

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Estructura de datos



MANUAL TÉCNICO

Práctica

Estudiante: José Luis Saloj

Carné: 201900081

Objetivos

- Hacer un uso correcto de la memoria dinámica y apuntadores en el lenguaje de programación C++.
- Aplicar los conocimientos adquiridos sobre estructuras de datos lineales.
- Utilizar la herramienta graphviz para la generación de reportes.

Partes del programa

Para ejecutar el programa debe escribir el siguiente código en la terminal de visual code. Con esto se crea un ejecutable

g++ main.cpp -o main.exe

El programa posee un archivo llamado “main.cpp”, la cual funciona como elemento de ejecución y funcionamiento del programa.

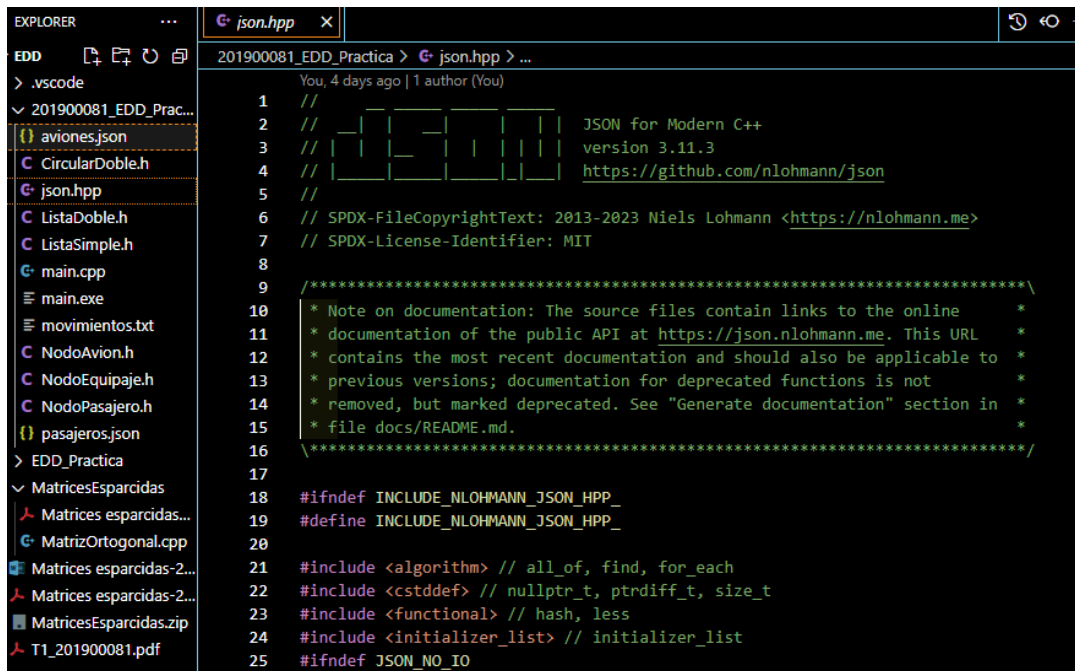
```
12 #include "json.hpp" // Incluir la biblioteca json.hpp
13
14 #include "CircularDoble.h"
15 #include "ListaSimple.h"
16 #include "ListaDoble.h"
17
18 using json = nlohmann::json;
19 using namespace std;
20
21 CircularDoble listaDisponible;
22 CircularDoble listaMantenimiento;
23
24 ListaSimple listaPasajeros;
25 ListaSimple listaConsulta;
26 ListaDoble listaEquipaje;
27 ListaDoble listaPasajerosOrdenados;
28
29 > void menu(){...
38
39 > void reportes(){...
48
49 > void reportesDespligue(){...
90
91 > void procesarLinea(const std::string& linea) {...
145
146 > void leerArchivoTxt(const std::string& nombreArchivo) {...
161
162 > void leerPasajeros(const std::string& nombreArchivo) {...
189
190
191 > void leerAviones(const std::string& nombreArchivo) {...
224
225 > int main() {...
```

Debe incluir la librería:

