

**Author: Абдуллаев Тимур Рашидович**

**Самоучитель программирования на "Liberty Basic"**

Абдуллаев Т.Р.

Программирование на  
языке «Liberty Basic»  
для начинающих

Второе издание  
исправленное и дополненное

Ташкент 2018  
УДК 355.23

Абдуллаев Т.Р.

Программирование на языке «Liberty Basic» для начинающих  
Самоучитель

Рассмотрены основы программирования на примере языка Liberty Basic версии 4.5.1, его базовые операторы и команды.

Самоучитель предназначен для самостоятельного изучения основ программирования. Он может быть полезным для преподавателей, студентов и школьников 7-8 классов.

## Оглавление

### Введение7

#### I.Основные операторы и команды11

##### 1.Описание среды программирования11

##### 2.Оператор присваивания «LET»13

##### 3.Оператор вывода данных «PRINT», команда «END»16

##### 4.Арифметические действия, операторы SIN(x), COS(x), TAN(x), ABS(x), EXP(x), LOG(x), INT(x), x MOD y, SQR(x), $x^y$ 18

##### 5.Оператор ввода данных «INPUT»20

##### 6.Действия со строковыми переменными, операторы: LEN(x\$), VAL(x\$), STR\$(x), TRIM\$(x\$), CHR\$(x), ASC(x\$), $x$+y$22$

##### 7.Оператор условия «IF», «THEN», «ELSE», логические операции «<», «>», «=», «<=», «>=», «<>»23

##### 8.Оператор безусловного перехода и условного перехода «GOTO», «IF..THEN GOTO», лейблы «»25

##### 9.Оператор цикла «FOR..TO..STEP..», «NEXT»26

##### 10.Оператор вызова подпрограммы «GOSUB», «RETURN»27

##### 11.Несколько операторов в одной строке, разделитель «:», комментарии «REM»28

##### 12.Функция «RND(1)»30

##### 13.Логические константы «TRUE», «FALSE»31

##### 14.Операторы «MID\$()», «INSTR()»32

##### 15.Оператор «INKEY\$»32

16. Вид оператора условия 33

17. Комбинации логических операций «AND», «OR», «XOR» и «NOT» 34

18. Операторы «SELECT CASE» и «CASE», команда «END SELECT» 35

19. Массивы данных, многомерные массивы, оператор «DIM» 37

20. Вложенные циклы 38

21. Функции «Function», «End Function», процедуры «Call Sub», «End Sub» и глобальные переменные «Global» 39

II. Основы графики 44

Введение 44

1. Графическое окно, оператор «OPEN», команды «WAIT», «NOMAINWIN» 44

2. Графическое подокно в окне, оператор «GRAPHICBOX» 49

3. Точка, операторы «SET», «SIZE» команды «DOWN», «UP», «FLUSH» 53

4. Цвет, операторы «COLOR», «FILL» 55

5. Линия, операторы «GO», «TURN», «GOTO», «LINE» команды «NORTH», «HOME» 58

6. Прямоугольник, операторы «BOX», «BOXFILLED», «BACKCOLOR», «PLACE» 61

7. Круг, операторы «CIRCLE», «CIRCLEFILLED» 64

8. Эллипс, операторы «ELLIPSE», «ELLIPSEFILLED» 65

9. Перехват ввода с клавиатуры, оператор «INKEY\$», команда «WHEN CHARACTERINPUT» 67

III. Интерфейсные программы 69

Введение 69

1.Текст или надпись, операторы «STATICTEXT», «FONT», команда «DISABLE»69

2.Кнопка, оператор «BUTTON»73

3.Текстовое окно, оператор «TEXTBOX», команды «CONTENTS?», «CLS», знак « \ »74

4.Группировка элементов, «GROUPBOX», команда «SETFOCUS»77

5.Отметка параметров, оператор «CHECKBOX», команды «SET», «RESET», «VALUE?», «TRAPCLOSE» и «CLOSE»78

6.Переключатель «радиокнопка», оператор «RADIOBUTTON»80

7.Выбор из списка параметров, операторы «LISTBOX», команда «SELECTION?»82

8.Выпадающий список параметров, оператор «COMBOBOX», команда «SELECTINDEX»84

9.Выпадающее меню, оператор «POPUPMENU», команда «WHEN RIGHTBUTTONUP», знаки «&», «\_» и «;»85

10.Меню в заголовке, оператор «MENU», знак «|»87

IV.Работа с файлами90

Введение90

1.Отображение нарисованной кнопки, операторы «BMPBUTTON» и «LOADBMP», команда «BITMAP»90

2.Отображение рисунка в формате «.bmp», оператор «DRAWBMP», команда «REDRAW»92

3.Проигрывание звукового файла в формате «.wav», оператор «PLAYWAVE»94

4.Проигрывание звукового файла в формате «.mid», операторы «PLAYMIDI» И «TIMER», команды «MIDIPOS()» и «STOPMIDI»96

5.Переименование файла, оператор «NAME..AS..»97

6.Удаление файла, оператор «KILL»98

7.Доступ к файлу только для чтения, режим «INPUT», команды «EOF()», «LINE INPUT», «INPUT» и «INPUT\$()»98

8.Создание файла и запись данных, режимы «OUTPUT» и «APPEND», команда «LOF()»103

9.Произвольная работа с файлом, режим «BINARY», оператор «SEEK», команда «LOK()»106

10.Сохранение рисунка, операторы «BMPSAVE» и «GETBMP»109

V.Работа со стандартными диалоговыми окнами и с системой112

Введение112

1.Диалоговое окно выбора цвета, оператор «COLORDIALOG»112

2.Диалоговое окно выбора шрифта, оператор «FONTDIALOG»114

3.Диалоговое окно открытия и сохранения файла, оператор «FILEDIALOG», команда «!CLS»116

4.Окно сообщения, оператор «NOTICE»119

5.Окно выбора, оператор «CONFIRM»120

6.Окно ввода строки, оператор «PROMPT»121

7.Печать текстовых данных через принтер, оператор «LPRINT», команда «DUMP»122

8.Окно печати, оператор «PRINTERDIALOG»123

9.Печать графики, оператор «PRINT»124

10.Перехват действий пользователя без остановки программы, команда «SCAN»127

11.Дата, оператор «DATE\$()»128

12.Время, оператор «TIME\$()»130

13.Запуск сторонней программы, оператор «RUN»131



14.Работа с манипулятором «мышь», команды «WHEN ...»  
переменные «MouseX» и «MouseY»132

15.Перехват сообщений об ошибке, оператор «ON ERROR»,  
команда «RESUME», переменные «Err\$» и «Err»134

16.Создание исполняемого автономного файла написанной  
программы и иконки135

17.Создание исполняемого автономного файла в конвертере  
«LBB» - Liberty Basic Booster версии 3.10148

Приложение153

Варианты решения задач153

Предметный указатель185

Введение

Данный самоучитель составлен с использованием материалов,  
представленных в англоязычных справках (файлы «LIBERTY4.hlp» и  
«LibertyBASIC\_4.htm») к программному обеспечению Liberty Basic  
версии 4.03 и 4.5.1.

Целью данного самоучителя является научить написанию  
программ и дать понятие об основных возможностях языка Liberty  
Basic и не преследует целью описать все возможности. Более  
глубокое изучение языка возложено на самих читателей, путем  
изучения вышеупомянутых справочных материалов.

Первая версия язык Liberty Basic была разработана в 1992 году  
Карлом Гюндедем (англ. Carl Gundel). Версия 4.5.1 была выпущена в  
мае 2017 года.

Среда программирования Liberty Basic предназначена для  
изучения программирования начинающими, при этом позволяет

создавать полноценный пользовательский интерфейс небольших программ.

Особенностью языка является простота его освоения.

Основным недостатком бейсико-подобных языков, как правило, является возможность использования переменных без их предварительного определения и возможность писать неструктурированные программы, а также возможность использования операторов, позволяющих переходить из любой части программы в любую другую. В совокупности все это запутывает и усложняет понимание конечного кода и, как следствие, поиск ошибок, но для небольших программ эти недостатки несущественны.

При этом если сразу приучить себя к следующему:

все переменные определять в начале кода,  
каждую переменную описывать комментариями,  
по возможности избегать использования операторов перехода,  
делить программный код на логические части,  
подписывать каждую часть,  
— то на бейсике вполне возможно писать довольно крупные программы.

Весь процесс программирования делится на несколько частей:  
представление конечного результата;  
формулирование основных частей программы;  
первый набросок интерфейса ввода данных (если имеется);  
написание логической части программы, содержащей ввод данных, их обработку и получение конечных данных;  
завершение интерфейса ввода и вывода данных;  
добавление графической составляющей.

Интерфейс - это визуальный способ программы запрашивать данные у пользователя, отображать различную информацию и выводить результат на экран монитора. Пример типичного интерфейса - это внешний вид калькулятора Microsoft.

На протяжении написания кода следует периодически «прогонять» программу, то есть проводить пробные запуски, используя так называемые «заглушки» (переменные с заранее определенными параметрами, предполагаемые в процессе выполнения ненаписанных частей программы). При прогоне, как

правило, выявляются и устраняются ошибки. При этом до 50% времени может уйти именно на устранение ошибок.

После написания и предварительной отладки кода, программа проверяется на «дурака», то есть проверяется возможность работы программы. Программа прогоняется с крайними возможными параметрами, а также проверяется устойчивость программы к вводу ошибочных данных.

Большую роль в написании программы играют комментарии к написанному коду, о них речь пойдет ниже.

В самоучителе операторы и команды выделены синим цветом, примеры программ приведены после слова «Пример:», результат выполнения программного кода следует после последней строки программного кода с горизонтальным подчеркиванием, задачи для самостоятельного решения выделены зеленым цветом, комментарии в программном коде выделены серым цветом.

Основные операторы и команды

Описание среды программирования

Запуск среды программирования Liberty basic осуществляется запуском программы «liberty.exe», после чего открывается основное окно программы.

