

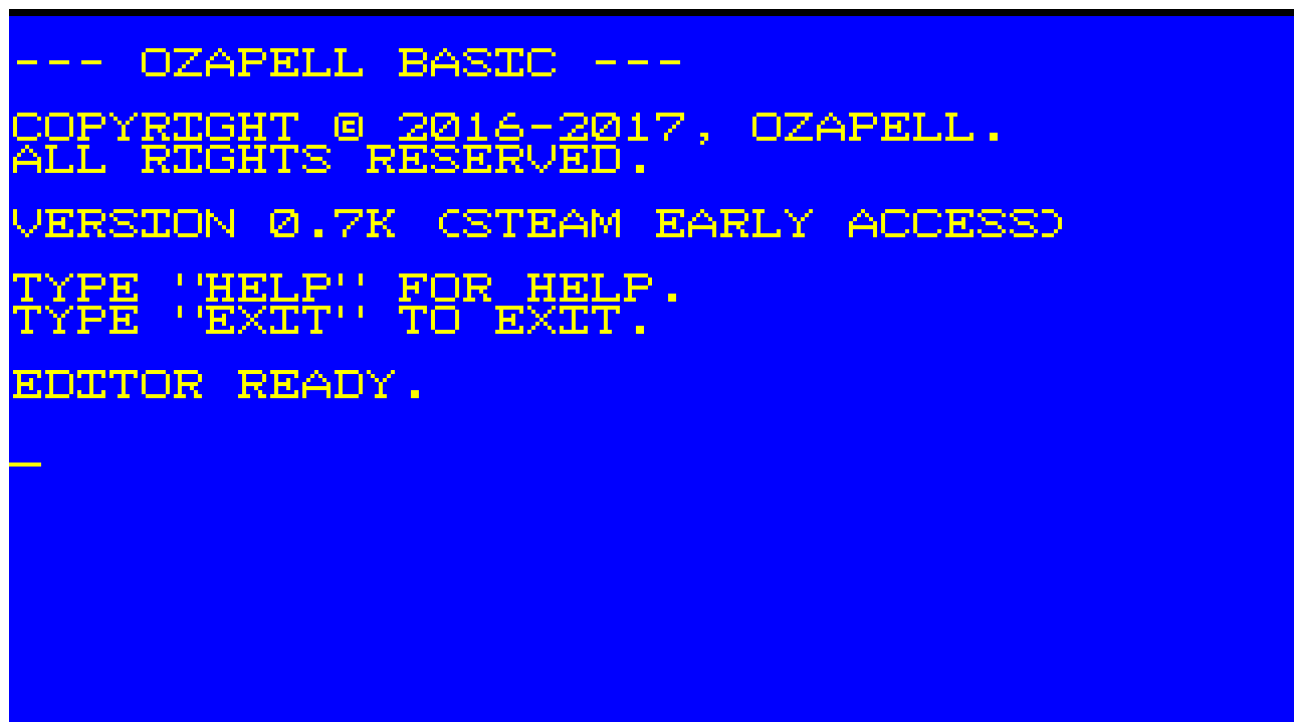
Ozapell Basic
Beginner's Manual

Copyright © 2017, Ozapell.
All Rights Reserved.

The Ozapell Basic Beginner's Manual is intended to get new users started with programming. The built-in help system contains all editor commands, keywords and supplied subroutines. Consult the help system (type "HELP") to list categories available. Then type "HELP [topic name]" for specific help about a command, keyword or subroutine.

Chapter 1. Editing

When Ozapell Basic is launched, it begins in editor mode. In this mode, programs can be created, loaded, edited and executed.



```
--- OZAPPELL BASIC ---  
COPYRIGHT © 2016-2017, OZAPPELL.  
ALL RIGHTS RESERVED.  
VERSION 0.7K (STEAM EARLY ACCESS)  
TYPE 'HELP' FOR HELP.  
TYPE 'EXIT' TO EXIT.  
EDITOR READY.  
_
```

Programs can be edited using the line editor, the fullscreen editor or an external text editor such as Notepad.

For now, we'll be using the line editor to demonstrate smaller programs.

To use a command in the editor, type the command and press [ENTER].

To clear the screen, type CLS and press [ENTER].

```
--- OZAPPELL BASIC ---  
COPYRIGHT © 2016-2017, OZAPPELL.  
ALL RIGHTS RESERVED.  
VERSION 0.7K (STEAM EARLY ACCESS)  
TYPE "HELP" FOR HELP.  
TYPE "EXIT" TO EXIT.  
EDITOR READY.  
CLS_
```



To program using the line editor, type the line number and then the code you want to enter.

```
0 PRINT 'HELLO'_
```

PRINT is a subroutine to display text on the screen. Subroutines will be discussed in a later chapter.

To list your program, type LIST and press [ENTER].

```
LIST  
0 PRINT 'HELLO'  
_
```

To save a program, type SAVE and the filename. Filenames may contains letters or numbers only. Ozapell Basic doesn't support extensions or paths to prevent access to other files.

