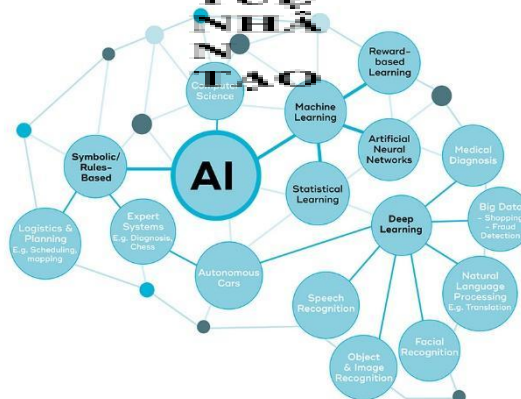


**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN**



**BỘ
CƠ
CẤU
HỌC
TẬP**

**MÔN
HỌC:
TRÍ
TUE
NHÀ
ĐÀO
TẠO**



Giảng viên bộ môn : Nguyễn Đình Cường
Sinh viên thực hiện : Nguyễn Văn Hải Long
Lớp : 60.CNTT-1
MSSV : 60136035

Tháng 8 năm 2021

PHỤ LỤC

1. Programming Heuristic	3
1.1. Thuật toán tìm kiếm A*(A star)	3
1.2. Chương trình demo: Puzzle 8 sử dụng thuật toán A*	3
2. Theory of game (Lý thuyết về trò chơi)	6
2.1. Thuật toán Minimax	6
2.2. Tinh chỉnh Alpha-beta (cắt tỉa Alpha- beta)	7
2.3. Chương trình demo: Cờ vua dùng thuật toán Minimax	7
3. Logic and Semantic (Logic và ngữ nghĩa)	9
3.1. Giải thuật Vương Hạo	9
3.2. Chương trình demo: Mô phỏng thuật toán Vương Hạo trong Prolog	10
4. Simulation (Mô phỏng)	12
4.1. Genetic Algorithm applied to optimization (Thuật toán di truyền áp dụng để tối ưu hoá)	12
4.2. Chương trình demo: Tìm đường đi sử dụng thuật toán di truyền	12
5. IoT and Robotics applied in with AI (IoT và Robotics được ứng dụng với AI)	14
5.1. Sử dụng Arduino để chế tạo cánh tay robot	14
5.2. Chương trình demo: Cánh tay robot	15

BÁO CÁO BÀI TẬP

1. Programming Heuristic

1.1. Thuật toán tìm kiếm A*(A star)

A* là một thuật toán tìm kiếm trong đồ thị. Thuật toán sẽ tìm đường từ 1 nút ban đầu đến 1 nút đích cho trước sao cho chi phí là tốt nhất (thấp nhất) và số bước duyệt là ít nhất (mang lại sự tối ưu cho bài toán).

A* là thuật toán cải thiện hiệu năng từ thuật toán Greedy Best-First Search. Khi Greedy Best-First Search chỉ tìm đường dựa trên hàm ước lượng từ điểm kế tiếp đến đích mà không tính độ dài quãng đường mà nó đã đi vì có thể ước lượng là nhỏ nhưng đường đã đi lại mang giá trị quá lớn.

Ý tưởng : Tránh việc xét các nhánh tìm kiếm đã xác định cho đến thời điểm hiện tại là có chi phí cao.

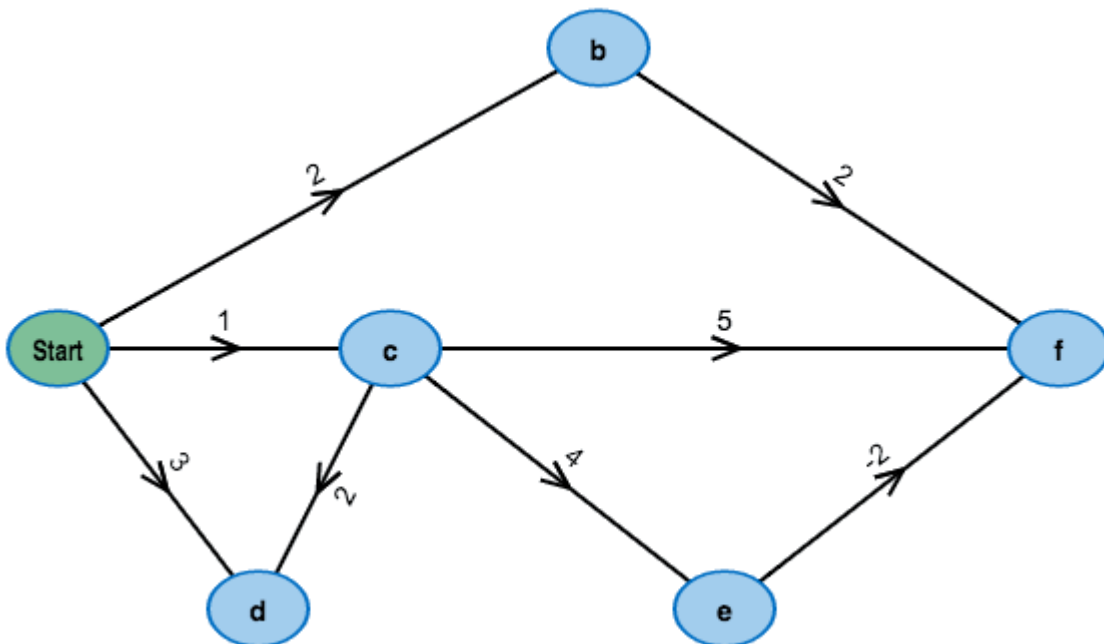
Sử dụng hàm đánh giá $f(x) = h(x) + g(x)$.

$h(x)$: là ước lượng từ nút hiện tại tới nút đích

$g(x)$: chi phí từ nút gốc tới nút hiện tại n

$f(x)$: là chi phí tổng thể ước lượng của đường đi qua nút hiện tại n đến đích.

Thuật toán A* sử dụng $f(x)$ làm tiêu chí chọn đỉnh tiếp theo trong đồ thị.



1.2. Chương trình demo: Puzzle 8 sử dụng thuật toán A*

Ngôn ngữ sử dụng: C#

Giao diện: Winform

Một số hình ảnh code:

+ Khởi tạo các giá trị đối tượng tính toán:

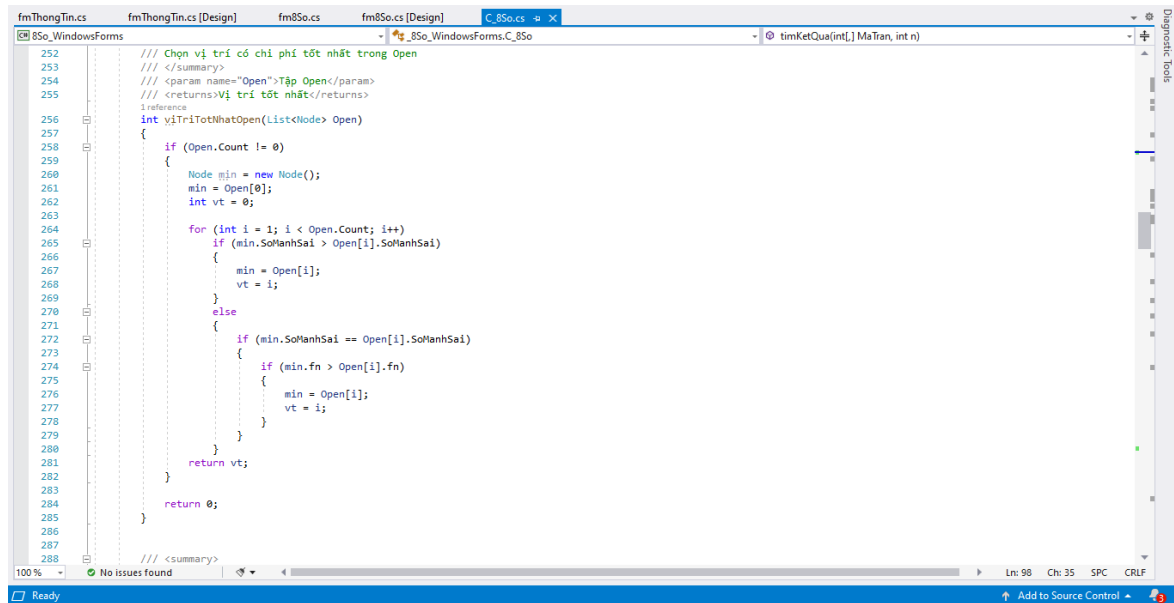
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace _8So_WindowsForms
7 {
8     35 references
9     public class Node
10     {
11         public int[,] MaTran; // ma trận 8 số
12         public int SoManhSai; // số mảnh sai vị trí của ma trận
13         public int ChiSo; // chỉ số của node
14         public int Cha; // cha của node, để truy vết kết quả
15         public int fn; // chỉ phí đi đến node đó
16     }
17
18     2 references
19     public class C_8So
20     {
21         private int ChiSo = 0; // chỉ số của Node sẽ tăng sau mỗi lần sinh ra 1 node
22         private int fn = 0; // Sau mỗi lần sinh ra các node thì chỉ phí các node tăng 1 đơn vị, tức node con sẽ có chỉ phí lớn hơn node cha 1 đơn vị
23         1 reference
24         public Stack<int[,]> timKetQua(int[,] MaTran, int n)
25         {
26             Stack<int[,]> gttKetQua = new Stack<int[,]>();
27
28             List<Node> Close = new List<Node>();
29             List<Node> Open = new List<Node>();
30
31             //khởi báo và khởi tạo cho node đầu tiên
32             Node tSo = new Node();
33             tSo.MaTran = MaTran;
34             tSo.SoManhSai = soMiangSaiViTri(MaTran);
35             tSo.ChiSo = 0;
36             tSo.Cha = -1;
37             tSo.fn = 0;
38         }
39     }
40 }
```

+ Điều chỉnh hướng di chuyển của đối tượng:

```
35 tSo.fn = 0;
36 //cho trạng thái đầu tiên vào Open;
37 Open.Add(tSo);
38
39 int t = 0;
40 while(Open.Count!=0)
41 {
42     1 reference
43     //chọn node tốt nhất trong tập Open và chuyển nó sang Close
44
45     //nếu node có số mảnh sai là 0, tức là đích thì thoát
46     if (tSo.SoManhSai == 0) break;
47     else
48     {
49         //sinh hướng đi của node hiện tại
50         List<Node> lstHuongDi = new List<Node>();
51         lstHuongDi = sinhHuongDi(tSo);
52
53         for (int i = 0; i < lstHuongDi.Count; i++)
54         {
55             //hướng đi không thuộc Open và Close
56             if (!haiNodeTrungNau(lstHuongDi[i], Open) && !haiNodeTrungNau(lstHuongDi[i], Close))
57             {
58                 Open.Add(lstHuongDi[i]);
59             }
60             else
61             {
62                 //nếu hướng đi thuộc Open
63                 if (haiNodeTrungNau(lstHuongDi[i], Open))
64                 {
65                     /*nếu hướng đi đó tốt hơn thì sẽ được cập nhật lại,
66                     ngược lại thì sẽ không cập nhật*/
67                     soSanhTotHon(lstHuongDi[i], Open);
68                 }
69                 else
70                 {
71                     //nếu hướng đi thuộc Close
72                     if (haiNodeTrungNau(lstHuongDi[i], Close))
73                     {
74                         //nếu hướng đi đó tốt hơn thì sẽ được cập nhật lại,
75                         ngược lại thì sẽ không cập nhật
76                     }
77                 }
78             }
79         }
80     }
81 }
```

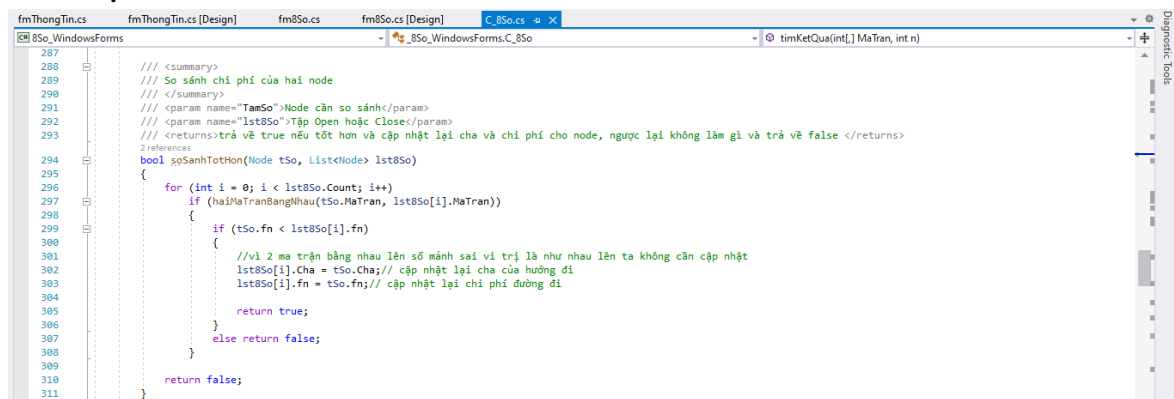
```

74
75 //nếu hướng đi thuộc Close
76 if (haiNodeTrungHau(lstHuongDi[i], Close))
77 {
78     //nếu hướng đi đó tốt hơn thì sẽ được cập nhật lại,
79     //ngược lại thì sẽ không cập nhật và chuyển từ Close sang Open*/
80     if (soSanhTotHon(lstHuongDi[i], Close))
81     {
82         Node temp = new Node();
83         temp = layNodeTrungTrongClose(lstHuongDi[i], Close);
84         Close.Remove(temp);
85         Open.Add(temp);
86     }
87 }
88
89
90
91 //chọn vị trí có phí tốt nhất trong Open
92 t = viTriTotNhatOpen(Open);
93
94 }
95
96
97 //truy vết kết quả tổng tập Close
98 stkKetQua = truyVetKetQua(Close);
99
100 return stkKetQua;
101 }
102
103 //truy vết kết quả đường đi trong tập Close
104 Stack<int,> ketQua = new Stack<int,>();
105 Stack<int,> truyVetKetQua(List<Node> Close)
106 {
107     Stack<int,> ketQua = new Stack<int,>();
108     int t = Close[Close.Count - 1].Cha;
109     Node temp = new Node();
110
111     //truy vết kết quả trong tập Close
112     stkKetQua = truyVetKetQua(Close);
113     return stkKetQua;
114 }
115
116 //truy vết kết quả đường đi trong tập Close
117 Stack<int,> truyVetKetQua(List<Node> Close)
118 {
119     Stack<int,> ketQua = new Stack<int,>();
120     int t = Close[Close.Count - 1].Cha;
121     Node temp = new Node();
122     ketQua.Push(Close[Close.Count - 1].MaTran);
123     while (t != -1)
124     {
125         for (int i = 0; i < Close.Count; i++)
126         {
127             if (t == Close[i].ChiSo)
128             {
129                 temp = Close[i];
130                 break;
131             }
132         }
133         ketQua.Push(temp.MaTran);
134         t = temp.Cha;
135     }
136     return ketQua;
137 }
138
139 /// <summary>
140 /// hàm sinh ra các hướng đi từ một node sinh ra các node con
141 /// </summary>
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
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
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
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
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
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
```



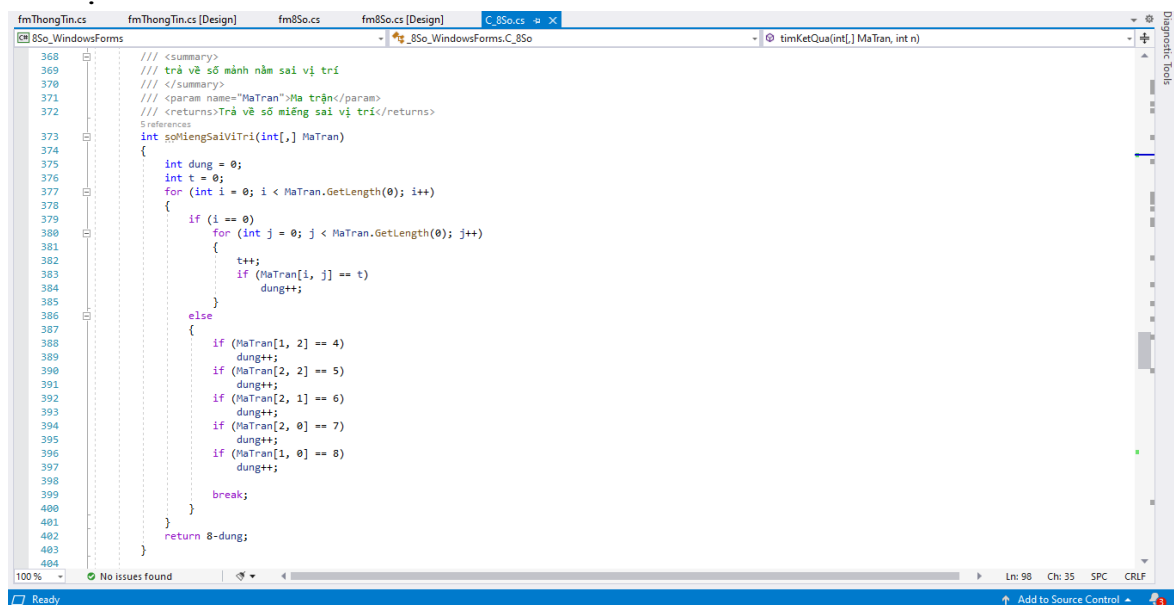
```
252 /// Chọn vị trí có chi phí tốt nhất trong Open
253 /// </summary>
254 /// <param name="Open">Tập Open</param>
255 /// <returns>Vị trí tốt nhất</returns>
256 int vịTrịTotWhatOpen(List<Node> Open)
257 {
258     if (Open.Count != 0)
259     {
260         Node min = new Node();
261         min = Open[0];
262         int vt = 0;
263
264         for (int i = 1; i < Open.Count; i++)
265             if (min.SoManhSai > Open[i].SoManhSai)
266             {
267                 min = Open[i];
268                 vt = i;
269             }
270             else
271             {
272                 if (min.SoManhSai == Open[i].SoManhSai)
273                 {
274                     if (min.fn > Open[i].fn)
275                     {
276                         min = Open[i];
277                         vt = i;
278                     }
279                 }
280             }
281         return vt;
282     }
283     return 0;
284 }
285
286
287
288 /// </summary>
```

+ So sánh vị trí Node:



```
287
288 /// </summary>
289 /// So sánh chi phí của hai node
290 /// </summary>
291 /// <param name="TamSo">Node cần so sánh</param>
292 /// <param name="lst8So">Tập Open hoặc Close</param>
293 /// <returns>trả về true nếu tốt hơn và chi phí cho node, ngược lại không làm gì và trả về false </returns>
294 bool soSanhTotHon(Node tso, List<Node> lst8So)
295 {
296     for (int i = 0; i < lst8So.Count; i++)
297         if (haiMaTranBangNhau(tso.MaTran, lst8So[i].MaTran))
298         {
299             if (tso.fn < lst8So[i].fn)
300             {
301                 //vì 2 ma trận bằng nhau lên số mảnh sai vị trí là như nhau lên ta không cần cập nhật
302                 lst8So[i].Cha = tso.Cha; // cập nhật lại cha của hướng đi
303                 lst8So[i].fn = tso.fn; // cập nhật lại chi phí đường đi
304             }
305             return true;
306         }
307         else return false;
308     }
309     return false;
310 }
311
```

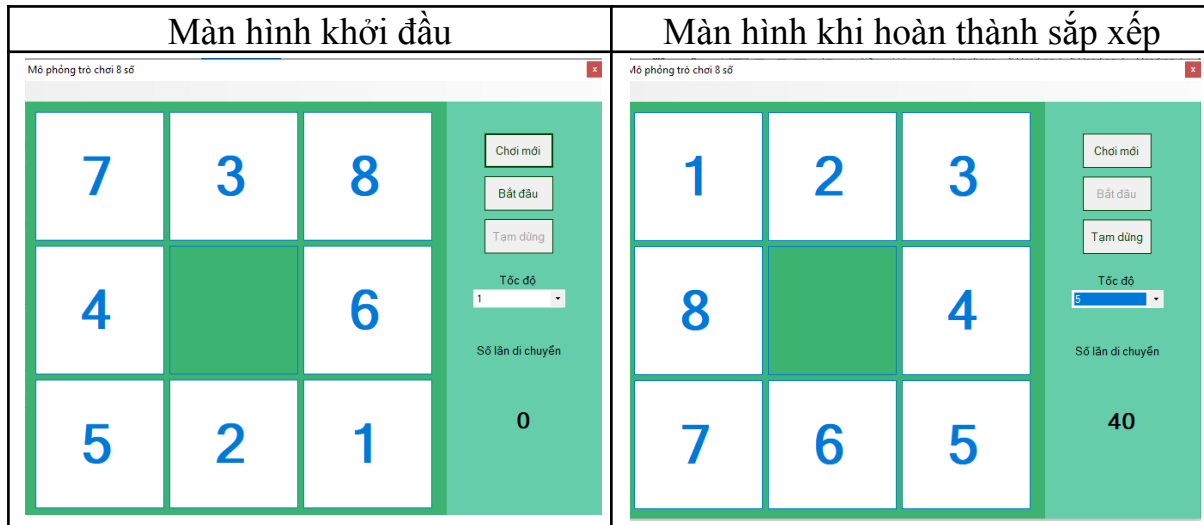
+ Trả về vị trí mảnh nằm sai:



```
368
369 /// </summary>
370 /// trả về số mảnh nằm sai vị trí
371 /// </summary>
372 /// <param name="MaTran">Ma trận</param>
373 /// <returns>Trả về số miếng sai vị trí</returns>
374 int soMiangSaiViTri(int[,] MaTran)
375 {
376     int dung = 0;
377     int t = 0;
378     for (int i = 0; i < MaTran.GetLength(0); i++)
379     {
380         if (i == 0)
381             for (int j = 0; j < MaTran.GetLength(0); j++)
382             {
383                 t++;
384                 if (MaTran[i, j] == t)
385                     dung++;
386             }
387         else
388         {
389             if (MaTran[1, 2] == 4)
390                 dung++;
391             if (MaTran[2, 2] == 5)
392                 dung++;
393             if (MaTran[2, 1] == 6)
394                 dung++;
395             if (MaTran[2, 0] == 7)
396                 dung++;
397             if (MaTran[1, 0] == 8)
398                 dung++;
399             break;
400         }
401     }
402     return 8 - dung;
403 }
404
```

Chú thích đã được thêm vào từng vùng code giảng viên có thể xem trong chương trình Demo.

Hình ảnh Demo:



Có kèm video Demo.

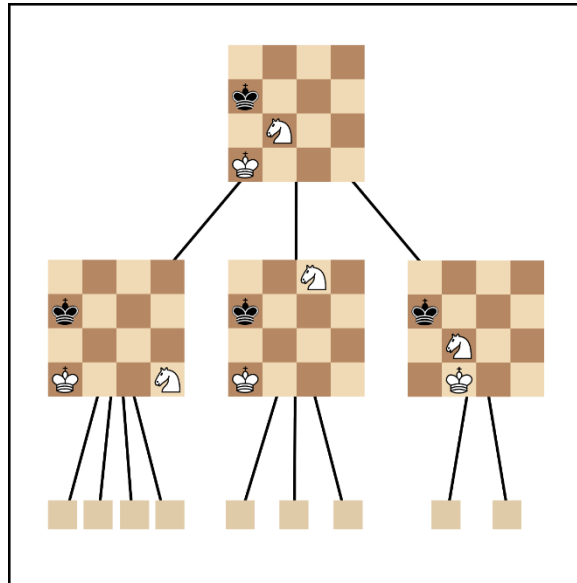
2. Theory of game (Lý thuyết về trò chơi)

2.1. Thuật toán Minimax

Minimax là giải thuật là một thuật toán đệ quy lựa chọn bước đi kế tiếp trong một trò chơi có hai người bằng cách định giá trị cho các Node trên cây trò chơi sau đó tìm Node có giá trị phù hợp để đi bước tiếp theo.

Quá trình chơi cờ là quá trình quân trắng và đen thay phiên nhau đưa ra quyết định, thực hiện một trong số các nước đi hợp lệ. Trên cây trò chơi, quá trình đó sẽ tạo ra đường đi từ gốc đến lá. Giả sử tới một thời điểm nào đó, đường đi đã dẫn tới đỉnh u . Nếu u là đỉnh trắng (đen) thì trắng (đen) cần chọn đi tới một trong các đỉnh đen (trắng) v là con của u . Tại đỉnh đen (trắng) v mà trắng (đen) vừa chọn, đen (trắng) sẽ phải chọn đi tới một trong các đỉnh trắng (đen) w là con của v . Quá trình trên sẽ dừng lại khi đạt tới một đỉnh là lá của cây.

Giả sử quân trắng cần tìm nước đi tại đỉnh u . Nước đi tối ưu cho quân trắng là nước đi dẫn tới đỉnh con của v là đỉnh tốt nhất (cho trắng) trong số các đỉnh con của u . Ta cần giả thiết rằng, đến lượt đối thủ chọn nước đi từ v , đen cũng sẽ chọn nước đi tốt nhất cho a ta. Như vậy, để chọn nước đi tối ưu cho trắng tại đỉnh u , ta cần phải xác định giá trị các đỉnh của trò chơi gốc u . Giá trị của các đỉnh lá là giá trị của hàm kết cuộc. Đỉnh có giá trị càng lớn càng tốt cho trắng, đỉnh có giá trị càng nhỏ càng tốt cho đen. Để xác định giá trị các đỉnh của cây trò chơi gốc u , ta đi từ mức thấp nhất lên gốc u . Giả sử v là đỉnh trong của cây và giá trị các đỉnh con của nó đã được xác định. Khi đó nếu v là đỉnh trắng thì giá trị của nó được xác định là giá trị lớn nhất trong các giá trị của các đỉnh con. Còn nếu v là đỉnh đen thì giá trị nhỏ nhất trong các giá trị của các đỉnh con.



Để cài đặt kĩ thuật Minimax, việc gán giá trị cho các đỉnh được thực hiện bởi các hàm đệ quy Maxval và Minval. Hàm MaxVal xác định giá trị cho các đỉnh trắng, hàm MinVal xác định giá trị cho các đỉnh đen.

2.2. Tinh chỉnh Alpha-beta (cắt tỉa Alpha- beta)

Việc tinh chỉnh Alpha-beta là một phương pháp tối ưu hóa thuật toán Minimax cho phép chúng ta bỏ qua một số hướng trong tất cả hướng đi có thể. Điều này giúp ta dự đoán hướng bằng minimax hiệu quả, trong khi sử dụng cùng một thuật toán.

Việc giảm thiểu alpha-beta dựa trên tình huống mà chúng ta có thể ngừng đưa ra hướng đi nếu chúng ta thấy hướng đi đó dẫn đến một kết quả không mong đợi. Việc điều chỉnh alpha-beta làm cho thuật toán nhanh hơn.

2.3. Chương trình demo: Cờ vua dùng thuật toán Minimax

Ngôn ngữ sử dụng: Javascript

Giao diện: Website (HTML+CSS)

Một số hình ảnh code:

Triển khai thuật toán Minimax: kèm theo các giá trị liên quan như alpha, beta (hỗ trợ cắt nhánh dư thừa) - file nguồn script.js.

