TRABALHO nº2: especificação da sintaxe da linguagem 2013.1

Construa **UMA ÚNICA** gramática, usando a notação BNF, que defina as regras sintáticas para escrever um programa na linguagem 2013.1, conforme as descrições abaixo. Considere que:

a) os símbolos terminais são os *tokens* especificados no trabalho nº1;
b) uma vez descrita a forma geral de uma estrutura sintática, a mesma "vale" para toda a linguagem, por exemplo, na forma geral de um programa foi descrita a `<lista comandos>`, sendo essa descrição válida para o comando de seleção e os comandos de repetição;
c) as regras sintáticas para o não-terminal **`<expressão>`** (expressões aritméticas, lógicas e relacionais) serão especificadas posteriormente.

---

forma geral de um programa

```
main module [ <lista de módulos> <lista de variáveis> <lista comandos> ]
```

onde:

- `<lista de módulos>` pode conter zero ou mais `<módulo>` s.
- em `<lista comandos>` podem ocorrer um ou mais `<comando>`s. Os `<comando>`s podem ser de atribuição, de entrada, de saída, de seleção ou de repetição

---

forma geral do `<módulo>`

```
module <identificador> <parâmetros>
[ <lista de variáveis> <lista comandos do módulo>  return <expressão>; ]
```

onde:

- `<identificador>` pode ser identificador de `int` ou um identificador de `float` ou um identificador de `string` ou um identificador de `bool`
- `<parâmetros>` pode ocorrer zero ou uma vez
- `<parâmetros>` tem a seguinte forma geral `:<lista de identificadores>`
- em `<lista de identificadores>` deve existir, no mínimo, um `<identificador>` (identificador de `int` ou um identificador de `float` ou um identificador de `string` ou um identificador de `bool`). Caso existam mais identificadores, os mesmos são separados uns dos outros por vírgula (,)
- em `<lista comandos do módulo>` podem ocorrer zero ou mais `<comando>`s. Os `<comando>`s podem ser de atribuição, de entrada, de saída, de seleção ou de repetição

---

forma geral da `<lista de variáveis>`

```
:<lista de identificadores>;
```

onde:

- `:<lista de identificadores>;` pode ocorrer zero ou mais vezes

---

forma geral do comando de atribuição

```
<identificador> <- <expressão>;
```

---

forma geral do comando de entrada de dados

```
in ( <lista de identificadores> );
```

---

forma geral do comando de saída de dados

```
out ( <lista de expressões> );
```

onde:

- em `<lista de expressões>` deve existir, no mínimo, uma `<expressão>`. Caso existam mais expressões, as mesmas são separadas umas das outras por vírgula (,)

---

forma geral do comando de seleção

```
if ( <expressão> ) isTrueDo : [ <lista comandos> ] isFalseDo : [ <lista comandos> ]
```

onde:

- a cláusula **`isFalseDo : [`** `<lista comandos>` **`]`** é opcional

---

forma geral do comando de repetição

```
while ( <expressão> ) isTrueDo : [ <lista comandos> ]
```

ou

```
while ( <expressão> ) isFalseDo : [ <lista comandos> ]
```