In this document will be shown the result of the execution of the Parser with different input files. After every execution, a comment on the Parser output clarifies the Parser execution.

The input programs used in this document are accessible in the **sample_programs** folder.

Well Formatted Inputs

Expression Evaluation

Input:

sample_programs > well_formatted > expr_evaluation.txt

```
1   int a = 12;
2   int b = 24;
3   double c = 24;
4
5   print(a/b);
6   print(a/c);
7   print((a/b) >= ((a/c)));
```

Output:

default.out

```
1   Type: integer, Value: 0
2   Type: double, Value: 0.500000
3   Type: boolean, Value: false
```

Comment on output:
This sample program shows that integer division is treated differently from double division, and provides a boolean expression evaluation example.

Variable Redefinition

Input:

sample_programs > well_formatted > var_redef.txt

```
1   int a = 12;
2   print a;
3
4   while(4>2) {
5     a = 21;
6     print(a);
7     int a = 30;
8     print a;
9   }
10  print(a);
11
```

Output:

default.out

```
1   Type: integer, Value: 12
2   Type: integer, Value: 21
3   Type: integer, Value: 30
4   Type: integer, Value: 21
5
```

Comment on output:
As it could be noticed, the assignment "*a = 21*" (line 5) reassigns the value associated to the variable "*a*" declared in the outer parent scope. Then, at line 7, the variable a is redeclared and this is possible, since the block of code enclosed by the curly brackets is a sub scope of the parent one. Therefore, variable redefinition is allowed. It could be noticed that the variable "*a*" is redeclared, since the version of "a" in the sub scope contains the vale 30 (as shown by the "print a" statement at line 8), whereas the version of "*a*" in the parent scope contains the value 21.

Conditional Statement

Input:

sample_programs > well_formatted > cond_stmt > a.txt

```
1   int a = -1;
2   if true {
3     int a = 1;
4     int c;
5   }
6   else {
7     print(a);
8     int a = 2;
9     print a;
10  }
11  print a;
```

Output:

default.out

```
1   Type: integer, Value: -1
2   Type: integer, Value: 2
3   Type: integer, Value: -1
4
```

Comment on output:
This example shows that the block statements associated to the if-branch and else-branch are independent from each other. Therefore, the redeclaration of the variable "a" in the if-branch (line 3), does not change influence the variable "a" in the else-branch.

Mixed Code

Input:

sample_programs > well_formatted > complex_code.txt

```
1   int a = 12;
2   double b = -a;
3   boolean c = a == b;
4
5   while(!c) {
6     print(a == (-b));
7     if(a == (-b)) {
8       print(a);
9     }
10    else {
11      print(b);
12    }
13  }
```

Output:

default.out

```
1   Type: boolean, Value: true
2   Type: integer, Value: 12
3   Type: double, Value: -12.000000
```

Comment on output: as it is possible to see, the expression at line 6 evaluates true. This is because an implicit type coercion is done when testing for equality (and in general for each operator) an integer value with a double value.

Badly formatted input

Multiline code-block not enclosed in curly brackets

Input:

sample_programs > badly_formatted > invalid_control_stmt.txt

```
1  if(12<=12.0)
2    print(28*12.0/(3+1));
3  else
4    int a = 0;
5    while(true){
6      print (a);
7    }
```

Output:

```
riccardo_rigoni@MacBook-Pro-di-Riccardo FL-C_finalProject % ./a.out -i sample_programs/badly_formatted/invalid_control_stmt.txt
sample_programs/badly_formatted/invalid_control_stmt.txt:6:12: error: Variable a has not been defined!
    print (a);
~~~~~~~~~~^
riccardo_rigoni@MacBook-Pro-di-Riccardo FL-C_finalProject %
```

Comment on output:
As the error message says, the variable “a” has not been declared. This happened since only the first line of code after the *else-branch* is associated to it. Furthermore, this line of code is considered to be a sub scope of the main one. Therefore, the *while-statement* cannot access the variable *a* declared at line 4.

Variable not declared

Input:

sample_programs > badly_formatted > var_not_decleared.txt

```
1  int a;
2  if(a > b){
3    print a;
4  }
```

Output:

```
riccardo_rigoni@MacBook-Pro-di-Riccardo FL-C_finalProject % ./a.out -i sample_programs/badly_formatted/var_not_decleared.txt
sample_programs/badly_formatted/var_not_decleared.txt:2:8: error: Variable b has not been defined!
if(a > b){
~~~~~~~^
riccardo_rigoni@MacBook-Pro-di-Riccardo FL-C_finalProject %
```

Invalid type assignment

Input:

sample_programs > badly_formatted > invalid_type.txt

```
1   int a = 1;
2   double b = a;
3   print(b);
4   boolean c = a;
5   print(c);
```

Output:

Comment on output:
The declaration at line 2 doesn't fail, since the integer type is compatible with the double type. Therefore, an implicit cast to double of the value contained in a is done prior to assign the value to be.
But the integer and the boolean types are not compatible, therefore, the output error message is generated