

Transcompilation

into



DConf 2019

London

Bastiaan

Veelo

Transcompilation

into



DConf 2019

London

Bastiaan

Veelo

Transcompilation

into



DConf 2019

London

Bastiaan

Veelo

Intermediate Representation / Computer Aided Porting

Transcompilation

into

A large, bold, red letter 'D' with a slight glow effect, positioned to the right of the word 'into'.

DConf 2019

London

Bastiaan

Veelo

Intermediate Representation / Computer Aided Porting

Transcompilation

into

A large, bold, red letter 'D' with a slight glow effect, positioned to the right of the word 'into'.

DConf 2019

London

Bastiaan

Veelo

Intermediate Representation / Computer Aided Porting

Transcompilation

DConf 2019

London

into

A large, bold, red letter 'D' with a white outline, positioned to the right of the word 'into'.

in

Bastiaan

Veelo

~~Intermediate Representation~~ / Computer Aided Porting



Extended Pascal → D

ca. 500 kloc

In parallel with development

Conversion of the DMD frontend from C++ to D



DConf 2015




```
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
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
```

```
246 unittest
247 {
248     ... assert(toD(EP(
249         `program·MyTest(output);
250
251     begin
252         ... writeln('Hello·D'·s·"World"!');
253     end.`))
254 ==·
255 `import·std·stdio;
256
257 ·
258
259 void·main(string[]·args)
260 {
261     ... writeln("Hello·D's·\"World\"!");
262 }`);
263 }
```



Pascal2D ▶ **D** p2d.d ▶ ...

```
1 import pegged.grammar;
2
3 // Extended Pascal grammar.
4 // Comments refer to the section numbers in ISO 10206
5 // http://pascal-central.com/docs/iso10206.pdf
6 mixin(grammar(`
7   EP:
8     .. CompileUnit <- Program eoi
9
10    # Token separators
11    .. _ <- (. WhiteSpace / Comment / InlineComment .) _ *
12    .. WhiteSpace <- ( " " / "\t" / endOfLine )+
13
14    # Comments:
15    .. CommentOpen <- . "{ " / "( *"
16    .. CommentClose <- . "}" / "*)"
17    .. CommentContent <- (. !CommentClose . .)*
18    .. InlineComment <- CommentOpen CommentContent CommentClose !endOfLine
19    .. Comment <- CommentOpen CommentContent CommentClose &endOfLine
20
21    # 6.1.1
22    .. Digit <- [0-9]
```

```
p2d.d
1
2 // Extended Pascal grammar.
3 // Comments refer to the section numbers in ISO 10206
4 // http://pascal-central.com/docs/iso10206.pdf
5
6 # Token separators
7
8 .. _ <- (. WhiteSpace / Comment / InlineComment .) _ *
9 .. WhiteSpace <- ( " " / "\t" / endOfLine )+
10
11 # Comments:
12 .. CommentOpen <- . "{ " / "( *"
13 .. CommentClose <- . "}" / "*)"
14 .. CommentContent <- (. !CommentClose . .)*
15 .. InlineComment <- CommentOpen CommentContent CommentClose !endOfLine
16 .. Comment <- CommentOpen CommentContent CommentClose &endOfLine
17
18 # 6.1.1
19 .. Digit <- [0-9]
```

```
108 # 6.9.1
109 Statement <- (Label ":" )? SimpleStatement
110
111 # 6.9.2.1
112 SimpleStatement <- ProcedureStatement
113
114 # 6.9.3.2
115 CompoundStatement <- "begin" StatementSequence "end"
```

```
108 # 6.9.1
109 Statement <- (Label ":" )? SimpleStatement
110
111 # 6.9.2.1
112 SimpleStatement <- ProcedureStatement
113
```

6.9.3.2 Compound-statements

A compound-statement shall specify execution of the statement-sequence of the compound-statement.

compound-statement = 'begin' statement-sequence 'end' .

```
120 # 6.9.3.2
121 CompoundStatement <- "begin" StatementSequence "end"
122
```

```
159 string toD(ParseTree p)
160 {
161     ... import std.container;
162     ... auto imports = new RedBlackTree!string;
163
164     string escapeString(string s)
165     { ...
166     }
167
168     string translateAny(ParseTree p)
169     { ...
170     }
171
172     ...
234 }
235
236 auto code = translateAny(p);
237
238 string importDeclaration;
239 foreach(imp; imports[]) {
240     importDeclaration ~= "import " ~ imp ~ ".\n";
241 }
242
243 return importDeclaration ~ "\n" ~ code;
244 }
245
```

```
159 string toD(ParseTree p)
160 {
161     ... import std.container;
162     ... auto imports = new RedBlackTree!string;
163
164     string escapeString(string s)
165     { ...
166     }
167
168     string translateAny(ParseTree p)
169     { ...
170     }
171
172     ...
234 }
235
236 auto code = translateAny(p);
237
238 string importDeclaration;
239 foreach(imp; imports[]) {
240     importDeclaration ~= "import " ~ imp ~ ".\n";
241 }
242
243 return importDeclaration ~ "\n" ~ code;
244 }
245
```

```
...
string toD(ParseTree p)
{
    ...
    string translateAny(ParseTree p)
    {
        ...
    }
    ...
}
...