```

/*The "AI" part starts here */
var minimaxRoot = function(depth, game, isMaximisingPlayer) {
    var newGameMoves = game.ugly_moves();
    var bestMove = -9999;
    var bestMoveFound;

    for(var i = 0; i < newGameMoves.length; i++) {
        var newGameMove = newGameMoves[i];
        game.ugly_move(newGameMove);
        var value = minimax(depth - 1, game, -10000, 10000, !isMaximisingPlayer);
        game.undo();
        if(value >= bestMove) {
            bestMove = value;
            bestMoveFound = newGameMove;
        }
    }
}

```

```

    }
    return bestMoveFound;
  };

  var minimax = function (depth, game, alpha, beta, isMaximisingPlayer) {
    positionCount++;
    if (depth === 0) {
      return -evaluateBoard(game.board());
    }

    var newGameMoves = game.ugly_moves();

    if (isMaximisingPlayer) {
      var bestMove = -9999;
      for (var i = 0; i < newGameMoves.length; i++) {
        game.ugly_move(newGameMoves[i]);
        bestMove = Math.max(bestMove, minimax(depth - 1, game, alpha, beta, !isMaximisingPlayer));
        game.undo();
        alpha = Math.max(alpha, bestMove);
        if (beta <= alpha) {
          return bestMove;
        }
      }
      return bestMove;
    } else {
      var bestMove = 9999;
      for (var i = 0; i < newGameMoves.length; i++) {
        game.ugly_move(newGameMoves[i]);
        bestMove = Math.min(bestMove, minimax(depth - 1, game, alpha, beta, !isMaximisingPlayer));
        game.undo();
        beta = Math.min(beta, bestMove);
        if (beta <= alpha) {
          return bestMove;
        }
      }
      return bestMove;
    }
  };

  var evaluateBoard = function (board) {

```

Khai báo các đối tượng trên bàn cờ: màu cờ (trắng-đen), loại cờ (tốt, xe, mã, tượng(tinh), hậu, vua) – file nguồn chess.js.

```

1
2
3 var Chess = function(fen) {
4
5   /* jshint indent: false */
6
7   var BLACK = 'b';
8   var WHITE = 'w';
9
10  var EMPTY = -1;
11
12  var PAWN = 'p';
13  var KNIGHT = 'n';
14  var BISHOP = 'b';
15  var ROOK = 'r';
16  var QUEEN = 'q';
17  var KING = 'k';
18
19  var SYMBOLS = 'pnbrqkPNBRQK';
20
21  var DEFAULT_POSITION = 'rnbqkbnr/pppppppp/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1';
22
23  var POSSIBLE_RESULTS = ['1-0', '0-1', '1/2-1/2', '**'];
24
25  var PAWN_OFFSETS = {
26    b: [16, 32, 17, 15],
27    w: [-16, -32, -17, -15]
28  };
29
30  var PIECE_OFFSETS = {
31    n: [-18, -33, -31, -14, 18, 33, 31, 14],
32    b: [-17, -15, 17, 15],
33    r: [-16, 1, 16, -1],
34    q: [-17, -16, -15, 1, 17, 16, 15, -1],
35    k: [-17, -16, -15, 1, 17, 16, 15, -1]
36  };
37
38  var ATTACKS = [
39    20, 0, 0, 0, 0, 0, 24, 0, 0, 0, 0, 0, 0, 20, 0,
40    0, 20, 0, 0, 0, 0, 0, 24, 0, 0, 0, 0, 0, 0, 20, 0,

```

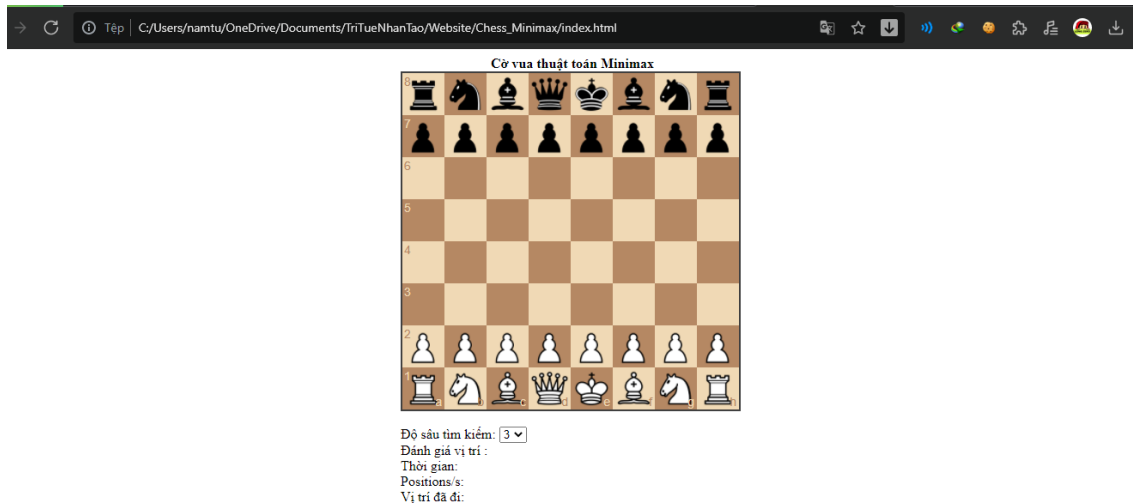
Lưu những vị trí tối ưu nhất (theo [Wikipedia](https://en.wikipedia.org/wiki/Chess_algorithm)) dưới dạng mảng nhằm đạt được hiệu quả cao nhất (những vị trí nằm giữa bàn cờ sẽ cho nhiều lựa chọn – đường đi tối ưu hơn) – file nguồn script.js.

```

87
88
89 var knightEval =
90 [
91   [-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0],
92   [-4.0, -2.0, 0.0, 0.0, 0.0, 0.0, -2.0, -4.0],
93   [-3.0, 0.0, 1.0, 1.5, 1.5, 1.0, 0.0, -3.0],
94   [-3.0, 0.5, 1.5, 2.0, 2.0, 1.5, 0.5, -3.0],
95   [-3.0, 0.0, 1.5, 2.0, 2.0, 1.5, 0.0, -3.0],
96   [-3.0, 0.5, 1.0, 1.5, 1.5, 1.0, 0.5, -3.0],
97   [-4.0, -2.0, 0.0, 0.5, 0.5, 0.0, -2.0, -4.0],
98   [-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0]
99 ];
100
101 var bishopEvalWhite = [
102   [-2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0],
103   [-1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0],
104   [-1.0, 0.0, 0.5, 1.0, 1.0, 0.5, 0.0, -1.0],
105   [-1.0, 0.5, 0.5, 1.0, 1.0, 0.5, 0.5, -1.0],
106   [-1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, -1.0],
107   [-1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, -1.0],
108   [-1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.5, -1.0],
109   [-2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0]
110 ];
111
112 var bishopEvalBlack = reverseArray(bishopEvalWhite);
113
114 var rookEvalWhite = [
115   [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
116   [ 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5],
117   [-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.5],
118   [-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.5],
119   [-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.5],
120   [-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.5],
121   [ 0.0, 0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.0]
122 ];

```

Hình ảnh Demo:



Video Demo được lưu trong thư mục Video.

3. Logic and Semantic (Logic và ngữ nghĩa)

3.1. Giải thuật Vương Hạo

Thuật giải Vương Hạo là một thuật giải khác để chứng minh việc suy dẫn mệnh đề (có cùng mục đích với Robinson):

Bước 1: Đưa bài toán cần chứng minh về dạng chuẩn:

$$GT_1, GT_2, \dots, GT_n \Rightarrow KL_1, KL_2, \dots, KL_m$$

Trong đó các GT_i và KL_j là các câu chỉ gồm các phép $\wedge$, $\vee$, $\neg$, không chứa phép $\Rightarrow$ hay $\Leftrightarrow$. Lưu ý: dấu phẩy (,) ở vế trái tương đương với $\wedge$, ở vế phải tương đương với $\vee$.

Bước 2: Nếu tồn tại một câu có phép $\neg$ ở đầu thì chuyển vế câu và loại bỏ phép $\neg$

Bước 3: Thay các dấu $\wedge$ ở vế trái và các dấu $\vee$ ở vế phải bằng dấu phẩy (,). Khi đó vế trái chỉ còn dấu $\vee$ và $\neg$, vế phải chỉ còn dấu $\wedge$ và $\neg$.

Bước 4:

Nếu dòng hiện hành có dạng:

$$GT_1, \dots, GT_k, A \vee B, GT_{k+1}, \dots, GT_n \Rightarrow KL_1, KL_2, \dots, KL_m$$

thì thay bằng hai dòng:

$$GT_1, \dots, GT_k, A, GT_{k+1}, \dots, GT_n \Rightarrow KL_1, KL_2, \dots, KL_m$$

và

$$GT_1, \dots, GT_k, B, GT_{k+1}, \dots, GT_n \Rightarrow KL_1, KL_2, \dots, KL_m$$

Nếu dòng hiện hành có dạng:

$$GT_1, GT_2, \dots, GT_n \Rightarrow KL_1, \dots, KL_k, A \wedge B, KL_{k+1}, \dots, KL_m$$

thì thay bằng hai dòng:

$$GT_1, GT_2, \dots, GT_n \Rightarrow KL_1, \dots, KL_k, A, KL_{k+1}, \dots, KL_m$$

và

$$GT_1, GT_2, \dots, GT_n \Rightarrow KL_1, \dots, KL_k, B, KL_{k+1}, \dots, KL_m$$

Bước 5: Một dòng được chứng minh nếu tồn tại một mệnh đề ở cả hai vế.

Bước 6: Một dòng không thể tách cũng không thể chuyển về dấu $\neg$ mà không có biến mệnh đề chung ở cả hai vế thì không được chứng minh.

Lặp lại bước 2 đến 6 cho tới khi mọi dòng được chứng minh hay tồn tại một dòng không được chứng minh. Bài toán ban đầu được chứng minh nếu mọi dòng tách ra từ nó được chứng minh.

3.2. Chương trình demo: Mô phỏng thuật toán Vương Hạo trong Prolog

Ngôn ngữ sử dụng: Prolog

Giao diện: SWI-Prolog.

Cách sử dụng:

Một số kí tự trong mệnh đề khá khó khăn để sử dụng nên phần mềm chỉ nhận những kí tự tương đương trong bảng:

Phép tính mệnh đề	Kí hiệu toán học	Kí hiệu chương trình
Phép phủ định (negation)	$\neg$	!
Phép hội (conjunction)	$\wedge$	&
Phép tuyển (disjunction)	$\vee$	v
Phép kéo theo (implication)	$\rightarrow$	->
Phép tương đương (equivalence)	$\leftrightarrow$	<->
Phép tuần tự (sequent)	$\vdash$	=

Một số hình ảnh code:

+ Xác định các toán tử