`#include "json.hpp"`

Sirve para la lectura de los archivos con formato “.json”

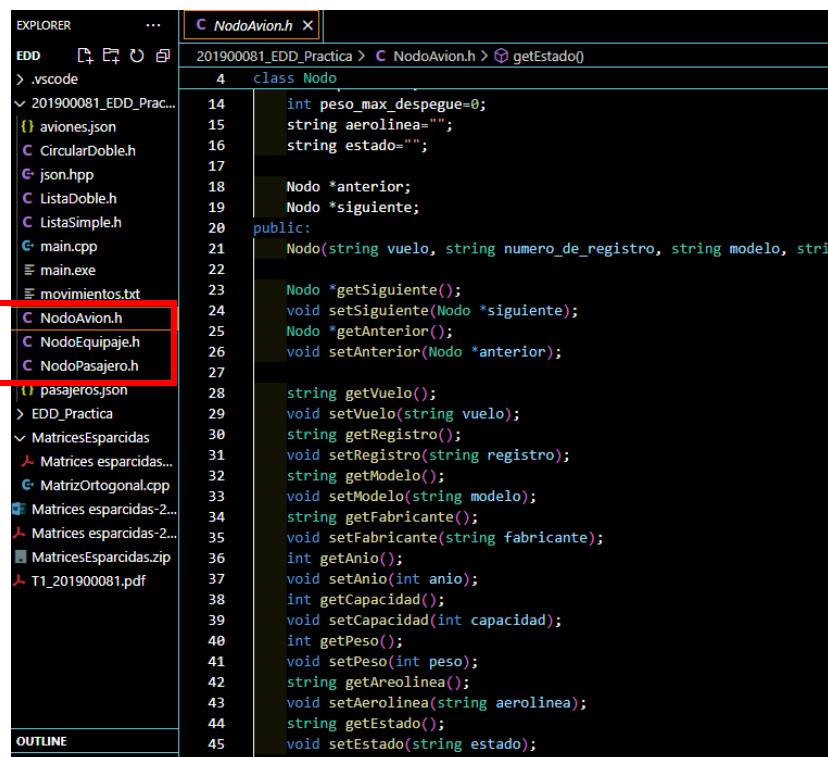
Para garantizar la lectura de los archivos de entradas se debe copiar las configuraciones de la librería “json” en la carpeta donde se desarrolla la práctica.



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left and the Editor window on the right. The Explorer sidebar shows a project structure with folders like '201900081_EDD_Practica' and files like 'aviones.json', 'CircularDoble.h', 'json.hpp', 'ListaDoble.h', 'ListaSimple.h', 'main.cpp', 'main.exe', 'movimientos.txt', 'NodoAvion.h', 'NodoEquipaje.h', 'NodoPasajero.h', 'pasajeros.json', 'MatricesEscarpidas', 'Matrices esparcidas-2...', 'Matrices esparcidas-2...', 'MatricesEscarpidas.zip', and 'T1_201900081.pdf'. The Editor window shows the content of 'json.hpp', which is a header file for the 'json' library. The code includes comments about the library's version (3.11.3) and license (MIT), and a block of documentation text. The code is as follows:

```
1 //
2 // _ _ _ _ _ JSON for Modern C++
3 // | | | | | version 3.11.3
4 // | | | | | https://github.com/nlohmann/json
5 //
6 // SPDX-FileCopyrightText: 2013-2023 Niels Lohmann <https://nlohmann.me>
7 // SPDX-License-Identifier: MIT
8
9 /*****
10  * Note on documentation: The source files contain links to the online
11  * documentation of the public API at https://json.nlohmann.me. This URL
12  * contains the most recent documentation and should also be applicable to
13  * previous versions; documentation for deprecated functions is not
14  * removed, but marked deprecated. See "Generate documentation" section in
15  * file docs/README.md.
16  *****/
17
18 #ifndef INCLUDE_NLOHMANN_JSON_HPP_
19 #define INCLUDE_NLOHMANN_JSON_HPP_
20
21 #include <algorithm> // all_of, find, for_each
22 #include <cstdint> // nullptr_t, ptrdiff_t, size_t
23 #include <functional> // hash, less
24 #include <initializer_list> // initializer_list
25 #ifndef JSON_NO_IO
```

Los archivos con extensión “.h” y nombre “Nodo” son los encargados de recibir y manipular los datos de los nodos a trabajar.



