Федеральное государственное бюджетное образовательное учреждение высшего образования

«Костромской государственный университет»

(КГУ)

Институт автоматизированных систем и технологий

Кафедра информационных систем и технологий

Пояснительная записка

по дисциплине «Организация ЭВМ и систем» на тему:

программа «SnakeVsDragon»

Выполнил: Манукянц А.Э.

Группа: 15-ИСбо-2

Проверил и утвердил: Прядкина Н.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017г.

Кострома

2017

Оглавление

[Назначение и область применения программы 3](#__RefHeading___Toc690_3078141354)

[Технические характеристики 3](#__RefHeading___Toc692_3078141354)

[Пользовательские методы 3](#__RefHeading___Toc694_3078141354)

[Описание и методы организации данных 3](#__RefHeading___Toc696_3078141354)

[Описание и обоснование выбора технических средств 4](#__RefHeading___Toc698_3078141354)

[Алгоритм выполнения программы 4](#__RefHeading___Toc700_3078141354)

[Тестирование 6](#__RefHeading___Toc702_3078141354)

Приложение1 11

# Назначение и область применения программы

Программа «SnakeVsDragon» является игрой типа «змейка», рассчитанной на двух игроков. Один игрок управляет змеёй, другой драконом. Побеждает тот игрок, который коснётся головы противника, имея более длинное тело.

Целью данной работы является ознакомление и углубленное изучение технологий программирования на языке Turbo Assembler с 16-битной архитектурой процессора.

# Технические характеристики

Для работы приложения необходим эмулятор DOSBox, эмулирующий работу с 16-битным процессором. Так же используется набор утилит из поставки программного продукта TASM для сборки проекта.

Необходимо наличие двух текстовых файлов с «экранами» для вывода:

1. level.txt
2. endfile.txt

# Пользовательские методы

Полный текст кода программы представлен в приложении.

Исходный код программы содержит запускаемую процедуру main, содержащую последовательный вызов следующих процедур:

1. open\_file – открытие файла и сохранение его дескриптора для последующих операций.
2. read\_file – чтение содержимого файла в объёме до 4095 символов. Запись содержимого в переменную raw\_content без обработки.
3. print\_raw\_content – вывод байтов области памяти raw\_content на экран.
4. read\_rule – получение данных с клавиатуры с помощью прерывания 16. Запись команд управления в память (переменные snake\_rule и dragon\_rule). Выход из программы произойдёт при нажатии клавиши “Esc”.
5. set\_eat – установка на поле еды для змейки и дракона. Еда устанавливается псевдослучайным образом. Работает счётчик шагов игроков, спустя некоторое количество, еда ставится в то место, где раньше была голова змеи или дракона. Количество шагов является константой, оно различно для каждого типа еды.
6. snake\_RUN – процедура, отвечающая за перемещение змейки.
7. dragon\_RUN – процедура, отвечающая за перемещение дракона. Аналогична предыдущей.
8. step – для перемещения змейки и дракона необходимо передвигать каждую часть их тел вперёд, на место следующей. Это реализуется данной процедурой. Происходит изменение переменных snake и dragon.
9. set\_snake – отрисовывает(помещает) змейку, хранящуюся в переменной snake, в области памяти, помеченной как raw\_content. Последняя часть тела змеи считается пустым местом. Это необходимо для отчистки полей, которые уже не являются змейкой(откуда она ушла).
10. set\_dragon – отрисовывает(помещает) дракона, хранящуюся в переменной dragon, в области памяти, помеченной, как raw\_content. Отличие от set\_snake в том, что происходит проверка, каким был последний символ дракона до отчистки. Может получиться так, что у хвоста дракона встанет змея, тогда поле не отчищается.
11. timer – для нормального восприятия игрового процесса человеком необходимы искусственные задержки. Это реализовано данной процедурой.
12. death – процедура вызывается при столкновении голов змеи и дракона, кем бы оно ни было произведено. Сравниваются параметры длинны змеи и дракона, побеждает наибольший. В случае, если длины равны, проигрывают оба.
13. end\_scrin – отрисовка в консоли финального экрана с четырьмя возможными концовками.
    1. Dragon VIN
    2. Snake VIN
    3. DOUBLE KILL
    4. END GAME

В работе данных процедур используются вспомогательные процедуры:

1. print – вызов команды вывода строки на экран.
2. exit – завершение работы программы.

# Описание и методы организации данных

Все файлы, с которыми должна работать программа должны находится в одной директории с исполняемым файлом.

# Описание и обоснование выбора технических средств

Технические средства были выбраны на основании требований изложенных в техническом задании.