Меню редактора

Меню редактора

Меню редактора

Меню редактора

Кнопки быстрого выполнения

Кнопки быстрого выполнения

Кнопки быстрого выполнения

Кнопки быстрого выполнения

Окно для написания кода программы

Окно для написания кода программы

Окно для написания кода программы

Окно для написания кода программы

Кнопки быстрого выполнения:

«Дебагер» - запуск в режиме пошагового выполнения

«Дебагер» - запуск в режиме пошагового выполнения

Вставить

Вставить

Вырезать

Вырезать

Настройки

Настройки

Распечатать текст программы

Распечатать текст программы

«Дебагер» - запуск в режиме пошагового выполнения

«Дебагер» - запуск в режиме пошагового выполнения

Вставить

Вставить

Вырезать

Вырезать

Настройки

Настройки

Распечатать текст программы

Распечатать текст программы

Помощь

Помощь

Помощь

Помощь

Открыть файл

Открыть файл

Открыть файл

Открыть файл

Новый файл

Новый файл

Заметки по возможностям Liberty Basic

Заметки по возможностям Liberty Basic

Новый файл

Новый файл

Заметки по возможностям Liberty Basic

Заметки по возможностям Liberty Basic

Сохранить файл

Сохранить файл

Сохранить файл

Сохранить файл

Запуск кода

Запуск кода

Копировать

Копировать

Найти, заменить

Найти, заменить

Перепрыгнуть к ...

Перепрыгнуть к ...

Сайт в Интернете

Сайт в Интернете

Запуск кода

Запуск кода

Копировать

Копировать

Найти, заменить

Найти, заменить

Перепрыгнуть к ...

Перепрыгнуть к ...

Сайт в Интернете

Сайт в Интернете

При запуске редактора в «окне написания кода программы» отображено описание Liberty Basic и код запуска справки.

Для написания собственного кода необходимо нажать кнопку «Новый файл», в окне для написания кода ввести свой код программы, для запуска программы нажать «Запуск кода», после запуска программы откроется основное окно выполнения программы.

При нажатии кнопки «Запуск в режиме пошагового выполнения» открывается окно так называемого «дебагера», то есть «вылавливателя ошибок».



Пошаговое выполнение с короткими автопаузами

Пошаговое выполнение с короткими автопаузами

Шаг через ...

Шаг через ...

Пошаговое выполнение

Пошаговое выполнение

Запуск без пауз

Запуск без пауз

Меню дебагера

Меню дебагера

Пошаговое выполнение с короткими автопаузами

Пошаговое выполнение с короткими автопаузами

Шаг через ...

Шаг через ...

Пошаговое выполнение

Пошаговое выполнение

Запуск без пауз

Запуск без пауз

Меню дебагера

Меню дебагера

Шаг на выход

Шаг на выход

Шаг на выход

Шаг на выход

Окно, где показаны значения переменных

Окно, где показаны значения переменных

Окно, где показаны значения переменных

Окно, где показаны значения переменных

Текущая строка выделена

Текущая строка выделена

Текущая строка выделена

Текущая строка выделена

Окно с текстом кода программы

Окно с текстом кода программы

Окно с текстом кода программы

Окно с текстом кода программы

Оператор присваивания «LET»

Основные типы данных (целые числа, вещественные числа с плавающей запятой, строковые переменные)

Целые числа - числа в диапазоне от  $-1*10512$  до  $1*10512$  без дробной части.

Вещественные числа с плавающей запятой - числа в диапазоне от  $-1.99999999*10256$  до  $1.99999999*10256$ .

! Компилятор обрабатывает только первые девять цифр вещественных чисел, остальное округляется.

Примеры (не программный код):

Целые числа:

A=1

B=3

C=A+B-10

C=-6

Вещественные числа:

A=0.123

B=-2.321

C=(A+B)/3

C=-0.73266667

A=1000.33333

B=0.00000032

C=A/B

C=3.12604166e9 (где e9 обозначает 10<sup>9</sup>)

В дальнейшем рабочий программный код в примерах приводится до горизонтальной линии, после горизонтальной линии показан результат работы кода.

Строковые переменные - переменные, содержащие символы, буквы, слова или предложения. Символы могут иметь значения от «0» до «255» согласно ASCII коду (American Standard Code for Information Interchange - Американский Стандартный Код для Информационного Обмена). Соответствие кодов символам представлены в «Таблице значений ASCII кода» ниже. Максимальная длина строковой переменной может достигать двух миллионов символов.

Строковые переменные после имени переменной обозначаются знаком «\$», значение переменной заключается в кавычки «"».

Примеры (не программный код):

Строковые переменные:

A\$="G"

B\$="408"

C\$=A\$+"-"+B\$

C\$="G-408"

Для хранения данных и значений используются переменные, имена которых должны содержать только латинские буквы и цифры.

Для присваивания переменным значения используется оператор «let». Оператор присваивания можно не писать.

Пример:

```
let A=21
```

```
let B=-32.21
```

```
C$="Answer"
```

```
D=A+B
```

В нижеприведенной таблице в колонке «Код» указаны значения ASCII кода символов, в колонке «Знач.» их печатные отображения.

Таблица значений ASCII кода

Код

Знач.

Код

Знач.

Код

Знач.

Код

Знач.

0

Служебные не печатные символы такие как табуляция, конец строки, кнопки вправо, влево, вверх, вниз и т.д.

32\*

” ”

64

@

96

、

1

33

!

65

A

97

a

2

34

"

66

B

98

b

3

35

#

67

C

99

c

4

36

\$

68

D

100

d

5

37

%

69

E

101

e

6

38

&

70

F

102

f

7

39

,

71

G

103

g

8

40

(

72

H

104

h

9

41

)

73

I

105

i

10

42

\*

74

J

106



j

11

43

+

75

K

107

k

12

44

,

76

L

108

l

13

45

—

77

M

109

m

14

46

.

78

N

110

n

15

47

/

79

O

111

o

16

48

0

80

P

112

p

17

49

1

81

Q

113

q

18

50

2

82

R

114

r

19

51

3

83

S

115

s

20

52

4

84

T

116

t

21

53

5

85

U

117

u

22

54

6

86

V

118

v

23

55

7

87

W

119

w

24

56

8

88

X

120

x

25

57

9

89

Y

121

y

26

58

:

90

Z

122

z

27

59

;

91

[

123

{

28

60

<

92

\

124

|

29

61

=

93

]

125

}

30

62

>

94

^

126

~

31

63

?

95

-

127\*\*

Код

Знач.

Код

Знач.

Код

Знач.

Код

Знач.

128

Ђ

160\*\*

192

А

224

а

129

Ѓ

161

Ў

193

Б

225

б

130

,

162

ÿ

194

В

226

в

131

ѓ

163

Ј

195

Г

227

г

132

”

164

ѣ

196

Д

228

д



133

...

165

Г

197

Е

229

е

134

†

166

‡

198

Ж

230

ж

135

‡

167

§

199

3

231

з

136

€

168

Ë

200

И

232

и

137

‰

169

©

201

Й

233

й

138

Ль

170

Є

202

К

234

к

139

‹

171

«

203

Л

235

л

140

Ѓ

172

¬

204

М

236

м

141

Ќ

173

205

Н

237

н

142

Ѕ

174

®

206

О

238

о

143

џ

175

ï

207

Π

239

π

144

ћ

176

◦

208

P

240

p

145

,

177

±

209

C

241

c

146

,

178

I

210

T

242

T

147

“

179

i

211

y

243

y

148

”

180

Г

212

Φ

244

φ

149

•

181

μ

213

X

245

x

150

-

182

¶

214

Ц

246

ц

151

—

183

.

215

Ч

247

ч

152\*\*

184

ë

216

Ш

248

ш

153

™

185

№

217

Щ

249

щ

154

ль

186

є

218

Ъ

250

Ъ

155

>

187

»

219

Ы

251

Ы

156

Ъ

188

ј

220

Ъ

252

Ъ

157\*\*

189

S

221

Э

253

э

158

Һ

190

һ

222

Ю

254

ю

159

Ц

191

ц

223

Я

255

я

\* - под 32-м кодом находится знак пробела.

\*\* - символы под кодами 127, 152, 157, 160 в данной версии  
Liberty Basic не отображаются.

Оператор вывода данных «PRINT», команда «END»

Способы вывода данных

Разделители «,», «;»



Оператор «print» предназначен для вывода данных и значений переменных.

Пример:

```
A=12.2
```

```
print 10
```

```
print A
```

```
print "123 стула"
```

```
10
```

```
12.2
```

```
123 стула
```

Разделитель «;» применяется для вывода нескольких значений в одном операторе.

Пример:

```
A=12.2
```

```
print 10;" ";A;" 123 number"
```

```
10 12.2 123 number
```

Разделитель «,» применяется для вывода нескольких значений в одном операторе, при этом между значениями вставляется 10 пробелов.

Пример:

```
A=12.2
```

```
print 10,A,"123 number"
```

```
10 12.2 123 number
```

Команда «end» говорит компьютеру, что выполнение программы закончено.

Пример:

```
A=12.2
```

```
print 10,A,"123 number"
```

end 'программа будет работать до этой строки и остановится

```
print "NNN" 'эта строка никогда не будет выполнена
```

```
10 12.2 123 number
```

Реализовать программу:

а) четырем разным переменным присвоить данные (имя, дату рождения, рост, вес) и вывести их значение на экран сперва в одной строке, потом в столбец. При выводе на экран все данные должны быть пояснены;

б) написать четыре строки с оператором «print», в каждой строке вывести произвольные данные, первую и третью строку завершить знаком «;», вторую строку завершить знаком «,». Получившийся результат проанализировать.

Арифметические действия, операторы SIN(x), COS(x), TAN(x), ABS(x), EXP(x), LOG(x), INT(x), x MOD y, SQR(x),  $x^y$

Арифметические действия можно выполнять в операторах «let» и «print».

Основные арифметические операции:

+ - сложение;

— - вычитание;

\* - произведение;

/ - отношение;

^ - степень.