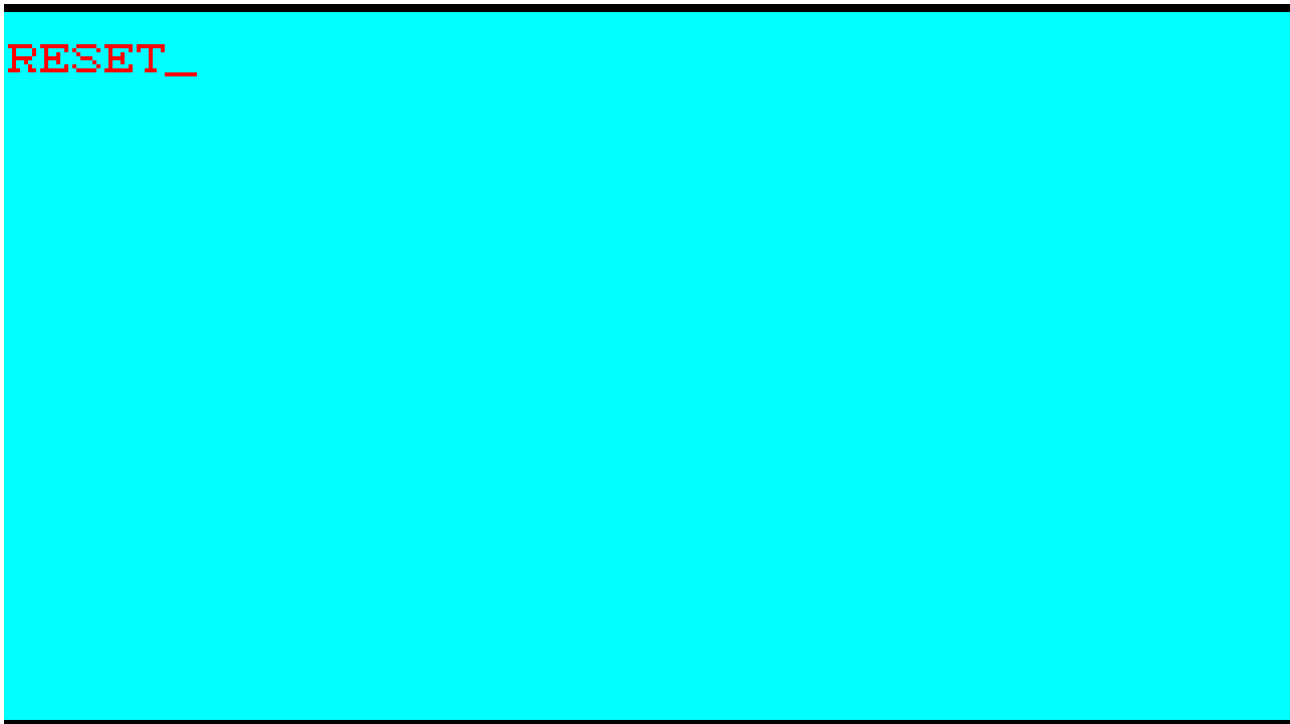
```
LIST
0 PRINT 'HELLO'
SAVE MYHELLO
MYHELLO SAVED. 1 LINES.
```

To load a program, type LOAD and the filename.

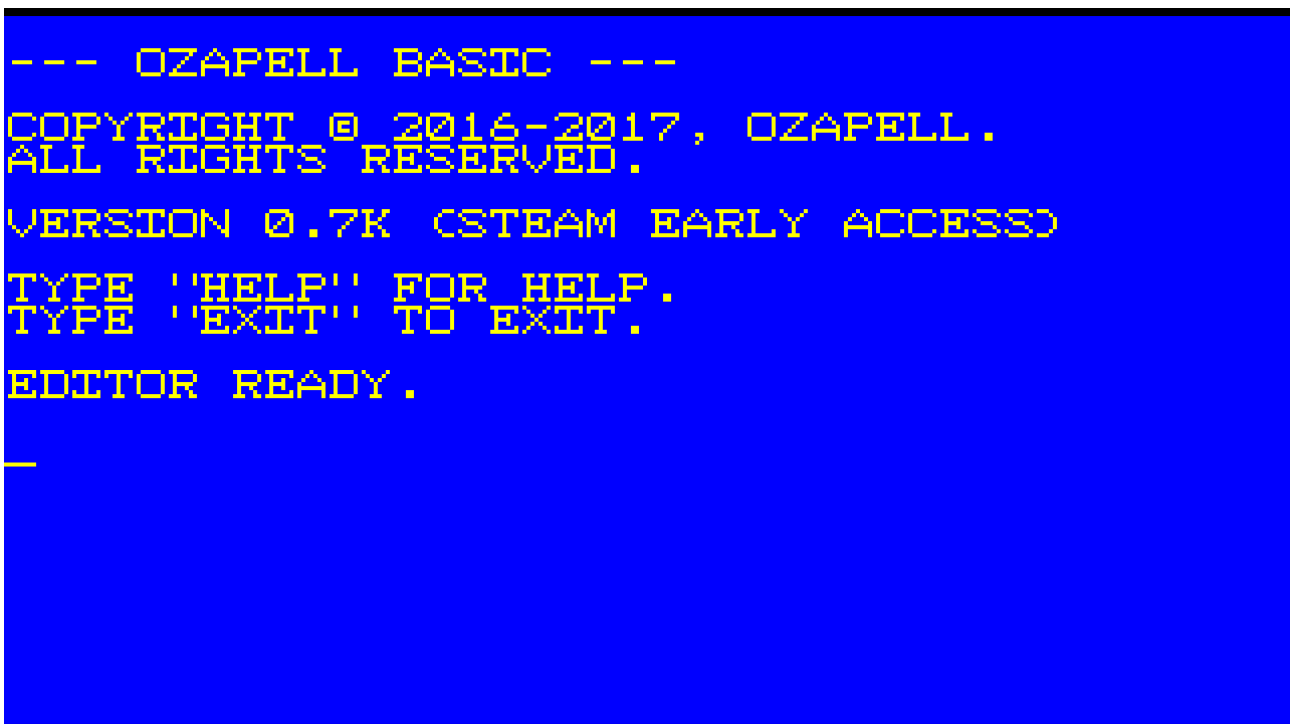
```
LIST
0 PRINT 'HELLO'
SAVE MYHELLO
MYHELLO SAVED. 1 LINES.
LOAD MYHELLO
MYHELLO LOADED. 1 LINES.
```

Type RUN and press [ENTER] to execute your program.

If the colors on your editing screen are unpleasant, you can type RESET and press [ENTER] to reset the interface to the default colors. RESET also resets the font to the default.



And after RESET.



When programming in Ozapell Basic, comments can be added to the code with the REM keyword. REM is the keyword for remark (comment). A line containing a comment is ignored by the compiler. It is used to provide additional information to the programmer or when sharing the source code with others.

```
--- OZAPPELL BASIC ---
COPYRIGHT © 2016-2017, OZAPPELL.
ALL RIGHTS RESERVED.

VERSION 0.7K (STEAM EARLY ACCESS)

TYPE 'HELP' FOR HELP.
TYPE 'EXIT' TO EXIT.

EDITOR READY.

0 REM PRINT DISPLAYS TEXT
1 PRINT "HELLO"
_
```

Here is a list of some keyboard shortcuts used throughout Ozapell Basic:

[ESC] = Quit listing or exit a running program
[SPACEBAR] = Pause or unpause a listing of a program
[PrtScr] = Copy the current screen to the clipboard
[Alt]+[Tab] or [Ctrl]+[Esc] or Windows Key = Minimize Ozapell Basic and return to desktop

While in fullscreen editing mode (type "EDIT"), these keys are used:

[ESC] = Exit fullscreen editing mode
Up Arrow = Move up in the program
[PGUP] = Move up a page in the program
Down Arrow = Move down in the program
[PGDN] = Move down a page in the program
Left Arrow = Move left in the program
Right Arrow = Move right in the program if there is text
[Home] = Move cursor to the beginning of the line
[End] = Move cursor to the end of the line
[DELETE] = Delete a character or line ahead of the cursor
[BACKSPACE] = Delete the previous character
[ENTER] = Add a new line

Chapter 2. Variables

A variable is a symbol representing a quantity.

