

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación.

IC5701 – Compiladores e Intérpretes.

Proyecto 3 Analizador – Ciruelas Extendido

Profesor: Aurelio Sanabria.

Estudiante:

Tamara Nicole Rodríguez Luna.

13 de mayo del 2025.

## Demostración y discusión

Primero tuve que modificar la clase de error del explorador para reutilizarla, además de crear una función específica para el reporte de errores.

```
1 # Analizador de Ciruelas (el lenguaje de programación)
2
3 from explorador.explorador import TipoComponente, ComponenteLéxico
4 from utils.árbol import ÁrbolSintaxisAbstracta, NodoÁrbol, TipoNodo
5
6 class Analizador:
7     4 hidden lines
8     11 componente_actual : ComponenteLéxico
9     12
10    13 def __init__(self, lista_componentes):
11    14
12    15 self.componentes_léxicos = lista_componentes
13    16 self.cantidad_componentes = len(lista_componentes)
14    17
15    20 self.asa = ÁrbolSintaxisAbstracta()
16    21
17    22
18    23 def imprimir_asa(self):
19    24
20    25 Imprime el árbol de sintaxis abstracta
21    26
22    27
23    28
24    29
25    30
26    31
27    32
28    33
29    34
30    35
31    36
32    37
33    38
34    39 self.asa.raiz = self.__analizar_programa()
35    40
36    41
37    42
38    43
39    44
40    45
41    46
42    47
43    48
44    49
45    50
46    51
47    52
48    53
49    54
50    55
51    56
52    57
53    58
54    59
55    60
56    61
57    62
58    63
59    64
60    65
61    66
62    67
63    68
64    69
65    70
66    71
67    72
68    73
69    74
70    75
71    76
72    77
73    78
74    79
75    80
76    81
77    82
78    83
79    84
80    85
81    86
82    87
83    88
84    89
85    90
86    91
87    92
88    93
89    94
90    95
91    96
92    97
93    98
94    99
95    100
96    101
97    102
98    103
99    104
100   105
101   106
102   107
103   108
104   109
105   110
106   111
107   112
108   113
109   114
110   115
111   116
112   117
113   118
114   119
115   120
116   121
117   122
118   123
119   124
120   125
121   126
122   127
123   128
124   129
125   130
126   131
127   132
128   133
129   134
130   135
131   136
132   137
133   138
134   139
135   140
136   141
137   142
138   143
139   144
140   145
141   146
142   147
143   148
144   149
145   150
146   151
147   152
148   153
149   154
150   155
151   156
152   157
153   158
154   159
155   160
156   161
157   162
158   163
159   164
160   165
161   166
162   167
163   168
164   169
165   170
166   171
167   172
168   173
169   174
170   175
171   176
172   177
173   178
174   179
175   180
176   181
177   182
178   183
179   184
180   185
181   186
182   187
183   188
184   189
185   190
186   191
187   192
188   193
189   194
190   195
191   196
192   197
193   198
194   199
195   200
196   201
197   202
198   203
199   204
200   205
201   206
202   207
203   208
204   209
205   210
206   211
207   212
208   213
209   214
210   215
211   216
212   217
213   218
214   219
215   220
216   221
217   222
218   223
219   224
220   225
221   226
222   227
223   228
224   229
225   230
226   231
227   232
228   233
229   234
230   235
231   236
232   237
233   238
234   239
235   240
236   241
237   242
238   243
239   244
240   245
241   246
242   247
243   248
244   249
245   250
246   251
247   252
248   253
249   254
250   255
251   256
252   257
253   258
254   259
255   260
256   261
257   262
258   263
259   264
260   265
261   266
262   267
263   268
264   269
265   270
266   271
267   272
268   273
269   274
270   275
271   276
272   277
273   278
274   279
275   280
276   281
277   282
278   283
279   284
280   285
281   286
282   287
283   288
284   289
285   290
286   291
287   292
288   293
289   294
290   295
291   296
292   297
293   298
294   299
295   300
296   301
297   302
298   303
299   304
300   305
301   306
302   307
303   308
304   309
305   310
306   311
307   312
308   313
309   314
310   315
311   316
312   317
313   318
314   319
315   320
316   321
317   322
318   323
319   324
320   325
321   326
322   327
323   328
324   329
325   330
326   331
327   332
328   333
329   334
330   335
331   336
332   337
333   338
334   339
335   340
336   341
337   342
338   343
339   344
340   345
341   346
342   347
343   348
344   349
345   350
346   351
347   352
348   353
349   354
350   355
351   356
352   357
353   358
354   359
355   360
356   361
357   362
358   363
359   364
360   365
361   366
362   367
363   368
364   369
365   370
366   371
367   372
368   373
369   374
370   375
371   376
372   377
373   378
374   379
375   380
376   381
377   382
378   383
379   384
380   385
381   386
382   387
383   388
384   389
385   390
386   391
387   392
388   393
389   394
390   395
391   396
392   397
393   398
394   399
395   400
396   401
397   402
398   403
399   404
400   405
401   406
402   407
403   408
404   409
405   410
406   411
407   412
408   413
409   414
410   415
411   416
412   417
413   418
414   419
415   420
416   421
417   422
418   423
419   424
420   425
421   426
422   427
423   428
424   429
425   430
426   431
427   432
428   433
429   434
430   435
431   436
432   437
433   438
434   439
435   440
436   441
437   442
438   443
439   444
440   445
441   446
442   447
443   448
444   449
445   450
446   451
447   452
448   453
449   454
450   455
451   456
452   457
453   458
454   459
455   460
456   461
457   462