Приоритет операций следующий:

() - выполняются действия, заключенные в скобках;

^ - выполняется возведение в степень, знак «^» - это делитель между числом и степенью;

\*,/ - произведение и отношение выполняются по порядку следования;

+,- - сложение и вычитание выполняются по порядку следования.

Пример решения функции «»:

x=111

Y=3.4\*x^2-32\*x+23\*(x-2)/x

print "Y=";Y 'решение функции через переменную

print "Y=";3.4\*x^2-32\*x+23\*(x-2)/x

'или решение функции непосредственно в операторе print

Y=38361.9856

Y=38361.9856

Некоторые арифметические операторы:

SIN(x) - возвращает (то есть получается в процессе выполнения) синус числа «x», число должно быть выражено в радианах;

COS(x) - возвращает косинус числа «x», число должно быть в радианах;

TAN(x) - возвращает тангенс числа «x», число должно быть в радианах;

ATN(x) - возвращает арктангенс числа «x», ответ в радианах;

ABS(x) - возвращает модуль числа «x», «|x|»;

EXP(x) - возвращает степень экспоненты «ex», где  $e=2.7182818$ ;

LOG(x) - возвращает десятичный логарифм «log10x» по основанию «10» от числа «x»;

INT(x) - возвращает целую часть от числа «x», не округляет, а именно отбрасывает дробную часть;

Пример:

A=123.987

B=INT(A)

print B

123

x MOD y - возвращает целый остаток от деления числа «x» на число «y»;

Пример:

A=15

B=6

C=A MOD B

print C

3

SQR(x) - возвращает квадратный корень числа «x», «»;

$x^y$  - возвращает степень числа «x» по основанию «y», «xy», степень «y» может быть как отрицательная, так и дробная.

Пример реализации формулы нахождения гипотенузы:

A=3

B=4

C=SQR(A<sup>2</sup>+B\*B)

```
print "C=";C
```

```
C=5
```

Реализовать формулы:

а) нахождения силы притяжения двух тел «», ответ в Ньютонах;

б) расчета эквивалентного сопротивления двух параллельно соединенных резисторов «», ответ в омах;

в) переменной присваиваем значение дробного числа, необходимо найти дробную часть числа и вывести её на экран;

г) находим степень двойки, степени присваиваем любое число (отрицательное, положительное или ноль).

Оператор ввода данных «INPUT»

Ввод данных разного типа

Ввод данных одним оператором последовательно в несколько переменных

Вывод текста оператором ввода данных

Оператор «input» предназначен для ввода данных в процессе выполнения программы.

Пример:

input A 'компьютер запросит пользователя значение переменной «A»

```
V=A^0.5
```

```
print V
```

```
?16
```

```
4
```

Компьютер в процессе выполнения программы выводит на экран символ «?» и ждет, когда пользователь введет значение переменной, например «16» и нажмет кнопку «Enter», после этого компьютер продолжит выполнение программы.

Оператор «input» поддерживает ввод значений в несколько переменных.

Пример:

input A,B,C\$ 'компьютер последовательно запросит значение переменной «A», «B» и строковой переменной «C\$»

D=A\*B

print C\$;"/";D

?1

??2

???привет

привет/2

Оператор «input» также поддерживает вывод текстового сообщения, заключенного в кавычках.

Пример:

input "Введите A,B,C для D=A\*B-C ";A,B,C

D=A\*B-C

print "D=";A;"\*";B;"-";C;"=";D

Введите A,B,C для D=A\*B-C 10

??23

???32

D=10\*23-32=198

Реализовать диалоговый режим компьютер-человек, то есть компьютер последовательно запрашивает у пользователя имя, затем приветствует его по имени, далее компьютер запрашивает требуемые данные для решения формул из 4-го параграфа I Главы.

Действия со строковыми переменными, операторы: LEN(x\$), VAL(x\$), STR\$(x), TRIM\$(x\$), CHR\$(x), ASC(x\$), x\$+y\$

Строковые операторы:

LEN(x\$) - возвращает длину символьной переменной.

Пример:

```
input "Введите своё имя:";A$
```

```
B=LEN(A$)
```

```
print "Длина вашего имени=";B
```

```
Введите своё имя:Timur
```

```
Длина вашего имени=5
```

VAL(x\$) - возвращает число, полученное преобразованием строковой переменной, содержащей цифры.

Пример:

```
A$="123.321"
```

```
B=VAL(A$)
```

```
print A$;"/";B
```

123.321/123.321

STR\$(x) - возвращает строку, полученную преобразованием числа в символы.

Пример:

```
A=123.321
```

```
B$=STR$(A)
```

```
print A;"/";B$
```

123.321/123.321

TRIM\$(x\$) - возвращает строку, у которой удалены пробелы в начале и в конце.

Пример:

```
A$=" Привет "
```

```
B$=TRIM$(A$)
```

```
print "/" ;A$;"/";B$;"/"
```

/ Привет /Привет/

CHR\$(x) - возвращает символ, соответствующий коду в «x» согласно ANSCII кодировке, воспринимает коды в диапазоне от «0» до «255»..

Пример:

```
A$=CHR$(192)
```

```
B=193
```

```
print A$+CHR$(B)
```

АБ



ASC(x\$) - возвращает число, соответствующее ANSCII коду символа в строковой переменной.

Пример:

```
A$="A"
```

```
B=ASC("Б")
```

```
print ASC(A$);"/";B
```

```
192/193
```

x\$+y\$ - производится сцепка двух строковых значений.

Пример:

```
A$="Лондон"
```

```
B$="прощай."
```

```
C$=A$+" "+B$
```

```
print C$
```

```
Лондон прощай.
```

Реализовать:

ввод одного целого и одного дробного положительных чисел, преобразование их в строку, сложение их, как строковых переменных, определение длины полученной строковой переменной, преобразование полученной строковой переменной в число.

ввод символа, определение его ASCII кода, вычитание из кода «32» и преобразование полученного кода обратно в символ.

Оператор условия «IF», «THEN», «ELSE», логические операции «<», «>», «=», «<=», «>=», «<>»

Оператор «if...then...» предназначен для решения логических выражений, например «if A>0 then A=1», где «if» непосредственно сам оператор, «A>0» логическое выражение, которое может быть верным или ложным, «then» служебное слово после которого следует оператор, который будет выполняться если логическое выражение оказалось верным.

Пример:

```
input "Введите А и В для C=A/V:";A,V
if V=0 then print "На ноль делить нельзя!"
if V<>0 then print "C=";A;"/";B;"=";A/V
Введите А и В для C=A/V:3
??0
На ноль делить нельзя!
```

Оператор «if...then...else...» аналогичен базовому формату, при этом если условие логического выражения не выполняется или ложное, то выполняется оператор, следующий после слова «else».

Пример:

```
input "Введите А и В для C=A/V:";A,V
if V=0 then print "На ноль делить нельзя!" else print
"C=";A;"/";B;"=";A/V
Введите А и В для C=A/V:3
??0
На ноль делить нельзя!
```

Базовые логические операции:

$x < y$  - x меньше y

$x > y$  -  $x$  больше  $y$

$x = y$  -  $x$  равно  $y$

$x \leq y$  -  $x$  меньше либо равно  $y$

$x \geq y$  -  $x$  больше либо равно  $y$

$x <> y$  -  $x$  не равно  $y$

Реализовать запрос двух чисел, их сравнение, и вывод ответа, которое из них больше, а которое меньше.

Оператор безусловного перехода и условного перехода «GOTO», «IF..THEN GOTO», лейблы «»

Оператор безусловного перехода «goto» предназначен для перехода в любое место программы, обозначенное так называемой лейблой «». Имя лейблы «L1», заключенное в квадратные скобки, может быть любое.

Пример:

A=10

goto

B=A\*2 'это место программы перепрыгнуто

print A

end

'компьютер продолжит выполнение программы с этого места

B=A+2

print B

12

Оператор условного перехода - это комбинация оператора условия и оператора безусловного перехода.

Пример:

```
input "Ведите А и В для А/В:";А,В
```

```
if В=0 then goto
```

```
print "А/В=";А/В
```

```
end
```

```
print "На ноль делить нельзя!"
```

```
Ведите А и В для А/В:3
```

```
??0
```

```
На ноль делить нельзя!
```

!Оператором безусловного и условного перехода необходимо пользоваться особенно внимательно во избежание зацикливания выполнения программы.

Пример:

```
А=10
```

```
В=В+А
```

```
goto
```

```
end
```

В данном примере программа не завершится и будет продолжать добавлять в переменную «В» переменную «А» пока память, отведенная под переменную «В», не переполнится.

Как избежать зацикливания программы описано в 10 параграфе V Главы.

!Операторами условного и безусловного перехода следует пользоваться в минимально разумных объемах, так как переход из одной части программы в другую и тем более возвращение в верхние части, затрудняют понимание конечного кода.

Реализовать нахождение корней квадратного уравнения « $ax^2 + bx + c = 0$ » по следующей ниже формуле.

$$D = b^2 - 4ac,$$

$$\text{если } D > 0, \text{ то } x_{1,2} = (-b \pm \sqrt{D}) / (2a)$$

$$\text{если } D = 0, \text{ то } x = -b / (2a)$$

если  $D < 0$ , то вывести ответ «корней нет»

Оператор цикла «FOR..TO..STEP..», «NEXT»

Оператор цикла предназначен для выполнения какой-то части программы несколько раз.

Пример:

$$A = 0$$

for i=1 to 5

$$A = A + i$$

next i

print "Сумма чисел от 1 до 5=";A

$$\text{Сумма чисел от 1 до 5} = 15$$

В данном примере «i» является переменной цикла с начальным значением «i=1», после слова «to» ставится конечное значение

переменной цикла «to 5», слово «next i» обозначает, что необходимо проверить достигнуто ли конечное значение цикла, если нет, - к переменной цикла добавить единицу и перейти в начало цикла.

Есть возможность использовать свой размер шага приращения переменной цикла.