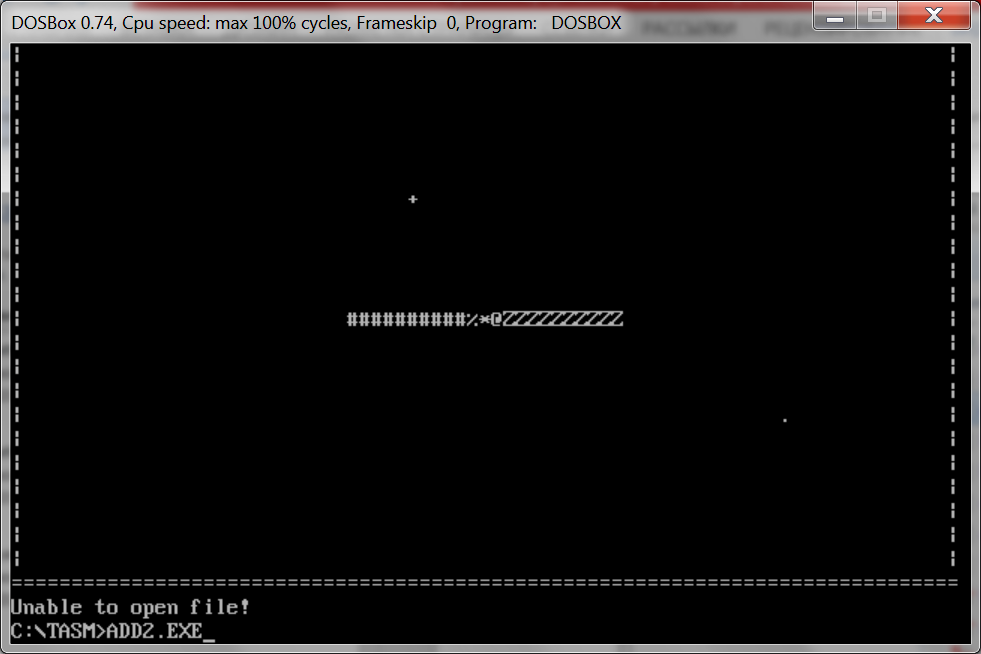
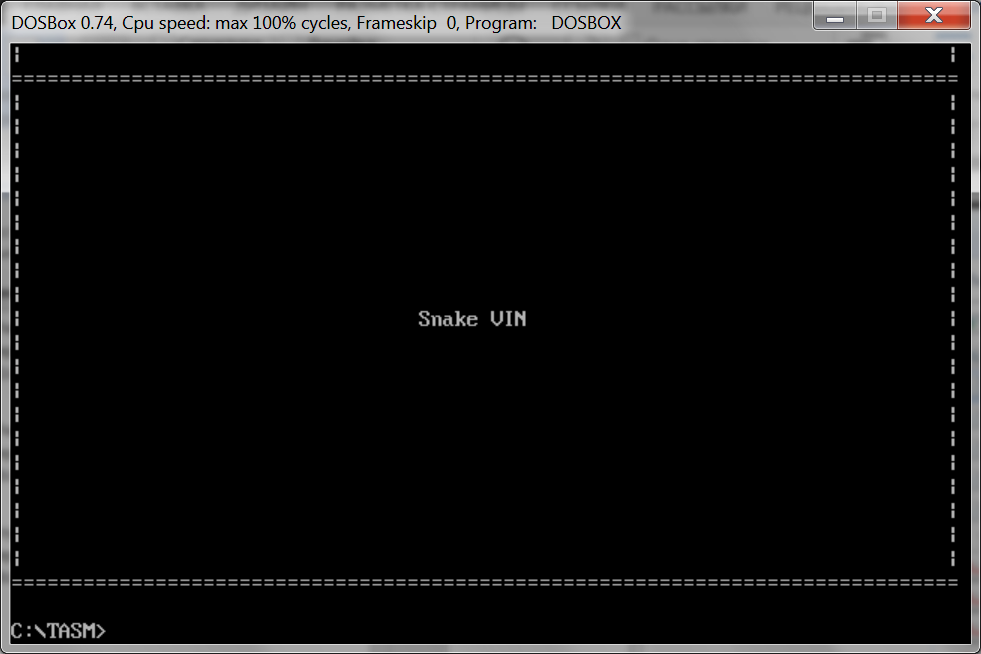
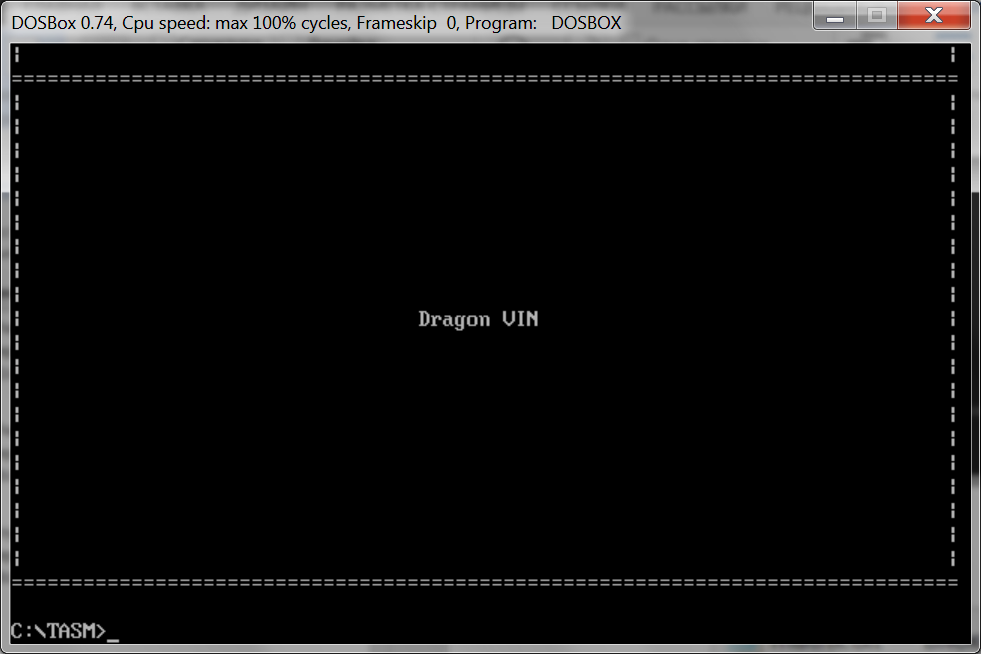
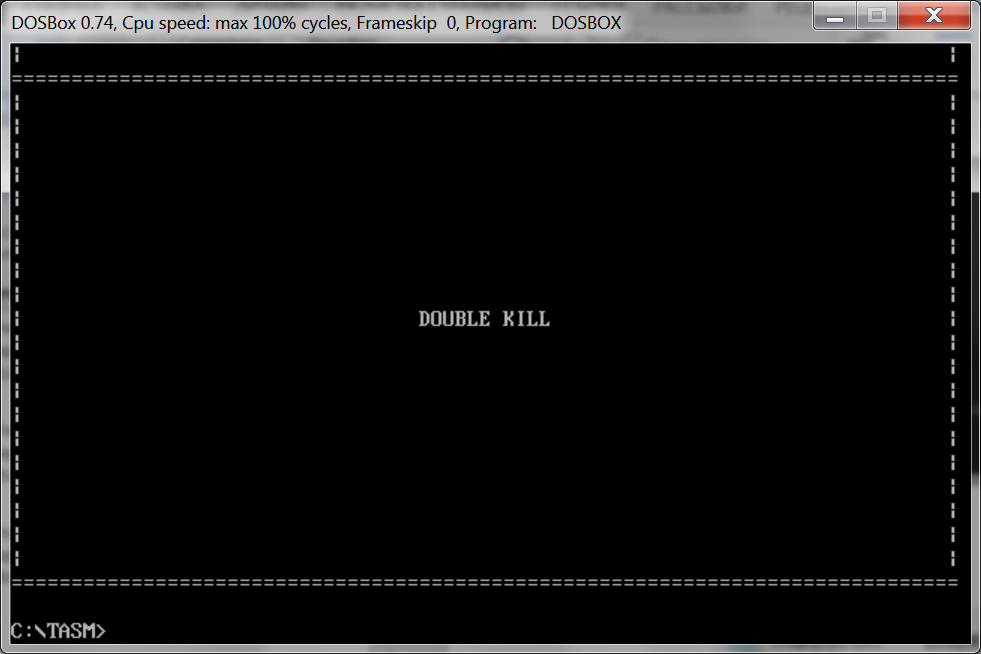
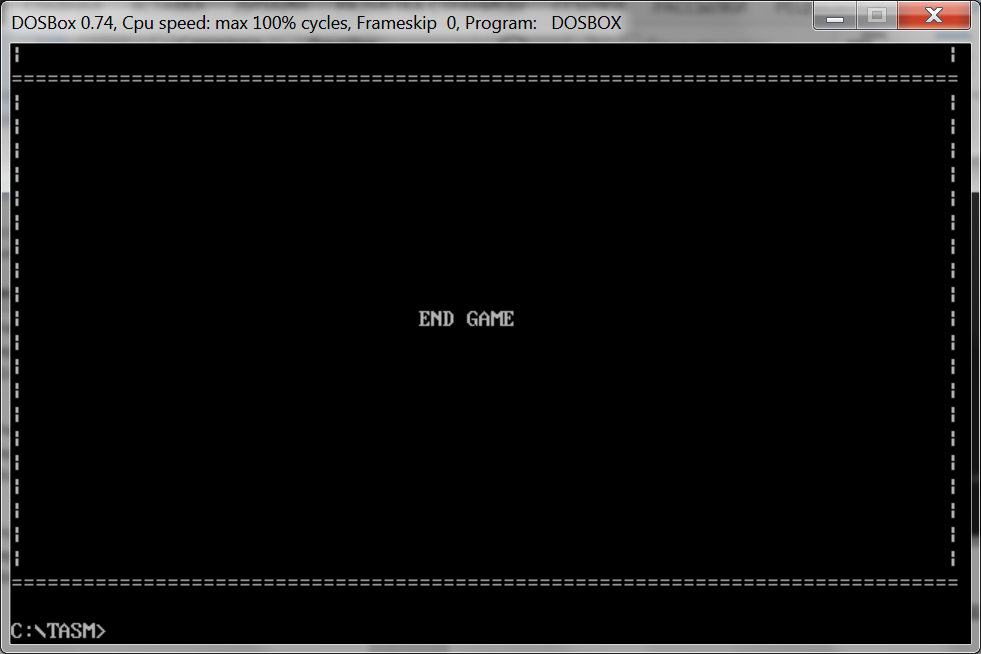
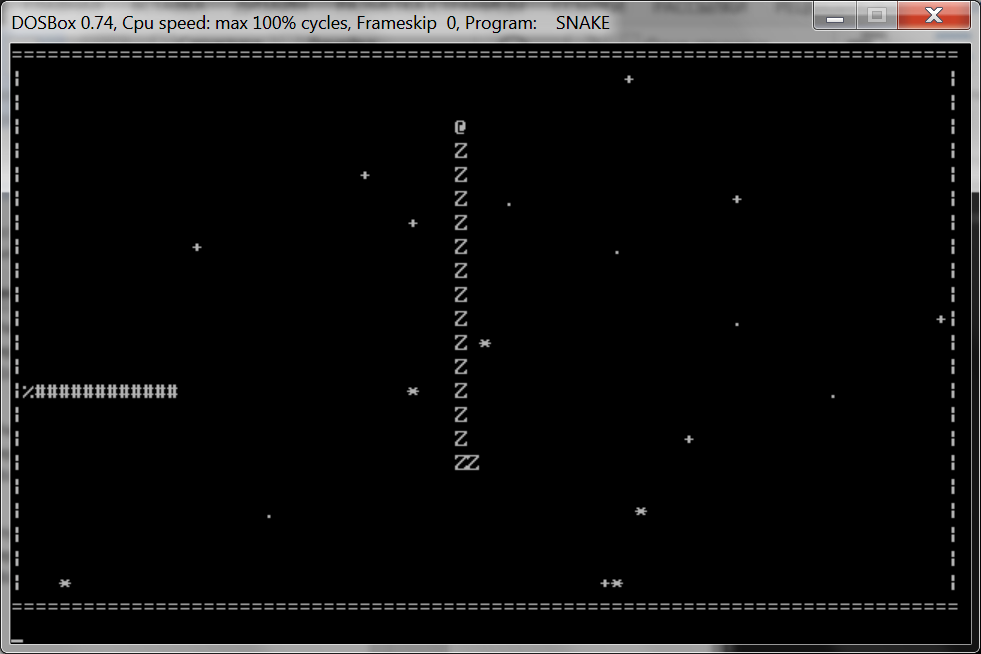
# Алгоритм выполнения программы

