

**Ciências
ULisboa**

Relatório do Projeto Aguda Compiler

Milestone 2

Ricardo Miguel Neto Sobral 56332

How to build your compiler

Para esta fase do projeto foi usado Python e um ambiente virtual (venv) para isolamento.

Para além disso foi utilizado Ply (versão 3.11), para tratar de ficheiros essenciais como o lexer e o parser.

É importante indicar que o ficheiro main encontra-se em *src/aguda-cabal/aguda/main.py*.

Posto isto, é necessário correr dois comandos para correr o container Docker:

```
docker-compose build
```

e de seguida:

```
docker-compose run aguda_app bash
```

E agora está tudo pronto para correr o compilador!

How to run a particular test

Para correr um único ficheiro de teste (.agu) precisamos correr o seguinte comando no terminal:

```
python3 src/aguda-cabal/aguda/main.py <path-to-test>.agu
```

Onde path-to-test, é o caminho para o teste específico, vejamos o seguinte exemplo exemplo:

```
(venv) venvricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 src/aguda-cabal/aguda/main.py test/valid/56332_array_sum/array_sum.agu
let arraySum (a) : ((Int[]) -> Int) -> (Int[]) -> Int =
  let i : Int = 0;
  let acc : Int = 0;
  while (i < length(a)) do
    set acc = (acc + a[i]);
    set i = (i + 1)
  acc

let a : Int[] = new Int[5|10];
let _ : Unit = print(arraySum(a))
(venv) venvricardo@MacBook-Pro-de-Ricardo aguda-testing %
```

How to run the whole test suite (valid and invalid programs)

Para correr vários testes temos várias opções:

- run_all_valid_tests.py

Para correr todos os testes validos, isto é, todos os testes dentro da pasta *test/valid*. Para correr este ficheiro usamos o seguinte comando:

```
(venv) venvricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 run_all_valid_tests.py
```

- run_invalid_tests.py

Para correr todos os testes invalidos, isto é, todos os testes dentro das pastas *test/invalid-syntax* e *test/inavlid-semantic*. Para correr este ficheiro usamos o seguinte comando:

```
(venv) venvricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 run_invalid_tests.py
```

Contudo este ficheiro não é muito útil para esta fase do projeto, dado que os compiladores não estão preparados para lidar com erros semânticos. Posto isto foi criado outro ficheiro Python para correr apenas alguns testes inválidos.

- run_invalid_syntax_tests.py

Este ficheiro, corre todos os ficheiros (.agu) dentro da pasta *test/invalid-syntax* que devia conter todos os testes inválidos, que apresentassem apenas erros léxicos e de sintaxe, no entanto após alguma reflexam e analyse, percebeu-se que havia testes dentro desta pasta com erros semânticos e outros que nem sequer tinham erros. Para correr este ficheiro use este comando:

```
Lexical Error: Lexical Error at line 3, column 11: Integer character '1'
(venv) venvricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 run_invalid_syntax_tests.py
```

- run_test_of_student.py

Para um melhor entendimento dos resultados, e para auxiliar a correção de erros no compilador, também foi criado um ficheiro Python, que corre os testes válidos de um aluno, passando o número de aluno como input. É de notar que também funciona para os testes do professor, que começam com tcomp000.

Para correr este ficheiro use este comando:

```
(venv) venvricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 run_test_of_student.py 56332  
=== Running test/valid/56332_factorial_in_aguda/factorial_in_aguda.agu ===
```

How to interpret the testing output (how many tests passed, which failed)

Para interpretar os resultados dos testes é simples. Para os casos em que os testes não passam na avaliação do compilador, é lançado um erro, indicado se o erro é um erro léxico ou sintático, como podemos ver no caso abaixo:

```
=== Running test/invalid-syntax/64854_non_letter_id/non_letter_id.agu ===  
Syntax Error: Syntax error at line 4, column 5 near '1'  
  
=== Running test/invalid-syntax/tcomp000_wrong_comment/wrong_comment.agu ===  
Lexical Error: Lexical error at line 3, column 1: illegal character '#'
```

Para além disso, o compilador também indica a linha e a coluna onde o erro ocorre.