Пример:

```
A=0
```

```
for i=1 to 5 step 2
```

```
A=A+i
```

```
next i
```

```
print "Сумма нечетных чисел от 1 до 5=";A
```

```
Сумма нечетных чисел от 1 до 5=9
```

Шаг приращения также может быть и отрицательным.

Пример:

```
A=0
```

```
for i=5 to 1 step -2
```

```
A=A+i
```

```
next i
```

```
print "Сумма нечетных чисел от 5 до 1=";A
```

```
Сумма нечетных чисел от 5 до 1=9
```

Реализовать:

Нахождение факториала.

Факториал - это произведение чисел от 1 до указанного, (Пример: Факториал «5», то есть  $5! = 1 * 2 * 3 * 4 * 5$ ).

Последовательный вывод символов ASCII кода с номером кодов от «32» до «255».

Оператор вызова подпрограммы «GOSUB », «RETURN»

Оператор вызова подпрограммы предназначен для выделения части программы в подпрограмму, к которой необходимо обращаться из разных частей основной программы. Оператор «gosub » дает команду на переход к подпрограмме, находящейся по лейбле «», сама подпрограмма завершается словом «return», после чего происходит возвращение к тому месту, откуда подпрограмма была вызвана.

Пример:

A=10

B=20

gosub 'первый вызов подпрограммы

print "C=";C

A=11

B=21

gosub 'второй вызов подпрограммы

print "C=";C

end

'подпрограмма

$C=A^2-B^{0.5}+A*B$

return

C=295.527864

C=347.417424

Реализовать решение формулы нахождения количества комбинаций спортлото «6 из 36» по формуле «», где  $m=6$   $n=36$ ,

функцию нахождения факториала вынести в подпрограмму, предусмотреть ввод параметров комбинации пользователем.

— Муж вывозил старшенького на дачу - к труду приучать. Поручил ему прополку грядки с луком и дал инструкцию: “Выдёргивать всё, кроме лука!”. Ванечка воспринял указание буквально и выдернул куст смородины, который рос в начале грядки. “Но это же не лук!” - оправдывался Ваня потом.

— Программист растёт.

— Муж вывозил старшенького на дачу - к труду приучать. Поручил ему прополку грядки с луком и дал инструкцию: “Выдёргивать всё, кроме лука!”. Ванечка воспринял указание буквально и выдернул куст смородины, который рос в начале грядки. “Но это же не лук!” - оправдывался Ваня потом.

— Программист растёт.

— Муж вывозил старшенького на дачу - к труду приучать. Поручил ему прополку грядки с луком и дал инструкцию: “Выдёргивать всё, кроме лука!”. Ванечка воспринял указание буквально и выдернул куст смородины, который рос в начале грядки. “Но это же не лук!” - оправдывался Ваня потом.

— Программист растёт.

— Муж вывозил старшенького на дачу - к труду приучать. Поручил ему прополку грядки с луком и дал инструкцию: “Выдёргивать всё, кроме лука!”. Ванечка воспринял указание буквально и выдернул куст смородины, который рос в начале грядки. “Но это же не лук!” - оправдывался Ваня потом.

— Программист растёт.



Несколько операторов в одной строке, разделитель «:», комментарии «REM»

Для сокращения записи, несколько операторов можно писать в одной строке, операторы при этом разделяются знаком «:».

Пример:

```
input "A=?,V=?:";A,V
if V=0 then print "На ноль делить нельзя":end
print A/V
```

```
A=?,V=?;3
```

```
??0
```

```
На ноль делить нельзя!
```

Во второй строке кода после слова «then» стоят два оператора, выполняющихся последовательно один за другим.

В больших программах в обязательном порядке необходимо использовать комментарии к тексту программы, для того что бы самому не запутаться в алгоритме.

Комментарии пишутся после слова «REM» или отделяются знаком «'». Текст комментариев компилятором не обрабатывается, а просто отбрасывается.

Пример:

REM определение значений переменных

A=10

B=20

gosub 'первый вызов подпрограммы

print "C=";C

A=11

B=21

gosub 'второй вызов подпрограммы

print "C=";C

end 'конец выполнения программы

'подпрограмма решения формулы

$C=A^2-B^{0.5}+A*B$

return 'возвращение из подпрограммы

C=295.527864

C=347.417424

Функция «RND(1)»

Функция «RND(1)» или генератор случайных чисел возвращает случайное вещественное число в промежутке от «0.000...01» до «0.999...9». Является основой создания динамичных игр.

Пример игры орел/решка:

A=RND(1)'генерируется случайное число от 0 до 1

B=INT(A+0.5)'случайное число сдвигается в промежуток

'от 0.5 до 1.5 и отбрасывается дробная часть, в

'результате получается или 0 или 1 с вероятностью 50%

input "Орел/решка? (орел=1, решка=0, выход=2):";C

if C=2 then end

if C=B then print "Угадал!" else print "Проиграл"

goto

Орел/решка? (орел=1, решка=0, выход=2):1

Угадал!

Орел/решка? (орел=1, решка=0, выход=2):1

Проиграл

Орел/решка? (орел=1, решка=0, выход=2):0

Угадал!

Орел/решка? (орел=1, решка=0, выход=2):0

Угадал!

Орел/решка? (орел=1, решка=0, выход=2):0

Проиграл

Орел/решка? (орел=1, решка=0, выход=2):2

Реализовать игру больше-меньше:

компьютер загадывает целое число от 1 до 100;

пользователь пытается угадать число последовательным вводом вариантов;

компьютер после ввода очередного варианта дает ответ «больше», «меньше» или «угадал».

### Логические константы «TRUE», «FALSE»

При использовании логических выражений таких как « $x > y$ », « $x = y$ » или « $x < y$ » и др., результатом их вычисления является логическая константа «true», равная единице, если выражение истинно, или «false» равная нулю если выражение ложно.

Пример:

```
input "a,b:";a,b
```

```
c=(a>b)
```

```
c=c+(a=b)
```

```
print "c=";c
```

```
a,b:3
```

```
??2
```

```
c=1
```

Американцы создали машину, которая переводит с русского на английский.

Машина сначала задымилась, а потом и взорвалась после разговора двух русских:

— Ты собираешься встречать Старый Новый Год?

— Да нет наверное...

— Ну а че точно там известно, не известно?

— Да поглядим, позже решим - будем не будем...

Американцы создали машину, которая переводит с русского на английский.

Машина сначала задымилась, а потом и взорвалась после разговора двух русских:

— Ты собираешься встречать Старый Новый Год?

— Да нет наверное...

— Ну а че точно там известно, не известно?

— Да поглядим, позже решим - будем не будем...

Американцы создали машину, которая переводит с русского на английский.

Машина сначала задымилась, а потом и взорвалась после разговора двух русских:

— Ты собираешься встречать Старый Новый Год?

— Да нет наверное...

— Ну а че точно там известно, не известно?

— Да поглядим, позже решим - будем не будем...

Американцы создали машину, которая переводит с русского на английский.

Машина сначала задымилась, а потом и взорвалась после разговора двух русских:

— Ты собираешься встречать Старый Новый Год?

— Да нет наверное...

— Ну а че точно там известно, не известно?

— Да поглядим, позже решим - будем не будем...

Реализовать функцию « $Y=f(x)$ » в одном выражении без использования оператора «if..then..» при следующих условиях:

$Y=1$ , если  $x>0$ ;  $Y=0$ , если  $x=0$ ;  $Y=-1$ , если  $x<0$

Операторы «MID\$()», «INSTR()»

Оператор «INSTR(A\$,B\$,n)» предназначен для поиска подстроки «B\$» в строке «A\$» начиная с позиции «n» и возвращает номер позиции начала подстроки, если начальная позиция «n» не указана, то поиск проводится, начиная с первой позиции.

Пример:

```
A$="кукуруза"
```

```
C=INSTR(A$,"y")
```

```
print C
```

```
C=INSTR(A$,"y",5)
```

```
print C
```

```
2
```

```
6
```

Оператор «MID\$(A\$,n,m)» предназначен для вырезания подстроки из строки «A\$» начиная с позиции «n» длиной «m».

Пример:

```
A$="прямоточно-последовательный"
```

```
i=INSTR(A$,"-")
```

```
B$=MID$(A$,1,i-1)
```

```
d=LEN(A$)
```

```
C$=MID$(A$,i+1,d-i)
```

```
print C$+"-"+B$
```

```
последовательный-прямоточно
```

Реализовать ввод дробного числа, преобразование его в строку, поиск разделителя дробной части от целой, выделение дробной части в отдельную строковую переменную.

Оператор «INKEY\$»

Компилятор Liberty Basic обрабатывает оператор «Inkey\$» не как классический Бейсик, дальнейшее описание дано для ознакомления. Работа оператора «Inkey\$» будет рассмотрена в 9 параграфе II Главы при изучении основ графики.

Оператор считывает один символ из буфера ввода клавиатуры и присваивает его значение символьной переменной. Если буфер ввода клавиатуры пустой, то будет присвоено значение без символа. Ожидания ввода символа с клавиатуры пользователем не происходит.

Пример: (программный код работать не будет)

```
K$=Inkey$  
if K$=" " then end  
if K$<>" " then print "/" ;K$; "/" ;  
goto
```

Вид оператора условия

Оператор условия «if..then..» может иметь следующий вид:

```
if..then  
..  
..  
else  
..  
..  
end if
```

Пример:

```
input "Введите A:";A  
if A=0 then  
print "A=0, повторите ввод."  
goto  
else  
B=SQR(ABS(A))  
print "B=";B  
end if  
Введите A:0
```



A=0, повторите ввод.

Введите A:3

B=1.73205081

При написании больших программ со множеством вложенных условий и циклов применяют отступ или табуляцию в начале строки для входящих в цикл или условие операторов для более наглядного отображения, как в показанном примере.

QvinA HAPXUST

В лет 7 меня научили готовить. Всё началось с яичницы. Мне предоставили такой порядок действий -> разбить яйцо -> жарить -> снять яйцо -> помыть сковороду.