```

```
159 string toD(ParseTree p)
160 {
161     ... import std::container;
162     ... auto imports = new RedBlackTree!string;
163
```

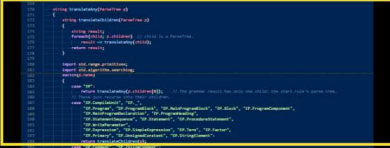
```
240 struct ParseTree
241 {
242     ... string name; /// The node name
243     ... bool successful; /// Indicates whether a parsing was
244     ... string[] matches; /// The matched input's parts. Some
245     ... string input; /// The input string that generated the
246     ... size_t begin, end; /// Indices for the matched part
247     ... ParseTree[] children; /// The sub-trees created by s
248
249
250
```

```
240     ... importDeclaration ~="import " ~ imp ~ ";" \n";
241     ... }
242
243     ... return importDeclaration ~ "\n" ~ code;
244 }
245
```



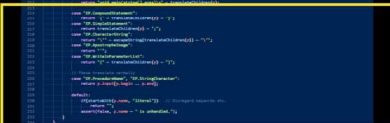
```
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
```

```
169 string translateAny(ParseTree p)
170 {
171     string translateChildren(ParseTree p)
172     {
173         string result;
174         foreach(child; p.children) // child is a ParseTree.
175             result += translateAny(child);
176         return result;
177     }
178
179     import std.algorithm.searching;
180     switch(p.name)
181     {
182         case "EP":
183             return translateAny(p.children[0]); // The grammar result has only
184             // These just recurse into their children.
185         case "EP.CompileUnit", "EP._",
186             "EP.Program", "EP.ProgramBlock", "EP.MainProgramBlock", "EP.Block",
187             "EP.ProgramComponent", "EP.MainProgramDeclaration",
188             "EP.ProgramHeading", "EP.StatementSequence", "EP.Statement",
189             "EP.ProcedureStatement", "EP.WriteParameter",
190             "EP.Expression", "EP.SimpleExpression", "EP.Term", "EP.Factor",
191             "EP.Primary", "EP.UnsignedConstant", "EP.StringElement":
192             return translateChildren(p);
```



```
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
```

```
212 case "EP.CompoundStatement":
213     return "{ ~ translateChildren(p) ~ }";
214 case "EP.SimpleStatement":
215     return translateChildren(p) ~ ";";
216 case "EP.CharacterString":
217     return "\"" ~ escapeString(translateChildren(p)) ~ "\"";
218 case "EP.ApostropheImage":
219     return "'";
220 case "EP.WriteLineParameterList":
221     return "(" ~ translateChildren(p) ~ ")";
222
223 // These translate verbally
224 case "EP.ProcedureName", "EP.StringCharacter":
225     return p.input[p.begin .. p.end];
226
227 default:
228     if(startsWith(p.name, "literal")) // Disregard keywords etc.
229         return "";
230     assert(false, p.name ~ " is unhandled.");
231 }
```



```
246 unittest
247 {
248     ... assert(toD(EP(
249         `program·MyTest(output);
250
251     begin
252         ... writeln('Hello·D'·s·"World"!');
253     end.`))
254 ==·
255     `import·std·stdio;
256
257     ·
258
259     void·main(string[]·args)
260     {
261         ... writeln("Hello·D's·\"World\"!");
262     }`);
263 }
```

```
246 unittest
247 {
248     ... assert(toD(EP(
249         `program·MyTest(output);
250
251     begin
252         ... writeln('Hello·D'·s·"World"!');
253     end.`))
254 ==·
255     `import·std·stdio;
256
257     ·
258
259     void·main(string[]·args)
260     {
261         ... writeln("Hello·D's·\"World\"!");
262     }`);
263 }
```

```

# 263
# 264
# 265
# 266
# 267
# 268
# 269

```



```
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
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
```

263 mixin(

264

265

266

267

268

269

```
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
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
```

```
263      mixin(`
264      program·MyTest(output);
265
266      begin
267      |···writeln('Hello·D' 's·"World"!');
268      end.
269      `
```

```
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
```

```
Command Prompt

C:\Users\bastiaan\Documents\D\Pascal2D>rdmd -m64 -I=..\Pegged p2d.d ..\Pegged\pegged.lib
Hello D's "World"!

C:\Users\bastiaan\Documents\D\Pascal2D>
```

```

263
264
265
266
267
268
269

```

```

263   mixin(`
264   program MyTest(output);
265
266   begin
267   |...writeln('Hello D's "World"!');
268   end.
269   ` .EP.toD);
```

Command Prompt

```
C:\Users\basti  
Hello D's "wor  
C:\Users\basti
```

Transcompiler

```
logged.lib
```

```
Id"!');
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Transcompiler
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello D's \"wor");
        }
    }
}
```


~~Transcompiler~~

parser

- ❌ Module import
- ❌ Symbol table
- ❌ Scope
- ❌ Semantic analysis
- ❌ Error reporting
- ❌ Optimization
- ❌ Code generation
- ✅ Rely on similarities
- ✅ Ambiguities OK
 $a * b$
- ✅ Microtranslation

Command Prompt

```
C:\Users\basti  
Hello D's "wor
```

```
C:\Users\basti
```

```
gged.lib
```

```
Id"!');
```