458   463
459   464
460   465
461   466
462   467
463   468
464   469
465   470
466   471
467   472
468   473
469   474
470   475
471   476
472   477
473   478
474   479
475   480
476   481
477   482
478   483
479   484
480   485
481   486
482   487
483   488
484   489
485   490
486   491
487   492
488   493
489   494
490   495
491   496
492   497
493   498
494   499
495   500
496   501
497   502
498   503
499   504
500   505
501   506
502   507
503   508
504   509
505   510
506   511
507   512
508   513
509   514
510   515
511   516
512   517
513   518
514   519
515   520
516   521
517   522
518   523
519   524
520   525
521   526
522   527
523   528
524   529
525   530
526   531
527   532
528   533
529   534
530   535
531   536
532   537
533   538
534   539
535   540
536   541
537   542
538   543
539   544
540   545
541   546
542   547
543   548
544   549
545   550
546   551
547   552
548   553
549   554
550   555
551   556
552   557
553   558
554   559
555   560
556   561
557   562
558   563
559   564
560   565
561   566
562   567
563   568
564   569
565   570
566   571
567   572
568   573
569   574
570   575
571   576
572   577
573   578
574   579
575   580
576   581
577   582
578   583
579   584
580   585
581   586
582   587
583   588
584   589
585   590
586   591
587   592
588   593
589   594
590   595
591   596
592   597
593   598
594   599
595   600
596   601
597   602
598   603
599   604
600   605
601   606
602   607
603   608
604   609
605   610
606   611
607   612
608   613
609   614
610   615
611   616
612   617
613   618
614   619
615   620
616   621
617   622
618   623
619   624
620   625
621   626
622   627
623   628
624   629
625   630
626   631
627   632
628   633
629   634
630   635
631   636
632   637
633   638
634   639
635   640
636   641
637   642
638   643
639   644
640   645
641   646
642   647
643   648
644   649
645   650
646   651
647   652
648   653
649   654
650   655
651   656
652   657
653   658
654   659
655   660
656   661
657   662
658   663
659   664
660   665
661   666
662   667
663   668
664   669
665   670
666   671
667   672
668   673
669   674
670   675
671   676
672   677
673   678
674   679
675   680
676   681
677   682
678   683
679   684
680   685
681   686
682   687
683   688
684   689
685   690
686   691
687   692
688   693
689   694
690   695
691   696
692   697
693   698
694   699
695   700
696   701
697   702
698   703
699   704
700   705
701   706
702   707
703   708
704   709
705   710
706   711
707   712
708   713
709   714
710   715
711   716
712   717
713   718
714   719
715   720
716   721
717   722
718   723
719   724
720   725
721   726
722   727
723   728
724   729
725   730
726   731
727   732
728   733
729   734
730   735
731   736
732   737
733   738
734   739
735   740
736   741
737   742
738   743
739   744
740   745
741   746
742   747
743   748
744   749
745   750
746   751
747   752
748   753
749   754
750   755
751   756
752   757
753   758
754   759
755   760
756   761
757   762
758   763
759   764
760   765
761   766
762   767
763   768
764   769
765   770
766   771
767   772
768   773
769   774
770   775
771   776
772   777
773   778
774   779
775   780
776   781
777   782
778   783
779   784
780   785
781   786
782   787
783   788
784   789
785   790
786   791
787   792
788   793
789   794
790   795
791   796
792   797
793   798
794   799
795   800
796   801
797   802
798   803
799   804
800   805
801   806
802   807
803   808
804   809
805   810
806   811
807   812
808   813
809   814
810   815
811   816
812   817
813   818
814   819
815   820
816   821
817   822
818   823
819   824
820   825
821   826
822   827
823   828
824   829
825   830
826   831
827   832
828   833
829   834
830   835
831   836
832   837
833   838
834   839
835   840
836   841
837   842
838   843
839   844
840   845
841   846
842   847
843   848
844   849
845   850
846   851
847   852
848   853
849   854
850   855
851   856
852   857
853   858
854   859
855   860
856   861
857   862
858   863
859   864
860   865
861   866
862   867
863   868
864   869
865   870
866   871
867   872
868   873
869   874
870   875
871   876
872   877
873   878
874   879
875   880
876   881
877   882
878   883
879   884
880   885
881   886
882   887
883   888
884   889
885   890
886   891
887   892
888   893
889   894
890   895
891   896
892   897
893   898
894   899
895   900
896   901
897   902
898   903
899   904
900   905
901   906
902   907
903   908
904   909
905   910
906   911
907   912
908   913
909   914
910   915
911   916
912   917
913   918
914   919
915   920
916   921
917   922
918   923
919   924
920   925
921   926
922   927
923   928
924   929
925   930
926   931
927   932
928   933
929   934
930   935
931   936
932   937
933   938
934   939
935   940
936   941
937   942
938   943
939   944
940   945
941   946
942   947
943   948
944   949
945   950
946   951
947   952
948   953
949   954
950   955
951   956
952   957
953   958
954   959
955   960
956   961
957   962
958   963
959   964
960   965
961   966
962   967
963   968
964   969
965   970
966   971
967   972
968   973
969   974
970   975
971   976
972   977
973   978
974   979
975   980
976   981
977   982
978   983
979   984
980   985
981   986
982   987
983   988
984   989
985   990
986   991
987   992
988   993
989   994
990   995
991   996
992   997
993   998
994   999
995   1000
```