Как-то раз утром мама попросила на всю семью пожарить 10 яиц...

Лоса69D

Маленький программист)

AndyHadlay

Иии чем закончилось??) я просто досих пор не очень умная)

QvinA HAPXUST

Я этот цикл 10 раз повторял хд через час мама таки спросила, что так долго. А я ток бе яйцо пожарил)000

(лично от автора: подсмотрел лет в 5-6 как жарить яйцо, ситуация осложнялась тем что газовая плита запитывалась от газового балона. Когда дома никого небыло - балон включил, яйцо разбил на сковороду...жарю. Смотрю яйцо пожарилось, а как выключить балон, что то я и запомнил. Шок, стресс, слезы, сгоревшее яйцо и все таки решился закрыть балон с газом.)

QvinA HAPXUST

В лет 7 меня научили готовить. Всё началось с яичницы. Мне предоставили такой порядок действий -> разбить яйцо -> жарить -> снять яйцо -> помыть сковороду.

Как-то раз утром мама попросила на всю семью пожарить 10 яиц...

Loca69D

Маленький программист)

AndyHadlay

Иии чем закончилось??) я просто досих пор не очень умная)

QvinA HAPXUST

Я этот цикл 10 раз повторял хд через час мама таки спросила, что так долго. А я ток бе яйцо пожарил)000

(лично от автора: подсмотрел лет в 5-6 как жарить яйцо, ситуация осложнялась тем что газовая плита запитывалась от газового балона. Когда дома никого небыло - балон включил, яйцо разбил на сковороду...жарю. Смотрю яйцо пожарилось, а как выключить балон, что то я и запомнил. Шок, стресс, слезы, сгоревшее яйцо и все таки решил закрыть балон с газом.)

QvinA HAPXUST

В лет 7 меня научили готовить. Всё началось с яичницы. Мне предоставили такой порядок действий -> разбить яйцо -> жарить -> снять яйцо -> помыть сковороду.

Как-то раз утром мама попросила на всю семью пожарить 10 яиц...

Loca69D

Маленький программист)

AndyHadlay

Иии чем закончилось??) я просто досих пор не очень умная)

QvinA HAPXUST

Я этот цикл 10 раз повторял хд через час мама таки спросила, что так долго. А я ток бе яйцо пожарил)000

(лично от автора: подсмотрел лет в 5-6 как жарить яйцо, ситуация осложнялась тем что газовая плита запитывалась от газового балона. Когда дома никого небыло - балон включил, яйцо разбил на сковороду...жарю. Смотрю яйцо пожарилось, а как выключить балон, что то я и запомнил. Шок, стресс, слезы, сгоревшее яйцо и все таки решился закрыть балон с газом.)

QvinA HAPXUST

В лет 7 меня научили готовить. Всё началось с яичницы. Мне предоставили такой порядок действий -> разбить яйцо -> жарить -> снять яйцо -> помыть сковороду.

Как-то раз утром мама попросила на всю семью пожарить 10 яиц...

Loca69D

Маленький программист)

AndyHadlay

Иии чем закончилось??) я просто досих пор не очень умная)

QvinA HAPXUST

Я этот цикл 10 раз повторял хд через час мама таки спросила, что так долго. А я ток бе яйцо пожарил)000

(лично от автора: подсмотрел лет в 5-6 как жарить яйцо, ситуация осложнялась тем что газовая плита запитывалась от газового балона. Когда дома никого небыло - балон включил, яйцо разбил на сковороду...жарю. Смотрю яйцо пожарилось, а как

выключить балон, что то я и запомнил. Шок, стресс, слезы, сгоревшее яйцо и все таки решился закрыть балон с газом.)

Комбинации логических операций «AND», «OR», «XOR» и «NOT»

Для выполнения сложных условий в одном логическом выражении применяют так называемое логическое умножение «AND», логическое сложение «OR» , сложение по модулю два «XOR» или инверсию «NOT».

Результат использования «AND»

if a AND b then ..

a

b

Результат

true

true

true

true

false

false

false

true

false

false

false

false

Результат использования «OR»

if a OR b then ..

a

b

Результат

true

true

true

true

false

true

false

true

true

false

false

false

Результат использования «XOR» в «if a XOR b then ..»

a

b

Результат

true

true

false

true

false

true

false

true

true

false

false

false

Результат использования «NOT» в «if NOT a then ..»

a

Результат

true

false

false

true

Пример:

```
print "C=SQR(A)/B"
```

```
input "Введите A и B:";A,B
```

```
if A>=0 AND B<>0 then C=SQR(A)/B:print "C=";C:end
```

```
goto
```

```
C=SQR(A)/B
```

```
Введите A и B:-3
```

```
??2
```

```
C=SQR(A)/B
```

```
Введите A и B:2
```

```
??0
```

```
C=SQR(A)/B
```

```
Введите A и B:2
```

```
??3
```

```
C=0.47140452
```

Реализовать:

проверку нахождения введенного числа в заданном диапазоне;

проверку на четность;

проверку нахождения введенного числа вне заданного диапазона.

## Операторы «SELECT CASE» и «CASE», команда «END SELECT»

Для организации разветвленного перехода используют оператор «select case num», где «select case» сам оператор, «num» переменная, на основании содержимого которой будет выбрана соответствующая строка программы. Если значение переменной не подходит ни под одну из строк, то будет выбрана строка с командой «case else». Оператор «select case» необходимо завершить командой «end select».

Пример:

```
input "Введите число от 1 до 3:";num
```

```
select case num
```

```
case 1
```

```
print "один"
```

```
case 2
```

```
print "два"
```

```
case 3
```

```
print "три"
```

```
case else
```

```
print "другое число"
```

```
end select
```

```
Введите число от 1 до 3:3
```

```
три
```

Значение переменной может быть символьное, а также для выбора одной строки можно перечислить несколько значений переменной.



Пример:

```
input "Введите день недели:";var$
select case var$
case "понедельник", "вторник", "среда", "четверг", "пятница"
print "рабочий день"
case "суббота", "воскресенье"
print "выходной"
case else
print "неизвестный день недели"
end select
```

```
Введите день недели:шестница
неизвестный день недели
```

Если одно и то же значение переменной есть для разных строк, то будет выбрана первая строка со значением переменной.

Пример:

```
num = 3
select case num
case 3, 5, 10 'будет выбрана эта строка
print "3, 5, 10" 'будет напечатана эта строка
case 3, 12, 14, 18
print "3, 12, 14, 18"
case else
print "Not evaluated."
end select
3, 5, 10
```

Выбор соответствующей строки может быть осуществлен и на основании логического выражения, в этом случае в операторе «select case» переменная не указывается, а логические выражения пишутся непосредственно в строках выбора.

Пример:

```
input "Введите число от 1 до 100: ";value
select case
case (value < 10) or (value > 50 and value < 60)
print "Первый case: ";value
case (value > 10) and (value < 50)
print "ВторойSecond case: ";value
case (value = 10) or (value = 50)
print "Третий case: ";value
case else
print "Число не попало в диапазон: ";value
end select
```

Введите число от 1 до 100:50

Третий case:50

Массивы данных, многомерные массивы, оператор «DIM»

Для хранения и обработки большого объема данных применяют переменные массивов.

```
DIM string$(20)
```

```
DIM number(30,15)
```

Оператор «DIM A(x)» применяется для определения количества элементов «x» массива «A» и выделения памяти под их хранение.

Массивы могут быть одномерные «A(x)» и многомерные: «A(x,y)», «A(x,y,z)» числовые «A(x)» и символьные «A\$(x)».

Пример:

```
DIM A(12)
```

```
print "Введите элементы массива"
```

```
for i=1 to 12
```

```
input A(i)
```

```
next i
```

```
S=0
```

```
for i=1 to 12
```

```
S=S+A(i)
```

```
next i
```

```
print "Сумма массива=";S
```

Введите элементы массива

?1

?4

?7

?45

?3

?5

?2

?5

?7

?2

?4

?6

Сумма массива=91

Реализовать заполнение массива размерностью 15 элементов случайными целыми числами от 1 до 36 и вывод их на экран.

Вложенные циклы

Для обработки, к примеру, многомерного массива можно использовать вложенные циклы.

Пример:

```
dim z(3,5)
```

```
for x=1 to 3'внешний цикл
```

```
for y=1 to 5'внутренний цикл
```

```
z(x,y)=x^2-x*y
```

```
print "z(";x;" ";y;"")=";z(x,y)
```

```
next y
```

```
next x
```

```
z(1,1)=0
```

```
z(1,2)=-1
```

```
z(1,3)=-2
```

```
z(1,4)=-3
```

```
z(1,5)=-4
```

```
z(2,1)=2
```

```
z(2,2)=0
```

```
z(2,3)=-2
```

```
z(2,4)=-4
```

```
z(2,5)=-6
```

$$z(3,1)=6$$

$$z(3,2)=3$$

$$z(3,3)=0$$

$$z(3,4)=-3$$

$$z(3,5)=-6$$

Реализовать:

заполнение массива размерностью 15 элементов случайными целыми числами от 1 до 15 без повторения чисел в массиве;

алгоритм сортировки массива, из предыдущего задания, по возрастанию.

Функции «Function», «End Function», процедуры «Call Sub», «End Sub» и глобальные переменные «Global»

Работу подпрограмм можно организовать через так называемые функции или процедуры.

Функция вызывается указанием ее имени «NameFunc(variable1, variable2)», где «NameFunc» имя функции, «(variable1, variable2)» имена переменных которые передаются для обработки в функцию. Между именем функции и скобкой с именами переменных не должно быть промежутка. Функция после завершения работы возвращает присвоенное ей значение. Код функции начинается с «Function» и заканчивается «End Function».

