Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ	Информатика и системы управления
КАФЕДРА	Программное обеспечение ЭВМ и информационные технологии

П О Я С Н И Т Е Л Ь Н А Я   З А П И С К А

к лабораторным работам 5 – 8

По курсу Конструирование компиляторов

Студент	____________	____________	**Радченко Д.В.**
		(Подпись, дата)	(И.О.Фамилия)
Студент	____________	____________	**Сулимов А.С.**
		(Подпись, дата)	(И.О.Фамилия)
Преподаватель	____________	____________	**Просуков Е.А.**
		(Подпись, дата)	(И.О.Фамилия)

Москва, 2011

Содержание

1 Пример программы на языке Cool

```
method void main() is
    declare integer i, j, k, m;
begin
      input >> i;
      input >> j;
      k = call summ(i, j);
      output << "SUMM result:";
      output << k;
      k = call substract(i, j);
      output << "Substract result:";
      output << k;
      call sub(i, j, ref k);
      output << "Substract result:";
      output << k;
end main

method integer summ(integer a, integer b) is
begin
      return a + b;
end main

method integer substract(integer a, integer b) is
begin
      return a - b;
end main

method void sub(integer a, integer b, ref integer c) is
begin
      c = a - b;
      return;
end main
```

2 Лабораторная работа №5. Лексический анализ

2.1 Постановка задачи

Аналитический анализ входной последовательности символов проводится с целью получения на выходе лексем (токенов).

2.2 Используемые инструменты

Gold Parser.

На вход программе подается LR(1) грамматика, на выходе получаем:

- класс разбора для необходимого нам языка программирования из предложенного списка (Ada, C, C#, C++, COBOL, D, Delphi, F#, Java, Object Pascal, Perl, Python, Visual Basic .NET, Visual Basic 6);
- бинарный файл грамматики в формате .egt.

После этого в проект подключаем библиотеку «GOLD Engine» и открываем сформированный egt – файл.