In computer programming, a variable represents a location in memory where a value is stored.

In Ozapell Basic, there are four simple types of variables. The type of the variable is determined by the data assigned to it.

Integers:

X=10

X is a variable assigned the value of 10. X is of the integer type because it is a whole number (doesn't contain a decimal point). 0 is also an integer. Negative whole numbers such as -1 or -50 are also integers.

```
LIST
0 X=10
1 PRINTLINE X
2 X=0
3 PRINTLINE X
4 X=-1
5 PRINTLINE X
6 X=-50
7 PRINTLINE X
```

```
RUN
AUTOSAVE SAVED. 8 LINES.
COMPILING...
LINKING...
RUNNING...

10
0
-1
-50

EDITOR READY.
```

Floating point numbers:

Y=10.5

Y is a variable assigned the value of 10.5. 10.5 is a floating point number because it contains a decimal point. 0.0 is also a floating point number as long as the decimal point is included. Negative floating point numbers contain a decimal point such as in -1.5 or -50.33.

```
LIST
0 X=10.5
1 PRINTLINE X
2 X=0.0
3 PRINTLINE X
4 X=-1.5
5 PRINTLINE X
6 X=-50.33
7 PRINTLINE X
|
```

```
RUN
AUTOSAVE SAVED. 8 LINES.
COMPILING...
LINKING...
RUNNING...
10.5
0.0
-1.5
-50.33
EDITOR READY.
|
```

Strings:

```
Z="HELLO"
```

Z is a variable assigned the text string HELLO. (Note: The quotation marks designate the begin and end of the string. These quotation marks are not included in the final string.)

```
HELLO="HELLO"  
WORLD="WORLD"  
PRINT HELLO  
PRINT  
PRINT WORLD  
PRINTLINE "!"  
RUN
```

AUTOSAVE SAVED. 6 LINES.

COMPILING...
LINKING...
RUNNING...

HELLO WORLD!

EDITOR READY.

—

Finally, the fourth simple type in Ozapell Basic is a boolean. Boolean variables contain either TRUE or FALSE. They are used by logic statements.

Booleans:

B=TRUE

B is a variable assigned the boolean value of TRUE.

```
A=TRUE  
B=FALSE  
PRINTLINE A  
PRINTLINE B  
RUN
```

AUTOSAVE SAVED. 4 LINES.

COMPILING...
LINKING...
RUNNING...

TRUE
FALSE

EDITOR READY.

—

TEXTCOLOR and BACKCOLOR are pre-defined variables used by Ozapell Basic. TEXTCOLOR

specifies the foreground color of text in text mode and BACKCOLOR specifies the background color of text. In graphics mode, TEXTCOLOR is used for all drawing commands while BACKCOLOR specifies the background color of the entire screen when CLS is used.

Here is a list of pre-defined colors in Ozapell Basic:

COLORBLACK
COLORWHITE
COLORRED
COLORYELLOW
COLORGREEN
COLORBLUE
COLORVIOLET
COLORPURPLE
COLORORANGE
COLORBROWN
COLORINDIGO
COLORCYAN
COLORGREY
COLORPINK
COLORLIGHTGREY
COLORDARKGREY
COLORPASTELBLUE
COLORPASTELGREEN
COLORPASTELYELLOW
COLORPASTELRED

Chapter 3. Subroutines

Subroutines are sections of code that can be executed once or many times. After calling the subroutine and it completes execution, the program returns to the same location where it had called the subroutine and continues program execution.

The included subroutines provide access to input, graphics, sound, text file support and more.

To call subroutine, simply put its name on a line of source code.

```
LIST
0 RANGE=100
1 RANDOM
2 PRINTLINE RESULT

RUN

AUTOSAVE SAVED. 3 LINES.

COMPILING...
LINKING...
RUNNING...

59

EDITOR READY.
```

The subroutine RANDOM uses the integer value in RANGE to return a random number in RESULT. If RANGE is 100, for example, RESULT will contain a random value between 0 and 99.

You'll notice the subroutine call to PRINTLINE includes a parameter. PRINT, PRINTLINE, PRINTXY and PRINTLINEWRAP provide a shortcut. They use the variable TEXT, but when the parameter is given, the content of that parameter will be copied to TEXT.

```
LIST
0 RANGE=100
1 RANDOM
2 TEXT=RESULT
3 PRINTLINE

RUN

AUTOSAVE SAVED. 4 LINES.

COMPILING...
LINKING...
RUNNING...

30

EDITOR READY.
```

```
TEXT=RESULT
PRINTLINE
```

Works the same as:

```
PRINTLINE RESULT
```

This does not apply to other subroutines.

All subroutines use zero or more variables, process the data and then return zero or more results in variables. Many of the included subroutines return data in the RESULT variable.

Users can also define their own subroutines. Use subroutines to divide source code up into more logical, organized parts. Subroutines can also be called many times, making it easier to handle the same task without repeating the same source code throughout the program.

To define a subroutine, use the keyword SUB and then the name of the subroutine. Subroutines can only contain letters in their names.

```
LIST
0 PRINTLINE 'HELLO'
1 MYSUBROUTINE
2 SUB MYSUBROUTINE
3 PRINTLINE 'GOODBYE'
RUN
AUTOSAVE SAVED. 4 LINES.
COMPILING...
LINKING...
RUNNING...
HELLO
GOODBYE
EDITOR READY.
```

In Ozapell Basic, all subroutines are defined after the main section of code. The main section is the first lines of code up until the first subroutine definition.

Chapter 4. FOR ... NEXT Loop

The FOR NEXT loop is a simple way to loop through one or more instructions for a specified number of times.

Example:

```
FOR X=1 to 6
PRINTLINE X
NEXT X
```

In this example, the numbers 1 through 6 will be displayed when the program is run. The variable X is first assigned the value of 1. When the program reaches NEXT X, the value in X is increased to 2


```

0 FOR X=6 TO 1
1 PRINTLINE X
2 NEXT X
RUN

AUTOSAVE SAVED. 3 LINES.

COMPILING...
LINKING...
RUNNING...

0004000
0040040
EDITOR READY.
|

```

The values used in a FOR statement must be integers (whole numbers) including 0 and negative numbers.

