывода →**print**(*списимён*)
списимён →*имя*{,*имя*}

Здесь целочисленные и вещественные литералы выглядят так (записаны соответствующие регулярные определения):

целое →*десятичное*|*восьмеричное*|*двоичное*|*шестнадцатеричное*
десятичное →*дес_цифра*('?*дес_цифра*)*
восьмеричное →0*восьм_цифра*('?*восьм_цифра*)*
двоичное →0(b|B)*двоичн_цифра*('?*двоичн_цифра*)*
шестнадцатеричное →0(x|X)*шестн_цифра*('?*шестн_цифра*)*
дес_цифра →0|1|2|3|4|5|6|7|8|9
восьм_цифра →0|1|2|3|4|5|6|7
двоичн_цифра →0|1
шестн_цифра →0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f|A|B|C|D|E|F
вещественное →*целая_часть*(*дробная_часть*)?(E|e)(+|-)?*порядок*
целая_часть →*десятичное*
дробная_часть →.*десятичное*
порядок →*десятичное*

Строковый литерал — это любая (в том числе и пустая) последовательность символов, заключённая в одинарные кавычки. Если в строковом литерале нужно указать саму одинарную кавычку, то эту кавычку нужно удвоить. Символьный литерал представляет собой либо заключённый в одинарные кавычки символ, либо символ, заданный своим кодом. Последнее представляет собой знак $, после которого идёт целое число, являющееся кодом символа. Число это может быть двоичным, восьмеричным, шестнадцатеричным, или десятичным Внутреннее представление строковых и символьных литералов — в кодироке Unicode, а именно, в её четырёхбайтовом варианте. Текст программы — только в кодировке UTF-8.

Идентификатор представляет собой последовательность букв, цифр, и знаков подчёркивания, и может начинаться либо с буквы, либо со знака подчёркивания. Под буквой понимается либо русская буква, либо латинская буква.

При реализации сканера для хранения целочисленного значения лексемы следует использовать значение типа `unsigned __int128`, а для хранения вещественного — значение типа `__float128`.

Задача 1.5. *С помощью генератора лексических анализаторов Мяука для приводимого ниже языка напишите и протестируйте лексический анализатор.*

программа → (*переменные*|*прототип*|*функция*){*переменные*|*прототип*|*функция*}
переменные →**перем** *списид*:*тип_перем*{;*списид*:*тип_перем*}
списид →*ид*{,*ид*}
тип_перем →**цел**|**вещ**|**лог**|**симв**|**строка**|**ничто**|**массив**[[*выр*]{,[*выр*]}]*тип_перем*
выр →*выр*$_0$[?*выр*$_0$:*выр*$_0$]
выр$_0$ →*выр*$_1$ {(||!||^^|!^^)*выр*$_1$}
выр$_1$ →*выр*$_2$ {(&&|!&&)*выр*$_2$}
выр$_2$ →{!}*выр*$_3$
выр$_3$ →*выр*$_4$ {(<|>|<=|>=|=|!=)*выр*$_4$}
выр$_4$ →*выр*$_5$ {(|~||^|~^)*выр*$_5$}
выр$_5$ →*выр*$_6$ {(&|~&|≪|≫)*выр*$_6$}
выр$_6$ →{~}*выр*$_7$
выр$_7$ →*выр*$_8$ {(+|-)*выр*$_8$}
выр$_8$ →*выр*$_9$ {(*|/|%|/.)*выр*$_9$}
выр$_9$ →*выр*$_{10}$[(**|**.)*выр*$_9$]