Грамматика для языка Cool, подаваемая на вход программы:

```
"Name"    = 'Cool'
"Version" = '1.0'
"Author"  = 'Dmitry Radchenko, Alexander Soulimov'
"About"   = 'The COOL Programming Language'

"Case Sensitive" = True
"Start Symbol"   = <PROGRAM>

Id = [_]*{Letter}+{Digit}*
Number = {Digit}+

{String Ch}      = {Printable} - ["]
{Char Ch}        = {Printable} - ['']
StringLiteral    = '"'( {String Ch} | '\'{Printable} )* '"'
CharLiteral      = '' ( {Char Ch} | '\'{Printable} )''

! =====================================================
! Comments
! =====================================================

Comment Block @= {Nesting = None, Advance = Character}

Comment Start = '/*'
Comment End   = '*/'
Comment Line  = '//'
```

```
!!
!! Global stuff. Module and body declaration.
!!

<PROGRAM>            ::= <CLASS> | <METHOD>
                     | <PROGRAM> <CLASS>
                     | <PROGRAM> <METHOD>

<BODY>               ::= <SUPER_INIT> <THIS_INIT> <BLOCK>
                     | <THIS_INIT> <BLOCK>
                     | <SUPER_INIT> <BLOCK>
                     | <BLOCK>

<THIS_INIT>          ::= this '(' <ARGLIST> ')'

<SUPER_INIT>         ::= super '(' <ARGLIST> ')'

<BLOCK>              ::= <VARDECS> begin <STATEMENTS> end | begin
<STATEMENTS> end

<VARDECLIST> ::= <TYPE> Id ';' | <TYPE> Id <VAR_TYPELIST>';'

<VAR_TYPELIST> ::= ',' Id | <VAR_TYPELIST> ',' Id

<VARDECS> ::= declare <VARDECLIST> | declare <VARDECLIST> <VARDECS>

<NAME>               ::=  Id
                     | <NAME> '.' Id


!
! EXPRESSIONS
!

<ASSIGNMENT>         ::=   <NAME> '=' <EXPRESSION>
                      | <NAME> '[' <EXPRESSION> ']' '=' <EXPRESSION>

<FACTOR>             ::= this
                     | super
                     | Number
                     | false
                     | true
                     | null
                     | <ALLOCATOR>
                     | <CAST_EXPR>

<ALLOCATOR>          ::= new <TYPE> '(' <ARGLIST> ')'
                     | new <TYPE> '(' ')'
                     | new <TYPE> '[' <EXPRESSION> ']'

<ARGLIST>            ::= <ARGUMENT>
                     | <ARGLIST> ',' <ARGUMENT>

<ARGUMENT>           ::= <EXPRESSION>
                     | ref <EXPRESSION>
```

```
<CAST_EXPR> ::= cast '(' <TYPE> ',' <EXPRESSION> ')'


<EXPRESSION>        ::= <EXPRESSION_TERM>
                    | <EXPRESSION> '+' <EXPRESSION_TERM>
                    | <EXPRESSION> '-' <EXPRESSION_TERM>

<EXPRESSION_TERM>   ::= <EXPRESSION_FACTOR>
                    | <EXPRESSION_TERM> '*' <EXPRESSION_FACTOR>
                    | <EXPRESSION_TERM> '/' <EXPRESSION_FACTOR>


<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
                    | <EXPRESSION_FACTOR> '%' <EXPRESSION_BINARY>
                    | <EXPRESSION_FACTOR> '>' <EXPRESSION_BINARY>
                    | <EXPRESSION_FACTOR> '<' <EXPRESSION_BINARY>
                    | <EXPRESSION_FACTOR> '>=' <EXPRESSION_BINARY>
                    | <EXPRESSION_FACTOR> '<=' <EXPRESSION_BINARY>
                    | <EXPRESSION_FACTOR> '==' <EXPRESSION_BINARY>
                    | <EXPRESSION_FACTOR> '#' <EXPRESSION_BINARY>

<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
                    | <EXPRESSION_BINARY> '&&' <EXPRESSION_UNARY>
                    | <EXPRESSION_BINARY> '||' <EXPRESSION_UNARY>

<EXPRESSION_UNARY>  ::= '+' <EXPRESSION_PRIMARY>
                    | '-' <EXPRESSION_PRIMARY>
                    | '!' <EXPRESSION_PRIMARY>
                    | <EXPRESSION_PRIMARY>
                    | <EXPRESSION_PRIMARY> '[' <EXPRESSION> ']'
                    | <EXPRESSION_PRIMARY> '(' <ARGLIST> ')'

<EXPRESSION_PRIMARY>    ::= <NAME>
                        | <FUNCTION_CALL>
                        | <FACTOR>
                        | '(' <EXPRESSION> ')'

!!
!! Statements
!!

<STATEMENTS>    ::=   <STATEMENT>
                    | <STATEMENTS> <STATEMENT>

<STATEMENT>     ::=   <BLOCK>
                    | <METHOD>
                    | <CLASS>
                    | <FUNCTION_CALL> ';'
                    | <ASSIGNMENT> ';'
                    | <INPUTSTMT> ';'
                    | <OUTPUTSTMT> ';'
                    | return <EXPRESSION> ';'
                    | return ';'
                    | continue ';'
                    | break ';'
```

```
                     | <IFSTMT>
                     | <TRYSTMT>
                     | loop <STATEMENTS> end loop
                     | exit ';'
                     | throw <EXPRESSION> ';'

<IFSTMT>             ::= if <EXPRESSION> then <STATEMENTS> end if
                     |   if <EXPRESSION> then <STATEMENTS> <ELSEPART>
end if
                     |   if <EXPRESSION> then <STATEMENTS>
<ELSEIF_PART> <ELSEPART> end if

<ELSEPART> ::= else <STATEMENTS>

<ELSEIF_PART> ::= elsif <EXPRESSION> then <STATEMENTS> | <ELSEIF_PART>
elsif <EXPRESSION> then <STATEMENTS>

<TRYSTMT> ::= try <STATEMENTS> <CATCH_CLAUSE> end try

<CATCH_CLAUSE> ::= catch '(' <TYPE> Id ')' <STATEMENTS> | catch '('
<TYPE> Id ')' <STATEMENTS> <CATCH_CLAUSE>

<OUTPUTSTMT> ::= output '<<' <EXPRESSION>
               | output '<<' StringLiteral
               | output '<<' CharLiteral

<INPUTSTMT> ::= input '>>' <NAME>

!! Types

<TYPE>               ::=  <STRUCTURE_TYPE>
                     | <PRIMITIVE_TYPE>
                     | <ARRAY_TYPE>

<PRIMITIVE_TYPE>     ::= integer
                     | boolean

<STRUCTURE_TYPE>     ::= Id

<ARRAY_TYPE>         ::= <STRUCTURE_TYPE> '[]'
                     | <PRIMITIVE_TYPE> '[]'



!!! Fields declaration

<ACCESS_SPEC>        ::= private
                     | protected
                     | public

<FIELD_DECL> ::= <ACCESS_SPEC> <TYPE> <FIELD_DECLLIST>';'

<FIELD_DECLLIST> ::= Id | <FIELD_DECLLIST> ',' Id

!!! Functions declaration
```

```
!!
!! Function stuff.
!!

<METHOD> ::= method <M_TYPE> <METHOD_ID>'('<PARAMETERS>')' is <BODY>
Id
            | method <M_TYPE> <METHOD_ID> '(' ')' is <BODY> Id

<METHOD_DECL> ::= <ACCESS_SPEC> method <M_TYPE> Id '('
<PARAMETERS_DECL> ')' ';'
                | <ACCESS_SPEC> method <M_TYPE> Id '(' ')' ';'

<METHOD_ID> ::= Id'::'Id
              | Id

<M_TYPE> ::= <TYPE>
           | void

<PARAMETERS>               ::= <PARAMETER> | <PARAMETERS> ','
<PARAMETER>

<PARAMETER>                ::= <TYPE> Id | ref <TYPE> Id

<PARAMETERS_DECL> ::= <PARAMETER_DECL>
                    | <PARAMETERS_DECL> ',' <PARAMETER_DECL>

<PARAMETER_DECL>  ::= <TYPE> Id
                   | <TYPE>


<FUNCTION_CALL>     ::= call <NAME> '(' ')'
                    | call <NAME> '(' <ARGLIST> ')'

!!! CLASS declaration

<CLASS> ::= class Id <SUPER_CLASS> is <CLASS_MEMBERLIST> end Id
        | class Id is <CLASS_MEMBERLIST> end Id

<CLASS_MEMBERLIST> ::= <CLASS_MEMBER> | <CLASS_MEMBERLIST>
<CLASS_MEMBER>

<CLASS_MEMBER> ::= <FIELD_DECL>
                 | <METHOD_DECL>

<SUPER_CLASS> ::= extends Id
```

2.3 Результат лексического анализа

В результате лексического анализа для набранного кода языка Cool генерируется XML – файл с найденными токенами, которые в дальнейшем выводятся в графическом пользовательском интерфейсе пользователя.