Example:

```

FOR Z=-3 TO 3
PRINTLINE Z
NEXT Z

```

When the program is run, the numbers -3 through 3 will be displayed including the 0 in the middle.

```

0 FOR Z=-3 TO 3
1 PRINTLINE Z
2 NEXT Z
RUN

AUTOSAVE SAVED. 3 LINES.

COMPILING...
LINKING...
RUNNING...

0004000
0040040
0080080
0120120
0160160
EDITOR READY.
|

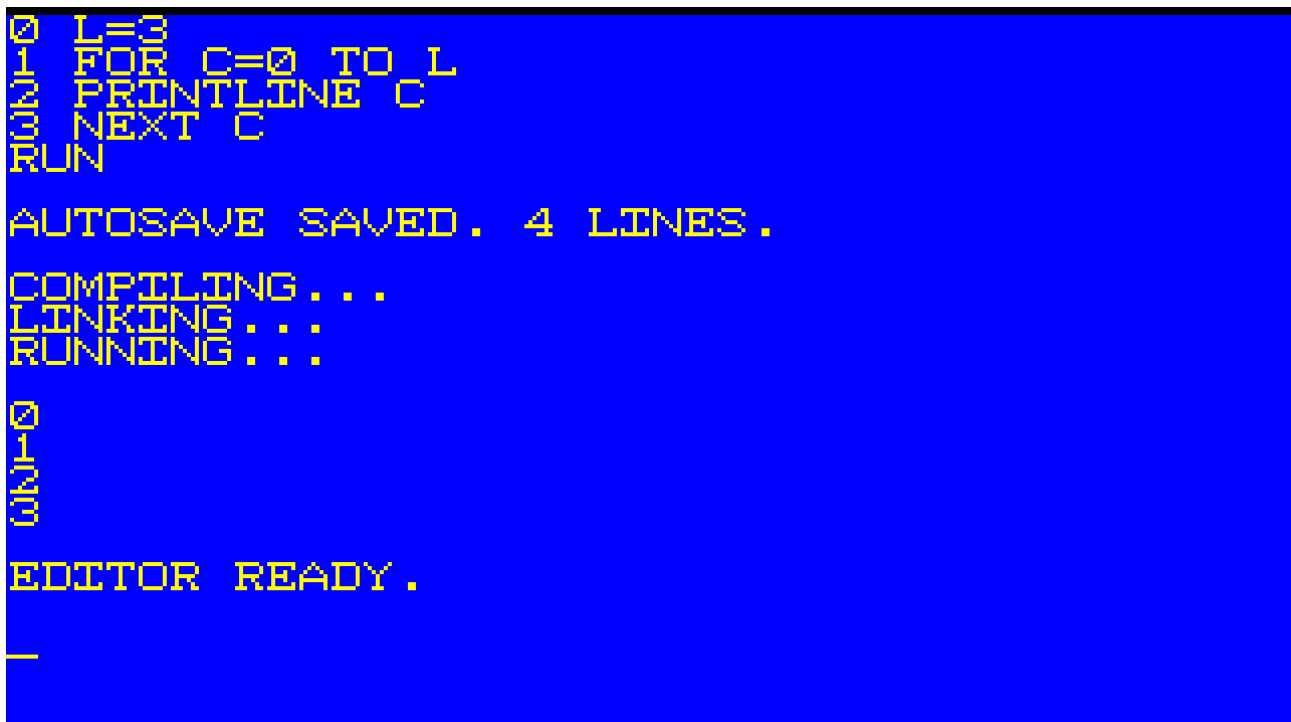
```

Finally, it should be noted that the values provided in the FOR statement can also be variables when the range needed is not known until run-time. This would be useful when looping through a line of text where the length of the text isn't known to the programmer.

Example:

```
L=3
FOR C=0 TO L
PRINTLINE C
NEXT C
```

In this example, the variable C will first be assigned the value of 0 and then loop until C reaches the value contained in the variable L.



```
0 L=3
1 FOR C=0 TO L
2 PRINTLINE C
3 NEXT C
4
5 RUN
6
7 AUTOSAVE SAVED. 4 LINES.
8
9 COMPILING...
10 LINKING...
11 RUNNING...
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
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

```



```
0 X=10
1 Y=X
2 Y+1
3 PRINTLINE RESULT
RUN

AUTOSAVE SAVED. 4 LINES.

COMPILING...
LINKING...
RUNNING...

11
EDITOR READY.

_
```

Logic would include any instruction that makes a decision based on data provided.

Input would include instructions that collect data from various sources such as a key press from the keyboard (INPUT) or a text file from a hard drive.

Output would include instructions that display text (PRINT) or graphics on the screen or write a text file to a hard drive.

Example:

```
REPEAT
INPUT
UNTIL RESULT
PRINTLINE RESULT
```

In this example program, the REPEAT and UNTIL will loop while waiting for a key press. While no key is pressed, the variable RESULT will be an empty string. An empty string in Ozapell Basic equates as FALSE. Any key press will equate as TRUE. UNTIL will jump back to its REPEAT until RESULT equals TRUE.

```
0 REPEAT
1 INPUT
2 UNTIL RESULT
3 PRINTLINE RESULT
RUN
AUTOSAVE SAVED. 4 LINES.
COMPILING...
LINKING...
RUNNING...
X
EDITOR READY.
_
```

Chapter 6. IF ... ENDIF and IFNOT ... ENDIFNOT

The simplest way to control what instruction execute or not is to use an IF or IFNOT statement.

IF checks a value or variable for TRUE. If TRUE, the following code will execute. If FALSE, the following code will be skipped until the ENDIF.

Boolean variables are checked for TRUE or FALSE.

Integers and floating point numbers are treated as FALSE if the value is 0 or 0.0. The value is TRUE otherwise.

Empty strings ("") are considered FALSE. A string containing one or more characters are considered TRUE.

```

0  A=TRUE
1  IF A
2  PRINTLINE 'A IS TRUE'
3  ENDIF
4  A=20
5  IF A
6  PRINTLINE 'A IS TRUE'
7  ENDIF
8  RUN
9
10 AUTOSAVE SAVED. 8 LINES.
11
12 COMPILING...
13 LINKING...
14 RUNNING...
15
16 A IS TRUE
17 A IS TRUE
18
19 EDITOR READY.

```

IFNOT is similar to IF except it checks if a value or variable is FALSE. If the variable or value is considered FALSE, the following code will execute. If the value or variable is TRUE, then the following code will be skipped until the ENDIFNOT.

```

0  IFNOT FALSE
1  PRINTLINE 'FALSE IS FALSE'
2  ENDIFNOT
3  B=""
4  IFNOT B
5  PRINTLINE 'B IS EMPTY STRING'
6  ENDIFNOT
7
8  RUN
9
10 AUTOSAVE SAVED. 7 LINES.
11
12 COMPILING...
13 LINKING...
14 RUNNING...
15
16 FALSE IS FALSE
17 B IS EMPTY STRING
18
19 EDITOR READY.