1. При запуске, сначала программа загружает карту. Происходит попытка открыть файл с именем из «level.txt». Обрабатывается ошибка, когда файл с данным именем не находится в директории вызова программы. После успешного открытия файла, его дескриптор записывается в переменную handle.
2. Далее из успешно открытого файла данные записываются в переменную программы raw\_content. Записываются максимум 4095 символов, иначе выводится ошибка количества символов.
3. Когда в переменную raw\_content успешно записаны данные программы, вся работа происходит с ней. Далее устанавливаются персонажи: змея и дракон. Они устанавливаются напротив друг друга с длинной тела 11, что означает 10 частиц тела и одиннадцатый пробел для отчистки хвостов. Части тел просто записываются в raw\_content.
4. Далее следует основной цикл программы run. Он отвечает за один шаг в игре. Шаг длится меньше секунды, в течение этого цикла игрок может указать следующее направление для своего персонажа. Сама игра происходит в рамках этих шагов.
5. Выход из цикла и, соответственно, игры совершается при нажатии игроком клавиши «Esc» или смерти одного из персонажей.
6. В самом цикле игры происходят следующие операции:
   1. Отображение на экран содержимого памяти по метке raw\_content
   2. Чтение новой управляющей команды от игроков. Принимаются только нажатия на клавиши «w», «d», «s», «a», «o», «;», «l», «k», «Esc». Первые 4 клавиши для управления змейкой, следующие 4 для управления драконом. Esc – выход из игры. Остальные нажатия игнорируются. Полученные управляющие символы записываются в переменные snake\_rule и dragon\_rule для последующих действий. При этом символы дракона преобразуются в соответствующие символы управления змейки. Чтение происходит посредством команд прерывания 16. Сначала происходит проверка на наличие нажатия, затем считывание. Однако, после это проверка повторяется. Таким образом берётся только последняя управляющая команда (последняя нажатая клавиша). С учётом динамичности игры, данный механизм позволяет управлять персонажами так, как этого интуитивно ожидают игроки.
   3. Далее происходит установка символов «\*», «+», «.» (еды). В течение шагов в данной процедуре идёт обратный счетчик шагов, начинающийся с 90. На большинстве из них ничего не происходит, однако на конкретных, задаваемых количествах шагов запоминается положение голов змеи и дракона. По истечении счётчика, в запомненные места устанавливается еда и счётчик обновляется на 90.
   4. Теперь остаётся разместить персонажей. Сначала устанавливается змейка. Оба персонажа, а вернее данные о их положении хранятся в переменных snake и dragon. Они хранятся в виде последовательности чисел, на которых находятся части тел персонажей. Для передвижения необходимо изменение данных этой переменной. Голова персонажа передвигается в сторону, на которую указывает управляющий символ. Затем в цикле каждая частица тела персонажа передвигается не место предыдущей. Последняя частичка отчищающая.  
      При попытке передвинуть голову персонажа происходит проверка на символ перед ним. Если это не пробел и не голова противника, то переход не совершается. Если это голова противника, вызывается процедура смерти.
   5. Процедура смерти. Сравнивает длины персонажей. Побеждает наибольший, но если они равны, то проигрывают оба. В зависимости от результата, на экран выводится соответствующая надпись.
   6. При завершении игры происходит попытка чтения файла «endfile.txt». Это шаблон экрана для завершения игры. В него вписывается фраза, подводящая итог игры. А затем происходит вывод на экран.
   7. Если же игра не завершена, происходит шаг или не происходит. Шаг завершён. Однако таким образом шаги происходят слишком быстро для человека, потому в каждом шаге отрабатывает таймер, замедляющий работу программы.