Для примера текста программы на языке Cool из пункта 1 содержимое сгенерированного XML – файла следующее:

```xml
<?xml version="1.0"?>
-<tokens>
      <token value="method" type="method" position="0" line="0"/>
      <token value="void" type="void" position="7" line="0"/>
      <token value="main" type="Id" position="12" line="0"/>
      <token value="(" type="(" position="16" line="0"/>
      <token value=")" type=")" position="17" line="0"/>
      <token value="is" type="is" position="19" line="0"/>
      <token value="declare" type="declare" position="4" line="1"/>
      <token value="integer" type="integer" position="12" line="1"/>
      <token value="i" type="Id" position="20" line="1"/>
      <token value="," type="," position="21" line="1"/>
      <token value="j" type="Id" position="23" line="1"/>
      <token value="," type="," position="24" line="1"/>
      <token value="k" type="Id" position="26" line="1"/>
      <token value="," type="," position="27" line="1"/>
      <token value="m" type="Id" position="29" line="1"/>
      <token value=";" type=";" position="30" line="1"/>
      <token value="begin" type="begin" position="0" line="2"/>
      <token value="input" type="input" position="1" line="3"/>
      <token value=">>" type=">>" position="7" line="3"/>
      <token value="i" type="Id" position="10" line="3"/>
      <token value=";" type=";" position="11" line="3"/>
      <token value="input" type="input" position="1" line="4"/>
      <token value=">>" type=">>" position="7" line="4"/>
      <token value="j" type="Id" position="10" line="4"/>
      <token value=";" type=";" position="11" line="4"/>
      <token value="k" type="Id" position="1" line="5"/>
      <token value="=" type="=" position="3" line="5"/>
      <token value="call" type="call" position="5" line="5"/>
      <token value="summ" type="Id" position="10" line="5"/>
      <token value="(" type="(" position="14" line="5"/>
      <token value="i" type="Id" position="15" line="5"/>
      <token value="," type="," position="16" line="5"/>
      <token value="j" type="Id" position="18" line="5"/>
      <token value=")" type=")" position="19" line="5"/>
      <token value=";" type=";" position="20" line="5"/>
      <token value="output" type="output" position="1" line="6"/>
      <token value="<<" type="<<" position="8" line="6"/>
      <token value=""SUMM result:"" type="StringLiteral" position="11" line="6"/>
      <token value=";" type=";" position="25" line="6"/>
      <token value="output" type="output" position="1" line="7"/>
      <token value="<<" type="<<" position="8" line="7"/>
      <token value="k" type="Id" position="11" line="7"/>
      <token value=";" type=";" position="12" line="7"/>
      <token value="k" type="Id" position="1" line="9"/>
      <token value="=" type="=" position="3" line="9"/>
      <token value="call" type="call" position="5" line="9"/>
      <token value="substract" type="Id" position="10" line="9"/>
      <token value="(" type="(" position="19" line="9"/>
      <token value="i" type="Id" position="20" line="9"/>
```

```
	<token value="," type="," position="21" line="9"/>
	<token value="j" type="Id" position="23" line="9"/>
	<token value=")" type=")" position="24" line="9"/>
	<token value=";" type=";" position="25" line="9"/>
	<token value="output" type="output" position="1" line="10"/>
	<token value="<<" type="<<" position="8" line="10"/>
	<token  value=""Substract  result:""  type="StringLiteral"  position="11"
line="10"/>
	<token value=";" type=";" position="30" line="10"/>
	<token value="output" type="output" position="1" line="11"/>
	<token value="<<" type="<<" position="8" line="11"/>
	<token value="k" type="Id" position="11" line="11"/>
	<token value=";" type=";" position="12" line="11"/>
	<token value="call" type="call" position="1" line="13"/>
	<token value="sub" type="Id" position="6" line="13"/>
	<token value="(" type="(" position="9" line="13"/>
	<token value="i" type="Id" position="10" line="13"/>
	<token value="," type="," position="11" line="13"/>
	<token value="j" type="Id" position="13" line="13"/>
	<token value="," type="," position="14" line="13"/>
	<token value="ref" type="ref" position="16" line="13"/>
	<token value="k" type="Id" position="20" line="13"/>
	<token value=")" type=")" position="21" line="13"/>
	<token value=";" type=";" position="22" line="13"/>
	<token value="output" type="output" position="1" line="14"/>
	<token value="<<" type="<<" position="8" line="14"/>
	<token  value=""Substract  result:""  type="StringLiteral"  position="11"
line="14"/>
	<token value=";" type=";" position="30" line="14"/>
	<token value="output" type="output" position="1" line="15"/>
	<token value="<<" type="<<" position="8" line="15"/>
	<token value="k" type="Id" position="11" line="15"/>
	<token value=";" type=";" position="12" line="15"/>
	<token value="end" type="end" position="0" line="16"/>
	<token value="main" type="Id" position="4" line="16"/>
	<token value="method" type="method" position="0" line="18"/>
	<token value="integer" type="integer" position="7" line="18"/>
	<token value="summ" type="Id" position="15" line="18"/>
	<token value="(" type="(" position="19" line="18"/>
	<token value="integer" type="integer" position="20" line="18"/>
	<token value="a" type="Id" position="28" line="18"/>
	<token value="," type="," position="29" line="18"/>
	<token value="integer" type="integer" position="31" line="18"/>
	<token value="b" type="Id" position="39" line="18"/>
	<token value=")" type=")" position="40" line="18"/>
	<token value="is" type="is" position="42" line="18"/>
	<token value="begin" type="begin" position="0" line="19"/>
	<token value="return" type="return" position="1" line="20"/>
	<token value="a" type="Id" position="8" line="20"/>
	<token value="+" type="+" position="10" line="20"/>
	<token value="b" type="Id" position="12" line="20"/>
	<token value=";" type=";" position="13" line="20"/>
	<token value="end" type="end" position="0" line="21"/>
	<token value="main" type="Id" position="4" line="21"/>
	<token value="method" type="method" position="0" line="23"/>
	<token value="integer" type="integer" position="7" line="23"/>
```

```
      <token value="substract" type="Id" position="15" line="23"/>
      <token value="(" type="(" position="24" line="23"/>
      <token value="integer" type="integer" position="25" line="23"/>
      <token value="a" type="Id" position="33" line="23"/>
      <token value="," type="," position="34" line="23"/>
      <token value="integer" type="integer" position="36" line="23"/>
      <token value="b" type="Id" position="44" line="23"/>
      <token value=")" type=")" position="45" line="23"/>
      <token value="is" type="is" position="47" line="23"/>
      <token value="begin" type="begin" position="0" line="24"/>
      <token value="return" type="return" position="1" line="25"/>
      <token value="a" type="Id" position="8" line="25"/>
      <token value="-" type="-" position="10" line="25"/>
      <token value="b" type="Id" position="12" line="25"/>
      <token value=";" type=";" position="13" line="25"/>
      <token value="end" type="end" position="0" line="26"/>
      <token value="main" type="Id" position="4" line="26"/>
      <token value="method" type="method" position="0" line="28"/>
      <token value="void" type="void" position="7" line="28"/>
      <token value="sub" type="Id" position="12" line="28"/>
      <token value="(" type="(" position="15" line="28"/>
      <token value="integer" type="integer" position="16" line="28"/>
      <token value="a" type="Id" position="24" line="28"/>
      <token value="," type="," position="25" line="28"/>
      <token value="integer" type="integer" position="27" line="28"/>
      <token value="b" type="Id" position="35" line="28"/>
      <token value="," type="," position="36" line="28"/>
      <token value="ref" type="ref" position="38" line="28"/>
      <token value="integer" type="integer" position="42" line="28"/>
      <token value="c" type="Id" position="50" line="28"/>
      <token value=")" type=")" position="51" line="28"/>
      <token value="is" type="is" position="53" line="28"/>
      <token value="begin" type="begin" position="0" line="29"/>
      <token value="c" type="Id" position="1" line="30"/>
      <token value="=" type="=" position="3" line="30"/>
      <token value="a" type="Id" position="5" line="30"/>
      <token value="-" type="-" position="7" line="30"/>
      <token value="b" type="Id" position="9" line="30"/>
      <token value=";" type=";" position="10" line="30"/>
      <token value="return" type="return" position="1" line="31"/>
      <token value=";" type=";" position="7" line="31"/>
      <token value="end" type="end" position="0" line="32"/>
      <token value="main" type="Id" position="4" line="32"/>
</tokens>
```