Пример:

```
Input "Введите катеты треугольника:";x,y
```

```
Z=Gipotenuza(x,y) 'Вызов функции
```

```

Print "Гипотенуза равна:";Z
Goto
End
Function Gipotenuza(a,b) '====Функция====
c=Sqr(a^2+b^2)
Gipotenuza=c
End Function
Введите катеты треугольника:3
??4
Гипотенуза равна:5
Введите катеты треугольника:4
??5
Гипотенуза равна:6.40312424

```

Процедура вызывается через «call NameSub variabl1, variabl2», где «call» оператор вызова процедуры, «NameSub» имя процедуры, «variabl1, variabl2» переменные, передающиеся в процедуру. Процедура после своего выполнения ничего не возвращает. Код процедуры начинается с «Sub» и заканчивается «End Sub».

Пример:

```

Input "Введите катеты треугольника:";x,y
Call Gipotenuza x,y 'Вызов процедуры
Goto
End
'====Процедура====
Sub Gipotenuza a,b
Z=Sqr(a^2+b^2)

```

```

Print "Гипотенуза равна:";Z
End Sub
Введите катеты треугольника:3
??4
Гипотенуза равна:5
Введите катеты треугольника:4
??5
Гипотенуза равна:6.40312424

```

Переменные, используемые в процедуре или функции, являются локальными, то есть в основном коде программы может быть переменная с таким же именем. Для использования переменной и в основном коде программы и в процедуре (функции) ее необходимо объявить как глобальную «Global variabl1», где «Global» команда для объявления переменной, «variabl1» имя переменной. Все массивы уже являются глобальными.

В данном примере и в функции и в основной программе используются переменные «n» и «m», значения которых не переходят друг к другу.

Пример:

```

print "Введите комбинацию спорт лото"
input "Сколько?";m
input "Из скольки?";n
if m>n Then goto
C=Factorial(n)/(Factorial(m)*Factorial(n-m))
print m;" из ";n;" имеет ";C;" комбинаций"
end

```

```
'=====
```

```
Function Factorial(m)
```

```
n=1
```

```
For i=1 To m
```

```
n=n*i
```

```
Next i
```

```
Factorial=n
```

```
End Function
```

Введите комбинацию спорт лото

Сколько?6

Из скольки?36

6 из 36 имеет 1947792 комбинаций

В примере показаном ниже переменные «G, M, R» являются глобальными и работают одинаково как в основной программе, так и в функции.

Пример:

```
Global G, M, R
```

```
G=6.74*10(-11)'Гравитационная постоянная
```

```
M=5.9726*10(27)'масса Земли в граммах
```

```
R=6371000'радиус Земли в метрах
```

```
print "Для нахождения силы притяжения Земли, введите:"
```

```
input "вашу массу тела в граммах:";m1
```

```
F=Power(m1)
```

```
print "F=";F;" Ньютон"
```

```
goto
```

```
end
```



```
'=====
```

```
Function Power(m1)
```

```
F=G*m1*M/R^2
```

```
Power=F
```

```
End Function
```

Для нахождения силы притяжения Земли, введите:

вашу массу тела в граммах:55000

F= 5.45469919e8 Ньютон

Одним из немало важных свойств процедур и функций является рекурсивный вызов. Рекурсия - вызов процедуры или функции самой себя.

В данном примере функция вызывает саму себя как «матрешка», но с меньшим коэффициентом, пока коэффициент не достигнет единицы.

Пример:

```
print "Введите комбинацию спорт лото"
```

```
input "Сколько?";m
```

```
input "Из скольки?";n
```

```
if m>n Then goto
```

```
C=Factorial(n)/(Factorial(m)*Factorial(n-m))
```

```
print m;" из ";n;" имеет ";C;" комбинаций"
```

```
end
```

```
'=====
```

```
Function Factorial(m)
```

```
If m>1 then
```

```
Factorial=m*Factorial(m-1)
```

```
'здесь функция вызывает саму себя
```

Else

Factorial=1

End If

End Function

Введите комбинацию спорт лото

Сколько?6

Из сколько?36

6 из 36 имеет 1947792 комбинаций

В следующем примере находится 15-тое число Фибоначи. Этот алгоритм самый популярный алгоритм по демонстрации рекурсии и самый низкопроизводительный.

Число Фибоначи равно сумме двух предыдущих чисел начиная с «1» и «1», то есть:  $Fib1=1$ ,  $Fib2=1$ ,  $Fib3=1+1=2$ ,  $Fib4=1+2=3$ ,  $Fib5=2+3=5$ ,  $Fib6=3+5=8$  и т.д.

Пример:

```
n=15
```

```
print fib(n)
```

```
end
```

```
'=====
```

```
function fib(n)
```

```
if n<3 Then
```

```
fib=1
```

```
else
```

```
fib=fib(n-1)+fib(n-2)
```

```
end if
```

```
end function
```

```
610
```

Реализуйте:

а) алгоритм поиска числа Фибоначи быстрый и без рекурсии и найдите 5000-ое число;

б) алгоритм нахождения корней квадратного уравнения из 8-го параграфа 1 Главы через процедуру.

Основы графики

Бабушка разговаривает по Скайпу с внуком.

— Сереженька, я тебе в фотошопе такой шарфик связала!

Бабушка разговаривает по Скайпу с внуком.

— Сереженька, я тебе в фотошопе такой шарфик связала!

Бабушка разговаривает по Скайпу с внуком.

— Сереженька, я тебе в фотошопе такой шарфик связала!

Бабушка разговаривает по Скайпу с внуком.

— Сереженька, я тебе в фотошопе такой шарфик связала!

Введение

Графические операторы и команды в Liberty Basic представлены довольно широкими возможностями.

Система координат для компьютерной графики определена:

начало координат - левый верхний угол;

ось X располагается слева на право;

ось Y располагается сверху вниз.

0

0

X

X

0

0

X

X

Y

Y

Y

Y

Рисование и отображение графики возможно в двух типах так называемых окон. В первом варианте дается команда отобразить графическое окно, во втором случае дается команда отобразить графическое подокно в обычном окне.

У каждого из типов отображения графики есть свои особенности.

Графическое окно, оператор «OPEN», команды «WAIT», «NOMAINWIN»

Указатель

Способы отображения графического окна

Параметры графического окна

Для наглядности приведем пример и разберем параметры данного примера.

Пример:

```
open "Drawing" for graphics as #handle  
wait
```

После выполнения примера откроется два пустых окна, основное окно и графическое окно.

Оператор «open "Drawing" for graphics as #handle» откроет графическое окно с заголовком «"Drawing"», с типом окна указанным после слова «for» в данном случае «graphics», после слова «as» указывается имя указателя на графическое окно «#handle».

Указатель - это переменная, которая хранит адрес объекта интерфейса, проще говоря, имя объекта. Перед именем указателя ставится знак «#».

Для того чтобы графическое окно не закрылось и можно было с ним взаимодействовать (перетаскивать, увеличивать, разворачивать, сворачивать и т.д), используется команда «wait».

Для того, чтобы не открывалось основное окно, используется команда «nomainwin»

Пример:

```
nomainwin  
open "Окно" for graphics as #wingraf  
wait
```

Существует несколько способов отображения графического окна.

По команде «graphics\_nsb» графическое окно откроется без вертикальной и горизонтальной прокрутки.

Пример:

```
nomainwin  
open "Окно" for graphics_nsb as #wingraf  
wait
```

По команде «graphics\_fs» графическое окно откроется развернутым по размеру экрана.

Пример:

```
nomainwin  
open "Окно" for graphics_fs as #wingraf  
wait
```

По команде «graphics\_fs\_nsb» графическое окно откроется без вертикальной и горизонтальной прокрутки развернутым по размеру экрана.

Пример:

```
nomainwin  
open "Окно" for graphics_fs_nsb as #wingraf  
wait
```

По команде «graphics\_nf\_nsb» графическое окно откроется без возможности поменять размер окна или развернуть на весь экран.

Пример:

```
nomainwin
```

```
open "Окно" for graphics_nf_nsb as #wingraf
```

```
wait
```

Для задания размеров окна используются следующие переменные.

Для определения горизонтального размера окна используют переменную «WindowWidth» определенную по умолчанию (то есть заранее), для горизонтального размера окна используют переменную «WindowHeight».

Пример:

```
nomainwin
```

```
WindowWidth = 200
```

```
WindowHeight = 100
```

```
open "Окно" for graphics_nf_nsb as #wingraf
```

```
wait
```

Для задания начального положения окна на экране компьютера, точнее, его верхнего левого угла, используют переменные, определенные по умолчанию, для горизонтальной



координаты X «UpperLeftX», для вертикальной координаты Y «UpperLeftY».

Пример:

```
nomainwin
```

```
WindowWidth = 200
```

```
WindowHeight = 100
```

```
UpperLeftX = 50
```

```
UpperLeftY = 50
```

```
open "Окно" for graphics_nf_nsb as #wingraf
```

```
wait
```

Для получения размеров текущего экрана определены переменные, для горизонтального размера «DisplayWidth», для вертикального размера «DisplayHeight»

Пример:

```
ShirEkрана=DisplayWidth
```

```
VisEkрана=DisplayHeight
```

```
print "Ширина экрана равна=";ShirEkрана
```

```
print "Высота экрана равна=";VisEkрана
```

```
Ширина экрана равна=1024
```

```
Высота экрана равна=768
```

Создать окно без возможности поменять его размер, габаритом в половину экрана компьютера и расположить его в центре.

Графическое подокно в окне, оператор «GRAPHICBOX»

Способы отображения графического подокна.

Для отображения графического подокна сперва описывают объект подокна «`graphicbox #win.gbox, 10, 10, 150, 150`», где «`#win.gbox`» составной указатель графического подокна «`gbox`», находящегося в окне «`win`», после составного указателя идут координаты графического подокна «`10, 10, 150, 150`», первые два числа указывают координаты «`X`» и «`Y`» левого верхнего угла, вторые два числа указывают ширину и высоту подокна.

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
wait
```

Существует несколько способов отображения основного окна.

По команде «`window_nf`» окно откроется без возможности поменять размер окна или развернуть на весь экран.

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window_nf as #win
wait
```

По команде «`window_popup`» окно откроется без обрамления, возможности поменять размер окна, развернуть или свернуть, а также закрыть.