Para os ficheiros .agu que passam na avaliação do compilador, é impresso no terminal o AST do respetivo programa, graças ao ficheiro auxiliar pretty_printer.py. Vejamos o exemplo:



```
=== Running test/valid/56332_simple_sum/simple_sum.agu ===  
  
let sum (a, b) : ((Int, Int) -> Int) -> (Int, Int) -> Int =  
  (a + b)  
  
let _ : Unit = print(sum(10, 5))
```

O conteúdo do ficheiro .agu em questão:

```
Users > ricardo > Desktop > fcul > Técnicas de Compilação > aguda-testing > te  
1  -- Author: 56332, Student Ricardo Sobral  
2  let sum (a, b) : (Int, Int) -> Int = a + b  
3  let _ : Unit = print(sum(10, 5))
```

Para além, disso também foi criado outros dois ficheiros Python para auxiliar na comparação dos resultados do compilador criado pelo professor. Esses ficheiros são:







-run_valid.py

Que corre os testes validos, tal como o ficheiro `run_all_valid_tests.py`, no entanto em vez de apresentar o tipo de erro ou imprimir o AST do programa, limita-se a indicar se o ficheiro `.agu` passa na verificação do compilador, imprimindo  OK, para os testes que passam na verificação, e  ERROR para os testes que não passam na verificação.

Para correr este ficheiro pode usar este comando:

```
(venv) venvricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 run_valid.py
```

Resultando em algo deste tipo:

```
=== Running test/valid/46494_recursive_factorial/recursive_factorial.agu ===  
 OK  
=== Running test/valid/46494_arrays/arrays.agu ===  
 OK  
=== Running test/valid/46494_palindromes/palindromes_check.agu ===  
 ERROR  
=== Running test/valid/46494_string_loop/mixing_string_integers_loops.agu ===  
 OK  
=== Running test/valid/46494_minimum_value/finding_the_minimum.agu ===  
 OK  
=== Running test/valid/46494_power_function/power_func.agu ===  
 OK  
=== Running test/valid/46494_simulate_chess_board/simulate_chess.agu ===  
 ERROR
```

No entanto, o intuito da criação deste ficheiro Python, era usar os resultados (expressos de um modo mais simples) para comparar com os resultados do do ficheiro log do compilador do professor (*test/valid-2015-04-09.log*). Sendo assim, sempre que este *run_valid.py* foi usado, utilizou-se este comando:

```
(venv) venvricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 run_valid.py > meu_teste_output.log
```

Para que o resultado do ficheiro *run_valid.py*, fosse impresso num ficheiro log, denominado *meu_teste_output.log*, que posteriormente é usado para fazer a comparação.

- `compare_with_log.py`

Este ficheiro é que é responsável por fazer a comparação entre o ficheiro de log *meu_teste_output.log* e o *valid-2015-04-09.log* (o ficheiro de log do professor). Para correr este ficheiro basta usar este comando:

```
(venv) venvricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 compare_with_log.py
```

E ele retorna algo deste tipo:

```

simple_sum.agu: ✓ OK (esperado ✓, obtido ✓)
simulate_chess.agu: ✓ OK (esperado ✗, obtido ✗)
somaSub.agu: ✓ OK (esperado ✗, obtido ✗)
stringLength.agu: ✓ OK (esperado ✓, obtido ✓)
string_concatenation.agu: ✓ OK (esperado ✓, obtido ✓)
string_printing.agu: ✓ OK (esperado ✓, obtido ✓)
sumArrayFunction.agu: ✓ OK (esperado ✓, obtido ✓)
sumNumInArray.agu: ✓ OK (esperado ✓, obtido ✓)
sumOfMinAndOfMaxOfArray.agu: ✓ OK (esperado ✓, obtido ✓)
sumTwoNumbers.agu: ✓ OK (esperado ✓, obtido ✓)
sum_array.agu: ✓ OK (esperado ✗, obtido ✗)
sum_function.agu: ✓ OK (esperado ✓, obtido ✓)
switch.agu: ✓ OK (esperado ✓, obtido ✓)
transpose_matrix.agu: ✓ OK (esperado ✓, obtido ✓)
two_sum.agu: ✓ OK (esperado ✓, obtido ✓)
unitFunc.agu: ✓ OK (esperado ✗, obtido ✗)
unit_as_input_and_output.agu: ✓ OK (esperado ✓, obtido ✓)
unit_type_test.agu: ✓ OK (esperado ✓, obtido ✓)
unused_parameter.agu: ✓ OK (esperado ✗, obtido ✗)
uv_mapping.agu: ✓ OK (esperado ✗, obtido ✗)
variables.agu: ✗ ERROR (esperado ✗, obtido ✓)
while_in_var.agu: ✓ OK (esperado ✗, obtido ✗)
while_loop_break.agu: ✓ OK (esperado ✓, obtido ✓)
while_loop_with_empty_body.agu: ✓ OK (esperado ✓, obtido ✓)
while_loop_with_mutations.agu: ✓ OK (esperado ✓, obtido ✓)
while_loops.agu: ✓ OK (esperado ✓, obtido ✓)
while_with_outer_var.agu: ✓ OK (esperado ✓, obtido ✓)
zeroLengthArray.agu: ✓ OK (esperado ✓, obtido ✓)
zip.agu: ✓ OK (esperado ✗, obtido ✗)

Resumo Final:
✓ 193 testes com mesmo resultado
✗ 10 testes com resultados diferentes

```

Com este ficheiro Pyhton, fui capaz de perceber algumas coisas interessantes sobre o meu compilador e o compilador do professor.