Luego usé la palabra clave “llamase” para diferenciar la asignación de la invocación

```
1d3920 - Se agrega palabra cl... (2 files) X
1 README
15 hidden lines
16 ## Ejecución del explorador
17
18 python3 ciruelas.py --solo-explorar docs/ejemplos/factorial.ciru
19
20 ## Otras opciones
21
16 hidden lines
15 hidden lines
16 ## Ejecución del explorador
17
18 python3 ciruelas.py --solo-explorar docs/ejemplos/factorial.ciru
19
20 ## Ejecución del analizador
21
22 python3 ciruelas.py --solo-analizar docs/ejemplos/factorial.ciru
23
24 ## Otras opciones
25
16 hidden lines
16 hidden lines
170 analizador.py analizador
120 hidden lines
121 elif self.componente_actual.texto == '(':
122     nodos_nuevos += [self.__analizar_expresión_matemática()]
123
124 # Acá tengo que decidir si es Invocación o solo un identificador
125 elif self.componente_actual.tipo == TipoComponente.IDENTIFICADOR:
126
127     if self.__componente_venidero().texto == '(':
128         nodos_nuevos += [self.__analizar_invocación()]
129     else:
130         nodos_nuevos += [self.__verificar_identificador()]
131
132 else:
133     self.__reportar_error("estructura de asignación inválida")
126 hidden lines
260 def __analizar_instrucción(self):
261     """
262     Instrucción ::= (Repetición | Bifurcación | Asignación | Invocación | Retorno)
263
264     Acá hay un error en la gramática por que no reconoce las
265     Invocaciones por la falta de corregir un error en la gramática LL
266
3 hidden lines
270 ...
271 La solución en código que yo presenté acá esta sería como algo así
272
273 Instrucción ::= (Repetición | Bifurcación | (Asignación | Invocación) | Retorno)
274
275 Ojo los paréntesis extra
276
277 """
278
279 nodos_nuevos = []
5 hidden lines
285 elif self.componente_actual.texto == 'diay siiii':
286     nodos_nuevos += [self.__analizar_bifurcación()]
287
288 elif self.componente_actual.tipo == TipoComponente.IDENTIFICADOR:
289
290
291     if self.__componente_venidero().texto == 'metale':
292         nodos_nuevos += [self.__analizar_asignación()]
293
294
120 hidden lines
121 elif self.componente_actual.texto == '(':
122     nodos_nuevos += [self.__analizar_expresión_matemática()]
123
124 elif self.componente_actual.texto == 'llamese':
125     nodos_nuevos += [self.__analizar_invocación()]
126
127 # Acá es solo un identificador
128 elif self.componente_actual.tipo == TipoComponente.IDENTIFICADOR:
129
130     nodos_nuevos += [self.__verificar_identificador()]
131
132 else:
133     self.__reportar_error("estructura de asignación inválida")
126 hidden lines
259 def __analizar_instrucción(self):
260     """
261     Instrucción ::= (Repetición | Bifurcación | Asignación | Invocación | Retorno)
262
263     Acá hay un error en la gramática por que no reconoce las
264     Invocaciones por la falta de corregir un error en la gramática LL
265
3 hidden lines
269 ...
270 La solución en código que yo presenté acá esta sería como algo así
271
272 Instrucción ::= (Repetición | Bifurcación | Asignación | Invocación | Retorno)
273
274
275 """
276
277 nodos_nuevos = []
5 hidden lines
281 elif self.componente_actual.texto == 'diay siiii':
282     nodos_nuevos += [self.__analizar_bifurcación()]
283
284 elif self.componente_actual.texto == 'llamese':
285     nodos_nuevos += [self.__analizar_invocación()]
286
287 elif self.componente_actual.tipo == TipoComponente.IDENTIFICADOR:
288
289     nodos_nuevos += [self.__analizar_asignación()]
```