3 Лабораторная работа №6. Синтаксический анализ

3.1 Постановка задачи

В процессе синтаксического анализа линейная последовательность лексем (токенов) языка сопоставляется с его формальной грамматикой.

3.2 Результат синтаксического анализа

Результатом обычно является дерево разбора (синтаксическое дерево).

Для примера текста программы на языке Cool из пункта 1 дерево разбора следующее:

```
No errors.
The parse tree is:

+-<PROGRAM> ::= <PROGRAM> <METHOD>
| +-<PROGRAM> ::= <PROGRAM> <METHOD>
| | +-<PROGRAM> ::= <PROGRAM> <METHOD>
| | | +-<PROGRAM> ::= <METHOD>
| | | | +-<METHOD> ::= method <M_TYPE> <METHOD_ID> '(' ')' is <BODY> Id
| | | | | +-method
| | | | | +-<M_TYPE> ::= void
| | | | | | +-void
| | | | | +-<METHOD_ID> ::= Id
| | | | | | +-main
| | | | | +-(
| | | | | +-)
| | | | | +-is
| | | | | +-<BODY> ::= <BLOCK>
| | | | | | +-<BLOCK> ::= <VARDECS> begin <STATEMENTS> end
| | | | | | | +-<VARDECS> ::= declare <VARDECLIST>
| | | | | | | | +-declare
| | | | | | | | +-<VARDECLIST> ::= <TYPE> Id <VAR_TYPELIST> ';'
| | | | | | | | | +-<TYPE> ::= <PRIMITIVE_TYPE>
| | | | | | | | | | +-<PRIMITIVE_TYPE> ::= integer
| | | | | | | | | | | +-integer
| | | | | | | | | +-i
| | | | | | | | | +-<VAR_TYPELIST> ::= <VAR_TYPELIST> ',' Id
| | | | | | | | | | +-<VAR_TYPELIST> ::= <VAR_TYPELIST> ',' Id
| | | | | | | | | | | +-<VAR_TYPELIST> ::= ',' Id
| | | | | | | | | | | | +-,
| | | | | | | | | | | | +-j
| | | | | | | | | | | +-,
| | | | | | | | | | | +-k
| | | | | | | | | | +-,
| | | | | | | | | | +-m
| | | | | | | | | +-;
| | | | | | | +-begin
| | | | | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | | | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | | | | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | | | | | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | | | | | | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | | | | | | | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | | | | | | | | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | | | | | | | | | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | | | | | | | | | | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | | | | | | | | | | | | | +-<STATEMENTS> ::= <STATEMENT>
| | | | | | | | | | | | | | | | | | +-<STATEMENT> ::= <INPUTSTMT> ';'
| | | | | | | | | | | | | | | | | | | +-<INPUTSTMT> ::= input '>>' <NAME>
| | | | | | | | | | | | | | | | | | | | +-input
| | | | | | | | | | | | | | | | | | | | +->>
| | | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | | +-i
| | | | | | | | | | | | | | | | | | | +-;
| | | | | | | | | | | | | | | | | +-<STATEMENT> ::= <INPUTSTMT> ';'
| | | | | | | | | | | | | | | | | | +-<INPUTSTMT> ::= input '>>' <NAME>
| | | | | | | | | | | | | | | | | | | +-input
| | | | | | | | | | | | | | | | | | | +->>
| | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | +-j
| | | | | | | | | | | | | | | | | | +-;
| | | | | | | | | | | | | | | | +-<STATEMENT> ::= <ASSIGNMENT> ';'
| | | | | | | | | | | | | | | | | +-<ASSIGNMENT> ::= <NAME> '=' <EXPRESSION>
| | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | +-k
| | | | | | | | | | | | | | | | | | +-=
| | | | | | | | | | | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <FUNCTION_CALL>
| | | | | | | | | | | | | | | | | | | | | | | | +-<FUNCTION_CALL> ::= call <NAME> '(' <ARGLIST> ')'
| | | | | | | | | | | | | | | | | | | | | | | | | +-call
| | | | | | | | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | | | | | | | +-summ
| | | | | | | | | | | | | | | | | | | | | | | | | +-(
| | | | | | | | | | | | | | | | | | | | | | | | | +-<ARGLIST> ::= <ARGLIST> ',' <ARGUMENT>
| | | | | | | | | | | | | | | | | | | | | | | | | | +-<ARGLIST> ::= <ARGUMENT>
| | | | | | | | | | | | | | | | | | | | | | | | | | | +-<ARGUMENT> ::= <EXPRESSION>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-
<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<NAME> ::=
Id
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-i
| | | | | | | | | | | | | | | | | | | | | | | | | | +-,
| | | | | | | | | | | | | | | | | | | | | | | | | | +-<ARGUMENT> ::= <EXPRESSION>
| | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION> ::=
<EXPRESSION_TERM>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::=
<EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR>
::= <EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY>
::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY>
::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-
<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-j
| | | | | | | | | | | | | | | | | | | | | | | | | +-)
| | | | | | | | | | | | | | | | | +-;
| | | | | | | | | | | | | | | +-<STATEMENT> ::= <OUTPUTSTMT> ';'