```

IF ... ENDIF and IFNOT ... ENDIFNOT statements can be nested.

```
0 X=TRUE
1 IF X
2 Y=TRUE
3 IF Y
4 PRINTLINE 'X AND Y ARE TRUE'
5 ENDEF
6 ENDEF
7 RUN

AUTOSAVE SAVED. 7 LINES.

COMPILING...
LINKING...
RUNNING...

X AND Y ARE TRUE

EDITOR READY.
```

Chapter 7. WHILE ... ENDWHILE and REPEAT ... UNTIL

In addition to the FOR NEXT statement(s) discussed previously, there are two more logical flow controlling statements.

The first is WHILE ENDWHILE. If the variable or data provided is TRUE, the code will execute. If it is FALSE, then WHILE will jump to the ENDWHILE and the loop will end.

Example:

```
VALID=TRUE
WHILE VALID
PRINTLINE VALID
VALID=FALSE
ENDWHILE
```

```
LIST
0  VALID=TRUE
1  WHILE VALID
2  PRINTLINE VALID
3  VALID=FALSE
4  ENDWHILE

RUN

AUTOSAVE SAVED. 5 LINES.

COMPILING...
LINKING...
RUNNING...

TRUE

EDITOR READY.
```

In this example, the loop will execute only once. If it had been WHILE TRUE, there would have been an infinite loop which could only be stopped by pressing [ESC] on the keyboard.

The second logical flow control statement(s) is REPEAT UNTIL. Unlike WHILE ENDWHILE which may never execute, REPEAT UNTIL will always execute its loop at least once. When the variable or data provided at UNTIL is TRUE, the loop will end.

Second example:

```
DONE=TRUE
REPEAT
PRINTLINE DONE
UNTIL DONE
```

```
0 DONE=TRUE
1 REPEAT
2 PRINTLINE DONE
3 UNTIL DONE
RUN

AUTOSAVE SAVED. 4 LINES.

COMPILING...
LINKING...
RUNNING...

TRUE

EDITOR READY.

_
```

In this example, the loop will execute only once. If DONE had been assigned FALSE, the loop would be infinite and would require the pressing of [ESC] on the keyboard to end.

Chapter 8. Simple Math

Mathematical operations in Ozapell Basic are done using subroutines from the built-in library. Insead of the usual operators, math operations are done with subroutine calls.

Example:

```
VARONE=10
VARTWO=20
ADD
PRINTLINE RESULT
```

```

0 VARONE=10
1 VARTWO=20
2 ADD
3 PRINTLINE RESULT
RUN
AUTOSAVE SAVED. 4 LINES.
COMPILING...
LINKING...
RUNNING...
30
EDITOR READY.
_

```

The subroutines SUBTRACT, MULTIPLY, DIVIDE, EQUALTO, NOTEQUALTO, LESSER and GREATER work in the same way.

EQUALTO, NOTEQUALTO, LESSER and GREATER return TRUE or FALSE in RESULT.

```

0 VARONE=500
1 VARTWO=0
2 EQUALTO
3 PRINTLINE RESULT
4 GREATER
5 PRINTLINE RESULT
RUN
AUTOSAVE SAVED. 6 LINES.
COMPILING...
LINKING...
RUNNING...
FALSE
TRUE
EDITOR READY.
_

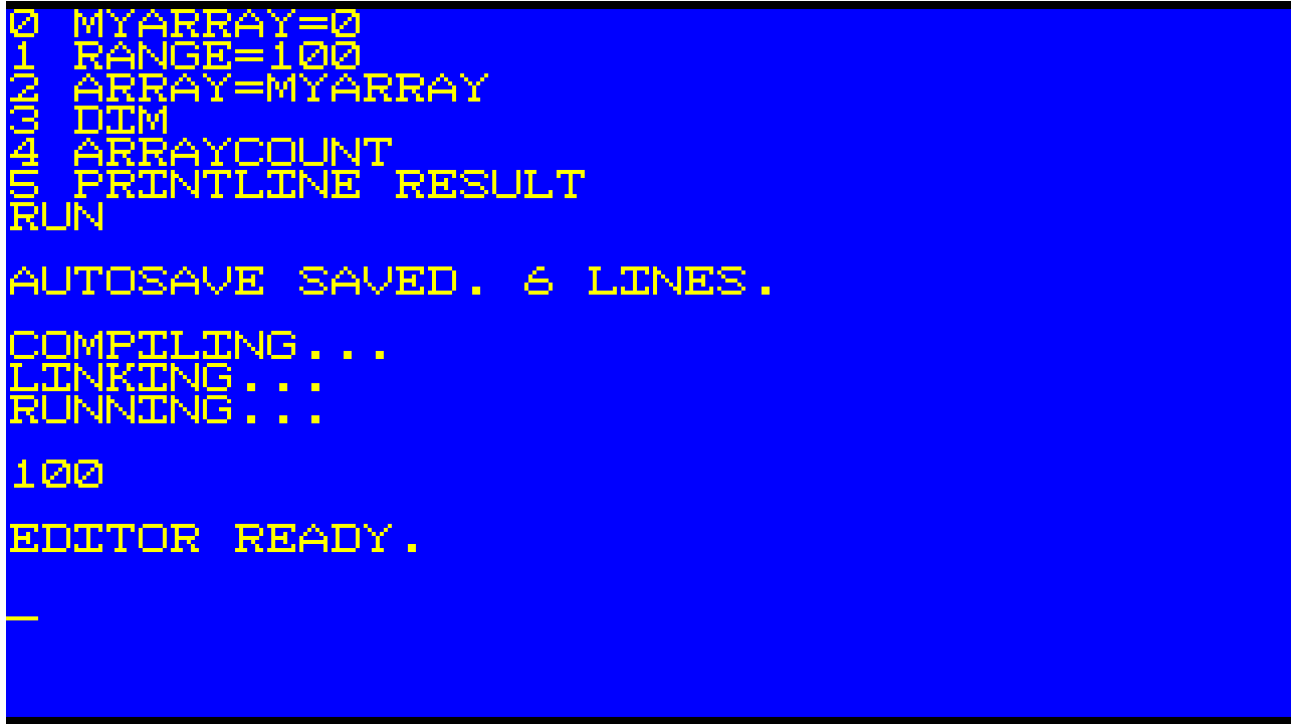
```

Chapter 9. Arrays

Arrays are similar to normal variables except they contain one or more entries. Each entry is treated like a variable.