Añadí el SWITCH-CASE

```
4462bff - Se agrega el correct... (6 files) X
analizador.py analizador
200     nodos_nuevos += [self.__analizar_repeticion()]
281
282     elif self.componente_actual.texto == 'diay siii':
16 hidden lines
299     # Acá yo debería volarme el nivel Instrucción por que no aporta nada
300     return NodoÁrbol(TipoNodo.INSTRUCCIÓN, nodos=nodos_nuevos)
381
302     def __analizar_repetición(self):
303     """
304
77 hidden lines
382     # porquería... pero a ustedes les toca arreglarlo
383
384     # opcional el AND o el OR
385     if componente_actual.texto == 'divorcio':
386         nodo = NodoÁrbol(TipoNodo.OPERADOR_LÓGICO, contenido='divorcio')
387         nodos_nuevos += [nodo]
388         self.__verificar('divorcio')
16 hidden lines
405     """
406     Comparación ::= Valor Comparador Valor
407     """
408     nodos_nuevos = []
409
410     # Sin opciones, todo se analiza
290     nodos_nuevos += [self.__analizar_repeticion()]
291
292     elif self.componente_actual.texto == 'diay siii':
16 hidden lines
309     # Acá yo debería volarme el nivel Instrucción por que no aporta nada
310     return NodoÁrbol(TipoNodo.INSTRUCCIÓN, nodos=nodos_nuevos)
311
312     def __analizar_switch_case(self):
313     """
314     SwitchCase ::= como está la vara ( Identificador ) { Movida+ (sino ni modo E
315     """
316     nodos_nuevos = []
317
318     # Verificar la estructura "como está la vara"
319     self.__verificar("como está la vara")
320
321     self.__verificar('(')
322     nodos_nuevos += [self.__verificar_identificador()]
323     self.__verificar(',')
324
325     self.__verificar('(')
326
327     # Al menos un caso (movida)
328     while self.componente_actual.texto == 'movida':
329         nodos_nuevos += [self.__analizar_movida()]
330
331     # Caso default opcional
332     if self.componente_actual.texto == 'sino ni modo':
333         nodos_nuevos += [self.__analizar_sino()]
334
335     self.__verificar('}')
336
337     return NodoÁrbol(TipoNodo.SWITCH_CASE, nodos=nodos_nuevos)
339     def __analizar_movida(self):
340     """
341     Movida ::= movida ~ Valor ~ { Instrucción+ }
342     """
343     print("[DEBUG] __analizar_movida")
344     nodos_nuevos = []
345
346     self.__verificar('movida')
347     nodos_nuevos += [self.__analizar_valor()]
348     nodos_nuevos += [self.__analizar_bloque_instrucciones()]
349
350     return NodoÁrbol(TipoNodo.MOVIDA, nodos=nodos_nuevos)
351
352     def __analizar_repetición(self):
353     """
431     # porquería... pero a ustedes les toca arreglarlo
432
433     # opcional el AND o el OR
434     if self.componente_actual.texto == 'divorcio':
435         nodo = NodoÁrbol(TipoNodo.OPERADOR_LÓGICO, contenido='divorcio')
436         nodos_nuevos += [nodo]
437         self.__verificar('divorcio')
16 hidden lines
454     """
455     Comparación ::= Valor Comparador Valor
456     """
457     print("[DEBUG] __analizar_comparación")
458     nodos_nuevos = []
459
460     # Sin opciones, todo se analiza
```