~~Transcompiler~~

parser

- ❌ Module import
- ❌ Symbol table
- ❌ Scope
- ❌ Semantic analysis
- ❌ Error reporting
- ❌ Optimization
- ❌ Code generation
- ✅ Rely on similarities
- ✅ Ambiguities OK
 $a * b$
- ✅ Microtranslation

Command Prompt

```
C:\Users\basti  
Hello D's "wor
```

```
C:\Users\basti
```

```
logged.lib
```

```
Id"!');
```

```
Command Prompt
C:\Users\bastiaan\Documents\D\Pascal2D>rdmd -m64 -I=..\Pegged p2d.d ..\Pegged\pegged.lib
Hello D's "World"!
C:\Users\bastiaan\Documents\D\Pascal2D>
```

```

263
264
265
266
267
268
269

```

```

263   mixin(`
264   program MyTest(output);
265
266   begin
267   |...writeln('Hello D' 's "World"!');
268   end.
269   ` .EP.toD);
```


0.8 kloc

19 kloc

source ▸ D epgrammar.d ▸ ...

```

20 enum EPgrammar = {
21     EP_
22 }

```

source ▸ D generate.d ▸ ...

```

23 5 import pegged.grammar;
24 6 import epgrammar;
25 7

```

source ▸ D make.d ▸ ...

```

8 4 void main(string[] opts) in (opts.length == 2)
9 5
10 6 {} dub.json ▸ [ ] preGenerateCommands
11 7

```

```

19     "excludedSourceFiles": [
20     "source/epgrammar.d",
21     "source/generate.d",
22     "source/make.d"
23     ],
24     "preGenerateCommands": [
25     "cd $PACKAGE_DIR/source && rdmd make.d $PEGGED_PACKAGE_DIR" (PT p)
26     ],
27     auto args = ["rdmd", "-I" ~ opts[1], "generate.d"]; (PT p)

```

source ▸ D epparser.d ▸ ...

```

18805 | ... }
18806 | }
18807 |
18808 | alias GenericEP!(ParseTree).EP EP;
18809 |
18810 |

```

examples/generated-source does not work. Regression in preGenerateCommmands. #1474

Open veelo opened this issue on 17 May 2018 · 3 comments

800 loc

```

source ▸ D epgrammar.d ▸ ...
20 enum EPgrammar = {
21 EP:
22     ... BNVCompileUnit <- Program eoi
23
24 # 6.1.1
25     ... Digit <- digit

```



19 kloc

```

source ▸ D epparser.d ▸ ...
18805 | ... }
18806 | }
18807
18808 alias GenericEP!(ParseTree).EP EP;
18809
18810

```

4 kloc

```

source ▸ D p2d.d ▸ toD
4145     ... lastImp = imp;
4146     }
4147     if (importDeclaration.length > 0)
4148     ... importDeclaration ~= ";"\n";
4149
4150     return moduleHeading ~ importDeclaration ~ "\n" ~ code;
4151 } // toD
4152

```


800 loc

```
source ▸ D epgrammar.d ▸ ...  
20 enum EPgrammar = {  
21 EP:  
22     ... BNVCompileUnit <- Program eoi  
23  
24 # 6.1.1  
25     ... Digit <- digit
```



19 kloc

```
source ▸ D epparser.d ▸ ...  
18805 | ... }  
18806 | }  
18807 |  
18808 | alias GenericEP!(ParseTree).EP EP;  
18809 |  
18810 |
```

4 kloc

```
source ▸ D p2d.d ▸ toD  
4145 | ... lastImp = imp;  
4146 | }  
4147 | ... if (importDeclaration.length > 0)  
4148 | ... importDeclaration ~= ";"\n";  
4149 |  
4150 | ... return moduleHeading ~ importDeclaration ~ "\n" ~ code;  
4151 | } // toD  
4152 |
```

80 loc

```
source ▸ D main.d ▸ ...  
65 | ... auto parsed = args.length == 3 ?  
66 | ... EP(readText(args[1]) ~ readText(args[2])) :  
67 | ... EP(readText(args[1]));  
68 | + | ... if (syntax_tree) ...  
70 | + | ... if (!parsed.successful) { ...  
76 | | ... }  
77 | ... writeln(toD(parsed));
```