# Тестирование

1. Отсутствует файл уровня:



1. Отсутствует файл завершения игры:  
   
2. Победа змейки  
   
3. Победа дракона  
   
4. Смерть обоих  
   
5. Завершение игры нажатием клавиши «Exc»  
   
6. Игра  
     
   
7. Запутавшаяся змея  
   

# Приложение1

Turbo Assembler Version 3.0 01/23/18 11:13:05 Page 1

snake.asm

1 0000 .model small

2 0000 .stack 100h

3 0000 .data

4 ; Messages

5 0000 20 45 4E 44 20 47 41+ end\_game\_mes db ' END GAME '

6 4D 45 20 20

7 000B 20 44 72 61 67 6F 6E+ dragon\_win\_mes db ' Dragon VIN'

8 20 56 49 4E

9 0016 20 53 6E 61 6B 65 20+ snake\_win\_mes db ' Snake VIN '

10 56 49 4E 20

11 0021 20 44 4F 55 42 4C 45+ double\_kill\_mes db ' DOUBLE KILL '

12 20 4B 49 4C 4C 20

13 002E 0D 0A 24 new\_line db 13, 10, '$'

14 0031 0000 mes\_offset dw 00

15

16 ; Errors

17 0033 55 6E 61 62 6C 65 20+ open\_error\_mes db 'Unable to open file!', '$'

18 74 6F 20 6F 70 65 6E+

19 20 66 69 6C 65 21 24

20 0048 57 72 6F 6E 67 20 66+ read\_error\_mes db 'Wrong file content!', '$'

21 69 6C 65 20 63 6F 6E+

22 74 65 6E 74 21 24

23 005C 57 72 6F 6E 67 20 73+ read\_count\_error\_mes db 'Wrong symbols count in file!', '$'

24 79 6D 62 6F 6C 73 20+

25 63 6F 75 6E 74 20 69+

26 6E 20 66 69 6C 65 21+

27 24

28

29 ; Animal:

30 0079 01 rule db 1

31 007A FF\*(0024) animal dw 0ffh dup('$')

32 0278 ???? animal\_length dw ?

33

34 ; Snake:

35 027A 64 snake\_rule db 'd'

36 027B FF\*(0024) snake dw 0ffh dup('$')

37 0479 ???? snake\_length dw ?

38

39 ; Dragon

40 047B 61 dragon\_rule db 'a'

41 047C FF\*(0024) dragon dw 0ffh dup('$')

42 067A ???? dragon\_length dw ?

43

44 067C 03F3 eat\_point\_1 dw 930+81

45 ; \*

46 067E 0258 eat\_point\_2 dw 600 ; +

47 0680 0550 eat\_point\_3 dw 1360 ; .

48 0682 0003 eat\_timeout dw 3

49

50 ; File info

51 0684 6C 65 76 65 6C 2E 74+ level\_file db 'level.txt', 0

52 78 74 00

53 068E 65 6E 64 66 69 6C 65+ end\_file db 'endfile.txt', 0

54 2E 74 78 74 00

55 069A ???? handle dw ?

56 069C 0FFF\*(24) raw\_content db 0fffh dup('$')

57 169B ???? content\_length dw ?