```
: -op(700,xfy,<->)
```

```
: -op(700,xfy,->)
```

```
: -op(600,xfy,v)
```

```
: -op(600,xfy,&)
```

```
: -op(500,fy,!)
```

+ Thực thi 7 luật suy diễn trong thuật toán Vương Hạo.

```
vuonghao.pl
File Edit Browse Compile Prolog Pce Help
vuonghao.pl
% Thuật toán Vương Hao.
% Xác định toán tử.
:-op(700,xfy,<->).
:-op(700,xfy,->).
:-op(600,xfy,v).
:-op(600,xfy,&).
:-op(500,fy,I).

% Lệnh gọi chính cho chương trình.
prove([],[]).
prove([L|P],[R|A]):-
    nl, ansi_format([bold],'Statement: ',[], write(L), write(' |= '), write(R), nl, nl,
    ansi_format([bold],'Attempting proof! ',[], nl, nl,
    wang(L,R),
    prove(P,A).

% Thủ tục thuật toán để chứng minh mệnh đề.
wang(L,R):-
    rules(L,R,[],0), nl,
    ansi_format([bold,fg(green)], 'Result: The given theorem is true.',[], nl, nl,
    ansi_format([bold, fg(red)], 'Proof failed for current step! ',[], nl, nl,
    ansi_format([bold,fg(red)], 'Result: The given theorem is false.',[], nl, nl.

% Di chuyển phụ định từ trái sang phải.
rules(L,R,S,T):-
    member(!X,L),
    delete(L,!X,Ld),
    append(S,[[['*Rule 1L',
    rules(Ld,[X|R],Sa,T).

% Di chuyển phụ định từ phải qua trái.
rules(L,R,S,T):-
    member(!X,R),
    delete(R,!X,Rd),
    append(S,[[['*Rule 1R',
    rules([X|L],Rd,Sa,T).

% Thay thế phép kéo theo bên trái.
rules(L,R,S,T):-
    member(X -> Y,L),
    delete(L,X -> Y,Ld),
    append(S,[[['*Rule 6L',
    rules([!X v Y|Ld],R,Sa,T).

% Thay thế phép kéo theo bên phải.
rules(L,R,S,T):-
    member(X -> Y,R),
    delete(R,X -> Y,Rd),
    append(S,[[['*Rule 6R',
    rules(L,[!X v Y|Rd],Sa,T).

% Thay thế phép tương đương bên trái.
rules(L,R,S,T):-
    member(X <-> Y,L),
    delete(L,X <-> Y,Ld),
    append(S,[[['*Rule 7L',
    rules([(X -> Y) & (Y -> X)|Ld],R,Sa,T).

% Thay thế phép tương đương bên phải.
rules(L,R,S,T):-
    member(X <-> Y,R),
    delete(R,X <-> Y,Rd),
    append(S,[[['*Rule 7R',
    rules(L,[L,' |= ',[(X -> Y) & (Y -> X)|Rd]],T]],Sa,T).

% So sánh hai vế mệnh đề.
rules(L,R,S,T):-
    append(S,[[['*Tautology?',
    ],[L,' |= ',R],T]],Sa,T),
    member(X,L),
    member(X,R),
    append(Sa,[[['*True.',
    ],[],T]],Sb,T),
    printprove(Sb).

% Thay thế phép hội bằng dấu phẩy ở bên trái.
rules(L,R,S,T):-
    member(X & Y,L),
    delete(L,X & Y,Ld),
    append(S,[[['*Rule 2',
    rules([X,Y|Ld],R,Sa,T).

% Thay thế phép tuyển bằng dấu phẩy ở bên phải.
rules(L,R,S,T):-
    member(X v Y,R),
    delete(R,X v Y,Rd),
    append(S,[[['*Rule 3',
    rules(L,[X,Y|Rd],Sa,T).

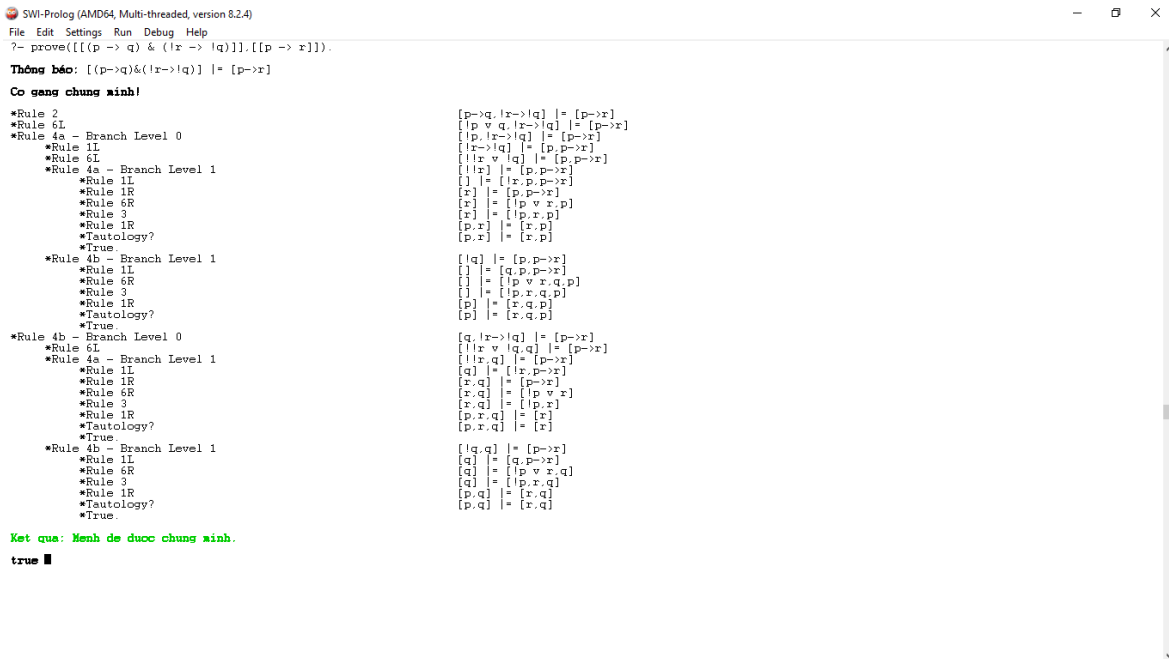
% Tách phép tuyển bên trái.
rules(L,R,S,T):-
    member(X v Y,L),
    delete(L,X v Y,Ld),
    Ta is T + 1,
    append(S,[[['*Rule 4a - Branch Level ',T],[[X|Ld], ' |= ',R],T]],Sa,T),
    rules([X|Ld],R,Sa,Ta),
    Tb is T + 1,
    append([],[[['*Rule 4b - Branch Level ',T],[[Y|Ld], ' |= ',R],T]],Sb,T),
    rules([Y|Ld],R,Sb,Tb).