Pero el último par de commits causaron un comportamiento inesperado en la detección de errores como reportar que no encontraba la función principal cuando sí estaba ahí, entonces cambié el manejo de los errores y archivos .ciru





Luego agregué FOR y TRY-CATCH

1df654 - Se agrega el análisis... (3 files) x

analizador.py analizador

299 hidden lines

300 if self.componente\_actual.texto == 'como está la vara':  
301 nodos\_nuevos += [self.\_\_analizar\_switch\_case()]  
302  
303 elif self.componente\_actual.texto == 'upee':  
304 nodos\_nuevos += [self.\_\_analizar\_repetición()]  
305  
306  
307  
308

17 hidden lines

323 # Acá yo debería volarme el nivel Instrucción por que no aporta nada  
324 return NodoArbol(TipoNodo.INSTRUCCIÓN, nodos=nodos\_nuevos)  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

261 hidden lines

590 nodos\_nuevos += [self.\_\_analizar\_instrucción()]  
591  
592 # Acá todo puede venir uno o más  
593 while self.componente\_actual.texto in ['upee', 'diay siii', 'sarpe', 'safis']  
594 or self.componente\_actual.tipo == TipoComponente.IDENTIFICADOR:  
595 nodos\_nuevos += [self.\_\_analizar\_instrucción()]  
596