Here is an example of creating an array in Ozapell Basic:

```
MYARRAY=0
RANGE=100
ARRAY=MYARRAY
DIM
ARRAYCOUNT
PRINTLINE RESULT
```



```
0 MYARRAY=0
1 RANGE=100
2 ARRAY=MYARRAY
3 DIM
4 ARRAYCOUNT
5 PRINTLINE RESULT
RUN
AUTOSAVE SAVED. 6 LINES.
COMPILING...
LINKING...
RUNNING...
100
EDITOR READY.
```

The variable MYARRAY is declared and given the value of 0. RANGE will specify how many entries will be in the array. The ARRAY variable is given MYARRAY. DIM changes the variable MYARRAY into an array containing 100 entries with the value of 0 in each. ARRAYCOUNT returns the number of entries in ARRAY as RESULT. PRINTLINE RESULT displays the number of entries.

If MYARRAY had been declared with a different initial value, then every entry would contain that value instead.

Here's an example of an array working with strings:

```
MYSTRINGARRAY="HELLO WORLD"
ARRAY=MYSTRINGARRAY
RANGE=20
DIM
```



```
0 MYSTRINGARRAY='HELLO WORLD'  
1 ARRAY=MYSTRINGARRAY  
2 RANGE=20  
3 DIM  
4 PRINTLINE MYSTRINGARRAY  
RUN  
  
AUTOSAVE SAVED. 5 LINES.  
  
COMPILING...  
LINKING...  
RUNNING...  
  
HELLO WORLD  
  
EDITOR READY.  
  
—
```

This code will result in MYSTRINGARRAY containing 20 entries with the string “HELLO WORLD” in them. It then uses PRINTLINE to display the first entry.

Here is an example of an array of colors:

```
COLORS=COLORBLACK  
RANGE=3  
ARRAY=COLORS  
DIM  
COLORS=COLORGREEN  
COLORS=COLORBLUE  
COLORS=COLORRED  
TEXTCOLOR=COLORS  
PRINTLINE "FIRST COLOR ENTRY IS GREEN"
```

```

0  COLORS=COLORBLACK
1  RANGE=3
2  ARRAY=COLORS
3  DIM
4  COLORS=COLORGREEN
5  COLORS=COLORBLUE
6  COLORS=COLORRED
7  TEXTCOLOR=COLORS
8  PRINTLINE "FIRST ENTRY IS GREEN"
9  RUN
AUTOSAVE SAVED. 9 LINES.
COMPILING...
LINKING...
RUNNING...
FIRST ENTRY IS GREEN
EDITOR READY.

```

In this example, an array called COLORS is created with 3 entries containing the color black. Then each entry is initialized with a different color. TEXTCOLOR is set to the first color in COLORS (green) and PRINTLINE displays the text.

Arrays in Ozapell Basic will automatically advance to the next entry when the array is read or new data is written to it. If the array reaches the end, it will automatically circle back to the first entry. The array index can also be set manually. This would be useful for random access to the array's data.

Assuming the previous array COLORS has been declared...

```

ARRAY=COLORS
INDEX=2
ARRAYINDEX
TEXTCOLOR=COLORS
PRINTLINE "INDEX 2 IS RED"

```

```

07 TEXTCOLOR=COLORS
08 PRINTLINE "FIRST ENTRY IS GREEN"
09 ARRAY=COLORS
10 INDEX=2
11 ARRAYINDEX
12 TEXTCOLOR=COLORS
13 PRINTLINE "INDEX 2 IS RED"

RUN

AUTOSAVE SAVED. 14 LINES.

COMPILING...
LINKING...
RUNNING...

FIRST ENTRY IS GREEN
INDEX 2 IS RED

EDITOR READY.

```

This example will change the index (next entry to be used) in the array to 0. In Ozapell Basic, all arrays are indexed starting from 0. The first entry in every array is 0, the second entry is 1, etc.

Chapter 10. INPUT, INPUTCODE and INPUTLINE

The subroutine INPUT returns a string in RESULT containing a single character if a key is pressed. Otherwise it returns an empty string (""). INPUT will only return printable characters. Use INPUTCODE for other key types such as the arrow keys.

The subroutine INPUTCODE is used to receive input from the keyboard or mouse. The key is returned in RESULT as an integer value. If no key was pressed, RESULT will contain 0.

Here is a list of a few of the codes that can be returned by INPUTCODE:

```

13 = [ENTER]
32 = [SPACEBAR]
48 to 57 = 0 to 9
65 to 90 = A to Z
97 to 122 = Shift + A to Z
200 = Left Mouse Button
201 = Right Mouse Button
202 = Middle Mouse Button
241 = [PGUP]
242 = [PGDN]
243 = [End]
244 = [Home]
245 = Left Arrow
246 = Up Arrow
247 = Right Arrow
248 = Down Arrow

```

Note: Mouse input requires the MOUSEON subroutine to be called first. This will display the

mouse cursor. When a mouse button is pressed, additional data will be found in the MOUSEX, MOUSEY and MOUSEOWNER variables. MOUSEX and MOUSEY indicate the location on the screen where the mouse button was pressed. MOUSEOWNER is used in graphics mode to return a value associated with a graphics image such as a sprite or button.

INPUTLINE is used to prompt a user for a line of text.

The variable Y specifies the text line to display the prompt. Or define Y as -1 to specify that the next line should be used.

TEXT is the starting string if needed. Otherwise, an empty string ("") can be used.

PROMPT will contain the string to display asking for text input. An empty string ("") can be used if no prompt string is needed.

INPUTLINE must be in a loop to work properly. This will update the TEXT variable as the user enters data. It will also display a flashing cursor.

Example:

```
Y=-1
PROMPT="ENTER TEXT: "
TEXT=""
REPEAT
INPUTLINE
UNTIL RESULT
```

RESULT will contain the final data entered after the user has pressed [ENTER]. Until [ENTER] is pressed, RESULT will return an empty string ("").

Chapter 11. Graphics Mode

Ozapell Basic has a text mode where the screen is divided into characters (40x22) and a graphics mode where the screen is divided in pixels (320x180).

To enter graphics mode, use the SETMODE subroutine with the variable MODE set to 1.

```
MODE=1
SETMODE
```

In graphics mode, the BACKCOLOR variable is only used to specify the background color of the screen. Set BACKCOLOR before calling SETMODE to clear the screen with that color. Also, CLS will clear the screen with the BACKCOLOR specified.

TEXTCOLOR is specifies the color of most subroutines used.

Draw a single pixel with COLORXY. For example, to draw a single white pixel in the lower right corner of the screen:

```
TEXTCOLOR=COLORWHITE
X=319
Y=179
```

COLORXY

DRAWLINE draws straight and diagonal lines. For example, to draw a diagonal line from the upper left corner to the lower right corner:

```
XONE=0  
YONE=0  
XTWO=319  
YTWO=179  
DRAWLINE
```

COLORBOX draws a rectangle using the current TEXTCOLOR. For example, to draw a rectangle that fills the entire left side of the screen:

```
XONE=0  
YONE=0  
XTWO=159  
YTWO=179  
COLORBOX
```

In graphics mode, PRINTXY is used to display text on the screen using the current TEXTCOLOR. X and Y specify the location in pixels. For example, to draw text in the upper left corner:

```
X=0  
Y=0  
PRINTXY "HELLO WORLD!"
```

Chapter 12. Delays

Sometimes you will need to add delays in your program.