Turbo Assembler Version 3.0 01/23/18 11:13:05 Page 2

snake.asm

58

59 169D .code

60 ; just print command

61 0000 print proc

62 0000 B4 09 mov ah,09h

63 0002 CD 21 int 21h

64 0004 C3 ret

65 0005 print endp

66

67 ; just exit command

68 0005 exit proc

69 0005 B4 4C mov ah,4Ch

70 0007 CD 21 int 21h

71 0009 exit endp

72

73 0009 end\_scrin proc

74 ; Open endfile and show endscrin

75 0009 BA 068Er mov dx, offset end\_file

76 000C E8 001D call open\_file

77 000F E8 002F call read\_file

78

79 ; Write end mess

80 0012 B9 000B mov cx,11 ; symbols count to write

81 ; setting symbols in raw\_content(writing)

82 0015 set\_symbol:

83 0015 8B F9 mov di,cx

84 0017 03 3E 0031r add di,mes\_offset

85 001B 8A 05 mov al,[di]

86 001D BF 03ED mov di,1005

87 0020 03 F9 add di,cx

88 0022 88 85 069Cr mov byte ptr raw\_content[di],al

89 0026 E2 ED loop set\_symbol

90

91 ; Show endscrin

92 0028 E8 0040 call print\_raw\_content

93

94 002B C3 ret

95 002C end\_scrin endp

96

97 002C open\_file proc

98 ; Открытие файла

99 002C B4 3D mov ah,3dh

100 002E B0 00 mov al,0h

101 0030 CD 21 int 21h

102 0032 72 04 jc open\_error

103 ; Запись дескриптора

104 0034 A3 069Ar mov handle,ax

105 0037 C3 ret

106

107 0038 open\_error:

108 0038 BA 0033r mov dx, offset open\_error\_mes

109 003B E8 FFC2 call print

110 003E E8 FFC4 call exit

111 0041 open\_file endp

112

113 0041 read\_file proc

114 ; Чтение файла

Turbo Assembler Version 3.0 01/23/18 11:13:05 Page 3

snake.asm

115 0041 B4 3F mov ah,3fh

116 0043 8B 1E 069Ar mov bx,handle

117 0047 B9 0FFF mov cx,0fffh

118 004A BA 069Cr mov dx,offset raw\_content

119 004D CD 21 int 21h

120

121 ; Обработка возможных ошибок

122 004F 72 08 jc read\_error

123 0051 3B C1 cmp ax, cx

124 0053 74 0D je read\_count\_error

125

126 ; Количество прочитанных символов

127 0055 A3 169Br mov content\_length, ax

128 0058 C3 ret

129

130 0059 read\_error:

131 0059 BA 0048r mov dx,offset read\_error\_mes

132 005C E8 FFA1 call print

133 005F E8 FFA3 call exit

134 0062 read\_count\_error:

135 0062 BA 005Cr mov dx,offset read\_count\_error\_mes

136 0065 E8 FF98 call print

137 0068 E8 FF9A call exit

138 006B read\_file endp

139

140 ; Print game in console

141 006B print\_raw\_content proc

142 006B BA 069Cr mov dx,offset raw\_content

143 006E E8 FF8F call print

144

145 0071 BA 002Er mov dx, offset new\_line

146 0074 E8 FF89 call print

147 0077 C3 ret

148 0078 print\_raw\_content endp

149

150 ; Wait some times

151 0078 timer proc

152 0078 BA 0000 mov dx,0

153 007B time\_next:

154 007B B9 00BF mov cx,0bfh

155 007E time\_in\_time:

156 007E E2 FE loop time\_in\_time

157

158 0080 42 inc dx

159 0081 81 FA 5000 cmp dx,5000h

160 0085 74 02 je time\_ret

161 0087 EB F2 jmp time\_next

162

163 0089 time\_ret:

164 0089 C3 ret

165 008A timer endp

166

167 ; read new symbol to rule, if it is

168 008A read\_rule proc

169 008A 33 C0 xor ax,ax

170 ; check symbol

171 008C B4 11 mov ah,11h

Turbo Assembler Version 3.0 01/23/18 11:13:05 Page 4

snake.asm

172 008E CD 16 int 16h

173 0090 75 01 jnz is\_tapped

174

175 0092 C3 ret

176

177 0093 is\_tapped:

178 ; get new rule

179 0093 B4 10 mov ah,10h

180 0095 CD 16 int 16h

181 ; protect of keyboard handling

182 0097 B4 11 mov ah,11h

183 0099 CD 16 int 16h

184 009B 75 F6 jnz is\_tapped

185

186 009D A2 0079r mov rule, al

187

188 ; check who is ruled

189 ; snake

190 00A0 3C 77 cmp al,'w'

191 00A2 74 25 je set\_snake\_rule

192 00A4 3C 64 cmp al,'d'

193 00A6 74 21 je set\_snake\_rule

194 00A8 3C 73 cmp al,'s'

195 00AA 74 1D je set\_snake\_rule

196 00AC 3C 61 cmp al,'a'

197 00AE 74 19 je set\_snake\_rule

198

199 ; dragon

200 00B0 3C 6F cmp al,'o'

201 00B2 B4 77 mov ah,'w'

202 00B4 74 17 je set\_dragon\_rule

203 00B6 3C 3B cmp al,';'

204 00B8 B4 64 mov ah,'d'

205 00BA 74 11 je set\_dragon\_rule

206 00BC 3C 6C cmp al,'l'

207 00BE B4 73 mov ah,'s'

208 00C0 74 0B je set\_dragon\_rule

209 00C2 3C 6B cmp al,'k'

210 00C4 B4 61 mov ah,'a'

211 00C6 74 05 je set\_dragon\_rule

212

213 ; no one is ruled. go out

214 00C8 C3 ret

215 00C9 set\_snake\_rule:

216 00C9 A2 027Ar mov snake\_rule,al

217 00CC C3 ret

218

219 00CD set\_dragon\_rule:

220 00CD 88 26 047Br mov dragon\_rule,ah

221 00D1 C3 ret

222 00D2 read\_rule endp

223

224 ; make step for one animal

225 00D2 step proc

226 00D2 8B 3E 007Ar mov di,animal

227 00D6 8B D7 mov dx,di

228

Turbo Assembler Version 3.0 01/23/18 11:13:05 Page 5

snake.asm

229 00D8 A0 0079r mov al,rule

230

231 00DB 3C 77 cmp al,'w'