выр$_{10}$ →[+|-]*выр*$_{11}$
выр$_{11}$→*выр*$_{12}$[#*выр*$_{12}$]
выр$_{12}$ →*целое*|*вещественное*|**истина**|**ложь**|*строковое*|(*выр*)|*имя*
имя →*ид*{([*списвыр*])|[*списвыр*]}
списвыр →*выр*{,*выр*}
прототип →**прото** *заголовок_функ*
заголовок_функ →**функ** *ид*(*списформпарам*):*тип_перем*
списформпарам →ε| *группа_парам*{;*группа_парам*}
группа_парам →[**перем**|**конст**]*списид*:*тип_перем*
функция →*заголовок_функ* {*тело_функ*}
тело_функ →{*переменные*|*операторы*}
операторы →[*оператор*]{;[*оператор*]}
оператор →*цикла*|*присваивания*|*перехода*|*ввода*|*вывода*|*условный*
условный →**если** *выр* **то**{*тело_функ*}{**инес** *выр* **то**{*тело_функ*}}[**иначе**{*тело_функ*}]**всё**
цикла →[*ид*::](**для** *ид* **из** *выр*{*тело_функ*}|**для** *ид* := *выр*,*выр*[,*выр*]{*тело_функ*}|
пока *выр*{*тело_функ*}|**повторяй**{*тело_функ*}**покуда** *выр*)
присваивания →*имя* (:=|+:=|-:=|*:=|/:=|/.:=|**:=|**.:=|%:=|&:=|~&:=|≪:=|≫:=||:=|~|:=|^:=|~^:=|
||:=|!|:=|^^:=|!^^:=|&&:=|!&&:=) *выр*
перехода →**выйди** [**из** *ид*]|**продолжи** [*ид*]|**возврат** [*выр*]
ввода →**ввод**(*списимён*)
вывода →**вывод**(*списимён*)
списимён →*имя*{,*имя*}

Здесь целочисленные и вещественные литералы выглядят так (записаны соответствующие регулярные определения):
целое →*десятичное*|*восьмеричное*|*двоичное*|*шестнадцатеричное*
десятичное →*дес_цифра*('?*дес_цифра*)*
восьмеричное →0o*восьм_цифра*('?*восьм_цифра*)*
двоичное →0(b|B)*двоичн_цифра*('?*двоичн_цифра*)*
шестнадцатеричное →0(x|X)*шестн_цифра*('?*шестн_цифра*)*
дес_цифра →0|1|2|3|4|5|6|7|8|9
восьм_цифра →0|1|2|3|4|5|6|7
двоичн_цифра →0|1
шестн_цифра →0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f|A|B|C|D|E|F
вещественное →*целая_часть*(*дробная_часть*)?(E|e)(+|-)?*порядок*
целая_часть →*десятичное*
дробная_часть →.*десятичное*
порядок →*десятичное*

Строковый литерал — это любая (в том числе и пустая) последовательность символов, заключённая в одинарные кавычки. Если в строковом литерале нужно указать саму одинарную кавычку, то эту кавычку нужно удвоить. Символьный литерал представляет собой либо заключённый в одинарные кавычки символ, либо символ, заданный своим кодом. Последнее представляет собой знак $, после которого идёт целое число, являющееся кодом символа. Число это может быть двоичным, восьмеричным, шестнадцатеричным, или десятичным Внутреннее представление строковых и символьных литералов — в кодироке Unicode, а именно, в её четырёхбайтовом варианте. Текст программы — только в кодировке UTF-8.

Идентификатор представляет собой последовательность букв, цифр, и знаков подчёркивания, и может начинаться либо с буквы, либо со знака подчёркивания. Под буквой понимается либо русская буква, либо латинская буква.

При реализации сканера для хранения целочисленного значения лексемы следует использовать значение типа `unsigned __int128`, а для хранения вещественного — значение типа `__float128`.

Задача 1.6. *С помощью генератора лексических анализаторов Мяука для приводимого ниже языка напишите и протестируйте лексический анализатор.*
программа → (*переменные*|*прототип*|*функция*){*переменные*|*прототип*|*функция*}
переменные →**var** *списид*:*тип_перем*{;*списид*:*тип_перем*}
списид →*ид*{,*ид*}
тип_перем →**int**|**float**|**bool**|**char**|**string**|**void**|**array**[[*выр*]{,[*выр*]}]*тип_перем*
выр →*выр*$_0$[?*выр*$_0$:*выр*$_0$]