% Tách phép hội bên phải.
rules(L,R,S,T):-
    member(X & Y,R),
    delete(R,X & Y,Rd),
    append(S,[[['*Rule 5a - Branch Level ',T],[L,' |= ',[X|Rd]],T]],Sa,T),
    Ta is T + 1,
    rules(L,[X|Rd],Sa,Ta),
    Tb is T + 1,
    append([],[[['*Rule 5b - Branch Level ',T],[L,' |= ',[Y|Rd]],T]],Sb,T),
    rules(L,[Y|Rd],Sb,Tb).
```

```
% Neu menh de dung thi in ra dieu phai chung minh.
printprove([]).
printprove([([P,Q,T]|S)):-
  tab(T*5), printlist(P), tab(40 - T*5), printlist(Q), nl,
  printprove(S).

% In danh sach theo de quy.
printlist([]).
printlist([H|T]):-
  write(H),
  printlist(T).
```

Hình ảnh Demo:



Video thực hiện thử một số mệnh đề được lưu trong thư mục Video.

4. Simulation (Mô phỏng)

4.1. Genetic Algorithm applied to optimization (Thuật toán di truyền áp dụng để tối ưu hoá)

Thuật toán di truyền (GA) là thuật toán tìm kiếm dựa trên các khái niệm về chọn lọc tự nhiên và di truyền. GAs là một tập con của một nhánh tính toán lớn hơn nhiều được gọi là Tính toán tiến hóa.

Trong GAs, ta có một nhóm các giải pháp khả thi cho vấn đề đã cho. Những dung dịch này sau đó trải qua quá trình tái tổ hợp và đột biến (giống như trong di truyền tự nhiên), tạo ra những đứa trẻ mới và quá trình này được lặp lại trong nhiều thế hệ khác nhau. Mỗi cá thể (hoặc giải pháp ứng cử viên) được chỉ định một giá trị phù hợp (dựa trên giá trị hàm mục tiêu của nó) và các cá thể lai tạo có cơ hội cao hơn để giao phối và sinh ra các cá thể phù hợp. Điều này phù hợp với Học thuyết của Darwin về sự sống còn của những cá thể khoẻ mạnh nhất.

Do đó, nó tiếp tục phát triển các cá nhân hoặc giải pháp tốt hơn qua nhiều thế hệ, cho đến khi đạt được tiêu chí dừng. Thuật toán di truyền về bản chất là đủ ngẫu nhiên, nhưng chúng hoạt động tốt hơn nhiều so với tìm kiếm cục bộ ngẫu nhiên (nơi ta chỉ thử các giải pháp ngẫu nhiên, theo dõi các giải pháp tốt nhất cho đến nay), vì chúng cũng khai thác thông tin lịch sử.

4.2. Chương trình demo: Tìm đường đi sử dụng thuật toán di truyền

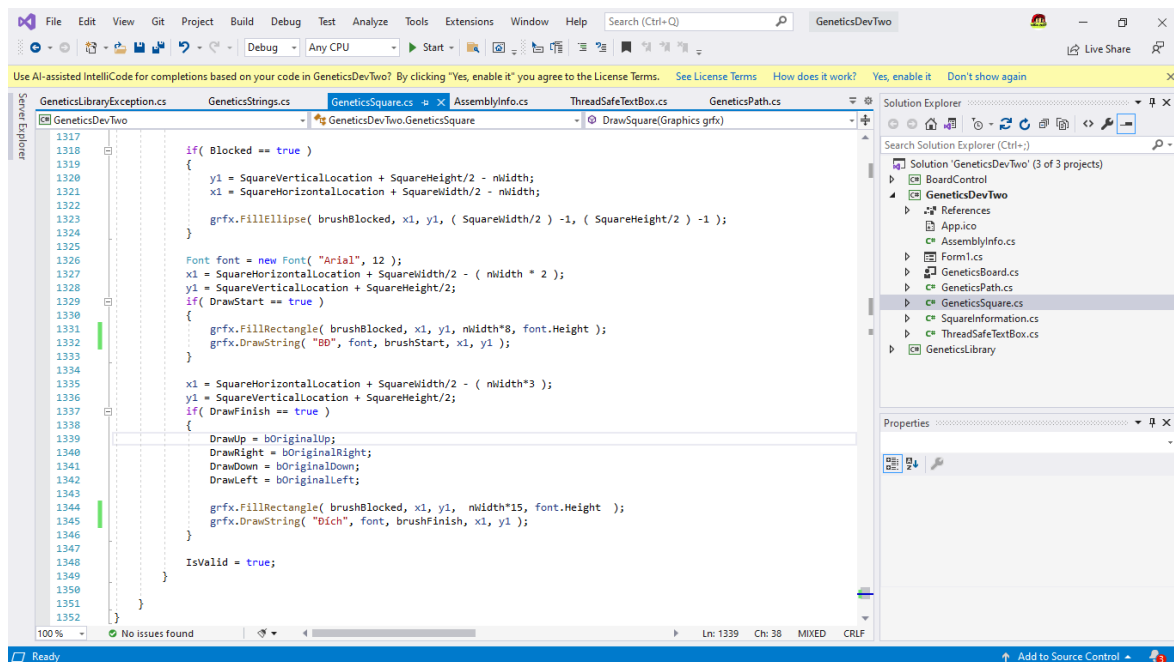
Ngôn ngữ sử dụng: C#

Giao diện: Winform

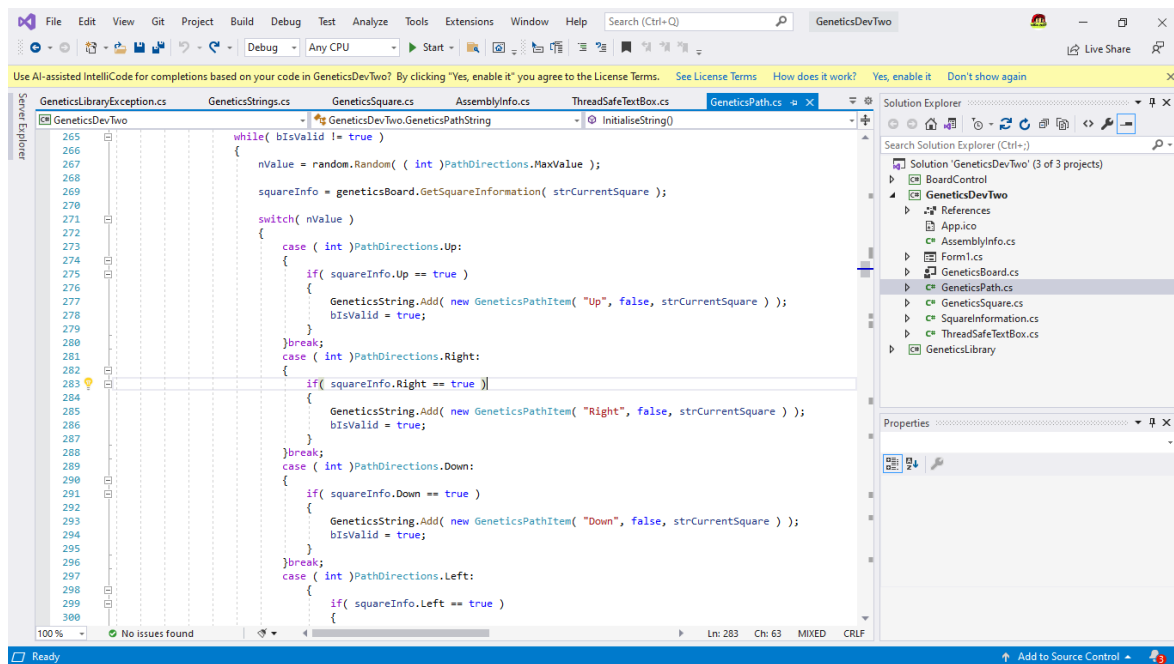
Mô tả: Project gồm nhiều file chức năng khác nhau phân là 3 mục chính: Control (điều khiển), GeneticsDevTwo (Chương trình chính) và Library (Thư viện chứa các chức năng của thuật toán).

Một số hình ảnh code:

- + Khai báo thông số cho điểm Bắt đầu (BD), điểm Đích cùng một vài thông số về vị trí – file GeneticsSquare.cs.



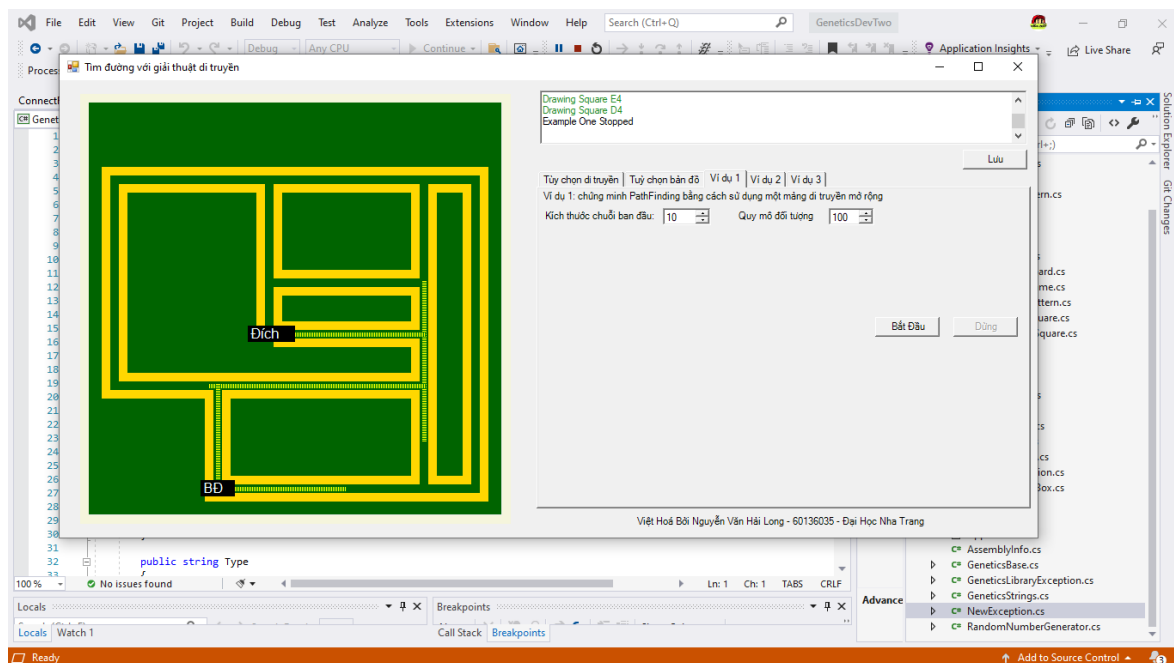
- + Thực thi các đường đi một cách ngẫu nhiên theo 4 hướng (tràn ra 4 hướng theo quy mô đối tượng) – file GeneticsPath.cs.



+ Các file thực thi khác:

- ThreadSafeTextBox.cs : TextBox chứa thông tin tìm đường (hướng đi).
- SquareInformation.cs: Nơi khai báo các biến mang giá trị Boolean.

Hình ảnh Demo:



Video demo được lưu trữ trong thư mục Video.

5. IoT and Robotics applied in with AI (IoT và Robotics được ứng dụng với AI)

5.1. Sử dụng Arduino để chế tạo cánh tay robot

Do tình hình dịch phức tạp nên em không có sử dụng được demo bằng phần cứng nên em gửi các file liên quan đến về chủ đề.

Các thiết bị cần thiết:

- Arduino UNO

- Module Bluetooth HC05
- Động cơ RC Servo MG996
- Động cơ RC Servo 9G
- Điện trở 1K Ohm
- Điện trở 2K Ohm
- Adapter 5V - 2A
- Jack cắm DC cái
- Breadboard 400 chân
- Dây bus đực - đực

Phần mềm yêu cầu:

- **Arduino IDE:** Arduino IDE là một phần mềm lập trình dựa trên ngôn ngữ lập trình C. Bằng cách viết những câu lệnh có cấu trúc tương tự cấu trúc câu lệnh trong C, chúng ta có thể điều khiển hoặc thu nhận các dữ liệu từ các module, cảm biến thông qua vi điều khiển arduino.
- **MIT App Inventor:** là một ứng dụng web nguồn mở ban đầu được cung cấp bởi Google và hiện tại được duy trì bởi Viện Công nghệ Massachusetts (MIT). Nền tảng cho phép nhà lập trình tạo ra các ứng dụng phần mềm cho hệ điều hành Android (OS). Bằng cách sử dụng giao diện đồ họa, nền tảng cho phép người dùng kéo và thả các khối mã (blocks) để tạo ra các ứng dụng có thể chạy trên thiết bị Android. Nguồn trang: <http://ai2.appinventor.mit.edu>.

5.2. Chương trình demo: Cánh tay robot

Lập trình với Arduino IDE:

Đầu tiên ta khai báo thư viện SoftwareSerial cho giao tiếp nối tiếp của mô-đun Bluetooth cũng như thư viện servo. Cả hai thư viện này đều được tích hợp sẵn với Arduino IDE nên ta không cần phải cài đặt chúng bên ngoài. Sau đó, ta cần xác định sáu servos, mô-đun Bluetooth HC-05 và một số biến để lưu trữ vị trí hiện tại và trước đó của servos, cũng như các mảng để lưu trữ các vị trí hoặc các bước cho chế độ tự động.