| | | | | | | | | | | | | | | | +-<OUTPUTSTMT> ::= output '<<' StringLiteral
| | | | | | | | | | | | | | | | | +-output
| | | | | | | | | | | | | | | | | +-<<
| | | | | | | | | | | | | | | | | +-"SUMM result:"
| | | | | | | | | | | | | | | | +-;
| | | | | | | | | | | | | | +-<STATEMENT> ::= <OUTPUTSTMT> ';'
| | | | | | | | | | | | | | | +-<OUTPUTSTMT> ::= output '<<' <EXPRESSION>
| | | | | | | | | | | | | | | | +-output
| | | | | | | | | | | | | | | | +-<<
| | | | | | | | | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::=
<EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | | | | +-k
| | | | | | | | | | | | | | | +-;
| | | | | | | | | | | | | +-<STATEMENT> ::= <ASSIGNMENT> ';'
| | | | | | | | | | | | | | +-<ASSIGNMENT> ::= <NAME> '=' <EXPRESSION>
| | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | +-k
| | | | | | | | | | | | | | | +-=
| | | | | | | | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::=
<EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <FUNCTION_CALL>
| | | | | | | | | | | | | | | | | | | | | +-<FUNCTION_CALL> ::= call <NAME> '('
<ARGLIST> ')'
| | | | | | | | | | | | | | | | | | | | | | +-call
| | | | | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | | | | +-substract
| | | | | | | | | | | | | | | | | | | | | | +-(
| | | | | | | | | | | | | | | | | | | | | | +-<ARGLIST> ::= <ARGLIST> ','
<ARGUMENT>
| | | | | | | | | | | | | | | | | | | | | | | +-<ARGLIST> ::= <ARGUMENT>
| | | | | | | | | | | | | | | | | | | | | | | | +-<ARGUMENT> ::= <EXPRESSION>
| | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION> ::=
<EXPRESSION_TERM>
| | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::=
<EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::=
<EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::=
<EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::=
<EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY>
::= <NAME>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-i
| | | | | | | | | | | | | | | | | | | | | | | +-,
| | | | | | | | | | | | | | | | | | | | | | | +-<ARGUMENT> ::= <EXPRESSION>
| | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION> ::=
<EXPRESSION_TERM>
| | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::=
<EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::=
<EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::=
<EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::=
<EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY>
::= <NAME>
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | +-j
| | | | | | | | | | | | | | | | | | | | | | +-)
| | | | | | | | | | | | | | +-;
| | | | | | | | | | | | +-<STATEMENT> ::= <OUTPUTSTMT> ';'
| | | | | | | | | | | | | +-<OUTPUTSTMT> ::= output '<<' StringLiteral
| | | | | | | | | | | | | | +-output
| | | | | | | | | | | | | | +-<<
| | | | | | | | | | | | | | +-"Substract result:"
| | | | | | | | | | | | | +-;
| | | | | | | | | | | +-<STATEMENT> ::= <OUTPUTSTMT> ';'
| | | | | | | | | | | | +-<OUTPUTSTMT> ::= output '<<' <EXPRESSION>
| | | | | | | | | | | | | +-output
| | | | | | | | | | | | | +-<<
| | | | | | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | +-k
| | | | | | | | | | | | +-;
| | | | | | | | | | +-<STATEMENT> ::= <FUNCTION_CALL> ';'
| | | | | | | | | | | +-<FUNCTION_CALL> ::= call <NAME> '(' <ARGLIST> ')'
| | | | | | | | | | | | +-call
| | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | +-sub
| | | | | | | | | | | | +-(
| | | | | | | | | | | | +-<ARGLIST> ::= <ARGLIST> ',' <ARGUMENT>
| | | | | | | | | | | | | +-<ARGLIST> ::= <ARGLIST> ',' <ARGUMENT>
| | | | | | | | | | | | | | +-<ARGLIST> ::= <ARGUMENT>
| | | | | | | | | | | | | | | +-<ARGUMENT> ::= <EXPRESSION>
| | | | | | | | | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::=
<EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | | | | +-i
| | | | | | | | | | | | | | +-,
| | | | | | | | | | | | | | +-<ARGUMENT> ::= <EXPRESSION>
| | | | | | | | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::=
<EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | | | +-j
| | | | | | | | | | | | | +-,
| | | | | | | | | | | | | +-<ARGUMENT> ::= ref <EXPRESSION>
| | | | | | | | | | | | | | +-ref
| | | | | | | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | | | | | +-k
| | | | | | | | | | | | +-)
| | | | | | | | | | | +-;
| | | | | | | | | +-<STATEMENT> ::= <OUTPUTSTMT> ';'
| | | | | | | | | | +-<OUTPUTSTMT> ::= output '<<' StringLiteral
| | | | | | | | | | | +-output
| | | | | | | | | | | +-<<
| | | | | | | | | | | +-"Substract result:"
| | | | | | | | | | +-;
| | | | | | | | +-<STATEMENT> ::= <OUTPUTSTMT> ';'