Let's assume you wrote a program in Ozapell Basic to draw a white pixel on a black screen in graphics mode:

```
BACKCOLOR=COLORBLACK  
MODE=1  
SETMODE  
TEXTCOLOR=COLORWHITE  
X=10  
Y=10  
COLORXY
```

```

EDITOR READY.

0 BACKCOLOR=COLORBLACK
1 MODE=1
2 SETMODE
3 TEXTCOLOR=COLORWHITE
4 X=10
5 Y=10
6 COLORXY
7 RUN

AUTOSAVE SAVED. 7 LINES.

COMPILING...
LINKING...
RUNNING...

```

```

EDITOR READY.

```

You run the program and nothing appears to happen. The program executed properly and drew its pixel, but then it immediately returned to text mode when the program ended.

This could also be a problem if a wave audio sound was loaded, played and the program ended. The sound would stop immediately and it might seem as though the sound wasn't playing correctly.

Here are some delays you can add to the end of your program:

```

REPEAT
UNTIL FALSE

```

```

BACKCOLOR=COLORBLACK
MODE=1
SETMODE
TEXTCOLOR=COLORWHITE
X=10
Y=10
COLORXY

REM INFINITE LOOP UNTIL [ESC] PRESSED
REPEAT
UNTIL FALSE

```

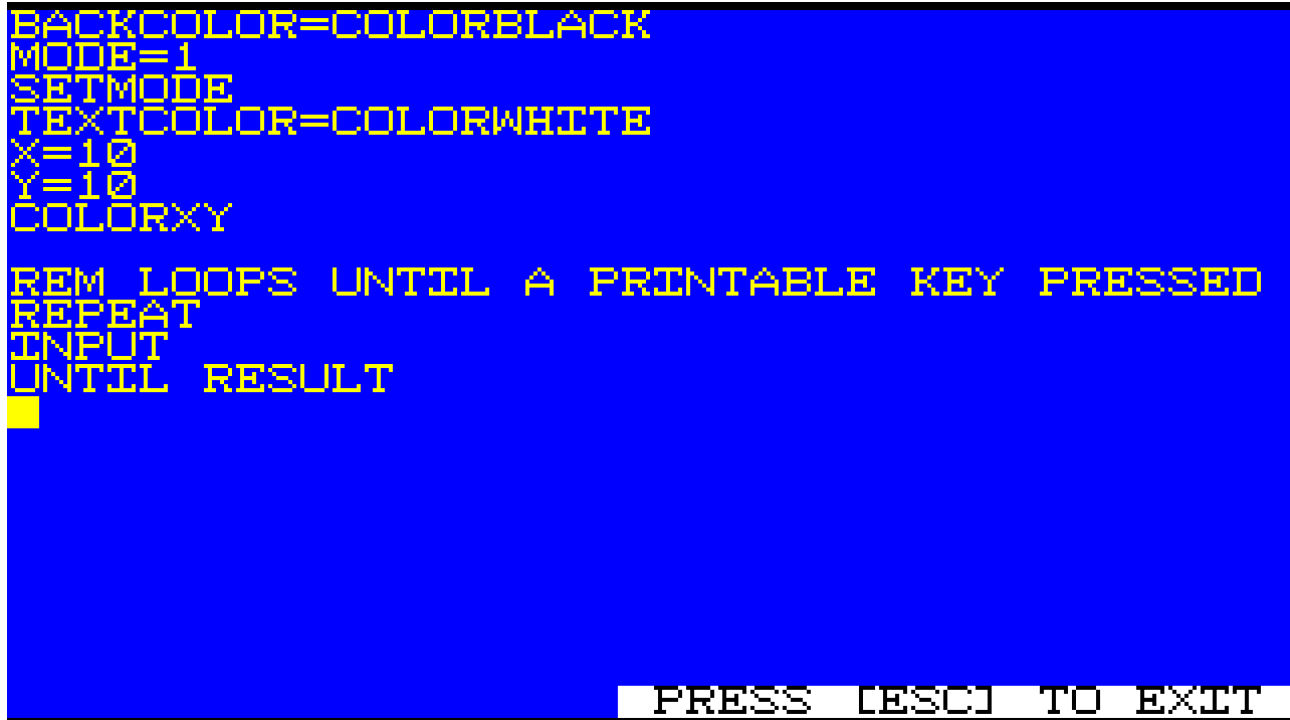
```

PRESS [ESC] TO EXIT

```

This will create an infinite loop. Only pressing the ESC key will end the program.

```
REPEAT
INPUT
UNTIL RESULT
```



```
BACKCOLOR=COLORBLACK
MODE=1
SETMODE
TEXTCOLOR=COLORWHITE
X=10
Y=10
COLORXY

REM LOOPS UNTIL A PRINTABLE KEY PRESSED
REPEAT
INPUT
UNTIL RESULT

```

PRESS [ESC] TO EXIT

This will loop until a key is pressed. INPUT will return a printable string character in RESULT. If it is an empty string (length of zero), the loop will continue. If a key is pressed (associated with a string character), then the loop will end.

```
TIME
OLDTIME=RESULT
REPEAT
TIME
RESULT-OLDTIME
RESULT>1000
UNTIL RESULT
```

```

0 TIME
1 OLDTIME=RESULT
2 REPEAT
3 TIME
4 RESULT-OLDTIME
5 RESULT>1000
6 UNTIL RESULT
7 RUN

AUTOSAVE SAVED. 7 LINES.

COMPILING...
LINKING...
RUNNING...

EDITOR READY.

```

This will loop for 1000 milliseconds (one second).

Chapter 13. Images

A bitmap image file can be loaded and displayed in graphics mode.

Here is an example of loading a bitmap file called DEMO.BMP. Note: all data files must be copied into the Ozapell Basic folder found in the Documents folder to be accessible by Ozapell Basic. When providing the filename, only the name is used. The path and the extension (.bmp) are assumed.

```

FILENAME="DEMO"
MYBITMAP=0
ARRAY=MYBITMAP
BITMAPLOAD
MYBITMAPWIDTH=WIDTH

```

In this example, DEMO.BMP is loaded into the array called MYBITMAP. RESULT will contain the number of pixels in the image which is also the number of entries in the array. WIDTH will contain the width of the image loaded. This value is used to display the image later.