232 00DD 74 0D je up

233 00DF 3C 64 cmp al,'d'

234 00E1 74 11 je rigth

235 00E3 3C 73 cmp al,'s'

236 00E5 74 15 je down

237 00E7 3C 61 cmp al,'a'

238 00E9 74 19 je left

239 00EB C3 ret

240

241 00EC up:

242 00EC 8B 3D mov di,[di]

243 00EE 83 EF 51 sub di,81

244 00F1 EB 19 90 jmp move

245 00F4 rigth:

246 00F4 8B 3D mov di,[di]

247 00F6 83 C7 01 add di,1

248 00F9 EB 11 90 jmp move

249 00FC down:

250 00FC 8B 3D mov di,[di]

251 00FE 83 C7 51 add di,81

252 0101 EB 09 90 jmp move

253 0104 left:

254 0104 8B 3D mov di,[di]

255 0106 83 EF 01 sub di,1

256 0109 EB 01 90 jmp move

257

258 010C move:

259 ; errors handling

260 010C 8A 85 069Cr mov al, byte ptr raw\_content[di]

261 0110 3C 3D cmp al, "="

262 0112 74 3C je do\_not\_move

263 0114 3C 7C cmp al, "|"

264 0116 74 38 je do\_not\_move

265 0118 3C 23 cmp al, "#"

266 011A 74 34 je do\_not\_move

267 011C 3C 25 cmp al, "%"

268 011E 74 3A je touth\_death

269 0120 3C 40 cmp al, "@"

270 0122 74 36 je touth\_death

271 0124 3C 5A cmp al, "Z"

272 0126 74 28 je do\_not\_move

273 0128 3C 2A cmp al, "\*"

274 012A 74 25 je eating

275 012C 3C 2E cmp al, "."

276 012E 74 21 je eating

277 0130 3C 2B cmp al, "+"

278 0132 74 1D je eating

279

280 0134 start\_moving:

281 0134 8B D7 mov dx,di

282 0136 8B 0E 0278r mov cx,animal\_length

283 013A 41 inc cx

284 013B BE 0000 mov si,0

285

Turbo Assembler Version 3.0 01/23/18 11:13:05 Page 6

snake.asm

286 013E move\_body:

287 ; get animal body index

288 013E 8B 3E 007Ar mov di,animal

289 0142 03 FE add di,si

290

291 ; change body

292 0144 8B 05 mov ax,[di]; remember

293 0146 89 15 mov [di],dx ; set

294 0148 8B D0 mov dx,ax ; remember new

295 014A 83 C6 02 add si, 2

296 014D E2 EF loop move\_body

297 014F C3 ret

298

299 0150 do\_not\_move:

300 0150 C3 ret

301

302 0151 eating:

303 0151 A1 0278r mov ax, animal\_length

304 0154 40 inc ax

305 0155 A3 0278r mov animal\_length,ax

306 0158 EB DA jmp start\_moving

307

308 015A touth\_death:

309 015A E8 0000 call death

310 015D step endp

311

312 ; if snake touch dragons head or dragon touch snakes head

313 015D death proc

314 015D A1 0479r mov ax,snake\_length

315 0160 3B 06 067Ar cmp ax,dragon\_length

316 0164 74 0E jz both\_death

317 0166 72 18 jb snake\_death

318

319 0168 B8 0016r mov ax, offset snake\_win\_mes

320 016B A3 0031r mov mes\_offset,ax

321 016E E8 FE98 call end\_scrin

322 0171 E8 FE91 call exit

323

324 0174 both\_death:

325 0174 B8 0021r mov ax, offset double\_kill\_mes

326 0177 A3 0031r mov mes\_offset,ax

327 017A E8 FE8C call end\_scrin

328 017D E8 FE85 call exit

329

330 0180 snake\_death:

331 0180 B8 000Br mov ax, offset dragon\_win\_mes

332 0183 A3 0031r mov mes\_offset,ax

333 0186 E8 FE80 call end\_scrin

334 0189 E8 FE79 call exit

335 018C death endp

336

337 018C set\_eat proc

338 018C 8B 0E 0682r mov cx, eat\_timeout

339 0190 49 dec cx

340 0191 83 F9 00 cmp cx,0

341 0194 74 32 jz new\_eat

342

Turbo Assembler Version 3.0 01/23/18 11:13:05 Page 7

snake.asm

343 0196 83 F9 23 cmp cx,35

344 0199 74 0D jz set\_eat\_point\_1

345

346 019B 83 F9 3C cmp cx,60

347 019E 74 11 jz set\_eat\_point\_2

348

349 01A0 83 F9 46 cmp cx,70

350 01A3 74 15 jz set\_eat\_point\_3

351

352 01A5 EB 1C 90 jmp end\_set\_eat

353

354 01A8 set\_eat\_point\_1:

355 01A8 A1 027Br mov ax, snake

356 01AB A3 067Cr mov eat\_point\_1,ax

357 01AE EB 13 90 jmp end\_set\_eat

358

359 01B1 set\_eat\_point\_2:

360 01B1 A1 047Cr mov ax, dragon

361 01B4 A3 067Er mov eat\_point\_2,ax

362 01B7 EB 0A 90 jmp end\_set\_eat

363

364 01BA set\_eat\_point\_3:

365 01BA A1 027Br mov ax, snake

366 01BD A3 0680r mov eat\_point\_3,ax

367 01C0 EB 01 90 jmp end\_set\_eat

368

369 01C3 end\_set\_eat:

370 01C3 89 0E 0682r mov eat\_timeout, cx

371 01C7 C3 ret

372

373 01C8 new\_eat:

374 01C8 B9 005A mov cx, 90

375 01CB 89 0E 0682r mov eat\_timeout, cx

376

377 01CF 8B 3E 067Cr mov di, eat\_point\_1

378 01D3 81 C7 069Cr add di, offset raw\_content

379 01D7 C6 05 2A mov byte ptr [di],'\*'

380

381 01DA 8B 3E 067Er mov di, eat\_point\_2

382 01DE 81 C7 069Cr add di, offset raw\_content

383 01E2 C6 05 2B mov byte ptr [di],'+'

384

385 01E5 8B 3E 0680r mov di, eat\_point\_3

386 01E9 81 C7 069Cr add di, offset raw\_content

387 01ED C6 05 2E mov byte ptr [di],'.'