pascal2d
transcompiler
dub

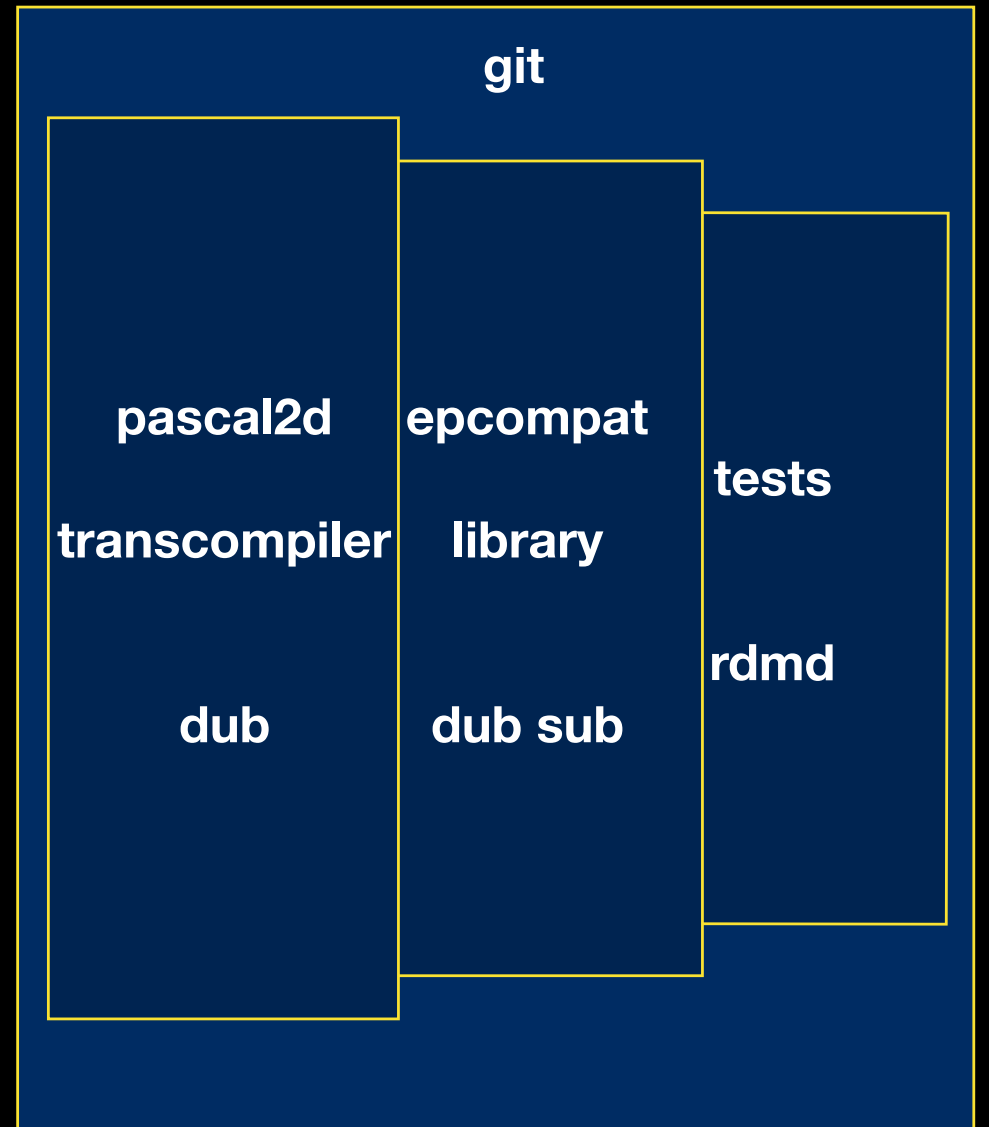
epcompat
library
dub sub

tests
rdmd

pascal2d
transcompiler
dub

epcompat
library
dub sub

tests
rdmd



dub
app

dub
lib

svn
Pascal

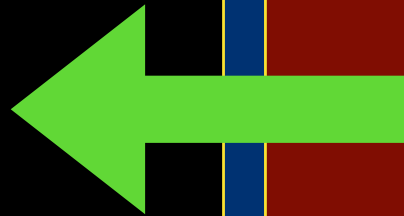
git
rdmd
transpile.d

git

pascal2d
transcompiler
dub

epcompat
library
dub sub

tests
rdmd



```
1 1
2 1
3 1
4 1
5 1
6 1
7 1
8 1
9 1
10 1
11 1
12 1
13 1
14 1
15 1
16 1
17 1
18 1
19 1
20 1
21 1
22 1
23 1
24 1
25 1
26 1
27 1
28 1
29 1
30 1
31 1
32 1
33 1
34 1
35 1
36 1
37 1
38 1
39 1
40 1
41 1
42 1
43 1
44 1
45 1
46 1
47 1
48 1
49 1
50 1
51 1
52 1
53 1
54 1
55 1
56 1
57 1
58 1
59 1
60 1
61 1
62 1
63 1
64 1
65 1
66 1
67 1
68 1
69 1
70 1
71 1
72 1
73 1
74 1
75 1
76 1
77 1
78 1
79 1
80 1
81 1
82 1
83 1
84 1
85 1
86 1
87 1
88 1
89 1
90 1
91 1
92 1
93 1
94 1
95 1
96 1
97 1
98 1
99 1
100 1
```

dub
app

dub
lib

svn
Pascal

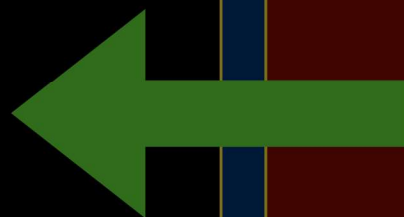
git
rdmd
transpile.d

git

pascal2d
transcompiler

epcompat
library

tests



```

1 // ...
2
3 // ...
4
5 // ...
6
7 // ...
8
9 // ...
10
11 // ...
12
13 // ...
14
15 // ...
16
17 // ...
18
19 // ...
20
21 // ...
22
23 // ...
24
25 // ...
26
27 // ...
28
29 // ...
30
31 // ...
32
33 // ...
34
35 // ...
36
37 // ...
38
39 // ...
40
41 // ...
42
43 // ...
44
45 // ...
46
47 // ...
48
49 // ...
50
51 // ...
52
53 // ...
54
55 // ...
56
57 // ...
58
59 // ...
60
61 // ...
62
63 // ...
64
65 // ...
66
67 // ...
68
69 // ...
70
71 // ...
72
73 // ...
74
75 // ...
76
77 // ...
78
79 // ...
80
81 // ...
82
83 // ...
84
85 // ...
86
87 // ...
88
89 // ...
90
91 // ...
92
93 // ...
94
95 // ...
96
97 // ...
98
99 // ...
100