Note: The only image format supported by BITMAPLOAD is an uncompressed, 24-bit color Windows BMP image.

To draw the image on the screen while in graphics mode, ARRAYDRAW or one of its variations are used:

```

ARRAY=MYBITMAP
WIDTH=MYBITMAPWIDTH
X=0
Y=0
ARRAYDRAW

```


In this example, the image contained in the array MYBITMAP is displayed in the upper-leftmost corner of the screen.

By default, all display subroutines draw on the screen immediately which is slower. For faster graphics or to compose complex scenes before displaying the end result, use the offscreen buffer. SCREENHIDE stops displays to the screen and, instead, displays all output to the offscreen buffer. SCREENSHOW displays the offscreen buffer to the screen and returns to the default of displaying immediately to the screen.

Bitmap images do not contain transparency information. To add transparency, it is easy to replace all instances of a color to make them transparent. For example, if a bitmap uses green as the background (transparency) color, then ARRAYREPLACE can be used to replace all instances of green to transparent:

```
ARRAY=MYBITMAP  
OLDVALUE=COLORGREEN  
NEWVALUE=-1  
ARRAYREPLACE
```

ARRAYDRAWOWNER works like ARRAYDRAW except the variable OWNER is also used. All non-transparent pixels associated with the image are given the owner value in OWNER. This is used in sprite collision detection as well as returned by mouse input.

Chapter 14. Sprite Collision

Sprite Collision Detection is easy with Ozapell Basic.

Switch to graphics mode:

```
MODE=1  
SETMODE
```

Assuming a bitmap image file called BITMAP.BMP is in the Documents -> Ozapell Basic folder:

```
BITMAP=0  
ARRAY=BITMAP  
FILENAME="BITMAP"  
BITMAPLOAD  
BITMAPWIDTH=WIDTH
```

The array BITMAP now contains all of the pixels for the image loaded. RESULT will return the number of entries (pixels) in the array. WIDTH will contain the width in pixels for the image.

```
ARRAY=BITMAP  
WIDTH=BITMAPWIDTH  
X=0  
Y=0  
OWNER=100  
ARRAYDRAWOWNER
```

Draws the image in BITMAP at 0,0 plus every pixel drawn is associated with the value in OWNER. (Pixels given the a negative value are transparent and will not be associated with an owner value.)

```
ARRAY=BITMAP  
WIDTH=BITMAPWIDTH  
OWNER=100  
X=0  
Y=0  
ARRAYDRAWCOLLIDE
```

RESULT will contain the highest owner value associated with any pixel that would be drawn over (collided) if ARRAYDRAW or ARRAYDRAWOWNER was used instead of ARRAYDRAWCOLLIDE. This value can be used to determine collisions such as walls, enemies and bonuses. OWNER provided with ARRAYDRAWCOLLIDE is necessary to prevent the sprite from colliding with itself. That is, use the same OWNER value used to draw the sprite you're checking for collision against other sprites.

It is possible to define a section of the screen with an owner using OWNERBOX.

```
XONE=0  
YONE=0  
XTWO=159  
YTWO=179  
OWNER=300  
OWNERBOX
```

In this example, the left half of the screen is given the owner value of 300.

There's a quick method to remove all instances of pixels associated with an owner:

```
OWNER=100  
OWNERCLEAR
```

In this example. all pixels (and owner data) with the owner value given in OWNER will be cleared from the screen. The pixels will be replaced with the current BACKCOLOR.

Chapter 15. Text Files

TEXTFILESAVE will create a text file in the Ozapell Basic folder in Documents. It saves each entry of an array as a line in the text file. It doesn't matter what type of data is in each entry of the array as it will be converted into a string.

The variable ARRAY needs to contain an array with one or more entries. FILENAME will contain a valid filename for the new text file. Paths and extensions are assumed and should not be used.

Example:

```
MYARRAY="HELLO"  
ARRAY=MYARRAY  
RANGE=32  
DIM
```

```
ARRAY=MYARRAY  
FILENAME="HELLO"  
TEXTFILESAVE
```

This example will create a file named "HELLO.TXT" and save 32 lines containing the word "HELLO" on each line.

RESULT will contain the number of lines saved.

TEXTFILELOAD is used to load a text file. FILENAME will contain the filename of the text file without a path or extension. ARRAY must specify a valid variable. The variable specified will be converted into an array with the same number of entries as lines in the text file. RESULT will return the number of lines loaded or 0 if the file doesn't exist.

Example:

```
MYDATA=0  
ARRAY=MYDATA  
FILENAME="HELLO"  
TEXTFILELOAD
```

In this example, the text file "HELLO.TXT" is loaded if it exists. MYDATA will now be an array with the number of entries equal to the number of lines in the text file. Each entry will contain the string for each line in the file.

Sometimes it is a good idea to convert the data in an array back to its native values instead of having them remain strings. This would be useful if the data must be modified by the program.

ARRAYVALUE will convert every entry in an array back to its native type. Specify the array to convert in ARRAY.

Example:

```
ARRAY=MYDATA  
ARRAYVALUE
```

In this example, each whole number integer, boolean (TRUE or FALSE) or floating point value will be converted back to its native type for use in the program.

Chapter 16. Sound

Ozapell Basic supports wave audio for sound.

Assuming there is a valid audio file called AUDIO.WAV in the Documents -> Ozapell Basic folder, then the following code will load it:

```
FILENAME="AUDIO"  
AUDIOARRAY=0  
ARRAY=AUDIOARRAY  
WAVEFILELOAD
```

Now the array AUDIOARRAY contains the audio data. This data can be analyzed or manipulated.

RESULT will contain the number of entries (samples) in the array.

Currently, the only wave audio format supported is Uncompressed PCM format, 16 bits per sample (CD quality), 44,100 samples per second (CD quality) and 1 channel (mono).

To copy the audio data from an array to a sound buffer:

```
ARRAY=AUDIOARRAY  
BUFFER=0  
WAVEFILEBUFFER
```

There are 32 sound buffers (0 to 31).

Audio data under 15 seconds in length will be considered a sound effect. 16 sound buffers will be created for these sound effects to allow for up to 16 simultaneous plays of the same sound.

Audio data over 15 seconds in length (with a current maximum of 5 minutes) will only be provided one sound buffer. Only one instance of this sound can be played from this buffer at one time. Playing the sound while it is already playing will cause it stop and start over from the beginning.

To play a sound:

```
BUFFER=0  
WAVEFILEPLAY
```

To loop a sound:

```
BUFFER=0  
WAVEFILELOOP
```

To stop all sound buffers for a specific sound:

```
BUFFER=0  
WAVEFILESTOP
```