144 hidden lines

299 hidden lines

300 if self.componente\_actual.texto == 'como está la vara':  
301 nodos\_nuevos += [self.\_\_analizar\_switch\_case()]  
302  
303 elif self.componente\_actual.texto == 'dele vuelta':  
304 nodos\_nuevos += [self.\_\_analizar\_dele\_vuelta()]  
305  
306 elif self.componente\_actual.texto == 'upee':  
307 nodos\_nuevos += [self.\_\_analizar\_repetición()]  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

261 hidden lines

624 nodos\_nuevos += [self.\_\_analizar\_instrucción()]  
625  
626 # Acá todo puede venir uno o más  
627 while self.componente\_actual.texto in ['upee', 'diay siii', 'sarpe', 'safis']  
628 or self.componente\_actual.tipo == TipoComponente.IDENTIFICADOR:  
629 nodos\_nuevos += [self.\_\_analizar\_instrucción()]  
630

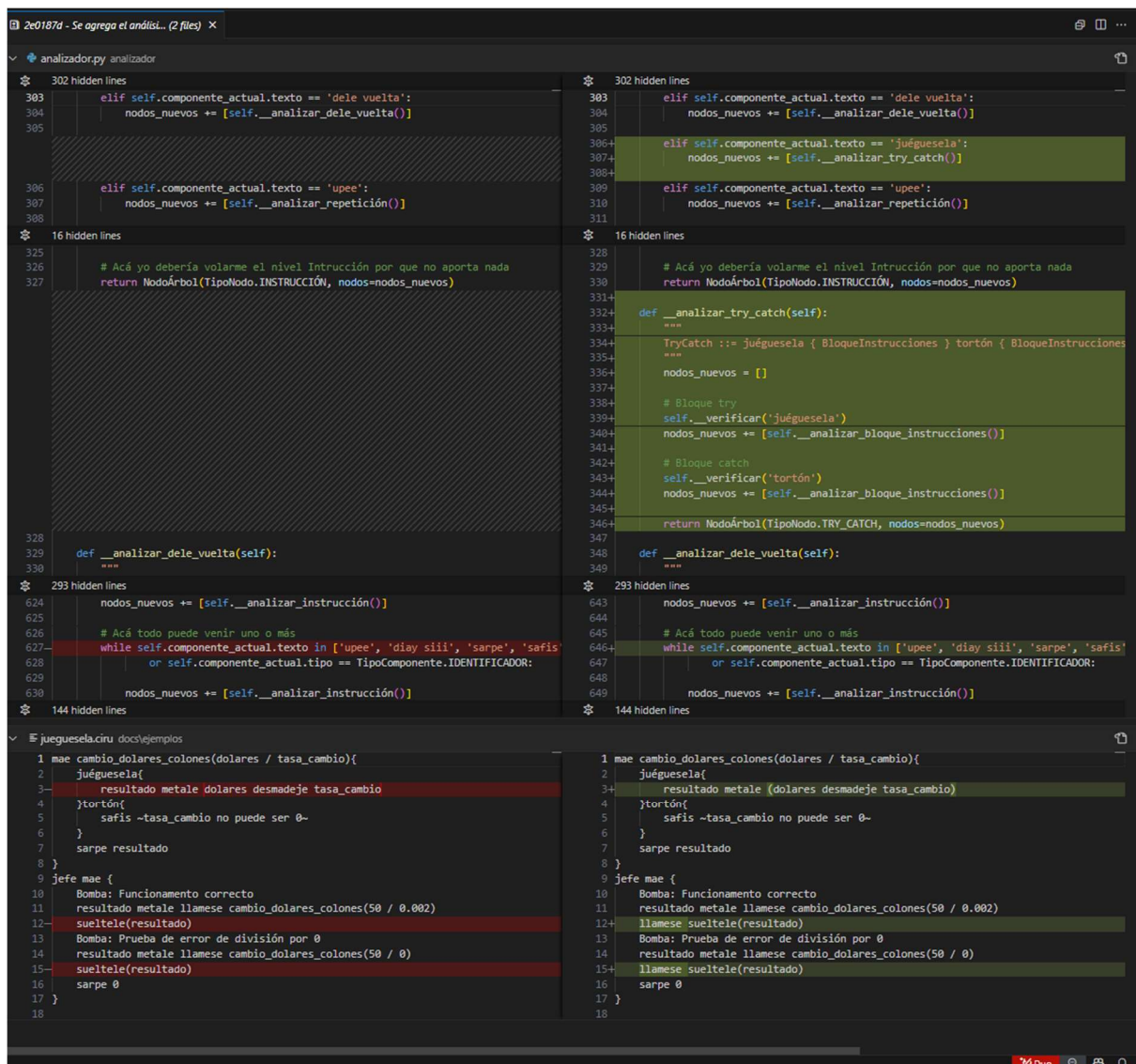
144 hidden lines

19 hidden lines

20 jefe mae {  
21 Bomba: Caso Puntarenas  
22 resultado metale llamese actividades\_provincia(~Puntarenas-)  