```

```

73     .... ["trunk/PIAS/source/sentinel.hdr"] ..... .tr("PIAS_D/source/sentinel.di"); // interface
74     .... ["trunk/PIAS/source/serial.hdr",
75     .... "trunk/PIAS/source/serial.pas"] ..... .tr("PIAS_D/source/serial.d");
76     .... ["trunk/PIAS/source/shmem.pas"] ..... .tr("PIAS_D/source/shmem.d");

```

dub
app

dub
lib

svn
Pascal

git
rdmd
transpile.d

git

```
5 immutable SysTime transpilerTime;
6 static this() { transpilerTime = timeLastModified("Pascal2D/pascal2d.exe"); }
7
8 void tr(string[] from, string to)
9 in (from.length > 0 && from.length <= 2)
10 {
11     immutable targetTime = timeLastModified(to, SysTime.min);
12     if (targetTime > transpilerTime &&
13         from.all!(a => targetTime > timeLastModified(a)))
14         return;
15     stderr.writeln("=== transpiling \"~ to.baseName ~\"");
16     (spawnProcess(["Pascal2D/pascal2d.exe"] ~ from, std.stdio.stdin,
17         File(to, "w")).wait == 0).enforce("Parse error.");
18     stderr.writeln("=== formatting \"~ to.baseName ~\"");
19     (spawnProcess(["dub", "run", "dfmt", "--", "--inplace", to]).wait == 0).enforce;
20 }
```


dub
app

dub
lib

svn
Pascal

git
rdmd
transpile.d

git

pascal2d

epcompat

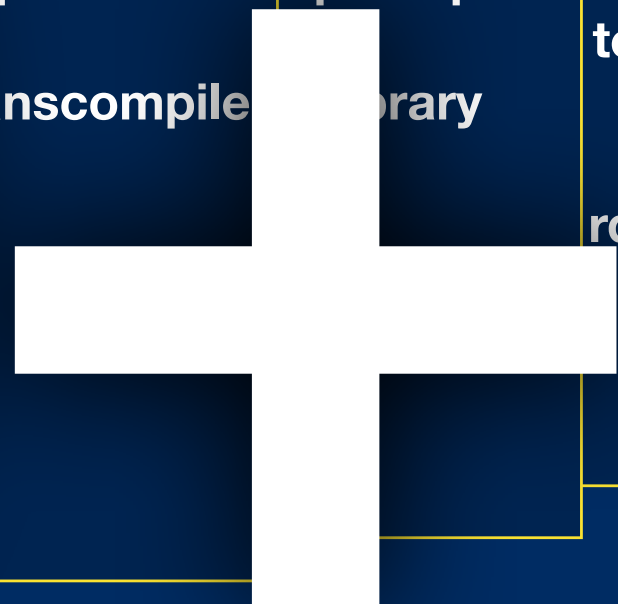
tests

transcompile

library

rdmd

40



1. Case sensitivity
2. Use of identifiers that are D keywords
3. Type conversions that are implicit in Pascal but need a cast in D
4. Dealing with mutable strings
5. Calls into the WinAPI that need an `&` on arguments in D



Passes

- read files
- lex
- parse
- create symbol table
- semantic 1
- semantic 2
- semantic 3
- inline
- glue
- optimize
- generate code
- write to object file

Windows systeemfoutmelding



Windows foutnummer 1812 (0x0000000000000714). De omschrijving hiervan luidt: The specified image file did not contain a resource section.

PIAS actie op dat moment: Ywin>Loading PIAS icon.
Dit is geen foutmelding van PIAS, maar van Windows.

OK

Command Prompt



```
C:\Users\bastiaan\Documents\D\PIAS_in_D\test\genmenu>test_genmenu.exe_
```



Tips & Tricks




```
1 program test;
2
3 procedure foo;
4 begin
5     ... {Complicated stuff}
6 end;
7
8 begin
9 end.
10
```

```
1 // Program name: test
2 void foo()
3 {
4     ... //Complicated stuff
5 }
6
7 void main(string[] args)
8 {
9 }
10
```

```
1 program test;
2
3 procedure foo;
4 begin
5     ... {$P2D_substitute_begin}
6     ... {Complicated stuff}
7     ... {$P2D_substitute_end}
8 end;
9
10 begin
11 end.
12
```

```
1 // Program name: test
2 void foo()
3 {
4 }
5
6 void main(string[] args)
7 {
8 }
9
```

```
1 program test;
2
3 procedure foo;
4 begin
5     ... {$P2D_substitute_begin
6     ... // Manual cleverness
7     ... }
8     ... {Complicated stuff}
9     ... {$P2D_substitute_end}
10 end;
11
12 begin
13 end.
14
```

```
1 // Program name: test
2 void foo()
3 {
4     ... // Manual cleverness
5 }
6
7 void main(string[] args)
8 {
9 }
10
```

```
1 program test;
2
3 procedure foo;
4 begin
5     ... {$P2D_substitute_begin
6     ... // Manual cleverness
7     ... }
8     ... {Complicated stuff}
9     ... {$P2D_substitute_end}
10 end;
11
12 begin
13     ... {$P2D_insert import std;
14     ... writeln("Running D.");}
15 end.
16
```

```
1 // Program name: test
2 void foo()
3 {
4     ... // Manual cleverness
5 }
6
7 void main(string[] args)
8 {
9     ... import std;
10
11     ... writeln("Running D.");
12 }
13
```

```
1 program test;
2
3 {$P2D_substitute_begin
4 void foo()
5 |
6 .....// Manual cleverness .
7 |
8 }
9 procedure foo;
10 begin
11 .....{Complicated stuff}
12 end;
13 {$P2D_substitute_end}
14
15 begin
16 end.
17
```