| | | | | | | | | +-<OUTPUTSTMT> ::= output '<<' <EXPRESSION>
| | | | | | | | | | +-output
| | | | | | | | | | +-<<
| | | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | | +-k
| | | | | | | | | +-;
| | | | | | | +-end
| | | | | +-main
| | | +-<METHOD> ::= method <M_TYPE> <METHOD_ID> '(' <PARAMETERS> ')' is <BODY> Id
| | | | +-method
| | | | +-<M_TYPE> ::= <TYPE>
| | | | | +-<TYPE> ::= <PRIMITIVE_TYPE>
| | | | | | +-<PRIMITIVE_TYPE> ::= integer
| | | | | | | +-integer
| | | | +-<METHOD_ID> ::= Id
| | | | | +-summ
| | | | +-(
| | | | +-<PARAMETERS> ::= <PARAMETERS> ',' <PARAMETER>
| | | | | +-<PARAMETERS> ::= <PARAMETER>
| | | | | | +-<PARAMETER> ::= <TYPE> Id
| | | | | | | +-<TYPE> ::= <PRIMITIVE_TYPE>
| | | | | | | | +-<PRIMITIVE_TYPE> ::= integer
| | | | | | | | | +-integer
| | | | | | | +-a
| | | | | +-,
| | | | | +-<PARAMETER> ::= <TYPE> Id
| | | | | | +-<TYPE> ::= <PRIMITIVE_TYPE>
| | | | | | | +-<PRIMITIVE_TYPE> ::= integer
| | | | | | | | +-integer
| | | | | | +-b
| | | | +-)
| | | | +-is
| | | | +-<BODY> ::= <BLOCK>
| | | | | +-<BLOCK> ::= begin <STATEMENTS> end
| | | | | | +-begin
| | | | | | +-<STATEMENTS> ::= <STATEMENT>
| | | | | | | +-<STATEMENT> ::= return <EXPRESSION> ';'
| | | | | | | | +-return
| | | | | | | | +-<EXPRESSION> ::= <EXPRESSION> '+' <EXPRESSION_TERM>
| | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | +-a
| | | | | | | | | +-+
| | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | +-b
| | | | | | | | +-;
| | | | | | +-end
| | | | +-main
| | +-<METHOD> ::= method <M_TYPE> <METHOD_ID> '(' <PARAMETERS> ')' is <BODY> Id
| | | +-method
| | | +-<M_TYPE> ::= <TYPE>
| | | | +-<TYPE> ::= <PRIMITIVE_TYPE>
| | | | | +-<PRIMITIVE_TYPE> ::= integer
| | | | | | +-integer
| | | +-<METHOD_ID> ::= Id
| | | | +-substract
| | | +-(
| | | +-<PARAMETERS> ::= <PARAMETERS> ',' <PARAMETER>
| | | | +-<PARAMETERS> ::= <PARAMETER>
| | | | | +-<PARAMETER> ::= <TYPE> Id
| | | | | | +-<TYPE> ::= <PRIMITIVE_TYPE>
| | | | | | | +-<PRIMITIVE_TYPE> ::= integer
| | | | | | | | +-integer
| | | | | | +-a
| | | | +-,
| | | | +-<PARAMETER> ::= <TYPE> Id
| | | | | +-<TYPE> ::= <PRIMITIVE_TYPE>
| | | | | | +-<PRIMITIVE_TYPE> ::= integer
| | | | | | | +-integer
| | | | | +-b
| | | +-)
| | | +-is
| | | +-<BODY> ::= <BLOCK>
| | | | +-<BLOCK> ::= begin <STATEMENTS> end
| | | | | +-begin
| | | | | +-<STATEMENTS> ::= <STATEMENT>
| | | | | | +-<STATEMENT> ::= return <EXPRESSION> ';'
| | | | | | | +-return
| | | | | | | +-<EXPRESSION> ::= <EXPRESSION> - <EXPRESSION_TERM>
| | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | +-a
| | | | | | | | +--
| | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | +-b
| | | | | | | +-;
| | | | | +-end
| | | +-main
| +-<METHOD> ::= method <M_TYPE> <METHOD_ID> '(' <PARAMETERS> ')' is <BODY> Id
| | +-method
| | +-<M_TYPE> ::= void
| | | +-void
| | +-<METHOD_ID> ::= Id
| | | +-sub
| | +-(
| | +-<PARAMETERS> ::= <PARAMETERS> ',' <PARAMETER>
| | | +-<PARAMETERS> ::= <PARAMETERS> ',' <PARAMETER>
| | | | +-<PARAMETERS> ::= <PARAMETER>
| | | | | +-<PARAMETER> ::= <TYPE> Id
| | | | | | +-<TYPE> ::= <PRIMITIVE_TYPE>
| | | | | | | +-<PRIMITIVE_TYPE> ::= integer
| | | | | | | | +-integer
| | | | | | +-a
| | | | +-,
| | | | +-<PARAMETER> ::= <TYPE> Id
| | | | | +-<TYPE> ::= <PRIMITIVE_TYPE>
| | | | | | +-<PRIMITIVE_TYPE> ::= integer
| | | | | | | +-integer
| | | | | +-b
| | | +-,
| | | +-<PARAMETER> ::= ref <TYPE> Id
| | | | +-ref
| | | | +-<TYPE> ::= <PRIMITIVE_TYPE>
| | | | | +-<PRIMITIVE_TYPE> ::= integer
| | | | | | +-integer
| | | | +-c
| | +-)
| | +-is
| | +-<BODY> ::= <BLOCK>
| | | +-<BLOCK> ::= begin <STATEMENTS> end
| | | | +-begin
| | | | +-<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
| | | | | +-<STATEMENTS> ::= <STATEMENT>
| | | | | | +-<STATEMENT> ::= <ASSIGNMENT> ';'
| | | | | | | +-<ASSIGNMENT> ::= <NAME> '=' <EXPRESSION>
| | | | | | | | +-<NAME> ::= Id
| | | | | | | | | +-c
| | | | | | | | +-=
| | | | | | | | +-<EXPRESSION> ::= <EXPRESSION> - <EXPRESSION_TERM>
| | | | | | | | | +-<EXPRESSION> ::= <EXPRESSION_TERM>
| | | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | | | +-a
| | | | | | | | | +--
| | | | | | | | | +-<EXPRESSION_TERM> ::= <EXPRESSION_FACTOR>
| | | | | | | | | | +-<EXPRESSION_FACTOR> ::= <EXPRESSION_BINARY>
| | | | | | | | | | | +-<EXPRESSION_BINARY> ::= <EXPRESSION_UNARY>
| | | | | | | | | | | | +-<EXPRESSION_UNARY> ::= <EXPRESSION_PRIMARY>
| | | | | | | | | | | | | +-<EXPRESSION_PRIMARY> ::= <NAME>
| | | | | | | | | | | | | | +-<NAME> ::= Id
| | | | | | | | | | | | | | +-b
| | | | | | | +-;
| | | | | +-<STATEMENT> ::= return ';'
| | | | | | +-return
| | | | | | +-;
| | | | +-end
| | +-main
```