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left and the Editor window on the right. The Explorer sidebar shows the same project structure as the previous screenshot. The Editor window shows the content of 'NodoAvion.h', which is a header file for the 'Nodo' class. The code is as follows:

```
1 class Nodo
2 {
3     14 int peso_max_despegue=0;
4     15 string aerolinea="";
5     16 string estado="";
6     17
7     18 Nodo *anterior;
8     19 Nodo *siguiente;
9     20
10    21 public:
11    22     Nodo(string vuelo, string numero_de_registro, string modelo, string
12    23
13    24     Nodo *getSiguiente();
14    25     void setSiguiente(Nodo *siguiente);
15    26     Nodo *getAnterior();
16    27     void setAnterior(Nodo *anterior);
17    28
18    29     string getVuelo();
19    30     void setVuelo(string vuelo);
20    31     string getRegistro();
21    32     void setRegistro(string registro);
22    33     string getModelo();
23    34     void setModelo(string modelo);
24    35     string getFabricante();
25    36     void setFabricante(string fabricante);
26    37     int getAnio();
27    38     void setAnio(int anio);
28    39     int getCapacidad();
29    40     void setCapacidad(int capacidad);
30    41     int getPeso();
31    42     void setPeso(int peso);
32    43     string getAerolinea();
33    44     void setAerolinea(string aerolinea);
34    45     string getEstado();
35    46     void setEstado(string estado);
36    47
37    48
38    49
39    50
40    51
41    52
42    53
43    54
44    55
45    56
46    57
47    58
48    59
49    60
50    61
51    62
52    63
53    64
54    65
55    66
56    67
57    68
58    69
59    70
60    71
61    72
62    73
63    74
64    75
65    76
66    77
67    78
68    79
69    80
70    81
71    82
72    83
73    84
74    85
75    86
76    87
77    88
78    89
79    90
80    91
81    92
82    93
83    94
84    95
85    96
86    97
87    98
88    99
90    100
91    101
92    102
93    103
94    104
95    105
96    106
97    107
98    108
99    109
100   110
101   111
102   112
103   113
104   114
105   115
106   116
107   117
108   118
109   119
110   120
111   121
112   122
113   123
114   124
115   125
116   126
117   127
118   128
119   129
120   130
121   131
122   132
123   133
124   134
125   135
126   136
127   137
128   138
129   139
130   140
131   141
132   142
133   143
134   144
135   145
136   146
137   147
138   148
139   149
140   150
141   151
142   152
143   153
144   154
145   155
146   156
147   157
148   158
149   159
150   160
151   161
152   162
153   163
154   164
155   165
156   166
157   167
158   168
159   169
160   170
161   171
162   172
163   173
164   174
165   175
166   176
167   177
168   178
169   179
170   180
171   181
172   182
173   183
174   184
175   185
176   186
177   187
178   188
179   189
180   190
181   191
182   192
183   193
184   194
185   195
186   196
187   197
188   198
189   199
190   200
191   201
192   202
193   203
194   204
195   205
196   206
197   207
198   208
199   209
200   210
201   211
202   212
203   213
204   214
205   215
206   216
207   217
208   218
209   219
210   220
211   221
212   222
213   223
214   224
215   225
216   226
217   227
218   228
219   229
220   230
221   231
222   232
223   233
224   234
225   235
226   236
227   237
228   238
229   239
230   240
231   241
232   242
233   243
234   244
235   245
236   246
237   247
238   248
239   249
240   250
241   251
242   252
243   253
244   254
245   255
246   256
247   257
248   258
249   259
250   260
251   261
252   262
253   263
254   264
255   265
256   266
257   267
258   268
259   269
260   270
261   271
262   272
263   273
264   274
265   275
266   276
267   277
268   278
269   279
270   280
271   281
272   282
273   283
274   284
275   285
276   286
277   287
278   288
279   289
280   290
281   291
282   292
283   293
284   294
285   295
286   296
287   297
288   298
289   299
290   300
291   301
292   302
293   303
294   304
295   305
296   306
297   307
298   308
299   309
300   310
301   311
302   312
303   313
304   314
305   315
306   316
307   317
308   318
309   319
310   320
311   321
312   322
313   323
314   324
315   325
316   326
317   327
318   328
319   329
320   330
321   331
322   332
323   333
324   334
325   335
326   336
327   337
328   338
329   339
330   340
331   341
332   342
333   343
334   344
335   345
336   346
337   347
338   348
339   349
340   350
341   351
342   352
343   353
344   354
345   355
346   356
347   357