388

389 01F0 C3 ret

390 01F1 set\_eat endp

391

392 01F1 set\_snake proc

393 ; Set head

394 01F1 8B 3E 027Br mov di,snake

395 01F5 C6 85 069Cr 25 mov raw\_content + di,'%'

396

397 ; Set body

398 01FA 8B 0E 0479r mov cx,snake\_length

399 01FE B3 02 mov bl,2

Turbo Assembler Version 3.0 01/23/18 11:13:05 Page 8

snake.asm

400 0200 set\_body:

401 0200 8B C1 mov ax,cx

402 0202 F6 E3 mul bl

403 0204 8B F8 mov di,ax

404 0206 8B BD 027Br mov di,snake[di]

405 020A C6 85 069Cr 23 mov raw\_content + di,'#'

406 020F E2 EF loop set\_body

407

408 ; Set tail void

409 0211 A1 0479r mov ax,snake\_length

410 0214 F6 E3 mul bl

411 0216 8B F8 mov di,ax

412 0218 8B BD 027Br mov di,snake[di]

413

414 021C C6 85 069Cr 20 mov raw\_content + di,' '

415

416 0221 C3 ret

417 0222 set\_snake endp

418

419 0222 set\_dragon proc

420 ; Set head

421 0222 8B 3E 047Cr mov di,dragon

422 0226 C6 85 069Cr 40 mov raw\_content + di,'@'

423

424 ; Set body

425 022B 8B 0E 067Ar mov cx,dragon\_length

426 022F 49 dec cx

427 0230 B3 02 mov bl,2

428 0232 set\_dragon\_body:

429 0232 8B C1 mov ax,cx

430 0234 F6 E3 mul bl

431 0236 8B F8 mov di,ax

432 0238 8B BD 047Cr mov di,dragon[di]

433 023C C6 85 069Cr 5A mov raw\_content + di,'Z'

434 0241 E2 EF loop set\_dragon\_body

435

436 ; Set tail void

437 0243 A1 067Ar mov ax,dragon\_length

438 0246 F6 E3 mul bl

439 0248 8B F8 mov di,ax

440 024A 8B BD 047Cr mov di,dragon[di]

441

442 ; check clearing

443 024E 8A 85 069Cr mov al,raw\_content + di

444 0252 3C 5A cmp al,'Z'

445 0254 74 01 jz do\_clear

446 0256 C3 ret

447

448 0257 do\_clear:

449 0257 C6 85 069Cr 20 mov raw\_content + di,' '

450 025C C3 ret

451 025D set\_dragon endp

452

453 ; all work with snake

454 025D snake\_RUN proc

455 ; move snake

456 025D C7 06 007Ar 027Br mov animal,offset snake

Turbo Assembler Version 3.0 01/23/18 11:13:05 Page 9

snake.asm

457 0263 A1 0479r mov ax,snake\_length

458 0266 A3 0278r mov animal\_length,ax

459 0269 A0 027Ar mov al,snake\_rule

460 026C A2 0079r mov rule,al

461

462

463 026F E8 FE60 call step

464 0272 E8 FF7C call set\_snake

465

466 ; set new snake length

467 0275 A1 0278r mov ax,animal\_length

468 0278 A3 0479r mov snake\_length,ax

469

470 027B C3 ret

471 027C snake\_RUN endp

472

473 ; all work with dragon

474 027C dragon\_RUN proc

475 ; move dragon

476 027C C7 06 007Ar 047Cr mov animal,offset dragon

477 0282 A1 067Ar mov ax,dragon\_length

478 0285 A3 0278r mov animal\_length,ax

479 0288 A0 047Br mov al,dragon\_rule

480 028B A2 0079r mov rule,al

481

482 028E E8 FE41 call step

483 0291 E8 FF8E call set\_dragon

484

485 ; set new dragon length

486 0294 A1 0278r mov ax,animal\_length

487 0297 A3 067Ar mov dragon\_length,ax

488

489 029A C3 ret

490 029B dragon\_RUN endp

491

492 029B main proc

493 029B B8 0000s mov ax,@data

494 029E 8E D8 mov ds,ax

495

496 02A0 BA 0684r mov dx, offset level\_file

497 02A3 E8 FD86 call open\_file

498 02A6 E8 FD98 call read\_file

499

500 02A9 C7 06 027Br 03DB mov snake, 1002-15

501 02AF C7 06 027Dr 03DA mov snake[1\*2], 1001-15

502 02B5 C7 06 027Fr 03D9 mov snake[2\*2], 1000-15

503 02BB C7 06 0281r 03D8 mov snake[3\*2], 999-15

504 02C1 C7 06 0283r 03D7 mov snake[4\*2], 998-15

505 02C7 C7 06 0285r 03D6 mov snake[5\*2], 997-15

506 02CD C7 06 0287r 03D5 mov snake[6\*2], 996-15

507 02D3 C7 06 0289r 03D4 mov snake[7\*2], 995-15

508 02D9 C7 06 028Br 03D3 mov snake[8\*2], 994-15

509 02DF C7 06 028Dr 03D2 mov snake[9\*2], 993-15

510 02E5 C7 06 028Fr 03D1 mov snake[10\*2], 992-15

511 02EB C7 06 0291r 03D0 mov snake[11\*2], 991-15

512 02F1 C7 06 0479r 000B mov snake\_length, 11

513

514 02F7 C7 06 047Cr 040B mov dragon, 1020+15

515 02FD C7 06 047Er 040C mov dragon[1\*2], 1021+15

516 0303 C7 06 0480r 040D mov dragon[2\*2], 1022+15

517 0309 C7 06 0482r 040E mov dragon[3\*2], 1023+15

518 030F C7 06 0484r 040F mov dragon[4\*2], 1024+15

519 0315 C7 06 0486r 0410 mov dragon[5\*2], 1025+15

520 031B C7 06 0488r 0411 mov dragon[6\*2], 1026+15

521 0321 C7 06 048Ar 0412 mov dragon[7\*2], 1027+15

522 0327 C7 06 048Cr 0413 mov dragon[8\*2], 1028+15

523 032D C7 06 048Er 0414 mov dragon[9\*2], 1029+15

524 0333 C7 06 0490r 0415 mov dragon[10\*2], 1030+15

525 0339 C7 06 0492r 0416 mov dragon[11\*2], 1031+15

526 033F C7 06 067Ar 000B mov dragon\_length, 11

527

528

529 0345 run:

530 ; print game in console

531 0345 E8 FD23 call print\_raw\_content

532

533 ; read new rule symbol

534 0348 E8 FD3F call read\_rule

535 ; check end

536 034B A0 0079r mov al,rule

537 034E 3C 1B cmp al,1Bh

538 0350 74 0E je ex

539

540 0352 E8 FE37 call set\_eat

541 0355 E8 FF05 call snake\_RUN

542 0358 E8 FF21 call dragon\_RUN

543

544 ; wait some time

545 035B E8 FD1A call timer

546 035E EB E5 jmp run

547

548 0360 ex:

549 0360 B8 0000r mov ax, offset end\_game\_mes

550 0363 A3 0031r mov mes\_offset,ax

551 0366 E8 FCA0 call end\_scrin

552 0369 E8 FC99 call exit

553

554 036C main endp

555 end main

Turbo Assembler Version 3.0 01/23/18 11:13:05 Page 11

Symbol Table

Symbol Name Type Value

??DATE Text "01/23/18"

??FILENAME Text "snake "

??TIME Text "11:13:05"

??VERSION Number 0300

@32BIT Text 0

@CODE Text \_TEXT

@CODESIZE Text 0

@CPU Text 0101H

@CURSEG Text \_TEXT

@DATA Text DGROUP

@DATASIZE Text 0

@FILENAME Text SNAKE

@INTERFACE Text 00H

@MODEL Text 2

@STACK Text DGROUP

@WORDSIZE Text 2

ANIMAL Word DGROUP:007A

ANIMAL\_LENGTH Word DGROUP:0278

BOTH\_DEATH Near \_TEXT:0174

CONTENT\_LENGTH Word DGROUP:169B

DEATH Near \_TEXT:015D

DOUBLE\_KILL\_MES Byte DGROUP:0021

DOWN Near \_TEXT:00FC

DO\_CLEAR Near \_TEXT:0257

DO\_NOT\_MOVE Near \_TEXT:0150

DRAGON Word DGROUP:047C

DRAGON\_LENGTH Word DGROUP:067A

DRAGON\_RULE Byte DGROUP:047B

DRAGON\_RUN Near \_TEXT:027C

DRAGON\_WIN\_MES Byte DGROUP:000B

EATING Near \_TEXT:0151

EAT\_POINT\_1 Word DGROUP:067C

EAT\_POINT\_2 Word DGROUP:067E

EAT\_POINT\_3 Word DGROUP:0680

EAT\_TIMEOUT Word DGROUP:0682

END\_FILE Byte DGROUP:068E

END\_GAME\_MES Byte DGROUP:0000

END\_SCRIN Near \_TEXT:0009

END\_SET\_EAT Near \_TEXT:01C3

EX Near \_TEXT:0360

EXIT Near \_TEXT:0005

HANDLE Word DGROUP:069A

IS\_TAPPED Near \_TEXT:0093

LEFT Near \_TEXT:0104

LEVEL\_FILE Byte DGROUP:0684

MAIN Near \_TEXT:029B

MES\_OFFSET Word DGROUP:0031

MOVE Near \_TEXT:010C

MOVE\_BODY Near \_TEXT:013E

NEW\_EAT Near \_TEXT:01C8

NEW\_LINE Byte DGROUP:002E

OPEN\_ERROR Near \_TEXT:0038

OPEN\_ERROR\_MES Byte DGROUP:0033

OPEN\_FILE Near \_TEXT:002C

PRINT Near \_TEXT:0000

PRINT\_RAW\_CONTENT Near \_TEXT:006B

RAW\_CONTENT Byte DGROUP:069C

READ\_COUNT\_ERROR Near \_TEXT:0062

READ\_COUNT\_ERROR\_MES Byte DGROUP:005C

READ\_ERROR Near \_TEXT:0059

READ\_ERROR\_MES Byte DGROUP:0048

READ\_FILE Near \_TEXT:0041

READ\_RULE Near \_TEXT:008A

RIGTH Near \_TEXT:00F4

RULE Byte DGROUP:0079

RUN Near \_TEXT:0345

SET\_BODY Near \_TEXT:0200

SET\_DRAGON Near \_TEXT:0222

SET\_DRAGON\_BODY Near \_TEXT:0232

SET\_DRAGON\_RULE Near \_TEXT:00CD

SET\_EAT Near \_TEXT:018C

SET\_EAT\_POINT\_1 Near \_TEXT:01A8

SET\_EAT\_POINT\_2 Near \_TEXT:01B1

SET\_EAT\_POINT\_3 Near \_TEXT:01BA

SET\_SNAKE Near \_TEXT:01F1

SET\_SNAKE\_RULE Near \_TEXT:00C9

SET\_SYMBOL Near \_TEXT:0015

SNAKE Word DGROUP:027B

SNAKE\_DEATH Near \_TEXT:0180

SNAKE\_LENGTH Word DGROUP:0479

SNAKE\_RULE Byte DGROUP:027A

SNAKE\_RUN Near \_TEXT:025D

SNAKE\_WIN\_MES Byte DGROUP:0016

START\_MOVING Near \_TEXT:0134

STEP Near \_TEXT:00D2

TIMER Near \_TEXT:0078

TIME\_IN\_TIME Near \_TEXT:007E

TIME\_NEXT Near \_TEXT:007B

TIME\_RET Near \_TEXT:0089

TOUTH\_DEATH Near \_TEXT:015A

UP Near \_TEXT:00EC

Groups & Segments Bit Size Align Combine Class

DGROUP Group

STACK 16 0100 Para Stack STACK

\_DATA 16 169D Word Public DATA

\_TEXT 16 036C Word Public CODE