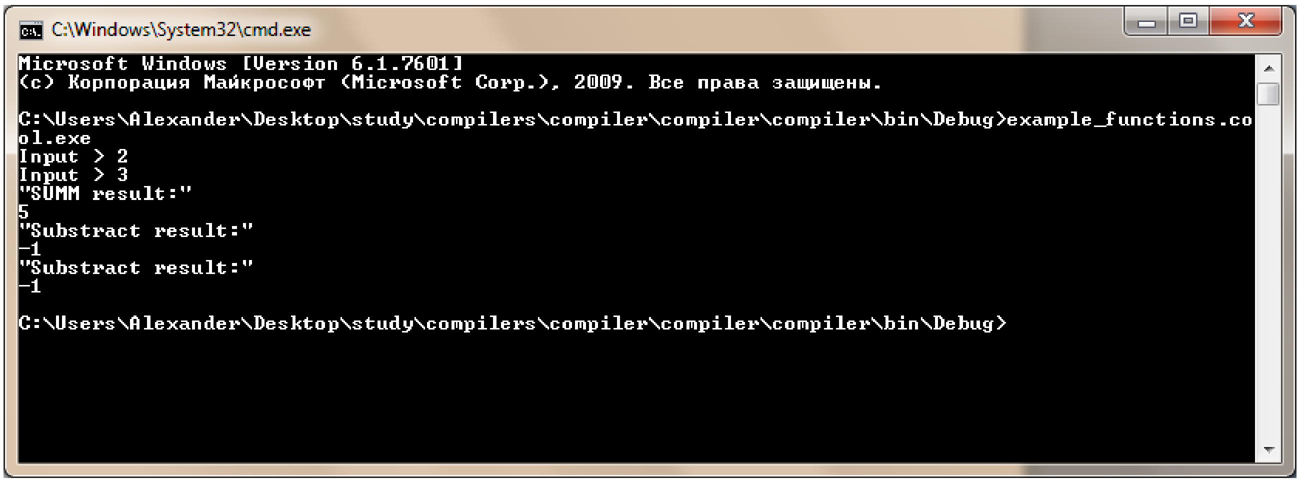
4 Лабораторная работа №7. Генератор кода