```

1.  #include <SoftwareSerial.h>
2.  #include <Servo.h>
3.
4.  Servo servo01;
5.  Servo servo02;
6.  Servo servo03;
7.  Servo servo04;
8.  Servo servo05;
9.  Servo servo06;
10.
11.  SoftwareSerial Bluetooth(3, 4); // Arduino(RX, TX) - HC-05 Bluetooth (TX, RX)
12.
13.  int servo1Pos, servo2Pos, servo3Pos, servo4Pos, servo5Pos, servo6Pos; // Vị trí hiện tại
14.  int servo1PPos, servo2PPos, servo3PPos, servo4PPos, servo5PPos, servo6PPos; // Vị trí trước
    đó
15.  int servo01SP[50], servo02SP[50], servo03SP[50], servo04SP[50], servo05SP[50],
    servo06SP[50]; // để lưu trữ các vị trí / bước
16.  int speedDelay = 20;
17.  int index = 0;
18.  String dataIn = "";

```

Trong phần thiết lập, ta cần khởi tạo servos và mô-đun Bluetooth và di chuyển cánh tay robot về vị trí ban đầu của nó. Sử dụng hàm write () di chuyển servo đến bất kỳ vị trí nào từ 0 đến 180 độ.

```
void setup() {
  servo01.attach(5);
  servo02.attach(6);
  servo03.attach(7);
  servo04.attach(8);
  servo05.attach(9);
  servo06.attach(10);
  Bluetooth.begin(38400); // Tốc độ truyền mặc định của mô-đun Bluetooth
  Bluetooth.setTimeout(1);
  delay(20);
  // Vị trí ban đầu của cánh tay robot
  servo1PPos = 90;
  servo01.write(servo1PPos);
  servo2PPos = 150;
  servo02.write(servo2PPos);
  servo3PPos = 35;
  servo03.write(servo3PPos);
  servo4PPos = 140;
  servo04.write(servo4PPos);
  servo5PPos = 85;
  servo05.write(servo5PPos);
  servo6PPos = 80;
  servo06.write(servo6PPos);
}
```

Tiếp đó khởi tạo vòng lặp, bằng cách sử dụng chức năng Bluetooth.available (), chương trình sẽ liên tục kiểm tra xem có bất kỳ dữ liệu nào đến từ Điện thoại thông minh hay không. Nếu đúng, sử dụng hàm readString (), chương trình đọc dữ liệu dưới dạng chuỗi và lưu trữ nó vào biến dataIn. Tùy thuộc vào dữ liệu đã đến, chương trình sẽ điều khiển cho cánh tay robot biết phải làm gì.

```
//Kiểm tra dữ liệu đến
if (Bluetooth.available() > 0) {
  dataIn = Bluetooth.readString(); // Đọc dữ liệu dưới dạng chuỗi
```

Sử dụng hàm startedWith (), chương trình kiểm tra tiền tố của mỗi dữ liệu đến và vì vậy nó biết phải làm gì tiếp theo.

```
// Nếu thanh trượt "Waist" đã thay đổi giá trị - Di chuyển Servo 1 đến vị trí

if (dataIn.startsWith("s1")) {
  String dataInS = dataIn.substring(2, dataIn.length()); // Chỉ trích xuất số
  servo1Pos = dataInS.toInt(); // Chuyển chuỗi sang số
```

Gọi hàm write () và servo sẽ đi đến vị trí đó, nhưng theo cách đó, servo sẽ chạy ở tốc độ tối đa, quá sức đối với cánh tay robot. Thay vào đó, ta cần kiểm soát tốc độ của servo, vì vậy nên sử dụng một số vòng lặp FOR để dần dần di chuyển servo từ vị trí trước đó đến vị trí hiện tại bằng cách triển khai thời gian trễ giữa mỗi lần lặp. Bằng cách thay đổi thời gian trễ, ta có thể thay đổi tốc độ của servo.

```
// Chúng tôi sử dụng vòng lặp for để chúng tôi có thể kiểm soát tốc độ của servo

// Nếu vị trí trước đó lớn hơn thì vị trí hiện tại
if (servo1PPos > servo1Pos) {
    for ( int j = servo1PPos; j >= servo1Pos; j--) { // Di chuyển servo xuống
        servo01.write(j);
        delay(20); // defines the speed at which the servo rotates
    }
}
// Nếu vị trí trước đó nhỏ hơn thì vị trí hiện tại
if (servo1PPos < servo1Pos) {
    for ( int j = servo1PPos; j <= servo1Pos; j++) { // Di chuyển servo lên
        servo01.write(j);
        delay(20);
    }
}
servo1PPos = servo1Pos; // Đặt vị trí hiện tại là vị trí trước đó
}
```

Thiết đặt lưu thông tin. Nếu chúng ta nhấn nút SAVE, vị trí của mỗi động cơ servo được lưu trong một mảng. Với mỗi lần nhấn, chỉ số tăng lên vì vậy mảng được lấp đầy từng bước.

