



Kierunek: Grafika komputerowa i elementy rzeczywistości mieszanej / Informatyka

Rok akademicki: 2022/2023

Przedmiot: Podstawy programowania zorientowane obiektowo

Semestr: II

Grupa dziekańska: 1

Sprawozdanie z: Laboratorium numer 2

data: 3 maja 2023

Grupa robocza: GrupaRobocza06

Skład grupy:

Osoba 1: Rostyslav Hrenitskyi

Osoba 2: Maksym Szczurko

Osoba 3 : Wojciech Gawlas

Osoba 4:

Aktywności realizowane przez członków grupy roboczej, czyli kto i za co był odpowiedzialny podczas laboratorium/zadania domowego oraz utworzenia sprawozdania (min 3 na osobę):

Osoba 1: Rostyslav Hrenitskyi

Zadanie 1, 2, 3, 4, 5.

Osoba 2: Maksym Szczurko

Zadanie 1, 2, 3, 4, 5.

Osoba 3: Wojciech Gawlas

Zadanie 1, 2, 3, 4, 5.

Osoba 4:

Sposoby i narzędzia komunikacji grupy roboczej:

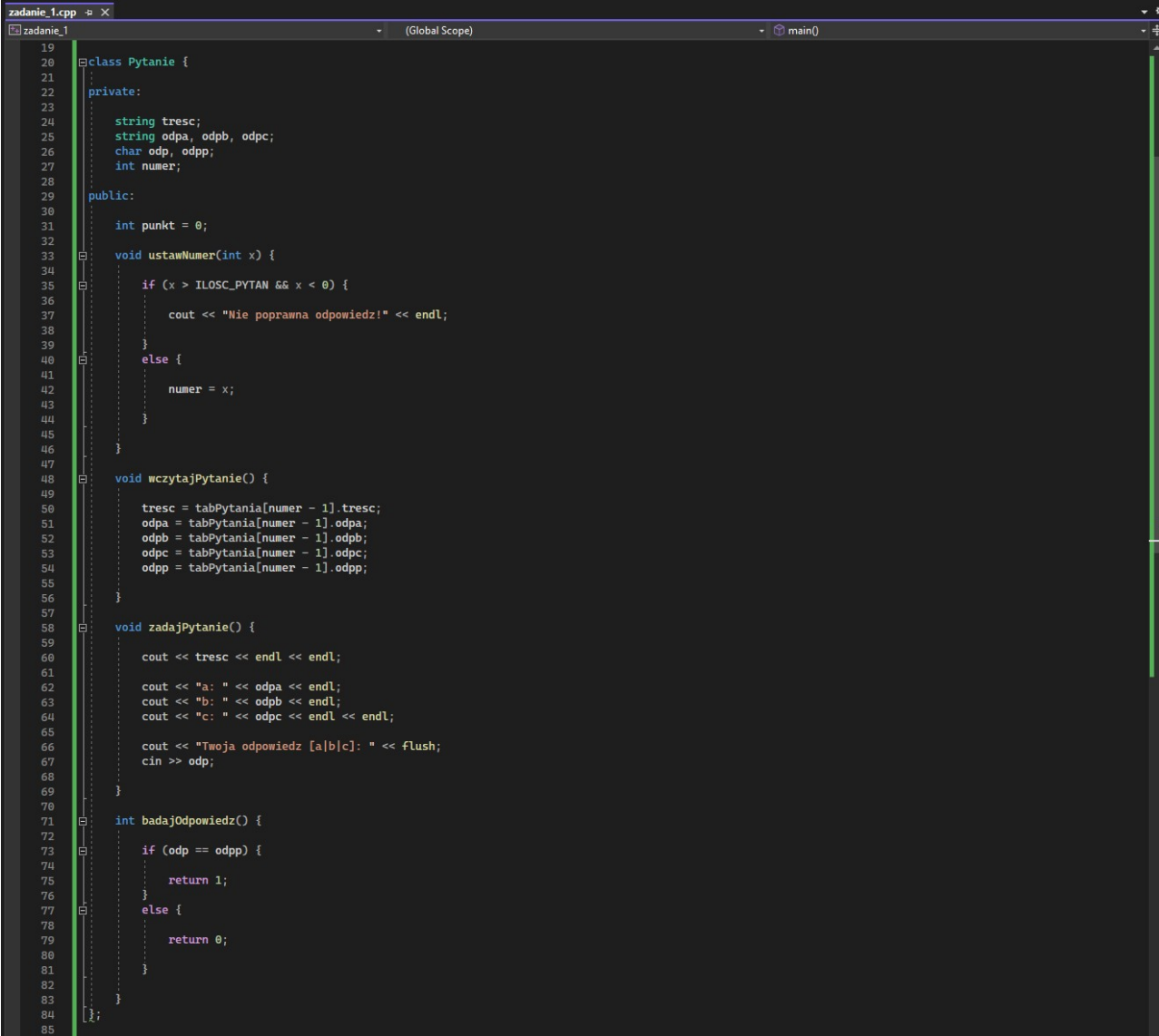
GitHub.

Zadanie 0

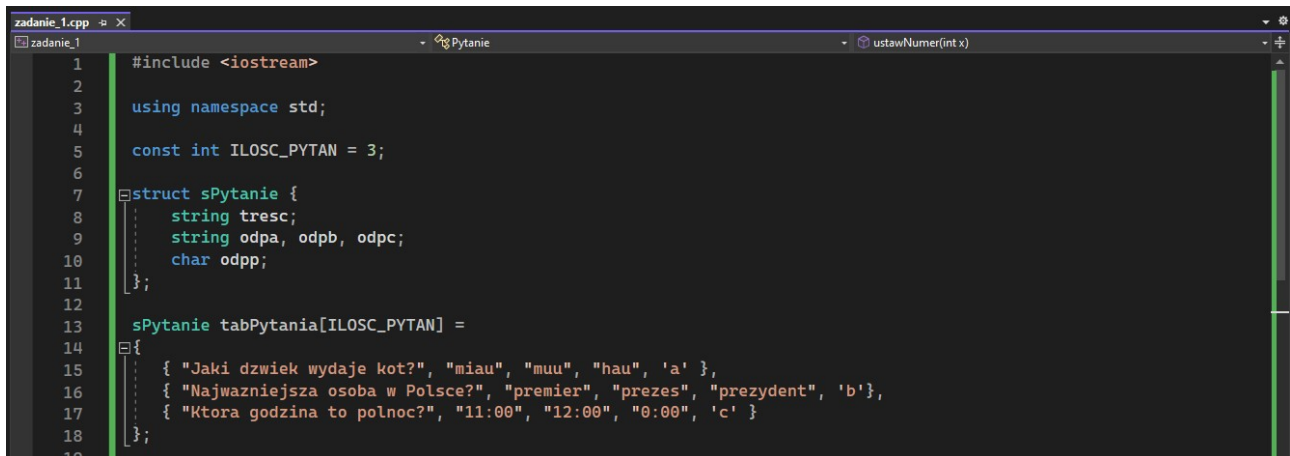
Napisać Prosty quizz zadający kilka pytań użytkownikowi. Każde pytanie to struktura umieszczona w kilkuelementowej tablicy.

Rozwiązanie zadania 0:

Klasa:

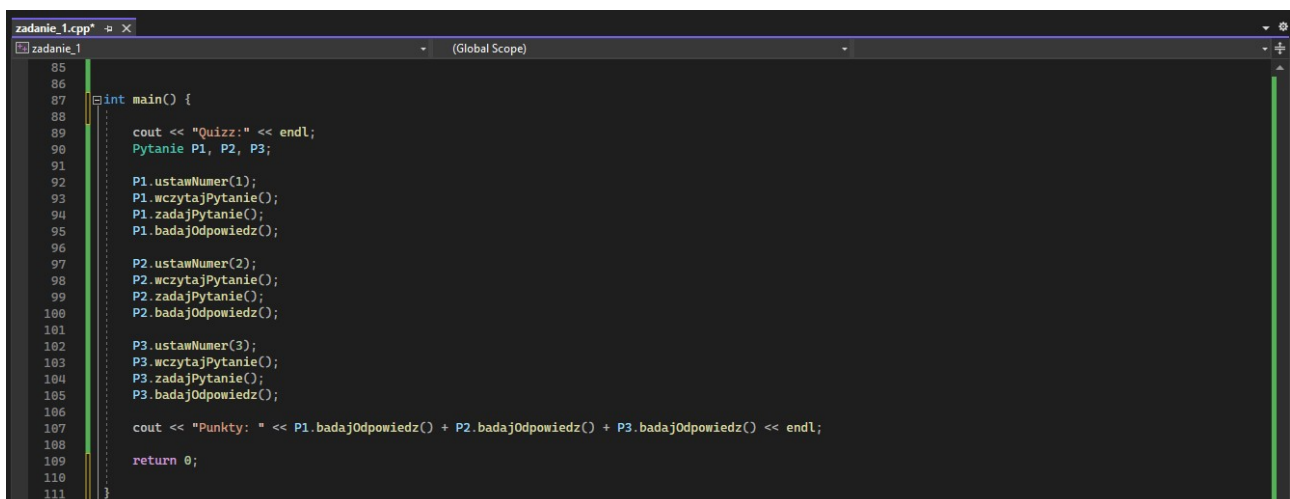


```
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
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
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
```

