Приложение. *Описание языка C♭*

1. Конструкции языка C♭

Мы начнем описание языка *Си-бемоль*(*C♭*) с общего описания конструкций, допустимых в этом языке (детали синтаксиса можно найти в прилагаемой грамматике, см. следующую страницу):

1. *Программа*: Программа состоит из описаний классов, статических глобальных функций и директив **using**.

2. *Описание класса*: класс состоит из описаний вложенных классов, описаний полей и методов. Единственным атрибутом полей и методов, который может быть явно указан, является **static**. Атрибуты видимости и прочие модификаторы не могут быть указаны явно (подразумевается **public**).

   Наследование допускается только от классов (не от интерфейсов).

3. *Методы*: заголовки методов являются более или менее обычными, но квалификаторы **ref**, **out** использоваться не могут. Также отсутствуют методы с переменным число параметров.

   Тело метода может содержать операторы-выражения (например, вызов метода), **if**, **while**, **for** и **return**, а также операторы блока и описания переменных.

4. В *выражениях* допустимо использование операций **+**, **-** , ***** , **/** , **%** , **&, |** , ^, вызова метода и индексации массива. Кроме того, реализованы операции сравнения выражений типов **int**, **char**, **float** на больше/меньше и (не)равенство и сравнение строк на равенство/неравенство.

   Также реализован оператор **new**, как для массивов, так и для простых объектов. Хотя описание собственных конструкторов с параметрами невозможно, вызов конструкторов стандартных классов с передачей им списка параметров допустим.

   Следует отметить, что оператор присваивания, в отличие от C#, возвращает значение типа **void**, то есть фактически может быть использован только как оператор-выражение.

   Особым образом следует сказать про вызов перегруженных методов в *C♭*: при вызове методов из стандартных классов используется стандартный алгоритм разрешения. При вызове методов из определенных в программе классов используется упрощенный алгоритм: выбор подходящего метода осуществляется только по числу параметров.

5. *Типы*: реализованы встроенные типы **void**, **int**, **char**, **string**, **float**, **object**. Также возможнен доступ к любым классам из библиотеки **mscorlib** (доступ к другим библиотекам не реализован) и массивы из этих типов.

2. Грамматика языка C♭

Язык *C♭* является подмножеством языка C#. Ниже приведена грамматика C-бемоль.

```
program ::= { class | method }
member  ::= class | method | fields
scope   ::= ["static"]
class   ::= "class" ident "{"  member { ";" member } "}"
fields  ::= scope type ident {"," ident } ";"

method  ::= scope type ident "(" [arg {"," arg}] ")" block
arg     ::= type ident

stmt     ::= ";" | decl | expr ";" | while | for | if | return | block
           | break | continue
block    ::= "{" {stmt} "}"
decl     ::= type ident [ = expr] {"," ident [ = expr]} ";"
while    ::= "while" "(" expr ")" stmt
for      ::= "for" "(" [expr] ";" [expr] ";" [expr] ")" stmt
if       ::= "if" "(" expr ")" stmt ["else" stmt]
return   ::= "return" expr ";"
break    ::= "break" ";"
continue ::= "continue" ";"

primary   ::= ident | int-const | float-const | char-const | string-const
            | "(" expr ")" | new-expr
args      ::= "(" [expr {"," expr } ] ")"
new-expr  ::= "new" type ("[" expr "]" | args)
postfix   ::= primary { "[" expr "]" | "." ident | args }
mul-expr  ::= [ mul-expr ("*" | "/" | "%") ] primary
add-expr  ::= [add-expr  ("+" | "-") ] mul-expr
rel-expr  ::= add-expr [ ("<" | ">" | "<=" | ">=" | "==" | "!=") add-expr ]
bool-expr ::= [ bool-expr ("&&" | "&" | "|" | "^") ] bool-expr
assign-expr ::= bool-expr [ "=" bool-expr]
expr ::= bool-expr

type ::= ident { "." ident } | type "[]" | "void" | "string" | "char" | "bool"
       | "int" | "float"
```