348   358
349   359
350   360
351   361
352   362
353   363
354   364
355   365
356   366
357   367
358   368
359   369
360   370
361   371
362   372
363   373
364   374
365   375
366   376
367   377
368   378
369   379
370   380
371   381
372   382
373   383
374   384
375   385
376   386
377   387
378   388
379   389
380   390
381   391
382   392
383   393
384   394
385   395
386   396
387   397
388   398
389   399
390   400
391   401
392   402
393   403
394   404
395   405
396   406
397   407
398   408
399   409
400   410
401   411
402   412
403   413
404   414
405   415
406   416
407   417
408   418
409   419
410   420
411   421
412   422
413   423
414   424
415   425
416   426
417   427
418   428
419   429
420   430
421   431
422   432
423   433
424   434
425   435
426   436
427   437
428   438
429   439
430   440
431   441
432   442
433   443
434   444
435   445
436   446
437   447
438   448
439   449
440   450
441   451
442   452
443   453
444   454
445   455
446   456
447   457
448   458
449   459
450   460
451   461
452   462
453   463
454   464
455   465
456   466
457   467
458   468
459   469
460   470
461   471
462   472
463   473
464   474
465   475
466   476
467   477
468   478
469   479
470   480
471   481
472   482
473   483
474   484
475   485
476   486
477   487
478   488
479   489
480   490
481   491
482   492
483   493
484   494
485   495
486   496
487   497
488   498
489   499
490   500
491   501
492   502
493   503
494   504
495   505
496   506
497   507
498   508
499   509
500   510
501   511
502   512
503   513
504   514
505   515
506   516
507   517
508   518
509   519
510   520
511   521
512   522
513   523
514   524
515   525
516   526
517   527
518   528
519   529
520   530
521   531
522   532
523   533
524   534
525   535
526   536
527   537
528   538
529   539
530   540
531   541
532   542
533   543
534   544
535   545
536   546
537   547
538   548
539   549
540   550
541   551
542   552
543   553
544   554
545   555
546   556
547   557
548   558
549   559
550   560
551   561
552   562
553   563
554   564
555   565
556   566
557   567
558   568
559   569
560   570
561   571
562   572
563   573
564   574
565   575
566   576
567   577
568   578
569   579
570   580
571   581
572   582
573   583
574   584
575   585
576   586
577   587
578   588
579   589
580   590
581   591
582   592
583   593
584   594
585   595
586   596
587   597
588   598
589   599
590   600
591   601
592   602
593   603
594   604
595   605
596   606
597   607
598   608
599   609
600   610
601   611
602   612
603   613
604   614
605   615
606   616
607   617
608   618
609   619
610   620
611   621
612   622
613   623
614   624
615   625
616   626
617   627
618   628
619   629
620   630
621   631
622   632
623   633
624   634
625   635
626   636
627   637
628   638
629   639
630   640
631   641
632   642
633   643
634   644
635   645
636   646
637   647
638   648
639   649
640   650
641   651
642   652
643   653
644   654
645   655
646   656
647   657
648   658
649   659
650   660
651   661
652   662
653   663
654   664
655   665
656   666
657   667
658   668
659   669
660   670
661   671
662   672
663   673
664   674
665   675
666   676
667   677
668   678
669   679
670   680
671   681
672   682
673   683
674   684
675   685
676   686
677   687
678   688
679   689
680   690
681   691
682   692
683   693
684   694
685   695
686   696
687   697
688   698
689   699
690   700
691   701
692   702
693   703
694   704
695   705
696   706
697   707
698   708
699   709
700   710
701   711
702   712
703   713
704   714
705   715
706   716
707   717
708   718
709   719
710   720
711   721
712   722
713   723
714   724
715   725
716   726
717   727
718   728
719   729
720   730
721   731
722   732
723   733
724   734
725   735
726   736
727   737
728   738
729   739
730   740
731   741
732   742
733   743
734   744
735   745
736   746
737   747
738   748
739   749
740   750
741   751
742   752
743   753
744   754
745   755
746   756
747   757
748   758
749   759
750   760
751   761
752   762
753   763
754   764
755   765
756   766
757   767
758   768
759   769
760   770
761   771
762   772
763   773
764   774
765   775
766   776
767   777
768   778
769   779
770   780
771   781
772   782
773   783
774   784
775   785
776   786
777   787
778   788
779   789
780   790
781   791
782   792
783   793
784   794
785   795
786   796
787   797
788   798
789   799
790   800
791   801
792   802
793   803
794   804
795   805
796   806
797   807
798   808