23 sueltele(resultado)  
24

19 hidden lines

20 jefe mae {  
21 Bomba: Caso Puntarenas  
22 resultado metale llamese actividades\_provincia(~Puntarenas-)  
23 llamese sueltele(resultado)  
24



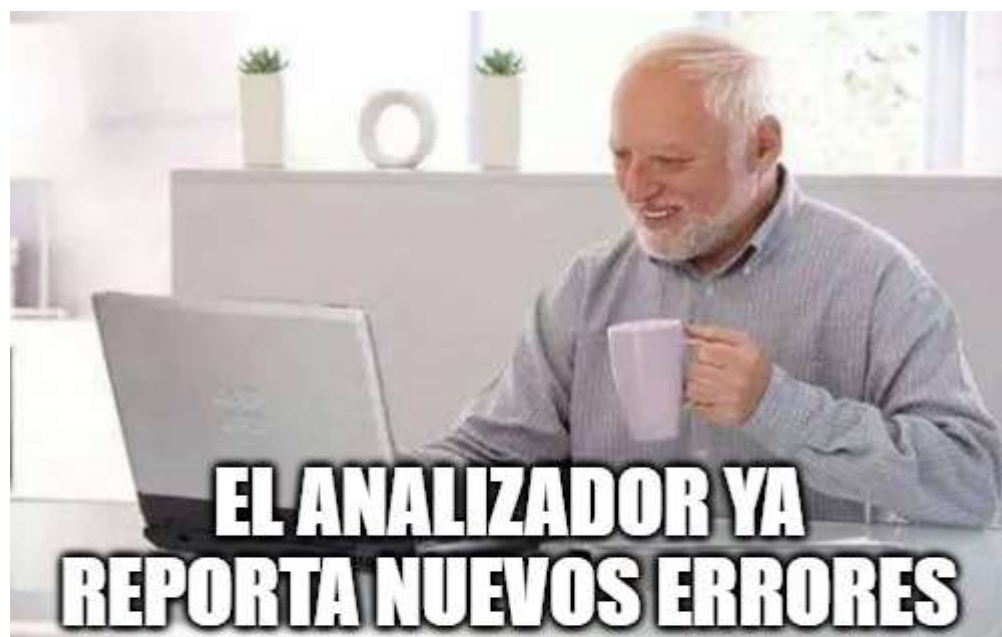
## Lecciones aprendidas

El analizador no me dio tantos problemas, creo que lo más importante que aprendí fue la importancia de prints para encontrar bugs, siempre son una herramienta simple y efectiva para saber en qué etapa está fallando. También es importante destacar que no porque el analizador muestre un error significa que está funcionando mal, varias veces me pasó que se me olvidó agregar la nueva palabra clave “llamase” antes de cualquier invocación.

## Memes







**EL ANALIZADOR YA  
REPORTA NUEVOS ERRORES**



**POR QUÉ FACTORIAL.CIRU NO SIRVE?**

imgflip.com



Haré las  
correcciones sugeridas



Pondré mis  
funciones nuevas



Haré pruebas



Por qué me  
dice que falta la  
función principal?