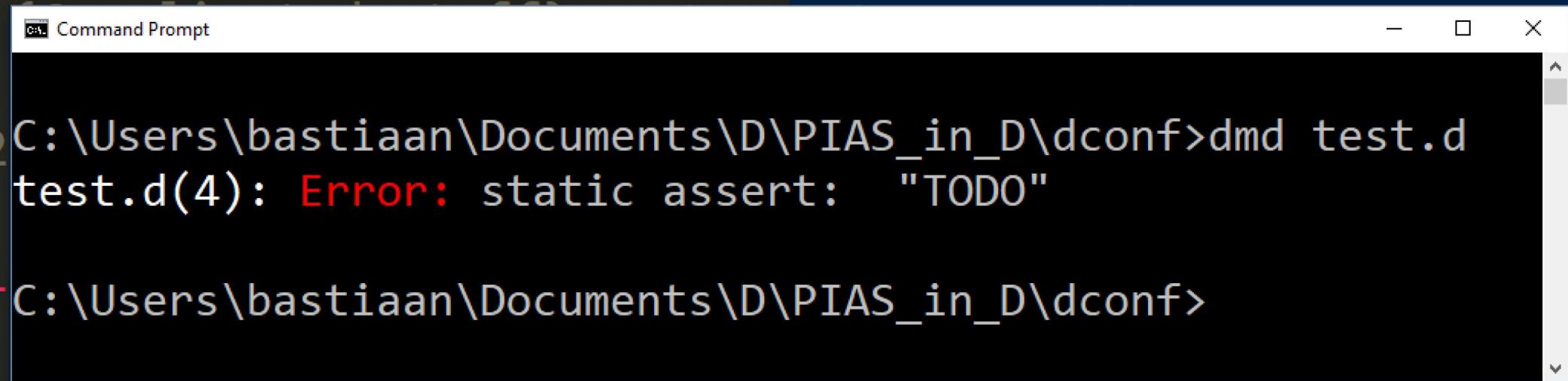
```
1 // Program name: test
2 void foo()
3 {
4 .....// Manual cleverness .
5 }
6
7 void main(string[] args)
8 {
9 }
10
```

```
1 program test;
2
3 {$P2D_substitute_begin
4 void foo()
5 |
6 ... assert(false, "TODO");
7 |
8 }
9 procedure foo;
10 begin
11 ... {Complicated stuff}
12 end;
13 {$P2D_substitute_end}
14
15 begin
16 end.
17
```

```
1 // Program name: test
2 void foo()
3 {
4 ... assert(false, "TODO");
5 }
6
7 void main(string[] args)
8 {
9 }
10
```

```
1 program test;
2
3 {$P2D_substitute_begin
4 void foo()
5 {
6     ... static assert(false, "TODO")
7 }
8 }
9 procedure foo;
10 begin
11     ...
12 end;
13 {$P2
14
15 begi
16 end.
17
```

```
1 // Program name: test
2 void foo()
3 {
4     ... static assert(false, "TODO")
5 }
6
7 void main(string[] args)
8 {
9 }
10
```



```
1 program test;
2
3 {$P2D_substitute_begin
4 deprecated("TODO")
5 void foo()
6 {
7     ... assert(false, "TODO");
8 }
9 }
```

```
10 procedure foo;
```

```
11 begi
```

```
12 ...
```

```
13 end;
```

```
14 {$P2
```

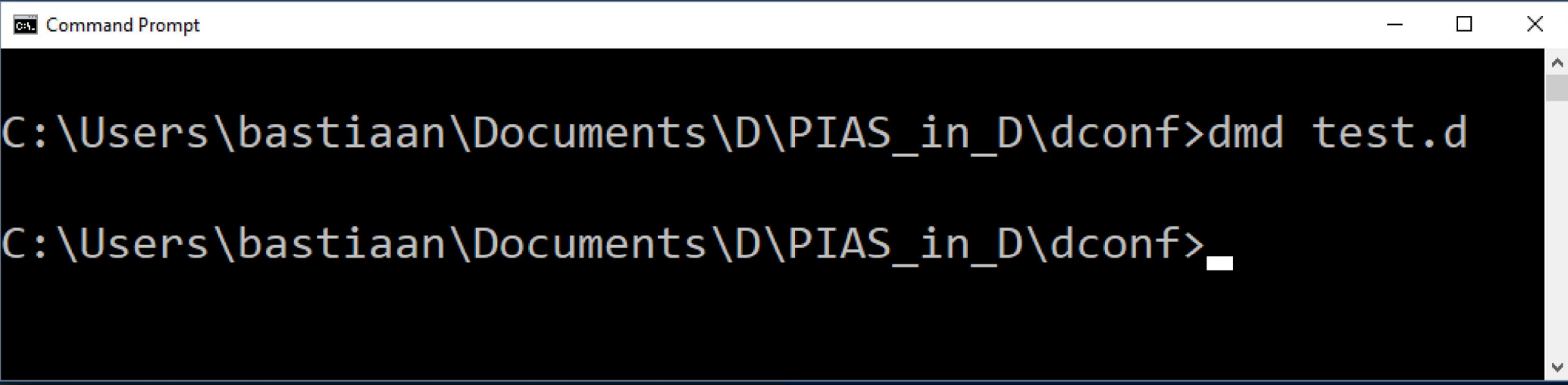
```
15
```

```
16 begi
```

```
17 end.
```