Struktura:

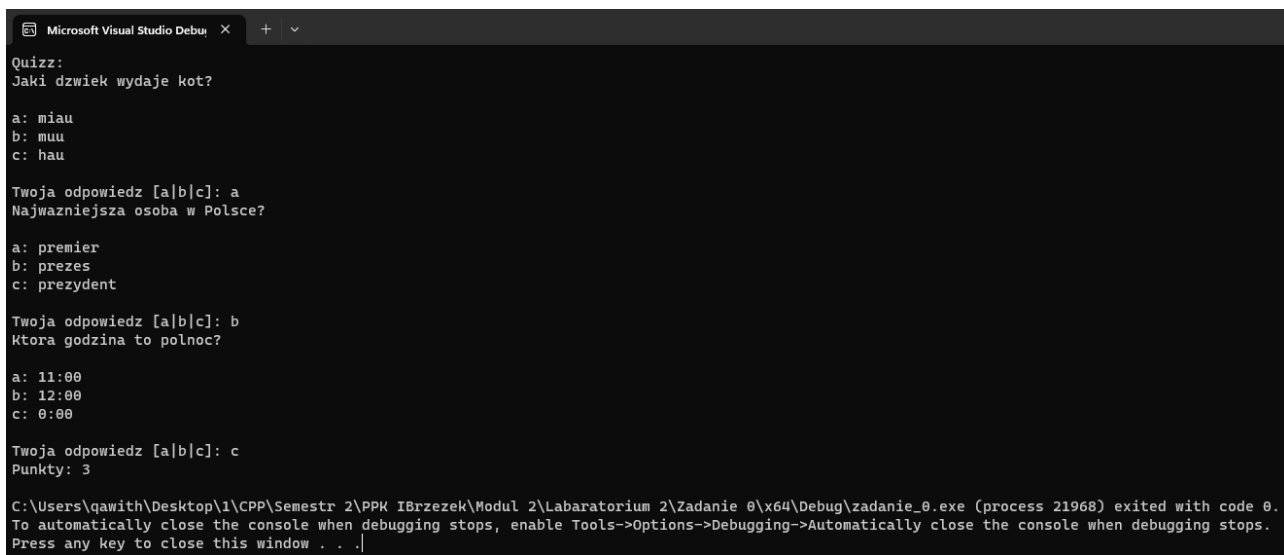
```
zadanie_1.cpp
zadanie_1
1  #include <iostream>
2
3  using namespace std;
4
5  const int ILOSC_PYTAN = 3;
6
7  struct sPytanie {
8      string tresc;
9      string odpa, odpb, odpc;
10     char odpp;
11 };
12
13 sPytanie tabPytania[ILOSC_PYTAN] =
14 {
15     { "Jaki dzwiek wydaje kot?", "miau", "muu", "hau", 'a' },
16     { "Najwazniejsza osoba w Polsce?", "premier", "prezes", "prezydent", 'b'},
17     { "Ktora godzina to polnoc?", "11:00", "12:00", "0:00", 'c' }
18 };
19
```

source: ...\\Modul 2\\Labaratorium 2\\Zadanie 0\\zadanie_0.cpp

Main:

```
zadanie_1.cpp
zadanie_1
85
86
87 int main() {
88
89     cout << "Quizz:" << endl;
90     Pytanie P1, P2, P3;
91
92     P1.ustawNumer(1);
93     P1.wczytajPytanie();
94     P1.zadajPytanie();
95     P1.badajOdpowiedz();
96
97     P2.ustawNumer(2);
98     P2.wczytajPytanie();
99     P2.zadajPytanie();
100    P2.badajOdpowiedz();
101
102    P3.ustawNumer(3);
103    P3.wczytajPytanie();
104    P3.zadajPytanie();
105    P3.badajOdpowiedz();
106
107    cout << "Punkty: " << P1.badajOdpowiedz() + P2.badajOdpowiedz() + P3.badajOdpowiedz() << endl;
108
109    return 0;
110
111 }
```

source: ...\\Modul 2\\Labaratorium 2\\Zadanie 0\\zadanie_0.cpp

Wyjście:

```
Microsoft Visual Studio Debug Console
Quiz:
Jaki dzwiek wydaje kot?

a: miau
b: muu
c: hau

Twoja odpowiedz [a|b|c]: a
Najwazniejsza osoba w Polsce?

a: premier
b: prezes
c: prezydent

Twoja odpowiedz [a|b|c]: b
Ktora godzina to polnoc?

a: 11:00
b: 12:00
c: 0:00

Twoja odpowiedz [a|b|c]: c
Punkty: 3

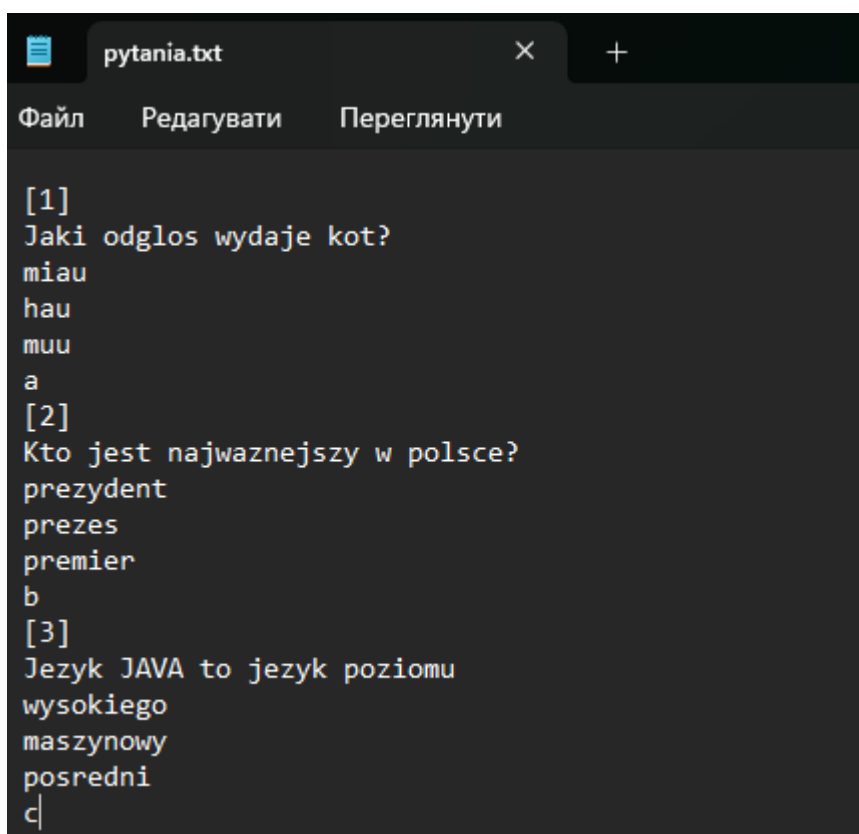
C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzeszek\Modul 2\Labaratorium 2\Zadanie 0\x64\Debug\zadanie_0.exe (process 21968) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Zadanie 1

Stosując OOP, napisać program prostego quizu. Program zadaje pytania odczytywane z pliku tekstowego. Każde pytanie posiada trzy odpowiedzi. Użytkownik odpowiada a program zlicza punkty.

Rozwiązanie zadania 1:

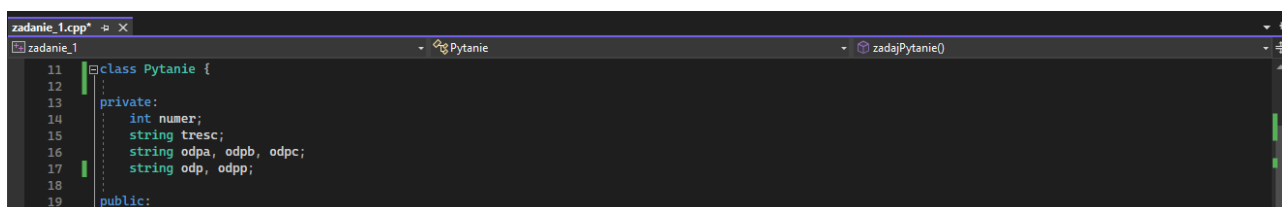
Plik tekstowy pytania.txt:



```
[1]
Jaki odgłos wydaje kot?
miau
hau
muu
a
[2]
Kto jest najważniejszy w polsce?
prezydent
prezes
premier
b
[3]
Język JAVA to język poziomu
wysokiego
maszynowy
pośredni
c
```

source: ...\\Modul 2\\Labaratorium 2\\Zadanie 1\\pytania.txt

Klasa:

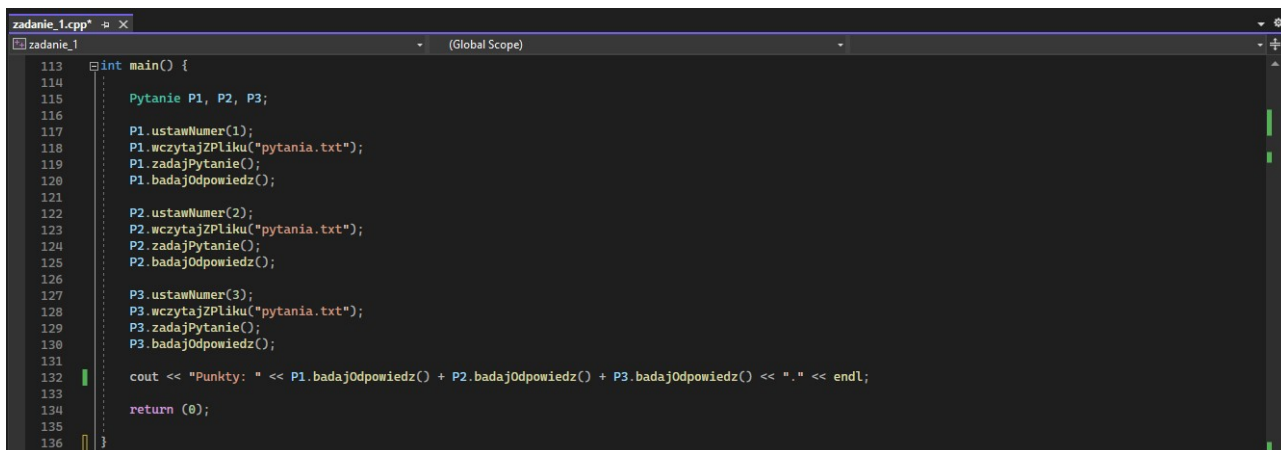


```
11 class Pytanie {
12
13 private:
14     int numer;
15     string tresc;
16     string odpa, odpb, odpc;
17     string odp, odpp;
18
19 public:
```

source: ...\\Modul 2\\Labaratorium 2\\Zadanie 2\\zadanie_1.cpp

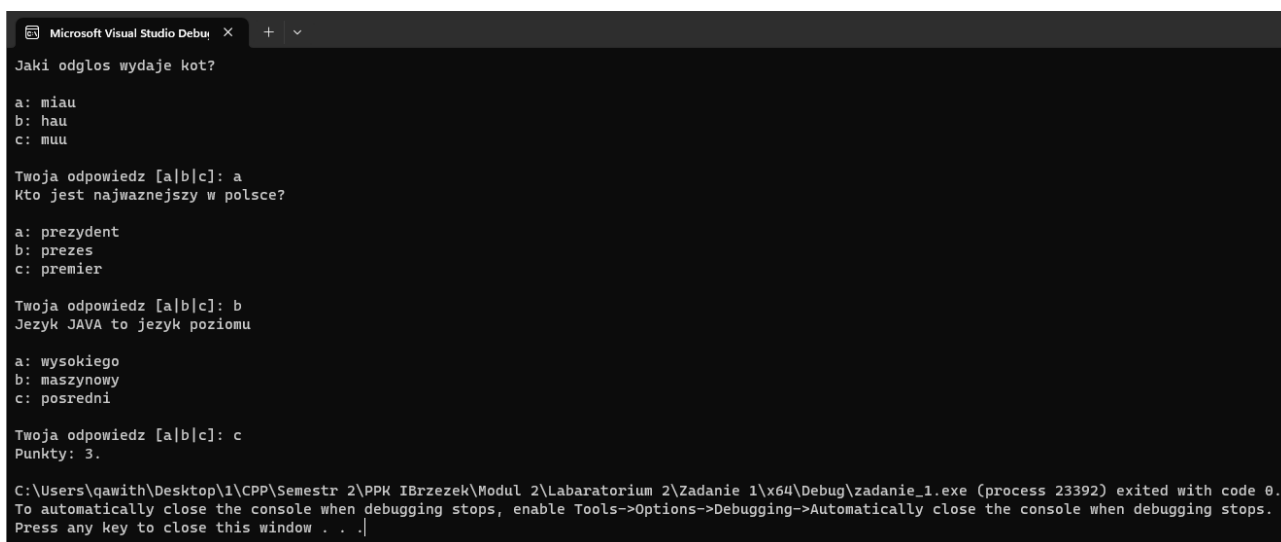
Metody:

```
zadanie_1.cpp
17     string odp, odpp;
18
19 public:
20
21     void ustawNumer(int t) {
22
23         if (t < 1 && t > ILOSC_PYTAN) {
24             cout << "Nie poprawny numer!" << endl;
25         }
26         else {
27             numer = t;
28         }
29     }
30
31     void wczytajZPliku(string nazwa) {
32
33         ifstream file(nazwa);
34
35         if (!file.is_open()) {
36             cout << "Blad otwarcia pliku!" << endl;
37             abort();
38         }
39         else if (file.is_open()) {
40             string line;
41             string snr = to_string(numer);
42             snr = "[" + snr + "]";
43
44             while (getline(file, line)) {
45                 if (line == snr) {
46                     getline(file, line);
47                     tresc = line;
48
49                     getline(file, line);
50                     odpa = line;
51
52                     getline(file, line);
53                     odpb = line;
54
55                     getline(file, line);
56                     odpc = line;
57
58                     getline(file, line);
59                     odpp = line;
60
61                     file.close();
62                 }
63             }
64         }
65     }
66
67     void zadajPytanie() {
68         cout << tresc << endl << endl;
69
70         cout << "a: " << odpa << endl;
71         cout << "b: " << odpb << endl;
72         cout << "c: " << odpc << endl << endl;
73
74         cout << "Twoja odpowiedz [a|b|c]: " << flush;
75         cin >> odp;
76     }
77
78     int badajOdpowiedz() {
79         if (odp == odpp) {
80             return 1;
81         }
82         else {
83             return 0;
84         }
85     }
86
87     ~Klasa() {
88     }
89
90     Klasa() {
91     }
92
93     Klasa() {
94     }
95
96     Klasa() {
97     }
98
99     Klasa() {
100     }
101
102     Klasa() {
103     }
104
105     Klasa() {
106     }
107
108     Klasa() {
109     }
110
111     Klasa() {
112     }
113
114     Klasa() {
115     }
116
117     Klasa() {
118     }
119
120     Klasa() {
121     }
122
123     Klasa() {
124     }
125
126     Klasa() {
127     }
128
129     Klasa() {
130     }
131
132     Klasa() {
133     }
134
135     Klasa() {
136     }
137
138     Klasa() {
139     }
140
141     Klasa() {
142     }
143
144     Klasa() {
145     }
146
147     Klasa() {
148     }
149
150     Klasa() {
151     }
152
153     Klasa() {
154     }
155
156     Klasa() {
157     }
158
159     Klasa() {
160     }
161
162     Klasa() {
163     }
164
165     Klasa() {
166     }
167
168     Klasa() {
169     }
170
171     Klasa() {
172     }
173
174     Klasa() {
175     }
176
177     Klasa() {
178     }
179
180     Klasa() {
181     }
182
183     Klasa() {
184     }
185
186     Klasa() {
187     }
188
189     Klasa() {
190     }
191
192     Klasa() {
193     }
194
195     Klasa() {
196     }
197
198     Klasa() {
199     }
200
201     Klasa() {
202     }
203
204     Klasa() {
205     }
206
207     Klasa() {
208     }
209
210     Klasa() {
211     }
212
213     Klasa() {
214     }
215
216     Klasa() {
217     }
218
219     Klasa() {
220     }
221
222     Klasa() {
223     }
224
225     Klasa() {
226     }
227
228     Klasa() {
229     }
230
231     Klasa() {
232     }
233
234     Klasa() {
235     }
236
237     Klasa() {
238     }
239
240     Klasa() {
241     }
242
243     Klasa() {
244     }
245
246     Klasa() {
247     }
248
249     Klasa() {
250     }
251
252     Klasa() {
253     }
254
255     Klasa() {
256     }
257
258     Klasa() {
259     }
260
261     Klasa() {
262     }
263
264     Klasa() {
265     }
266
267     Klasa() {
268     }
269
270     Klasa() {
271     }
272
273     Klasa() {
274     }
275
276     Klasa() {
277     }
278
279     Klasa() {
280     }
281
282     Klasa() {
283     }
284
285     Klasa() {
286     }
287
288     Klasa() {
289     }
290
291     Klasa() {
292     }
293
294     Klasa() {
295     }
296
297     Klasa() {
298     }
299
300     Klasa() {
301     }
302
303     Klasa() {
304     }
305
306     Klasa() {
307     }
308
309     Klasa() {
310     }
311
312     Klasa() {
313     }
314
315     Klasa() {
316     }
317
318     Klasa() {
319     }
320
321     Klasa() {
322     }
323
324     Klasa() {
325     }
326
327     Klasa() {
328     }
329
330     Klasa() {
331     }
332
333     Klasa() {
334     }
335
336     Klasa() {
337     }
338
339     Klasa() {
340     }
341
342     Klasa() {
343     }
344
345     Klasa() {
346     }
347
348     Klasa() {
349     }
350
351     Klasa() {
352     }
353
354     Klasa() {
355     }
356
357     Klasa() {
358     }
359
360     Klasa() {
361     }
362
363     Klasa() {
364     }
365
366     Klasa() {
367     }
368
369     Klasa() {
370     }
371
372     Klasa() {
373     }
374
375     Klasa() {
376     }
377
378     Klasa() {
379     }
380
381     Klasa() {
382     }
383
384     Klasa() {
385     }
386
387     Klasa() {
388     }
389
390     Klasa() {
391     }
392
393     Klasa() {
394     }
395
396     Klasa() {
397     }
398
399     Klasa() {
400     }
401
402     Klasa() {
403     }
404
405     Klasa() {
406     }
407
408     Klasa() {
409     }
410
411     Klasa() {
412     }
413
414     Klasa() {
415     }
416
417     Klasa() {
418     }
419
420     Klasa() {
421     }
422
423     Klasa() {
424     }
425
426     Klasa() {
427     }
428
429     Klasa() {
430     }
431
432     Klasa() {
433     }
434
435     Klasa() {
436     }
437
438     Klasa() {
439     }
440
441     Klasa() {
442     }
443
444     Klasa() {
445     }
446
447     Klasa() {
448     }
449
450     Klasa() {
451     }
452
453     Klasa() {
454     }
455
456     Klasa() {
457     }
458
459     Klasa() {
460     }
461
462     Klasa() {
463     }
464
465     Klasa() {
466     }
467
468     Klasa() {
469     }
470
471     Klasa() {
472     }
473
474     Klasa() {
475     }
476
477     Klasa() {
478     }
479
480     Klasa() {
481     }
482
483     Klasa() {
484     }
485
486     Klasa() {
487     }
488
489     Klasa() {
490     }
491
492     Klasa() {
493     }
494
495     Klasa() {
496     }
497
498     Klasa() {
499     }
500
501     Klasa() {
502     }
503
504     Klasa() {
505     }
506
507     Klasa() {
508     }
509
510     Klasa() {
511     }
512
513     Klasa() {
514     }
515
516     Klasa() {
517     }
518
519     Klasa() {
520     }
521
522     Klasa() {
523     }
524
525     Klasa() {
526     }
527
528     Klasa() {
529     }
530
531     Klasa() {
532     }
533
534     Klasa() {
535     }
536
537     Klasa() {
538     }
539
540     Klasa() {
541     }
542
543     Klasa() {
544     }
545
546     Klasa() {
547     }
548
549     Klasa() {
550     }
551
552     Klasa() {
553     }
554
555     Klasa() {
556     }
557
558     Klasa() {
559     }
560
561     Klasa() {
562     }
563
564     Klasa() {
565     }
566
567     Klasa() {
568     }
569
570     Klasa() {
571     }
572
573     Klasa() {
574     }
575
576     Klasa() {
577     }
578
579     Klasa() {
580     }
581
582     Klasa() {
583     }
584
585     Klasa() {
586     }
587
588     Klasa() {
589     }
590
591     Klasa() {
592     }
593
594     Klasa() {
595     }
596
597     Klasa() {
598     }
599
600     Klasa() {
601     }
602
603     Klasa() {
604     }
605
606     Klasa() {
607     }
608
609     Klasa() {
610     }
611
612     Klasa() {
613     }
614
615     Klasa() {
616     }
617
618     Klasa() {
619     }
620
621     Klasa() {
622     }
623
624     Klasa() {
625     }
626
627     Klasa() {
628     }
629
630     Klasa() {
631     }
632
633     Klasa() {
634     }
635
636     Klasa() {
637     }
638
639     Klasa() {
640     }
641
642     Klasa() {
643     }
644
645     Klasa() {
646     }
647
648     Klasa() {
649     }
650
651     Klasa() {
652     }
653
654     Klasa() {
655     }
656
657     Klasa() {
658     }
659
660     Klasa() {
661     }
662
663     Klasa() {
664     }
665
666     Klasa() {
667     }
668
669     Klasa() {
670     }
671
672     Klasa() {
673     }
674
675     Klasa() {
676     }
677
678     Klasa() {
679     }
680
681     Klasa() {
682     }
683
684     Klasa() {
685     }
686
687     Klasa() {
688     }
689
690     Klasa() {
691     }
692
693     Klasa() {
694     }
695
696     Klasa() {
697     }
698
699     Klasa() {
700     }
701
702     Klasa() {
703     }
704
705     Klasa() {
706     }
707
708     Klasa() {
709     }
710
711     Klasa() {
712     }
713
714     Klasa() {
715     }
716
717     Klasa() {
718     }
719
720     Klasa() {
721     }
722
723     Klasa() {
724     }
725
726     Klasa() {
727     }
728
729     Klasa() {
730     }
731
732     Klasa() {
733     }
734
735     Klasa() {
736     }
737
738     Klasa() {
739     }
740
741     Klasa() {
742     }
743
744     Klasa() {
745     }
746
747     Klasa() {
748     }
749
750     Klasa() {
751     }
752
753     Klasa() {
754     }
755
756     Klasa() {
757     }
758
759     Klasa() {
760     }
761
762     Klasa() {
763     }
764
765     Klasa() {
766     }
767
768     Klasa() {
769     }
770
771     Klasa() {
772     }
773
774     Klasa() {
775     }
776
777     Klasa() {
778     }
779
780     Klasa() {
781     }
782
783     Klasa() {
784     }
785
786     Klasa() {
787     }
788
789     Klasa() {
790     }
791
792     Klasa() {
793     }
794
795     Klasa() {
796     }
797
798     Klasa() {
799     }
800
801     Klasa() {
802     }
803
804     Klasa() {
805     }
806
807     Klasa() {
808     }
809
810     Klasa() {
811     }
812
813     Klasa() {
814     }
815
816     Klasa() {
817     }
818
819     Klasa() {
820     }
821
822     Klasa() {
823     }
824
825     Klasa() {
826     }
827
828     Klasa() {
829     }
830
831     Klasa() {
832     }
833
834     Klasa() {
835     }
836
837     Klasa() {
838     }
839
840     Klasa() {
841     }
842
843     Klasa() {
844     }
845
846     Klasa() {
847     }
848
849     Klasa() {
850     }
851
852     Klasa() {
853     }
854
855     Klasa() {
856     }
857
858     Klasa() {
859     }
860
861     Klasa() {
862     }
863
864     Klasa() {
865     }
866
867     Klasa() {
868     }
869
870     Klasa() {
871     }
872
873     Klasa() {
874     }
875
876     Klasa() {
877     }
878
879     Klasa() {
880     }
881
882     Klasa() {
883     }
884
885     Klasa() {
886     }
887
888     Klasa() {
889     }
890
891     Klasa() {
892     }
893
894     Klasa() {
895     }
896
897     Klasa() {
898     }
899
900     Klasa() {
901     }
902
903     Klasa() {
904     }
905
906     Klasa() {
907     }
908
909     Klasa() {
910     }
911
912     Klasa() {
913     }
914
915     Klasa() {
916     }
917
918     Klasa() {
919     }
920
921     Klasa() {
922     }
923
924     Klasa() {
925     }
926
927     Klasa() {
928     }
929
930     Klasa() {
931     }
932
933     Klasa() {
934     }
935
936     Klasa() {
937     }
938
939     Klasa() {
940     }
941
942     Klasa() {
943     }
944
945     Klasa() {
946     }
947
948     Klasa() {
949     }
950
951     Klasa() {
952     }
953
954     Klasa() {
955     }
956
957     Klasa() {
958     }
959
960     Klasa() {
961     }
962
963     Klasa() {
964     }
965
966     Klasa() {
967     }
968
969     Klasa() {
970     }
971
972     Klasa() {
973     }
974
975     Klasa() {
976     }
977
978     Klasa() {
979     }
980
981     Klasa() {
982     }
983
984     Klasa() {
985     }
986
987     Klasa() {
988     }
989
990     Klasa() {
991     }
992
993     Klasa() {
994     }
995
996     Klasa() {
997     }
998
999     Klasa() {
1000     }
1001
1002     Klasa() {
1003     }
1004
1005     Klasa() {
1006     }
1007
1008     Klasa() {
1009     }
1010
1011     Klasa() {
1012     }
1013
1014     Klasa() {
1015     }
1016
1017     Klasa() {
1018     }
1019
1020     Klasa() {
1021     }
1022
1023     Klasa() {
1024     }
1025
1026     Klasa() {
1027     }
1028
1029     Klasa() {
1030     }
1031
1032     Klasa() {
1033     }
1034
1035     Klasa() {
1036     }
1037
1038     Klasa() {
1039     }
1040
1041     Klasa() {
1042     }
1043
1044     Klasa() {
1045     }
1046
1047     Klasa() {
1048     }
1049
1050     Klasa() {
1051     }
1052
1053     Klasa() {
1054     }
1055
1056     Klasa() {
1057     }
1058
1059     Klasa() {
1060     }
1061
1062     Klasa() {
1063     }
1064
1065     Klasa() {
1066     }
1067
1068     Klasa() {
1069     }
1070
1071     Klasa() {
1072     }
1073
1074     Klasa() {
1075     }
1076
1077     Klasa() {
1078     }
1079
1080     Klasa() {
1081     }
1082
1083     Klasa() {
1084     }
1085
1086     Klasa() {
1087     }
1088
1089     Klasa() {
1090     }
1091
1092     Klasa() {
1093     }
1094
1095     Klasa() {
1096     }
1097
1098     Klasa() {
1099     }
1100
1101     Klasa() {
1102     }
1103
1104     Klasa() {
1105     }
1106
1107     Klasa() {
1108     }
1109
1110     Klasa() {
1111     }
1112
1113     Klasa() {
1114     }
1115
1116     Klasa() {
1117     }
1118
1119     Klasa() {
1120     }
1121
1122     Klasa() {
1123     }
1124
1125     Klasa() {
1126     }
1127
1128     Klasa() {
1129     }
1130
1131     Klasa() {
1132     }
1133
1134     Klasa() {
1135     }
1136
1137     Klasa() {
1138     }
1139
1140     Klasa() {
1141     }
1142
1143     Klasa() {
1144     }
1145
1146     Klasa() {
1147     }
1148
1149     Klasa() {
1150     }
1151
1152     Klasa() {
1153     }
1154
1155     Klasa() {
1156     }
1157
1158     Klasa() {
1159     }
1160
1161     Klasa() {
1162     }
1163
1164     Klasa() {
1165     }
1166
1167     Klasa() {
1168     }
1169
1170     Klasa() {
1171     }
1172
1173     Klasa() {
1174     }
1175
1176     Klasa() {
1177     }
1178
1179     Klasa() {
1180     }
1181
1182     Klasa() {
1183     }
1184
1185     Klasa() {
1186     }
1187
1188     Klasa() {
1189     }
1190
1191     Klasa() {
1192     }
1193
1194     Klasa() {
1195     }
1196
1197     Klasa() {
1198     }
1199
1200     Klasa() {
1201     }
1202
1203     Klasa() {
1204     }
1205
1206     Klasa() {
1207     }
1208
1209     Klasa() {
1210     }
1211
1212     Klasa() {
1213     }
1214
1215     Klasa() {
1216     }
1217
1218     Klasa() {
1219     }
1220
1221     Klasa() {
1222     }
1223
1224     Klasa() {
1225     }
1226
1227     Klasa() {
1228     }
1229
1230     Klasa() {
1231     }
1232
1233     Klasa() {
1234     }
1235
1236     Klasa() {
1237     }
1238
1239     Klasa() {
1240     }
1241
1242     Klasa() {
1243     }
1244
1245     Klasa() {
1246     }
1247
1248     Klasa() {
1249     }
1250
1251     Klasa() {
1252     }
1253
1254     Klasa() {
1255     }
1256
1257     Klasa() {
1258     }
1259
1260     Klasa() {
1261     }
1262
1263     Klasa() {
1264     }
1265
1266     Klasa() {
1267     }
1268
1269     Klasa() {
1270     }
1271
1272     Klasa() {
1273     }
1274
1275     Klasa() {
1276     }
1277
1278     Klasa() {
1279     }
1280
1281     Klasa() {
1282     }
1283
1284     Klasa() {
1285     }
1286
1287     Klasa() {
1288     }
1289
1290     Klasa() {
1291     }
1292
1293     Klasa() {
1294     }
1295
1296     Klasa() {
1297     }
1298
1299     Klasa() {
1300     }
1301
1302     Klasa() {
1303     }
1304
1305     Klasa() {
1306     }
1307
1308     Klasa() {
1309     }
1310
1311     Klasa() {
1312     }
1313
1314     Klasa() {
1315     }
1316
1317     Klasa() {
1318     }
1319
1320     Klasa() {
1321     }
1322
1323     Klasa() {
1324     }
1325
1326     Klasa() {
1327     }
1328
1329     Klasa() {
1330     }
1331
1332     Klasa() {
1333     }
1334
1335     Klasa() {
1336     }
1337
1338     Klasa() {
1339     }
1340
1341     Klasa() {
1342     }
1343
1344     Klasa() {
1345     }
1346
1347     Klasa() {
1348     }
1349
1350     Klasa() {
1351     }
1352
1353     Klasa() {
1354     }
1355
1356     Klasa() {
1357     }
1358
1359     Klasa() {
1360     }
1361
1362     Klasa() {
1363     }
1364
1365     Klasa() {
1366     }
1367
1368     Klasa() {
1369     }
1370
1371     Klasa() {
1372     }
1373
1374     Klasa() {
1375     }
1376
1377     Klasa() {
1378     }
1379
1380     Klasa() {
1381     }
1382
1383     Klasa() {
1384     }
1385
1386     Klasa() {
1387     }
1388
1389     Klasa() {
1390     }
1391
1392     Klasa() {
1393     }
1394
1395     Klasa() {
1396     }
1397
1398     Klasa() {
1399     }
1400
1401     Klasa() {
1402     }
1403
1404     Klasa() {
1405     }
1406
1407     Klasa() {
1408     }
1409
1410     Klasa() {
1411     }
1412
1413     Klasa() {
1414     }
1415
1416     Klasa() {
1417     }
1418
1419     Klasa() {
1420     }
1421
1422     Klasa() {
1423     }
1424
1425     Klasa() {
1426     }
1427
1428     Klasa() {
1429     }
1430
1431     Klasa() {
1432     }
1433
1434     Klasa() {
1435     }
1436
1437     Klasa() {
1438     }
1439
1440     Klasa() {
1441     }
1442
1443     Klasa() {
1444     }
1445
1446     Klasa() {
1447     }
1448
1449     Klasa() {
1450     }
1451
1452     Klasa() {
1453     }
1454
1455     Klasa() {
1456     }
1457
1458     Klasa() {
1459     }
1460
1461     Klasa() {
1462     }
1463
1464     Klasa() {
1465     }
1466
1467     Klasa() {
1468     }
1469
1470     Klasa() {
1471     }
1472
1473     Klasa() {
1474     }
1475
1476     Klasa() {
1477     }
1478
1479     Klasa() {
1480     }
1481
1482     Klasa() {
1483     }
1484
1485     Klasa() {
1486     }
1487
1488     Klasa() {
1489     }
1490
1491     Klasa() {
1492     }
1493
1494     Klasa() {
1495     }
1496
1497     Klasa() {
1498     }
1499
1500     Klasa() {
1501     }
1502
1503     Klasa() {
1504     }
1505
1506     Klasa() {
1507     }
1508
1509     Klasa() {
1510     }
1511
1512     Klasa() {
1513     }
1514
1515     Klasa() {
1516     }
1517
1518     Klasa() {
1519     }
1520
1521     Klasa() {
1522     }
1523
1524     Klasa() {
1525     }
1526
1527     Klasa() {
1528     }
1529
1530     Klasa() {
1531     }
1532
1533     Klasa() {
1534     }
1535
1536     Klasa() {
1537     }
1538
1539     Klasa() {
1540     }
1541
1542     Klasa() {
1543     }
1544
1545     Klasa() {
1546     }
1547
1548     Klasa() {
1549     }
1550
1551     Klasa() {
1552     }
1553
1554     Klasa() {
1555     }
1556
1557     Klasa() {
1558     }
1559
1560     Klasa() {
1561     }
1562
1563     Klasa() {
1564     }
1565
1566     Klasa() {
1567     }
1568
1569     Klasa() {
1570     }
1571
1572     Klasa() {
1573     }
1574
1575     Klasa() {
1576     }
1577
1578     Klasa() {
1579     }
1580
1581     Klasa() {
1582     }
1583
1584     Klasa() {
1585     }
1586
1587     Klasa() {
1588     }
1589
1590     Klasa() {
1591     }
1592
1593     Klasa() {
1594     }
1595
1596     Klasa() {
1597     }
1598
1599     Klasa() {
1600     }
1601
1602     Klasa() {
1603     }
1604
1605     Klasa() {
1606     }
1607
1608     Klasa() {
1609     }
1610
1611     Klasa() {
1612     }
1613
1614     Klasa() {
1615     }
1616
1617     Klasa() {
1618     }
1619
1620     Klasa() {
1621     }
1622
1623     Klasa() {
1624     }
1625
1626     Klasa() {
1627     }
1628
1629     Klasa() {
1630     }
1631
1632     Klasa() {
1633     }
1634
1635     Klasa() {
1636     }
1637
1638     Klasa() {
1639     }
1640
1641     Klasa() {
1642     }
1643
1644     Klasa() {
1645     }
1646
1647     Klasa() {
1648     }
1649
1650     Klasa() {
1651     }
1652
1653     Klasa() {
1654     }
1655
1656     Klasa() {
1657     }
1658
1659     Klasa() {
1660     }
1661
1662     Klasa() {
1663     }
1664
1665     Klasa() {
1666     }
1667
1668     Klasa() {
1669     }
1670
1671     Klasa() {
1672     }
1673
1674     Klasa() {
1675     }
1676
1677     Klasa() {
1678     }
1679
1680     Klasa() {
1681     }
1682
1683     Klasa() {
1684     }
1685
1686     Klasa() {
1687     }
1688
1689     Klasa() {
1690     }
1691
1692     Klasa() {
1693     }
1694
1695     Klasa() {
1696     }
1697
1698     Klasa() {
1699     }
1700
1701     Klasa() {
1702     }
1703
1704     Klasa() {
1705     }
1706
1707     Klasa() {
1708     }
1709
1710     Klasa() {
1711     }
1712
1713     Klasa() {
1714     }
1715
1716     Klasa() {
1717     }
1718
1719     Klasa() {
1720     }
1721
1722     Klasa() {
1723     }
1724
1725     Klasa() {
1726     }
1727
1728     Klasa() {
1729     }
1730
1731     Klasa() {
1732     }
1733
1734     Klasa() {
1735     }
1736
1737     Klasa() {
1738     }
1739
1740     Klasa() {
1741     }
1742
1743     Klasa() {
1744     }
1745
1746     Klasa() {
1747     }
1748
1749     Klasa() {
1750     }
1751
1752     Klasa() {
1753     }
1754
1755     Klasa() {
1756     }
1757
1758     Klasa() {
1759     }
1760
1761     Klasa() {
1762     }
1763
1764     Klasa() {
1765     }
1766
1767     Klasa() {
1768     }
1769
1770     Klasa() {
1771     }
1772
1773     Klasa() {
1774     }
1775
1776     Klasa() {
1777     }
1778
1779     Klasa() {
1780     }
1781
1782     Klasa() {
1783     }
1784
1785     Klasa() {
1786     }
1787
1788     Klasa() {
1789     }
1790
1791     Klasa() {
1792     }
1793
1794    
```

Main:

```
113 int main() {  
114     Pytanie P1, P2, P3;  
115  
116     P1.ustawNumer(1);  
117     P1.wczytajZPliku("pytania.txt");  
118     P1.zadajPytanie();  
119     P1.badajOdpowiedz();  
120  
121     P2.ustawNumer(2);  
122     P2.wczytajZPliku("pytania.txt");  
123     P2.zadajPytanie();  
124     P2.badajOdpowiedz();  
125  
126     P3.ustawNumer(3);  
127     P3.wczytajZPliku("pytania.txt");  
128     P3.zadajPytanie();  
129     P3.badajOdpowiedz();  
130  
131     cout << "Punkty: " << P1.badajOdpowiedz() + P2.badajOdpowiedz() + P3.badajOdpowiedz() << "." << endl;  
132  
133     return (0);  
134 }  
135  
136
```

source: ...\\Modul 2\\Labaratorium 2\\Zadanie 1\\zadanie_1.cpp

Wyjście:

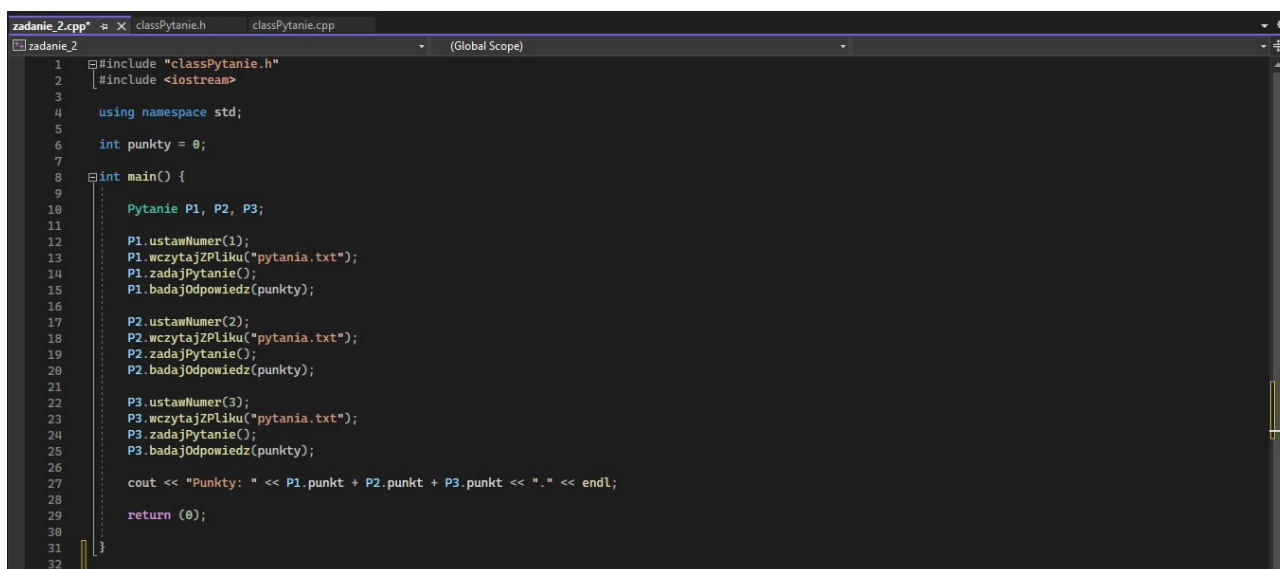
```
Microsoft Visual Studio Debug Console  
-----  
Jaki odglos wydaje kot?  
a: miau  
b: hau  
c: muu  
Twoja odpowiedz [a|b|c]: a  
Kto jest najwazniejszy w polsce?  
a: prezydent  
b: prezes  
c: premier  
Twoja odpowiedz [a|b|c]: b  
Jezyk JAVA to jezyk poziomemu  
a: wysokiego  
b: maszynowy  
c: posredni  
Twoja odpowiedz [a|b|c]: c  
Punkty: 3.  
C:\\Users\\qawith\\Desktop\\1\\CPP\\Semestr 2\\PPK IBrzechek\\Modul 2\\Labaratorium 2\\Zadanie 1\\x64\\Debug\\zadanie_1.exe (process 23392) exited with code 0.  
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.  
Press any key to close this window . . .|
```


Zadanie 2

Powyższy program zapisać w plikach: nagłówkowym oraz modułu. W pliku nagłówkowym znajduje się opis klasy a w pliku modułu ciała metod. Opisać przeznaczenie i budowę takich plików.

Rozwiązanie zadania 2:

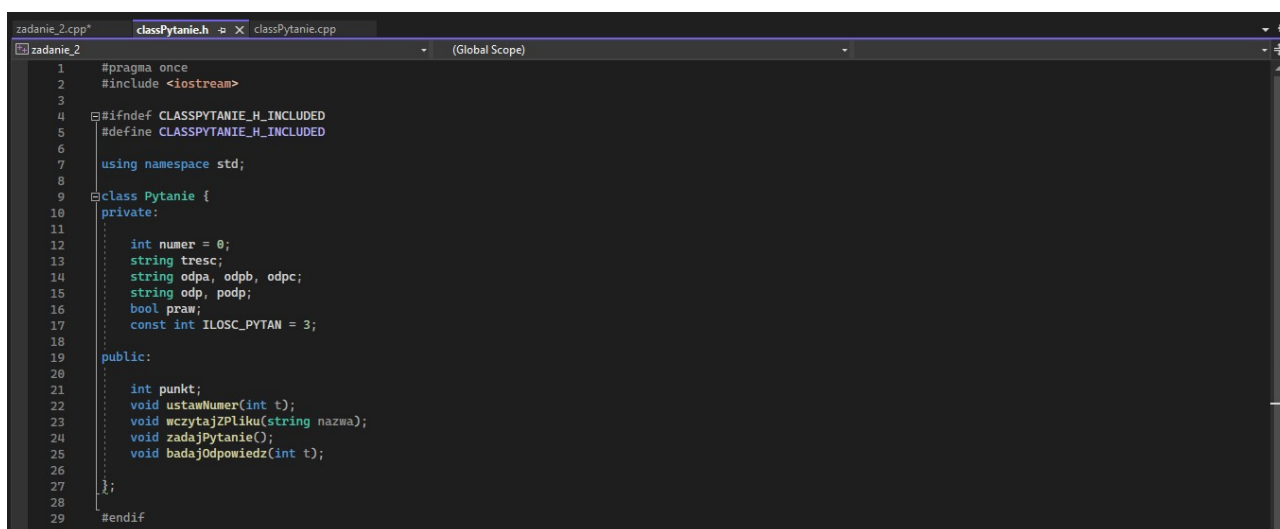
Plik główny zadanie_2.cpp:



```
1 #include "classPytanie.h"
2 #include <iostream>
3
4 using namespace std;
5
6 int punkty = 0;
7
8 int main() {
9
10     Pytanie P1, P2, P3;
11
12     P1.ustawNumer(1);
13     P1.wczytajZPliku("pytania.txt");
14     P1.zadajPytanie();
15     P1.badaJodpowiedz(punkty);
16
17     P2.ustawNumer(2);
18     P2.wczytajZPliku("pytania.txt");
19     P2.zadajPytanie();
20     P2.badaJodpowiedz(punkty);
21
22     P3.ustawNumer(3);
23     P3.wczytajZPliku("pytania.txt");
24     P3.zadajPytanie();
25     P3.badaJodpowiedz(punkty);
26
27     cout << "Punkty: " << P1.punkt + P2.punkt + P3.punkt << "." << endl;
28
29     return 0;
30 }
31
32
```

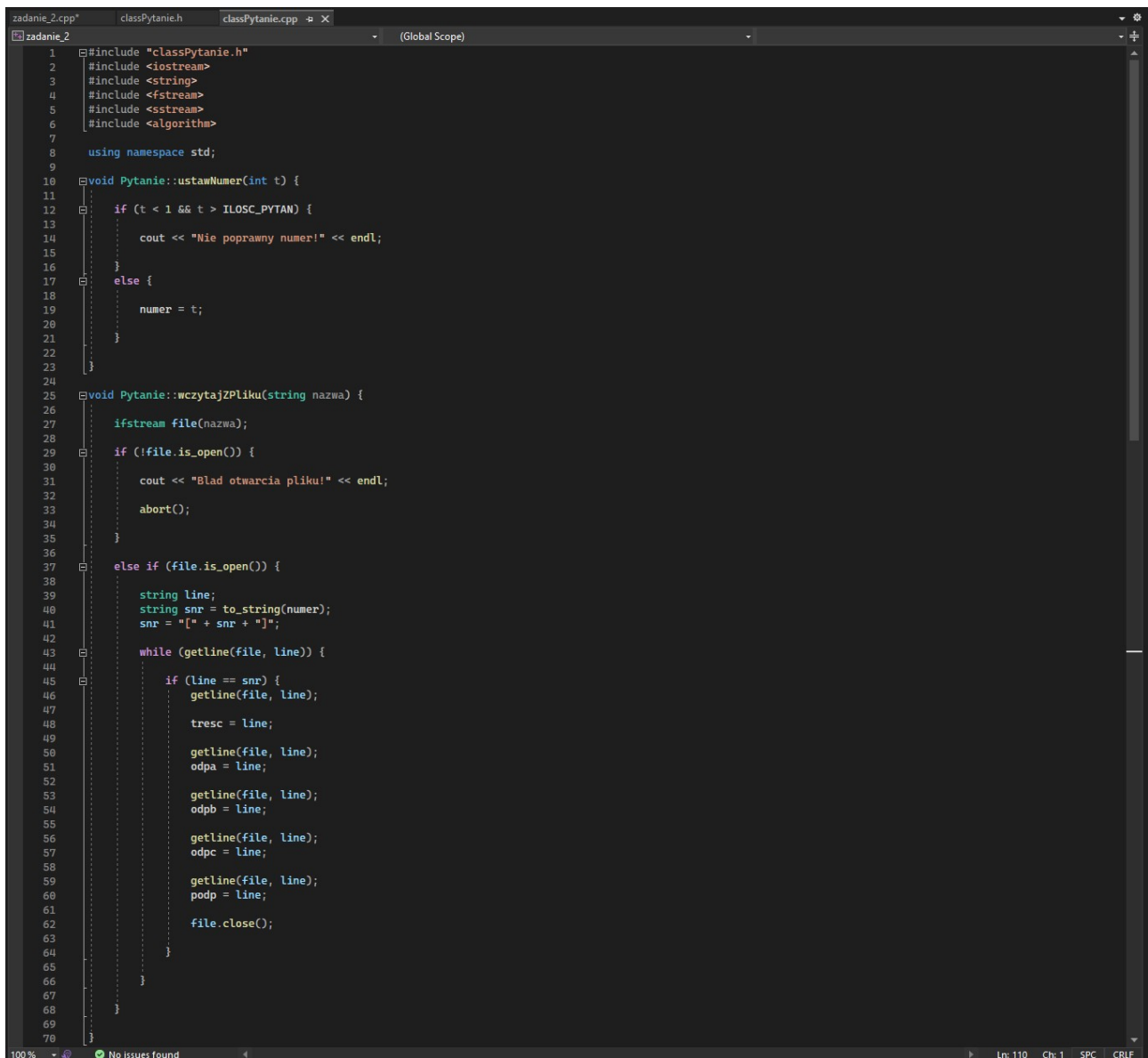
source: ...\\Modul 2\\Labaratorium 2\\Zadanie 2\\zadanie_2.cpp

Plik nagłówkowy classPytanie.h:



```
1 #pragma once
2 #include <iostream>
3
4 #ifndef CLASSPYTANIE_H_INCLUDED
5 #define CLASSPYTANIE_H_INCLUDED
6
7 using namespace std;
8
9 class Pytanie {
10 private:
11
12     int numer = 0;
13     string tresc;
14     string odpa, odpb, odpc;
15     string odp, podp;
16     bool praw;
17     const int ILOSC_PYTAN = 3;
18
19 public:
20
21     int punkt;
22     void ustawNumer(int t);
23     void wczytajZPliku(string nazwa);
24     void zadajPytanie();
25     void badaJodpowiedz(int t);
26
27 };
28
29 #endif
30
```

source: ...\\Modul 2\\Labaratorium 2\\Zadanie 1\\classPytanie.h

Plik modul classPytanie.cpp:

```
1 #include "classPytanie.h"
2 #include <iostream>
3 #include <string>
4 #include <fstream>
5 #include <sstream>
6 #include <algorithm>
7
8 using namespace std;
9
10 void Pytanie::ustawNumer(int t) {
11     if (t < 1 && t > ILOSC_PYTAN) {
12         cout << "Nie poprawny numer!" << endl;
13     }
14     else {
15         numer = t;
16     }
17 }
18
19 void Pytanie::wczytajZPliku(string nazwa) {
20     ifstream file(nazwa);
21     if (!file.is_open()) {
22         cout << "Blad otwarcia pliku!" << endl;
23         abort();
24     }
25     else if (file.is_open()) {
26         string line;
27         string snr = to_string(numer);
28         snr = "[" + snr + "]";
29         while (getline(file, line)) {
30             if (line == snr) {
31                 getline(file, line);
32                 tresc = line;
33                 getline(file, line);
34                 odpa = line;
35                 getline(file, line);
36                 odpb = line;
37                 getline(file, line);
38                 odpc = line;
39                 getline(file, line);
40                 podp = line;
41                 file.close();
42             }
43         }
44     }
45 }
```

```
zadanie_2.cpp  classPytanie.h  classPytanie.cpp  (Global Scope)
72 void Pytanie::zadajPytanie() {
73
74     if (number > ILOSC_PYTAN) {
75
76         cout << "Nie poprawny numer!" << endl;
77     }
78
79     else if (number < ILOSC_PYTAN + 1 && number > 0) {
80
81         cout << "Pytanie numer " << number << ":" << endl;
82         cout << tresc << endl << endl;
83
84         cout << "a. " << odpa << endl;
85         cout << "b. " << odpb << endl;
86         cout << "c. " << odpd << endl << endl;
87
88         cout << "Twoja odpowiedz: "; cin >> odp; cout << endl;
89         cout << "Odpowiedz zapisana." << endl << endl;
90     }
91 }
92
93
94
95 void Pytanie::badajOdpowiedz(int t) {
96
97     if (odp == podp) {
98
99         punkt = 1;
100     }
101     else {
102
103         punkt = 0;
104     }
105
106 }
107
108
109
```

source: ...\\Modul 2\\Laboratorium 2\\Zadanie 1\\classPytanie.cpp

Wyjście:

```
Microsoft Visual Studio Debug
Pytanie numer 1:
Jaki odglos wydaje kot?

a. miau
b. hau
c. muu

Twoja odpowiedz: a

Odpowiedz zapisana.

Pytanie numer 2:
Kto jest najwazniejszy w polsce?

a. prezydent
b. prezes
c. premier

Twoja odpowiedz: b

Odpowiedz zapisana.

Pytanie numer 3:
Jezyk JAVA to jezyk poziomu

a. wysokiego
b. maszynowy
c. posredni

Twoja odpowiedz: c

Odpowiedz zapisana.

Punkty: 3.

C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 2\Laboratorium 2\Zadanie 2\x64\Debug\zadanie_2.exe (process 19696) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

W pliku nagłówkowym znajduje się opis klasy a w pliku modułu ciała metod. Opisać przeznaczenie i budowę takich plików.

Plik nagłówkowy to plik tekstowy, który zawiera deklaracje funkcji, klas i zmiennych, które są używane w innym module lub pliku źródłowym.