выр$_0$ →*выр*$_1${(**||**|**!||**|^^|**!^^**)*выр*$_1$}
выр$_1$ →*выр*$_2${(**&&**|**!&&**)*выр*$_2$}
выр$_2$ →{**!**}*выр*$_3$
выр$_3$ →*выр*$_4${(**<**|**>**|**<=**|**>=**|**=**|**!=**)*выр*$_4$}
выр$_4$ →*выр*$_5${(**|**|**~|**|^|**~^**)*выр*$_5$}
выр$_5$ →*выр*$_6${(**&**|**~&**|≪|≫)*выр*$_6$}
выр$_6$ →{**~**}*выр*$_7$
выр$_7$ →*выр*$_8${(**+**|**-**)*выр*$_8$}
выр$_8$ →*выр*$_9${(*****|**/**|**%**|**/.**)*выр*$_9$}
выр$_9$ →*выр*$_{10}$[(******|****.**)*выр*$_9$]
выр$_{10}$ →[**+**|**-**]*выр*$_{11}$
выр$_{11}$→*выр*$_{12}$[**#***выр*$_{12}$]
выр$_{12}$ →*целое*|*вещественное*|**true**|**false**|*строковое*|(*выр*)|*имя*
имя →*ид*{([*списвыр*])|[*списвыр*]}
списвыр →*выр*{,*выр*}
прототип →**proto** *заголовок_функ*
заголовок_функ →**func** *ид*(*списформпарам*):*тип_перем*
списформпарам →ε| *группа_парам*{;*группа_парам*}
группа_парам →[**var**|**const**]*списид*:*тип_перем*
функция →*заголовок_функ* {*тело_функ*}
тело_функ →{*переменные*|*операторы*}
операторы →[*оператор*]{;[*оператор*]}
оператор →*цикла*|*присваивания*|*перехода*|*ввода*|*вывода*|*условный*
условный →**if**(*выр*){*тело_функ*}{**elif**(*выр*){*тело_функ*}}[**else**{*тело_функ*}]**endif**
цикла →[*ид*::](**for** *ид* **in** *выр*{*тело_функ*}[**else**{*тело_функ*}]|
for *ид* := *выр*,*выр*[,*выр*]{*тело_функ*}[**else**{*тело_функ*}]|
while *выр*{*тело_функ*}|**repeat**{*тело_функ*}**until** *выр*)
присваивания →*имя* (**:=**|**+:=**|**-:=**|***:=**|**/:=**|**/.:=**|****:=**|****.:=**|**%:=**|**&:=**|**~&:=**|**≪:=**|**≫:=**|**||:=**|**~|:=**|**^:=**|**~^:=**|
||:=|**!||:=**|**^^:=**|**!^^:=**|**&&:=**|**!&&:=**) *выр*
перехода →**exit** [*ид*]|**continue** [*ид*]|**return** [*выр*]
ввода →**read**(*списимён*)
вывода →**print**(*списимён*)
списимён →*имя*{,*имя*}

Здесь целочисленные и вещественные литералы выглядят так (записаны соответствующие регулярные определения):
целое →*десятичное*|*восьмеричное*|*двоичное*|*шестнадцатеричное*
десятичное →*дес_цифра*('?*дес_цифра*)*
восьмеричное →0*восьм_цифра*('?*восьм_цифра*)*
двоичное →0(b|B)*двоичн_цифра*('?*двоичн_цифра*)*
шестнадцатеричное →0(x|X)*шестн_цифра*('?*шестн_цифра*)*
дес_цифра →0|1|2|3|4|5|6|7|8|9
восьм_цифра →0|1|2|3|4|5|6|7
двоичн_цифра →0|1
шестн_цифра →0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f|A|B|C|D|E|F
вещественное →*целая_часть*(*дробная_часть*)?(E|e)(+|-)?*порядок*
целая_часть →*десятичное*
дробная_часть →.*десятичное*
порядок →*десятичное*

Строковый литерал — это любая (в том числе и пустая) последовательность символов, заключённая в одинарные кавычки. Если в строковом литерале нужно указать саму одинарную кавычку, то эту кавычку нужно удвоить. Символьный литерал представляет собой либо заключённый в одинарные кавычки символ, либо символ, заданный своим кодом. Последнее представляет собой знак $, после которого идёт целое число, являющееся кодом символа. Число это может быть двоичным, восьмеричным, шестнадцатеричным, или десятичным Внутреннее представление строковых и символьных литералов — в кодироке Unicode, а именно, в её четырёхбайтовом варианте. Текст программы — только в кодировке UTF-8.

Идентификатор представляет собой последовательность букв, цифр, и знаков подчёркивания, и может начинаться либо с буквы, либо со знака подчёркивания. Под буквой понимается либо русская буква, либо латинская буква.

При реализации сканера для хранения целочисленного значения лексемы следует использовать значение типа `unsigned __int128`*, а для хранения вещественного — значение типа* `__float128`*.*