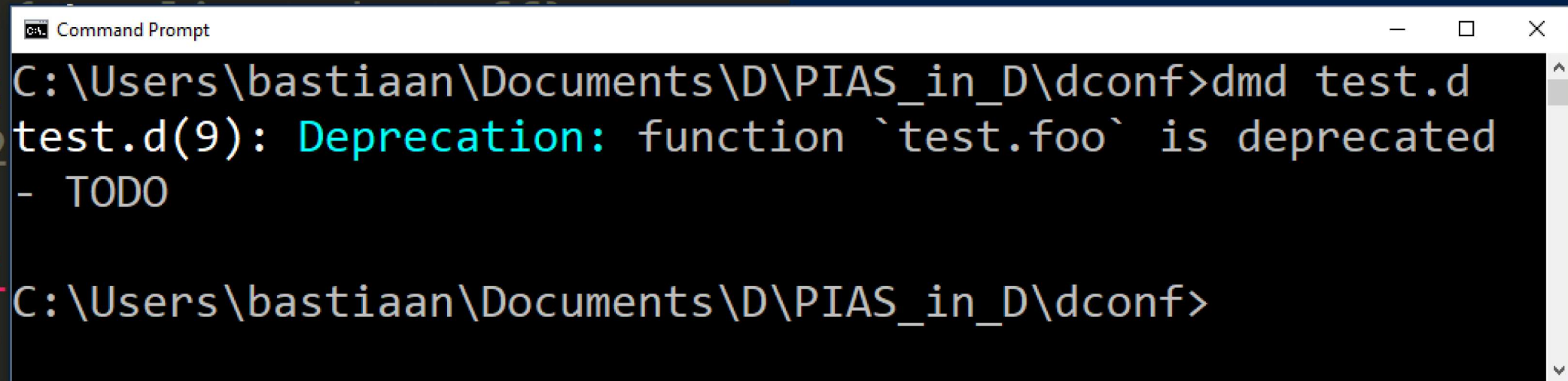
```
18
```

```
1 // Program name: test
2 deprecated("TODO") void foo()
3 {
4     ... assert(false, "TODO");
5 }
6
7 void main(string[] args)
8 {
9 }
10
```




```
1 program test;
2
3 {$P2D_substitute_begin
4 deprecated("TODO") void foo()
5 }
6 ... assert(false, "TODO");
7 }
8 }
9 procedure foo;
10 begin
11 ...
12 end;
13 {$P2
14
15 begi
16 ...
17 end.
18
```

```
1 // Program name: test
2 deprecated("TODO") void foo()
3 {
4 ... assert(false, "TODO");
5 }
6
7 void main(string[] args)
8 {
9 ... foo;
10 }
```



```
1 program test(output);
2
3 var i, j : boolean;
4 begin
5     if i and j then
6         writeln('both true');
7     end.
8
```

```
1 import epcompat;
2 import std.stdio;
3 import std.traits : isCallable, ReturnTy
4
5 // Program name: test
6 bool i, j;
7 void main(string[] args)
```

Command Prompt

```
test ~master: building configuration "application"...
test.d(9,5): Error: static assert: "Cannot be bitwise factor, AND should be AND_THEN in the Pascal source."
C:\D\dmd2\windows\bin\dmd.exe failed with exit code 1.
```

```
C:\Users\bastiaan\Documents\D\PIAS_in_D\dconf>
```

```
14     ....."Cannot be bitwise factor, A
15     if (i & j)
16     {
17         ....."both true");
18     }
19 }
```

```
1 program test(output);
2
3 var i, j : boolean;
4 begin
5     if i and then j then
6         writeln('both true');
7 end.
8
```

```
1 import epcompat;
2 import std.stdio;
3
4 // Program name: test
5 bool i, j;
6 void main(string[] args)
7 {
8     if (i && j)
9         writeln("both true");
10 }
11
```

```
1 program test(output);
2
3 var i : boolean;
4 begin
5     if not i then
6         writeln('not i');
7     end.
8
```

```
1 import epcompat;
2 import std.stdio;
3 import std.traits : isBoolean, Bool
4
5 // Program name: test
6 bool i;
7 void main(string[] args)
8 {
9     static assert(is(BooleanTypeOf!
10     | | | | | | | | | | (isCallable!i &&
11     | | | | | | | | | | "Boolean type tes
12     if (!i)
13     {
14     | | | | | | | | | | writeln("not i");
15     }
16 }
17
```

```
1 program test(output);
2
3 var i, j : integer;
4 begin
5     j := not i;
6 end.
7
```

```
1 import std.stdio;
2 import std.traits :: isBoolean, BooleanTypeOf, is
3
4 // Program name: test
5 int i, j;
6 void main(string[] args)
7 {
8     static assert(is(BooleanTypeOf!(typeof(i)))
9     | (isCallable!i && isBoolean!(Re
10     | "Boolean type test failed. If
11     j = !i;
12 }
```