Plik ciała metod to plik źródłowy, który zawiera definicje funkcji i klas, które zostały zadeklarowane w pliku nagłówkowym. Plik ciała metod zawiera implementację metod i funkcji, które zostały zadeklarowane w pliku nagłówkowym

Zadanie 3

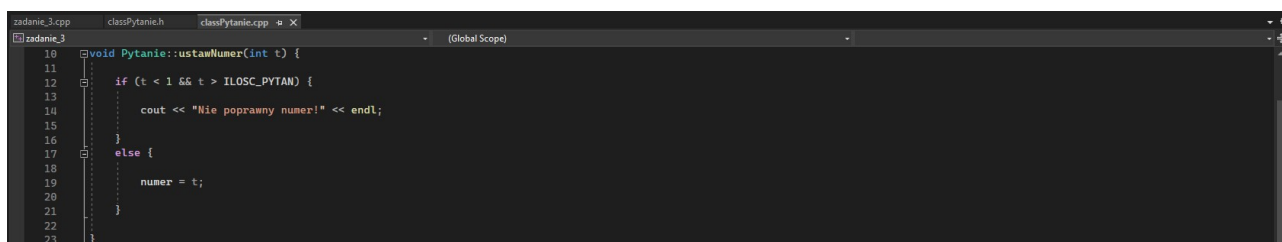
W powyższym programie zabezpieczyć dostęp do atrybutów klasy. Czy metody powinny być publiczne czy prywatne? Czy wszystkie atrybuty muszą być prywatne, a może powinny być publiczne?

Rozwiązanie zadania 3:

Czy metody powinny być publiczne czy prywatne? Czy wszystkie atrybuty muszą być prywatne, a może powinny być publiczne?

W tym programie dostęp potrzebny tylko do zmiennej **numer**. Inne atrybuty muszą być prywatne, żeby nie możliwe było zmienić parametry quiz'a.

Metoda ustawNumer():



```
10 void Pytanie::ustawNumer(int t) {
11
12     if (t < 1 && t > ILOSC_PYTAN) {
13
14         cout << "Nie poprawny numer!" << endl;
15
16     }
17     else {
18
19         numer = t;
20
21     }
22
23 }
```

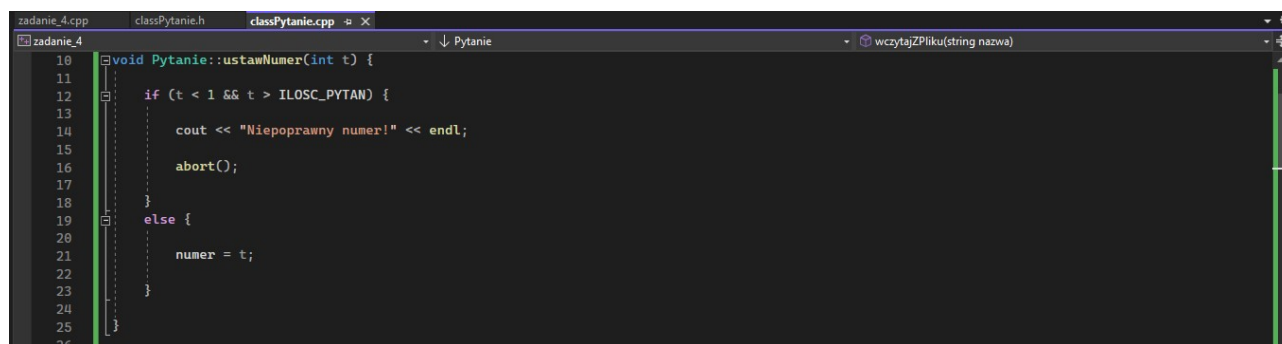
source: ...\\Modul 2\\Labaratorium 2\\Zadanie 3\\classPytanie.cpp

Zadanie 4:

Zabezpieczyć cechy klasy tak, aby nie było bezpośredniego dostępu do żadnej z nich z zewnątrz ale aby można było także bezpiecznie ustalać wartość numeru pytania oraz pobierać (tylko pobierać czyli ReadOnly) punktację. Wprowadzić i opisać pojęcie getter i setter. W kwestii ustalenia numeru pytania dla danego obiektu zastanowić się czy taka metoda ustalająca powinna coś zwracać? Jeśli tak to jakiego typu oraz jak użyć takiej informacji zwrotnej w programie?

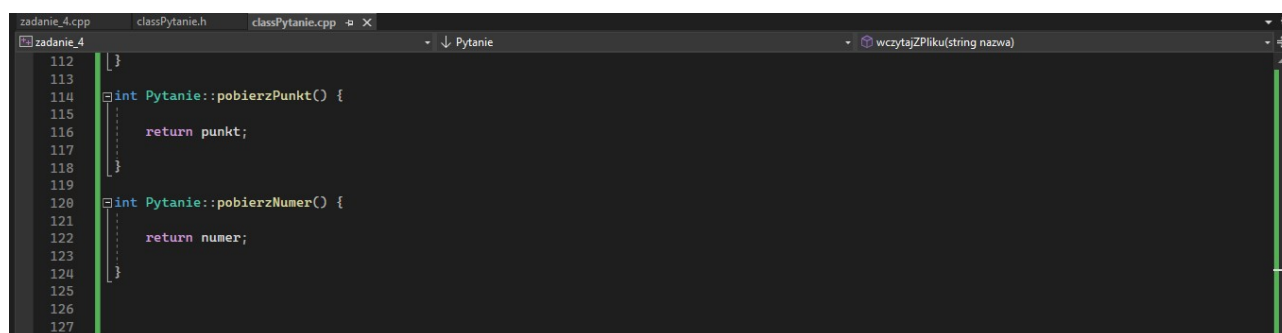
Rozwiązanie zadania 4:

Zabezpieczyć cechy klasy tak, aby nie było bezpośredniego dostępu do żadnej z nich z zewnątrz ale aby można było także bezpiecznie ustalać wartość numeru pytania oraz pobierać (tylko pobierać czyli ReadOnly) punktację.



```
10 void Pytanie::ustawNumer(int t) {
11
12     if (t < 1 && t > ILOSC_PYTAN) {
13
14         cout << "Niepoprawny numer!" << endl;
15
16         abort();
17
18     }
19     else {
20
21         numer = t;
22
23     }
24 }
25
26
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 4\\classPytanie.cpp



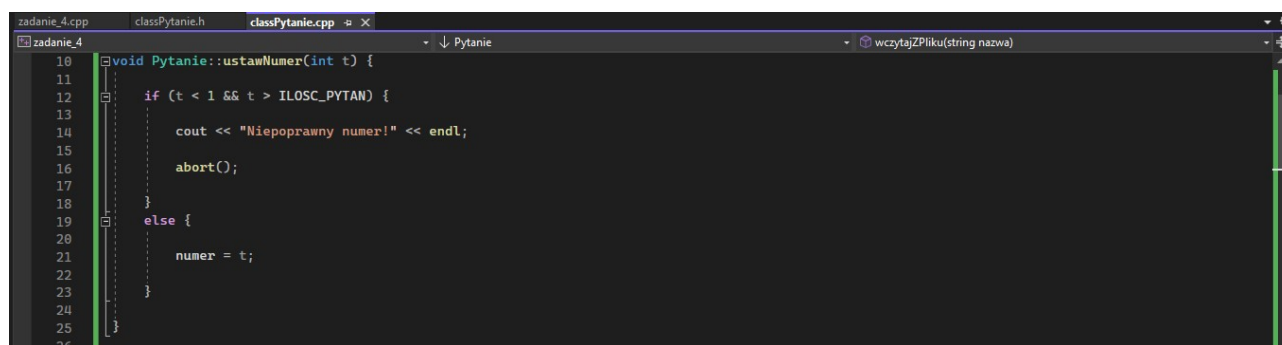
```
112
113
114 int Pytanie::pobierzPunkt() {
115
116     return punkt;
117
118 }
119
120 int Pytanie::pobierzNumer() {
121
122     return numer;
123
124 }
125
126
127
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 4\\classPytanie.cpp

Wprowadzić i opisać pojęcie getter i setter.

Getter i setter to nazwy metod, które służą do uzyskiwania i ustawiania wartości zmiennych składowych obiektów klasy.

Przykład implementacji settera dla zmiennej składowej numer w klasie Pytanie:



```
10 void Pytanie::ustawNumer(int t) {
11     if (t < 1 && t > ILOSC_PYTAN) {
12         cout << "Niepoprawny numer!" << endl;
13         abort();
14     }
15     else {
16         numer = t;
17     }
18 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 4\\classPytanie.cpp

W kwestii ustalenia numeru pytania dla danego obiektu zastanowić się czy taka metoda ustalająca powinna coś zwracać?

Metoda ta przyjmuje jako argument int t, który jest nową wartością prywatnej zmiennej składowej numer dla danego obiektu. W metodzie tej dodano również warunek, który sprawdza, czy wartość numeru jest zgodna z zakresem **ILOSC_PYTAN** i w przypadku nieprawidłowej wartości wyświetla stosowny komunikat. W tym konkretnym przypadku metoda setter nie zwraca żadnej informacji zwrotnej, ponieważ nie jest to potrzebne do dalszych operacji programu. Setter jedynie ustawia wartość prywatnej zmiennej składowej numer dla danego obiektu.