799   809
800   810
801   811
802   812
803   813
804   814
805   815
806   816
807   817
808   818
809   819
810   820
811   821
812   822
813   823
814   824
815   825
816   826
817   827
818   828
819   829
820   830
821   831
822   832
823   833
824   834
825   835
826   836
827   837
828   838
829   839
830   840
831   841
832   842
833   843
834   844
835   845
836   846
837   847
838   848
839   849
840   850
841   851
842   852
843   853
844   854
845   855
846   856
847   857
848   858
849   859
850   860
851   861
852   862
853   863
854   864
855   865
856   866
857   867
858   868
859   869
860   870
861   871
862   872
863   873
864   874
865   875
866   876
867   877
868   878
869   879
870   880
871   881
872   882
873   883
874   884
875   885
876   886
877   887
878   888
879   889
880   890
881   891
882   892
883   893
884   894
885   895
886   896
887   897
888   898
889   899
890   900
891   901
892   902
893   903
894   904
895   905
896   906
897   907
898   908
899   909
900   910
901   911
902   912
903   913
904   914
905   915
906   916
907   917
908   918
909   919
910   920
911   921
912   922
913   923
914   924
915   925
916   926
917   927
918   928
919   929
920   930
921   931
922   932
923   933
924   934
925   935
926   936
927   937
928   938
929   939
930   940
931   941
932   942
933   943
934   944
935   945
936   946
937   947
938   948
939   949
940   950
941   951
942   952
943   953
944   954
945   955
946   956
947   957
948   958
949   959
950   960
951   961
952   962
953   963
954   964
955   965
956   966
957   967
958   968
959   969
960   970
961   971
962   972
963   973
964   974
965   975
966   976
967   977
968   978
969   979
970   980
971   981
972   982
973   983
974   984
975   985
976   986
977   987
978   988
979   989
980   990
981   991
982   992
983   993
984   994
985   995
986   996
987   997
988   998
989   999
1000  1000
1001  1001
1002  1002
1003  1003
1004  1004
1005  1005
1006  1006
1007  1007
1008  1008
1009  1009
1010  1010
1011  1011
1012  1012
1013  1013
1014  1014
1015  1015
1016  1016
1017  1017
1018  1018
1019  1019
1020  1020
1021  1021
1022  1022
1023  1023
1024  1024
1025  1025
1026  1026
1027  1027
1028  1028
1029  1029
1030  1030
1031  1031
1032  1032
1033  1033
1034  1034
1035  1035
1036  1036
1037  1037
1038  1038
1039  1039
1040  1040
1041  1041
1042  1042
1043  1043
1044  1044
1045  1045
1046  1046
1047  1047
1048  1048
1049  1049
1050  1050
1051  1051
1052  1052
1053  1053
1054  1054
1055  1055
1056  1056
1057  1057
1058  1058
1059  1059
1060  1060
1061  1061
1062  1062
1063  1063
1064  1064
1065  1065
1066  1066
1067  1067
1068  1068
1069  1069
1070  1070
1071  1071
1072  1072
1073  1073
1074  1074
1075  1075
1076  1076
1077  1077
1078  1078
1079  1079
1080  1080
1081  1081
1082  1082
1083  1083
1084  1084
1085  1085
1086  1086
1087  1087
1088  1088
1089  1089
1090  1090
1091  1091
1092  1092
1093  1093
1094  1094
1095  1095
1096  1096
1097  1097
1098  1098
1099  1099
1100  1100
1101  1101
1102  1102
1103  1103
1104  1104
1105  1105
1106  1106
1107  1107
1108  1108
1109  1109
1110  1110
1111  1111
1112  1112
1113  1113
1114  1114
1115  1115
1116  1116
1117  1117
1118  1118
1119  1119
1120  1120
1121  1121
1122  1122
1123  1123
1124  1124
1125  1125
1126  1126
1127  1127
1128  1128
1129  1129
1130  1130
1131  1131
1132  1132
1133  1133
1134  1134
1135  1135
1136  1136
1137  1137
1138  1138
1139  1139
1140  1140
1141  1141
1142  1142
1143  1143
1144  1144
1145  1145
1146  1146
1147  1147
1148  1148
1149  1149
1150  1150
1151  1151
1152  1152
1153  1153
1154  1154
1155  1155
1156  1156
1157  1157
1158  1158
1159  1159
1160  1160
1161  1161
1162  1162
1163  1163
1164  1164
1165  1165
1166  1166
1167  1167
1168  1168
1169  1169
1170  1170
1171  1171
1172  1172
1173  1173
1174  1174
1175  1175
1176  1176
1177  1177
1178  1178
1179  1179
1180  1180
1181  1181
1182  1182
1183  1183
1184  1184
1185  1185
1186  1186
1187  1187
1188  1188
1189  1189
1190  1190
1191  1191
1192  1192
1193  1193
1194  1194
1195  1195
1196  1196
1197  1197
1198  1198
1199  1199
1200  1200
1201  1201
1202  1202
1203  1203
1204  1204
1205  1205
1206  1206
1207  1207
1208  1208
1209  1209
1210  1210
1211  1211
1212  1212
1213  1213
1214  1214
1215  1215
1216  1216
1217  1217
1218  1218
1219  1219
1220  1220
1221  1221
1222  1222
1223  1223
1224  1224
1225  1225
1226  1226
1227  1227
1228  1228
1229  1229
1230  1230
1231  1231
1232  1232
1233  1233
1234  1234
1235  1235
1236  1
```