Пример:

```
nomainwin
```

```
graphicbox #win.gbox, 10, 10, 150, 150
```

```
open "Drawing" for window_popup as #win
```

```
wait
```

По команде «dialog» окно откроется без возможности поменять размер окна, развернуть или свернуть.

Пример:

```
nomainwin
```

```
graphicbox #win.gbox, 10, 10, 150, 150
```

```
open "Drawing" for dialog as #win
```

```
wait
```

По команде «dialog\_nf» окно откроется без возможности поменять размер окна или развернуть на весь экран.

Пример:

```
nomainwin
```

```
graphicbox #win.gbox, 10, 10, 150, 150
```

```
open "Drawing" for dialog_nf as #win
```

```
wait
```

По команде «dialog\_fs» окно откроется на весь экран, без возможности поменять размер окна или свернуть.

Пример:

```
nomainwin
```

```
graphicbox #win.gbox, 10, 10, 150, 150
```

```
open "Drawing" for dialog_fs as #win
```

```
wait
```

По команде «dialog\_popup» окно откроется без обрамления, возможности поменять размер окна, развернуть или свернуть, а также закрыть.

Пример:

```
nomainwin
```

```
graphicbox #win.gbox, 10, 10, 150, 150
```

```
open "Drawing" for dialog_popup as #win
```

```
wait
```

Создать графическое подокно размером с окно типа «dialog» на весь экран, без возможности поменять размер окна или свернуть.

Точка, операторы «SET», «SIZE» команды «DOWN», «UP», «FLUSH»

Понятие «перо»

Рисование точки

Все операторы для рисования привязаны через оператор «print».

Механизм рисования в Liberty Basic организован через так называемое «перо», которое перед началом рисования необходимо опустить и по окончании рисования поднять.

Для того чтобы нарисовать точку, используют оператор «print #handle, "set 10 10"», где «#handle» - указатель (или составной указатель) на графическое окно, «"set 10 10"» - непосредственно оператор рисования точки, заключенный в кавычки «"» с координатами точки.

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down" 'опустить перо
print #win.gbox, "set 75 75"
'нарисовать точку с координатами x=75, y=75
print #win.gbox, "up" 'поднять перо
wait
```

Координаты точки можно задать через переменные.

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down" 'опустить перо
X=74
Y=76
```

```
print #win.gbox, "set ";X;" ";Y 'нарисовать точку с координатами
x=74, y=76
print #win.gbox, "up" 'поднять перо
wait
```

Для задания размера пера используют оператор «print #handle, "size 4"», где «4» - размер «пера», по умолчанию размер «пера» установлен равным «1».

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, "size 4" 'установка размера пера
print #win.gbox, "set 71 72"
print #win.gbox, "up"
wait
```

Для того что бы нарисованная картинка при сворачивании и разворачивании окна или при перекрытии другими окнами не стиралась, используют команду «"flush"»

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, "size 4"
print #win.gbox, "set 71 72"
```

```
print #win.gbox, "up"  
print #win.gbox, "flush"  
wait
```

Нарисовать окружность точками в центре графического подокна, для определения координат использовать формулы: « $X_1$ », « $Y_1$ », где « $X_1$ » и « $Y_1$ » - координаты центра окружности, « $R$ » - радиус окружности, « $a$ » - угол в радианах. Угол « $a$ » необходимо менять в диапазоне от «0» до «6.283» для получения окружности.

Цвет, операторы «COLOR», «FILL»

Цвет

Предопределенные цвета

Цвет фона

Цвет рисования

Liberty Basic оперирует более чем 16 700 000 оттенками цветов. Цвет может быть определен компонентно « $R G B$ », где « $R$ » - красный компонент, « $G$ » - зеленый компонент, « $B$ » - синий компонент, параметры « $R G B$ » могут принимать значения от «0» до «255».

Просыпаются утром программист с женой, за окном – грустный серый осенний день.

Жена:

— Какой сегодня серый день...

Программист:

— Да. Палитра слетела...

Просыпаются утром программист с женой, за окном – грустный серый осенний день.

Жена:

— Какой сегодня серый день...

Программист:

— Да. Палитра слетела...

Просыпаются утром программист с женой, за окном – грустный серый осенний день.

Жена:

— Какой сегодня серый день...

Программист:

— Да. Палитра слетела...

Просыпаются утром программист с женой, за окном – грустный серый осенний день.

Жена:

— Какой сегодня серый день...

Программист:

— Да. Палитра слетела...



Есть так же predetermined цвета:

yellow - желтый;

brown - коричневый;

red - красный;

darkred - темнокрасный;

pink - фиолетовый;

darkpink - темнофиолетовый;

blue - синий;

darkblue - темносиний;

green - зеленый;

darkgreen - темнозеленый;

cyan - голубой;

darkcyan - темноголубой;

white - белый;

black - черный;

lightgray - светлосерый;

darkgray - темносерый.

Заходит компьютерщик в булочную после бессонной ночи у компьютера, провозившись с установкой кривой видеокарты, и говорит:

— Мне, пожалуйста, буханку черно-белого хлеба и батон цветного....

Заходит компьютерщик в булочную после бессонной ночи у компьютера, провозившись с установкой кривой видеокарты, и говорит:

— Мне, пожалуйста, буханку черно-белого хлеба и батон цветного....

Заходит компьютерщик в булочную после бессонной ночи у компьютера, провозившись с установкой кривой видеокарты, и говорит:

— Мне, пожалуйста, буханку черно-белого хлеба и батон цветного....

Заходит компьютерщик в булочную после бессонной ночи у компьютера, провозившись с установкой кривой видеокарты, и говорит:

— Мне, пожалуйста, буханку черно-белого хлеба и батон цветного....

Для задания цвета рисования используется оператор «“color 255 255 255”» или «“color red”».

Пример:

```
nomainwin
```

```
graphicbox #win.gbox, 10, 10, 150, 150
```

```
open "Drawing" for window as #win
```

```
print #win.gbox, "down"
print #win.gbox, "size 10" 'установка размера пера
print #win.gbox, "color 255 0 0" 'компонентный цвет
print #win.gbox, "set 71 65"
print #win.gbox, "color yellow" 'предопределенный цвет
print #win.gbox, "set 71 75"
Re=0
Gr=255
Bl=0
print #win.gbox, "color ";Re;" ";Gr;" ";Bl
'компонентный цвет через переменные
print #win.gbox, "set 71 85"
print #win.gbox, "up"
print #win.gbox, "flush"
wait
```

Для задания цвета фона используется оператор «"fill 255 255 255"»

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, "size 6"
print #win.gbox, "fill 255 255 0" 'цвет фона
print #win.gbox, "color 255 0 255"
print #win.gbox, "set 75 75"
```

```
print #win.gbox, "up"  
print #win.gbox, "flush"  
wait
```

Нарисовать на черном фоне разноцветные точки со случайной координатой.

Линия, операторы «GO», «TURN», «GOTO», «LINE» команды «NORTH», «HOME»

### Способы рисования линии

Линию можно нарисовать несколькими способами.

Оператор «"go 50"», рисует линию от последней позиции рисования на указанную величину под углом, определенным до этого, по умолчанию угол равен «0», отсчет углов начинается от направления на «12 часов» по часовой стрелке.

Пример:

```
nomainwin  
graphicbox #win.gbox, 10, 10, 150, 150  
open "Drawing" for window as #win  
print #win.gbox, "down"  
print #win.gbox, "size 6"  
print #win.gbox, "set 75 75"  
print #win.gbox, "go 65"  
print #win.gbox, "up"  
print #win.gbox, "flush"  
wait
```

Для того чтобы изменить направление рисования, используют оператор «turn 360», угол задается в градусах, может быть как положительным, так и отрицательным. Оператор устанавливает направление рисования от текущего, то есть если уже был определен угол рисования, например «10», и затем еще раз задается угол, например «15», то суммарный угол составит «25».

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "home" 'перо перемещается в центр
print #win.gbox, "down"
print #win.gbox, "size 6"
print #win.gbox, "turn 30" 'первый угол
print #win.gbox, "go 25"
print #win.gbox, "turn 30" 'второй угол
print #win.gbox, "go 25"
print #win.gbox, "up"
print #win.gbox, "flush"
wait
```

В приведенном примере команда «home», перемещает перо в центр графического окна.

а) Написать программу рисования спирали с использованием цикла.

Для определения направления на «0» используется команда «north».

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, "size 6"
print #win.gbox, "set 75 75"
print #win.gbox, "turn 30"
print #win.gbox, "go 25"
print #win.gbox, "north" 'угол становится равным «0»
print #win.gbox, "go 25"
print #win.gbox, "up"
print #win.gbox, "flush"
wait
```

Оператор «goto 10 100», рисует линию от последней позиции рисования до указанных координат.

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, "size 6"
```

```
print #win.gbox, "set 75 75"  
print #win.gbox, "goto 10 100"  
print #win.gbox, "flush"  
wait
```

Оператор «"line 10 70 90 80"» рисует линию от первой координаты «10 70» до второй координаты «90 80».

Пример:

```
nomainwin  
graphicbox #win.gbox, 10, 10, 150, 150  
open "Drawing" for window as #win  
print #win.gbox, "down"  
print #win.gbox, "size 6"  
print #win.gbox, "line 10 70 140 80"  
print #win.gbox, "flush"  
wait
```

Все параметры графических операторов можно выразить через переменные, как было описано в 4 параграфе II Главы.

б) Написать программу рисования множества линий со случайной координатой и случайным цветом.

Примерный результат выполнения программы:

Прямоугольник, операторы «BOX», «BOXFILLED», «BACKCOLOR», «PLACE»

### Рисование и закрашивание прямоугольника

Оператор «"box 25 52"» рисует прямоугольник от текущей координаты до указанной.