Command Prompt

```
test.d(8,5): Error: static assert: "Boolean type test failed. If it should be Boolean then make
sure to not use a custom type like "boolean0". To force bitwise negation, use "{$P2D_bitwise}NO
T" in the Pascal source."
```

```
C:\D\dmd2\windows\bin\dmd.exe failed with exit code 1.
```

```
C:\Users\bastiaan\Documents\D\PIAS_in_D\dconf>
```

```
1 program test(output);
2
3 var i, j : integer;
4 begin
5     j := {$P2D_bitwise}not i;
6 end.
7
```

```
1 import std.stdio;
2 import std.traits : isCallable, ReturnTy
3
4 // Program name: test
5 int i, j;
6 void main(string[] args)
7 {
8     static assert(is(IntegralTypeOf!(typ
9     |.....|.....|.....|.....| (isCallable!i && isInt
10    |.....|.....|.....|.....| "Cannot be bitwise neg
11    |.....|.....|.....|.....| j = ~i;
12 }
13
```

```
1 program test:
```

```
1 // Program name: test
```

Command Prompt



```
test.d(13,13): Error: function test.call_dlg(void delegate() dlg) is not callable using argument  
types (void)
```

```
test.d(13,13): cannot pass argument my_dlg() of type void to parameter void delegate() dl  
g
```

```
C:\D\dmd2\windows\bin\dmd.exe failed with exit code 1.
```

```
C:\Users\bastiaan\Documents\D\PIAS_in_D\dconf>
```

```
8 procedure too;
```

```
9  
10 procedure my_dlg;
```

```
11 begin
```

```
12 end;
```

```
14 begin
```

```
15 call_dlg(my_dlg);
```

```
16 end;
```

```
18 begin
```

```
19 end.
```

```
20
```

```
8
```

```
{
```

```
9
```

```
void my_dlg()
```

```
10
```

```
{
```

```
11
```

```
}
```

```
12
```

```
13
```

```
call_dlg(my_dlg);
```

```
14
```

```
}
```

```
15
```

```
16
```

```
void main(string[] args)
```

```
17
```

```
{
```

```
18
```

```
}
```

```
19
```

```
1 program test;
2
3 procedure call_dlg(procedure dlg);
4 begin
5     ... dlg;
6 end;
7
8 procedure foo;
9
10     procedure my_dlg;
11     begin
12     end;
13
14 begin
15     call_dlg({$P2D_proc}my_dlg);
16 end;
17
18 begin
19 end.
20
```

```
1 // Program name: test
2 void call_dlg(void delegate() dlg)
3 {
4     ... dlg;
5 }
6
7 void foo()
8 {
9     ... void my_dlg()
10     {
11     }
12
13     call_dlg(&my_dlg);
14 }
15
16 void main(string[] args)
17 {
18 }
19
```



```
test.d(13,13): Error: function test.call_dlg(void delegate() dlg) is not callable using argument
types (void function())
test.d(13,13):      cannot pass argument & my_dlg of type void function() to parameter void de
legate() dlg
C:\D\dmd2\windows\bin\dmd.exe failed with exit code 1.

C:\Users\bastiaan\Documents\D\PIAS_in_D\dconf>_
```

```
8 procedure my_dlg;
9 begin
10 end;
11
12 procedure foo;
13 begin
14     ... call_dlg({$P2D_proc}my_dlg);
15 end;
16
17 begin
18 end.
19
```

```
7 void my_dlg()
8 {
9 }
10
11 void foo()
12 {
13     ... call_dlg(&my_dlg);
14 }
15
16 void main(string[] args)
17 {
18 }
19
```

```
1 program test;
2
3 procedure call_dlg(procedure dlg);
4 begin
5     ... dlg;
6 end;
7
8 procedure my_dlg;
9 begin
10 end;
11
12 procedure foo;
13 begin
14     ... call_dlg({$P2D_toDelegate}my_d
15 end;
16
17 begin
18 end.
19
```

```
1 import std.functional : toDelegate;
2
3 // Program name: test
4 void call_dlg(void delegate() dlg)
5 {
6     ... dlg;
7 }
8
9 void my_dlg()
10 {
11 }
12
13 void foo()
14 {
15     ... call_dlg(toDelegate(&my_dlg));
16 }
17
18 void main(string[] args)
19 {
```

{P2D_preserve_index}

{P2D_UDA}

{P2D_string}

{P2D_cat}

{P2D_array}

{P2D_packed}

{P2D_method}

Strings

1. Mutability (in place manipulation and passing to WinAPI)
2. Indexing
3. file i/o
4. winAPI (UTF16 W vs A, taking address of slice, lifetime of toUTF16z)
5. fixed capacity
6. String pointers and newing strings

Take any small opportunity
do your best
to make it a bigger one