Por exemplo, percebi que o meu compilador não avalia corretamente programas terminados em “;”, como podemos ver por este exemplo:

```

Users > ricardo > Desktop > fcul > Técnicas de Compilação > aguda-testing > test > valid > 61015_boolean > boolean.agu
1  -- Author: 61015, Filipe Lopes
2
3  let printAndReturn (msg, b) : (String, Bool) -> Bool =
4      print(msg);
5      b;
6
7  if false && printAndReturn("This should not print", true) then
8      print("Result is true")
9  else
10     print("Result is false");|

```

O resultado do meu compilador:

```

(venv) venvricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 src/aguda-cabal/aguda/main.py test/valid/61015_boolean/boolean.agu
Syntax Error: Syntax error at EOF

```

Pelo que consegui apurar, o meu compilador, sempre que encontra “;”, está à espera que exista algum conteúdo para além do “;”, e por isso lança o erro sintático em EOF (end-of-file).

Contudo parece-me que o compilador do professor também apresenta, pelo menos até à data do ficheiro log (*test/valid-2015-04-09.log*). Vejamos os exemplos:

```

1  -- Author: 58215, Manuel Lourenço
2
3  let minValInArr (a,size) : Int[] -> Int =
4      let i : Int = 0;
5      let min : Int = a[0];
6      while i < size do (
7          if min < a[i] then
8              set min = a[i]
9          else
10             unit;
11             set i = i + 1
12         );
13         min
14
15 let main : Unit =
16     let arr : Int[] = new Int[4 | 4];
17     set arr[1] = 3;
18     set arr[2] = 2;
19     set arr[3] = 1;
20     print(minValInArr,4)

```

Neste ficheiro .agu (minArrAndElseUnit.agu), que era suposto ser um teste válido, passa na avaliação do compilador do professor:

```

126 /Users/vv/workspace/aguda-testing/test/valid/58215_minArrAndElseUnit/minArrAndElseUnit.agu [✓]
127 /Users/vv/workspace/aguda-testing/test/valid/58215_arrAndEven/arrAndEven.agu [✓]

```

No entanto existe um erro de sintaxe no final do ficheiro (linha 20), pelo que a função minValInArr é chamada, sem sequer usar parenteses, logo é um erro sintático e não é semântico, dado que o erro não é causado pela ausência de argumentos.

Output do compilador:

```

(env) venricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 src/aguda-cabal/aguda/main.py test/valid/58215_minArrAndElseUnit/minArrAndElseUnit.agu
Syntax Error: Syntax error at line 20, column 22 near ', '

```

Outro exemplo, é no ficheiro index_of.agu:

```

test > valid > 64371_index_of > index_of.agu
1  -- Author: 64371, Ricardo Costa
2
3  -- returns the index of the first occurrence of x in arr, or -1 if not found
4  let indexOf(arr, x): (Int[], Int) -> Int =
5      let index: Int = -1;
6      let i: Int = 0;
7      let length: Int = length(arr);
8      let continue: Bool = true;
9      while i < length && continue do (
10         if arr[i] == x then (
11             set index = i;
12             set continue = false -- break
13         );
14         set i = i + 1
15     );
16     index
17
18 let main: Unit =
19     let n: Int = 5;
20     let i: Int = 0;
21     let arr: Int[] = new Int[n | 0];
22     while i < n do ( -- fill array with 1..n
23         set arr[i] = i + 1;
24         set i = i + 1
25     );
26     let res: Int = indexOf(arr, 3);
27     print(res) -- 2

```

Em que é usada a palavra reservada `length`, como um `id`, pelo que o compilador não deveria permitir isso.

Resultado do compilador criado:

```
SyntaxError: Syntax error at line 7, column 9 near 'length'
(venv) ven@ricardo@MacBook-Pro-de-Ricardo aguda-testing % python3 src/aguda-cabal/aguda/main.py test/valid/64371_index_of/index_of.agu
(venv) ven@ricardo@MacBook-Pro-de-Ricardo aguda-testing %
```

Resultado do compilador do professor:

```
193 /Users/vv/workspace/aguda-testing/test/valid/64371_index_of/index_of.agu [✓]
194 /Users/vv/workspace/aguda-testing/test/valid/64371_multiples_3_or_5/multiples_3_or_5.agu [✓]
```

Estes não são os únicos casos em que possivelmente o compilador do professor esteja a avaliar os testes indevidamente, portanto deixei um ficheiro denominado *para_o_professor.txt*, com alguns possíveis testes indevidamente avaliados segundo a minha análise.