El archivo de nombre “CircularDoble.h” se encarga de darle funcionamiento a las listas circulares utilizados para el almacenamiento y visualización de la información de los estados de los aviones.

El archivo de nombre “ListaDoble.h” se encarga del almacenamiento de la información de los pasajeros.

El archivo de nombre “ListaSimple.h” funciona como un contenedor de los datos iniciales de ingreso de pasajeros.



```
122 void ListaSimple::visualizarLista(){
140     actual = actual->getSiguiente();
141 }
142 }
143 }
144
145 void ListaSimple::visualizarListaDot() { You, 20 hours ago * parte graphviz de col
146     std::ofstream archivo("lista_pasajeros.dot");
147
148     archivo << "digraph ListaPasajeros {" << std::endl;
149     archivo << "    rankdir=LR;" << std::endl; // Opcional: Organiza los nodos de izqui
150
151     if (estaVacia()) {
152         archivo << "        vacia [label=\"La lista está vacía\", shape=box];" << std::endl
153     } else {
154         NodoPasajero* actual = primero;
155         while (actual != nullptr) {
156             archivo << "        \"" << actual->getNombre() << "\" [label=\"" << actual->get
157                 if (actual->getSiguiente() != nullptr) {
158                     archivo << "        \"" << actual->getNombre() << "\" -> \"" << actual->get
159                 }
160             actual = actual->getSiguiente();
161         }
162     }
```