В качестве виртуальной машины в работе используется виртуальная машина .Net.

Для рассматриваемого примера получаем следующий код на языке MSIL:

```
//  Microsoft (R) .NET Framework IL Disassembler.  Version 4.0.30319.1
//  Copyright (c) Microsoft Corporation.  All rights reserved.



// Metadata version: v4.0.30319
.assembly extern mscorlib
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )                         // .z\V.4..
  .ver 4:0:0:0
}
.assembly 'Sharp Code Assembly'
{
  .hash algorithm 0x00008004
  .ver 0:0:0:0
}
.module example_functions.cool
// MVID: {70A1E1ED-2468-453E-A6B0-88D5F6A44D23}
.imagebase 0x00400000
.file alignment 0x00000200
.stackreserve 0x00100000
.subsystem 0x0003       // WINDOWS_CUI
.corflags 0x00000001    //  ILONLY
// Image base: 0x0000000001D00000


// ================== GLOBAL METHODS =========================

.method public static void  main() cil managed
{
  //
  .maxstack  0
  .locals init (int32 V_0,
           int32 V_1,
           int32 V_2,
           int32 V_3)
  IL_0000:  ldstr      "Input > "
  IL_0005:  call       void [mscorlib]System.Console::Write(string)
  IL_000a:  call       string [mscorlib]System.Console::ReadLine()
  IL_000f:  call       int32 [mscorlib]System.Int32::Parse(string)
  IL_0014:  stloc.0
  IL_0015:  ldstr      "Input > "
  IL_001a:  call       void [mscorlib]System.Console::Write(string)
  IL_001f:  call       string [mscorlib]System.Console::ReadLine()
  IL_0024:  call       int32 [mscorlib]System.Int32::Parse(string)
  IL_0029:  stloc.1
  IL_002a:  ldloc.0
  IL_002b:  ldloc.1
  IL_002c:  call       int32 summ(int32,
                                  int32)
  IL_0031:  stloc.2
  IL_0032:  ldstr      "\"SUMM result:\""
  IL_0037:  call       void [mscorlib]System.Console::WriteLine(string)
  IL_003c:  ldloc.2
  IL_003d:  call       void [mscorlib]System.Console::WriteLine(int32)
  IL_0042:  ldloc.0
  IL_0043:  ldloc.1
  IL_0044:  call       int32 substract(int32,
                                       int32)
  IL_0049:  stloc.2
  IL_004a:  ldstr      "\"Substract result:\""
  IL_004f:  call       void [mscorlib]System.Console::WriteLine(string)
  IL_0054:  ldloc.2
  IL_0055:  call       void [mscorlib]System.Console::WriteLine(int32)
  IL_005a:  ldloc.0
  IL_005b:  ldloc.1
  IL_005c:  ldloca.s   V_2
  IL_005e:  call       void 'sub'(int32,
                                  int32,
                                  int32&)
  IL_0063:  ldstr      "\"Substract result:\""
  IL_0068:  call       void [mscorlib]System.Console::WriteLine(string)
  IL_006d:  ldloc.2
  IL_006e:  call       void [mscorlib]System.Console::WriteLine(int32)
} // end of global method main

.method public static int32  summ(int32 a,
                                  int32 b) cil managed
{
  //
  .maxstack  2
  IL_0000:  ldarg.s    a
  IL_0002:  nop
  IL_0003:  nop
  IL_0004:  nop
  IL_0005:  ldarg.s    b
  IL_0007:  nop
  IL_0008:  nop
  IL_0009:  nop
  IL_000a:  add
  IL_000b:  ret
} // end of global method summ

.method public static int32  substract(int32 a,
                                       int32 b) cil managed
{
  //
  .maxstack  2
  IL_0000:  ldarg.s    a
  IL_0002:  nop
  IL_0003:  nop
  IL_0004:  nop
  IL_0005:  ldarg.s    b
  IL_0007:  nop
  IL_0008:  nop
  IL_0009:  nop
  IL_000a:  sub
  IL_000b:  ret
} // end of global method substract

.method public static void  'sub'(int32 a,
                                  int32 b,
                                  int32& c) cil managed
{
  //
  .maxstack  3
  IL_0000:  ldarg.s    c
  IL_0002:  nop
  IL_0003:  nop
  IL_0004:  nop
  IL_0005:  ldarg.s    a
  IL_0007:  nop
  IL_0008:  nop
  IL_0009:  nop
  IL_000a:  ldarg.s    b
  IL_000c:  nop
  IL_000d:  nop
  IL_000e:  nop
  IL_000f:  sub
  IL_0010:  stind.i4
  IL_0011:  ret
} // end of global method 'sub'

.method public static void  Main() cil managed
{
  .entrypoint
  //
  .maxstack  0
  IL_0000:  call       void main()
} // end of global method Main


// =============================================================


// =============== CLASS MEMBERS DECLARATION ===================

.class private auto ansi Global
       extends [mscorlib]System.Object
{
  .method public specialname rtspecialname
          instance void  .ctor() cil managed
  {
    //
    .maxstack  2
    IL_0000:  ldarg.0
    IL_0001:  call       instance void [mscorlib]System.Object::.ctor()
    IL_0006:  ret
  } // end of method Global::.ctor
} // end of class Global


// ===============================================================

//
```

5 Лабораторная работа №8. Исполнение кода



C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\Alexander\Desktop\study\compilers\compiler\compiler\compiler\bin\Debug>example_functions.co
ol.exe
Input > 2
Input > 3
"SUMM result:"
5
"Substract result:"
-1
"Substract result:"
-1

C:\Users\Alexander\Desktop\study\compilers\compiler\compiler\compiler\bin\Debug>
```