```
// Nếu nút "SAVE" được nhấn
if (dataIn.startsWith("SAVE")) {
    servo01SP[index] = servo1PPos; // lưu vị trí vào mảng
    servo02SP[index] = servo2PPos;
    servo03SP[index] = servo3PPos;
    servo04SP[index] = servo4PPos;
    servo05SP[index] = servo5PPos;
    servo06SP[index] = servo6PPos;
    index++; // Tăng chỉ số mảng
}
```

Khai báo nút RUN và thao tác thực thi, gọi hàm tùy chỉnh `runningervo()` chạy các bước được lưu trữ. Ở đây chương trình chạy đi chạy lại các bước đã lưu trữ cho đến khi chúng tôi nhấn nút RESET. Sử dụng vòng lặp FOR, chạy qua tất cả các vị trí được lưu trữ trong các mảng và đồng thời hàm kiểm tra xem có bất kỳ dữ liệu đến từ điện thoại thông minh hay không. Dữ liệu này có thể là nút RUN / PAUSE, nút này sẽ tạm dừng rô bốt và nếu được nhấp lại sẽ tiếp tục với các chuyển động tự động. Ngoài ra nếu điều khiển tốc độ, chương trình sẽ sử dụng giá trị đó để thay đổi thời gian trễ giữa mỗi lần lặp trong vòng lặp FOR bên dưới, điều khiển tốc độ của động cơ servo.

```

// Chức năng tùy chỉnh chế độ tự động - chạy các bước đã lưu
void runservo() {
    while (dataIn != "RESET") { // Chạy đi chạy lại các bước cho đến khi nhấn nút "RESET"
        for (int i = 0; i <= index - 2; i++) { // Chạy qua tất cả các bước (chỉ mục)
            if (Bluetooth.available() > 0) { // Kiểm tra dữ liệu nhập
                dataIn = Bluetooth.readString();
                if (dataIn == "PAUSE") { // Nếu nút "PAUSE" được nhấn
                    while (dataIn != "RUN") { // Chờ cho đến khi "RUN" được nhấn lại
                        if (Bluetooth.available() > 0) {
                            dataIn = Bluetooth.readString();
                            if (dataIn == "RESET") {
                                break;
                            }
                        }
                    }
                }
            }
        }
        // Nếu thanh trượt SPEED được thay đổi
        if (dataIn.startsWith("ss")) {
            String dataInS = dataIn.substring(2, dataIn.length());
            speedDelay = dataInS.toInt(); // Thay đổi tốc độ servo (thời gian trễ)
        }
        // Servo 1
        if (servo01SP[i] == servo01SP[i + 1]) {
        }
        if (servo01SP[i] > servo01SP[i + 1]) {
            for (int j = servo01SP[i]; j >= servo01SP[i + 1]; j--) {
                servo01.write(j);
                delay(speedDelay);
            }
        }
        if (servo01SP[i] < servo01SP[i + 1]) {
            for (int j = servo01SP[i]; j <= servo01SP[i + 1]; j++) {
                servo01.write(j);
                delay(speedDelay);
            }
        }
    }
}

```

Di chuyển các servos đến vị trí tiếp theo của chúng. Sau cùng nếu nhấn nút RESET, chương trình sẽ xóa tất cả dữ liệu từ các mảng và cũng đặt lại chỉ mục về 0 để ta có thể thực hiện các chuyển động mới.

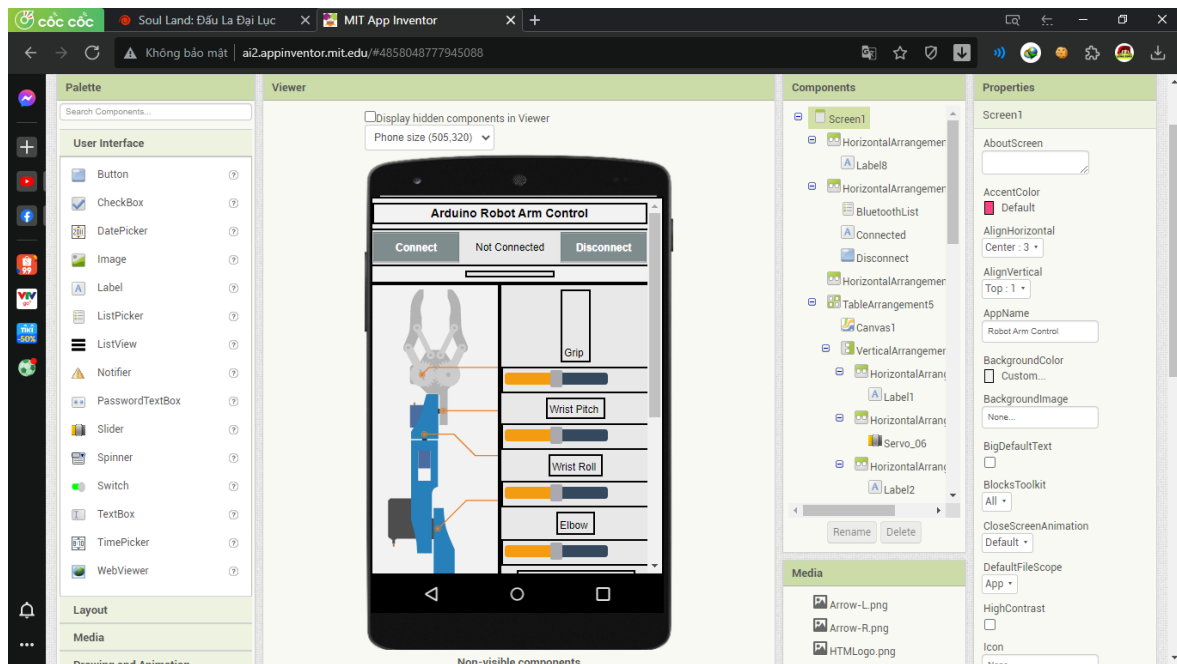
```

// Nếu nút "RESET" được nhấn
if (dataIn == "RESET") {
    memset(servo01SP, 0, sizeof(servo01SP)); // Xóa dữ liệu mảng thành 0
    memset(servo02SP, 0, sizeof(servo02SP));
    memset(servo03SP, 0, sizeof(servo03SP));
    memset(servo04SP, 0, sizeof(servo04SP));
    memset(servo05SP, 0, sizeof(servo05SP));
    memset(servo06SP, 0, sizeof(servo06SP));
    index = 0; // Chỉ mục thành 0
}

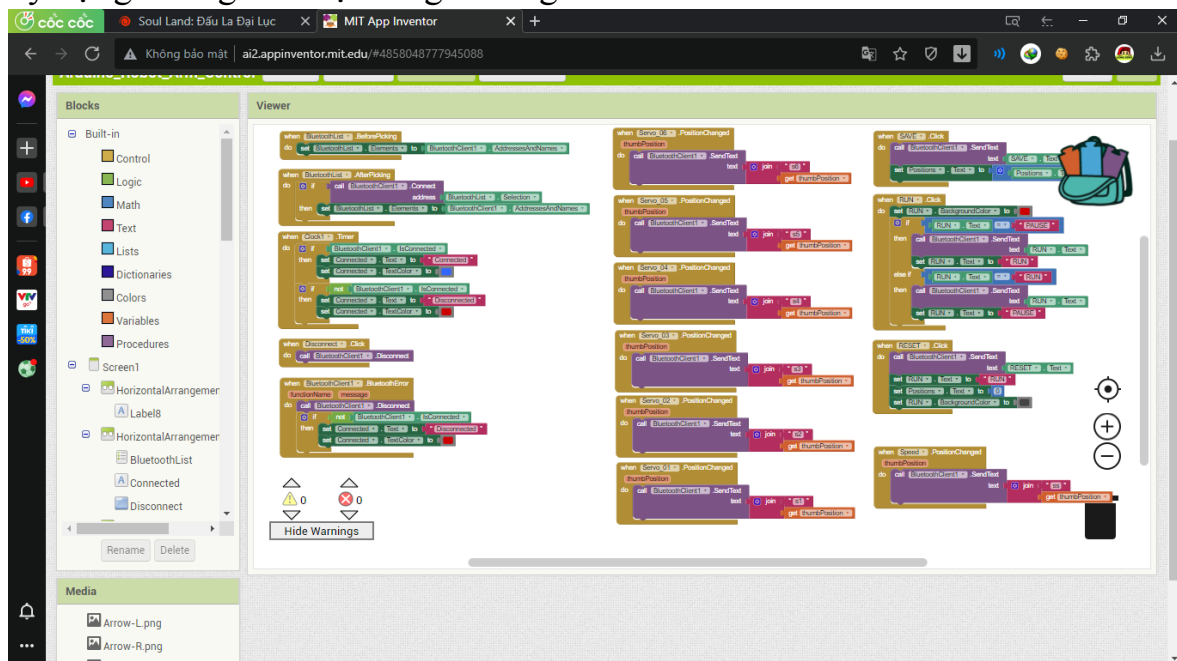
```

Lập trình MIT App Inventor:

Sử dụng biểu tượng bằng cách kéo thả để tạo giao diện



Xây dựng những khối lệnh logic trong Blocks



File thực thi được lưu trong thư mục B5