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, "size 4"
print #win.gbox, "set 10 60"
print #win.gbox, "box 140 90"
print #win.gbox, "flush"
wait
```

Для рисования прямоугольника закрашенным используют оператор «"boxfilled 25 52"», цвет закрашивания определяется оператором «"backcolor 255 255 255"»

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
```



```
print #win.gbox, "down"  
print #win.gbox, "size 4"  
print #win.gbox, "set 10 60"  
print #win.gbox, "backcolor 0 255 255"  
'определяем цвет закрашивания  
print #win.gbox, "boxfilled 140 90"  
'рисуем прямоугольник с закрашиванием  
print #win.gbox, "flush"  
wait
```

Для изменения текущей координаты рисования используют оператор «place 30 40».

Пример:

```
nomainwin  
graphicbox #win.gbox, 10, 10, 150, 150  
open "Drawing" for window as #win  
print #win.gbox, "down"  
print #win.gbox, "size 4"  
print #win.gbox, "set 10 60" 'ставим точку  
print #win.gbox, "backcolor 0 255 255"  
print #win.gbox, "boxfilled 140 90"  
print #win.gbox, "place 60 10" 'перемещаем перо  
print #win.gbox, "backcolor 255 255 0"  
print #win.gbox, "boxfilled 90 140"  
print #win.gbox, "flush"  
wait
```

Нарисовать:

график функции « $Y=X^2$ », должны быть ось абсцисс и ось ординат, кривая функции, координатная сетка;

случайный лабиринт, для рисования лабиринта необходимо:

определить область рисования лабиринта;

весь лабиринт условно разбить на квадраты;

в каждом квадрате случайно рисовать или левую сторону условного квадрата или нижнюю сторону или левую и нижнюю сторону условного квадрата вместе или оставлять пустым.

Круг, операторы «CIRCLE», «CIRCLEFILLED»

Рисование и закрашивание круга

Оператор «“circle 255”» рисует круг указанным радиусом в центре с текущей координатой.

Пример:

```
nomainwin
```

```
graphicbox #win.gbox, 10, 10, 150, 150
```

```
open "Drawing" for window as #win
```

```
print #win.gbox, "down"
```

```
print #win.gbox, "size 4"
```

```
print #win.gbox, "place 75 75"
```

```
print #win.gbox, "circle 25"
```

```
print #win.gbox, "flush"
```

wait

Для рисования круга закрашенным используют оператор «circlefilled 25»»

Пример:

```
nomainwin
```

```
graphicbox #win.gbox, 10, 10, 150, 150
```

```
open "Drawing" for window as #win
```

```
print #win.gbox, "down"
```

```
print #win.gbox, "size 4"
```

```
print #win.gbox, "place 75 75"
```

```
print #win.gbox, "backcolor 128 255 128"
```

```
print #win.gbox, "circlefilled 30"
```

```
print #win.gbox, "flush"
```

```
wait
```

Эллипс, операторы «ELLIPSE», «ELLIPSEFILLED»

Рисование и закрашивание эллипса.

Оператор «ellipse 5 20»» рисует эллипс с шириной «5» и высотой «20» в центре с текущей координатой.

Пример:

```
nomainwin
```

```
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, "size 3"
print #win.gbox, "place 75 75"
print #win.gbox, "ellipse 40 70"
print #win.gbox, "ellipse 80 30"
print #win.gbox, "flush"
wait
```

Нарисовать эллипсы с одним центром, последовательно меняя ширину в диапазоне от 5 до 100 и высоту от 100 до 5 с шагом 10.

Для рисования эллипса закрашенным используют оператор «"ellipsefilled 25 33"»

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, fill 200 255 255
print #win.gbox, "size 3"
print #win.gbox, "place 75 75"
print #win.gbox, "backcolor 10 170 30"
print #win.gbox, "ellipsefilled 80 130"
print #win.gbox, "flush"
wait
```

Замечание: если фон графического окна не был определен заранее, то при сворачивании или перекрытии другими окнами фон перекрасится.

Рекомендуется определять фон во избежание его перекрашивания.

Пример:

```
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, "fill 200 255 255" 'определяем фон
print #win.gbox, "size 3"
print #win.gbox, "place 75 75"
print #win.gbox, "backcolor 10 170 30"
print #win.gbox, "ellipsefilled 80 130"
print #win.gbox, "flush"
wait
```

Перехват ввода с клавиатуры, оператор «INKEY\$», команда «WHEN CHARACTERINPUT»

Как говорилось в 15 параграфе I Главы, для перехвата ввода символов с клавиатуры используется оператор «key\$ = Inkey\$», где

«key\$» - символьная переменная, которой присваивается введенная клавиша. Оператор «Inkey\$» работает только в графических окнах или графических подокнах. Для определения, было ли произведено нажатие клавиши, используется команда «print #graph, "when characterInput "», где «#graph» - графическое окно, за которым происходит наблюдение, «when characterInput» - непосредственно команда отслеживания нажатия клавиши, «» - лейбл, куда необходимо перейти при нажатии клавиши.

Пример:

```
print "Для выхода из программы нажмите пробел"  
open "Пример работы Inkey$" for graphics as #graph  
print #graph, "when characterInput "  
wait  
  
key$ = Inkey$  
if key$=" " Then end  
print "Нажата клавиша: "; key$  
wait
```

Для того чтобы приведенный пример работал, необходимо, чтобы графическое окно или подокно было активным.

Приходит программист к другу-пианисту – посмотреть на новый рояль. Долго ходит вокруг, хмыкает, потом заявляет:

— Клава неудобная – всего 84 клавиши, половина функциональных, ни одна не подписана; хотя... шифт нажимать ногой – оригинально.

Приходит программист к другу-пианисту – посмотреть на новый рояль. Долго ходит вокруг, хмыкает, потом заявляет:

— Клава неудобная – всего 84 клавиши, половина функциональных, ни одна не подписана; хотя... шифт нажимать ногой – оригинально.

Приходит программист к другу-пианисту – посмотреть на новый рояль. Долго ходит вокруг, хмыкает, потом заявляет:

— Клава неудобная – всего 84 клавиши, половина функциональных, ни одна не подписана; хотя... шифт нажимать ногой – оригинально.

Приходит программист к другу-пианисту – посмотреть на новый рояль. Долго ходит вокруг, хмыкает, потом заявляет:

— Клава неудобная – всего 84 клавиши, половина функциональных, ни одна не подписана; хотя... шифт нажимать ногой – оригинально.

Реализовать программу рисования при помощи клавиш, где «ц» - рисование вверх, «ы» - рисование вниз, «ф» - рисование влево, «в» - рисование вправо, «к» - определяет белый цвет, «а» - определяет черный цвет, «м» - определяет красный цвет.

Интерфейсные программы

## Введение

Интерфейс позволяет компьютеру в более наглядной форме обмениваться информацией с пользователем. Окно интерфейса, как правило, состоит из стандартного набора элементов, таких как: текст или надпись, кнопки, выпадающие списки, текстовые окна, радиокнопки, кнопки проверки.

Все элементы интерфейса определяются до открытия основного окна.



Текст или надпись, операторы «STATICTEXT», «FONT», команда «DISABLE»

Вывод статического текста

Параметры текста

Для вывода статического текста используют оператор «statictext #win, "Текст", 5, 10, 100, 15», где «statictext» - сам оператор, «#win» - указатель, на каком окне выводить текст, «"Текст"»- непосредственно содержание текста, заключенное в кавычки, «5, 10» - координаты левого верхнего угла, «100, 15» - ширина и высота окна, отведенного для отображения текста.

Пример:

```
nomainwin
```

```
WindowWidth = 300 'устанавливаем ширину окна
```

```
WindowHeight = 150 'устанавливаем высоту окна
```

```
'определяем размер экрана и координаты левого верхнего угла  
окна
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
statictext #win, "Текст", 5, 10, 100, 15
```

```
'определяем текст
```

```
open "Интерфейс" for dialog as #win
```

```
wait
```

Для определения шрифта и размера надписи используют оператор «print #win, "font times\_new\_roman 20 30"», где «font» -

оператор для определения шрифта, «times\_new\_roman» - название шрифта или файла шрифта, в котором все пробелы в названии заменены на подчеркивание, «20 30» - ширина и высота шрифта в «пикселях» (пиксель - элементарная точка отображения на экране компьютера), ширину можно не указывать. Нормально работают шрифты с расширением «.ttf».

Пример:

```
nomainwin
```

```
WindowWidth = 300
```

```
WindowHeight = 150
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
statictext #win, "Текст", 5, 10, 130, 30
```

```
open "Интерфейс" for dialog as #win
```

```
print #win, "font times_new_roman 20 30"
```

```
'устанавливаем тип шрифта и его размер
```

```
wait
```

Если нужно непосредственно определить свойства текстового окна, то можно дать имя объекту текстового окна «statictext #win.ttt, "Текст", 5, 10, 130, 30», где в указателе «#win.ttt» определено имя текстового окна «ttt»

Пример:

```
nomainwin
```

```
WindowWidth = 300
```

```
WindowHeight = 150
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
statictext #win.ttt, "Текст", 5, 10, 130, 30
open "Интерфейс" for dialog as #win
print #win, "font times_new_roman 20 30"
#win.ttt "!Disable"
'устанавливаем текстовое окно не активным
wait
```

а) Реализовать программу вывода одного четверостишья:

У Лукоморья дуб зеленый.

Златая цепь на дубе том.

И днем и ночью кот ученый,

Все ходит по цепи кругом.

Для определения шрифта для каждого элемента меню в отдельности используют оператор «print #win.st1, "!font times\_new\_roman 20 30"», где знак «!» говорит программе, что шрифт применяется только для конкретного элемента интерфейса.

Пример:

```
nomainwin
```

```
WindowWidth = 300
```

```
WindowHeight = 150
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
statictext #win.st1, "", 5, 10, 130, 30
```

```
statictext #win.st2, "", 5, 50, 130, 30
```

```
open "Интерфейс" for dialog as #win
```

```
print #win.st1, "!font times_new_roman 20 30"
```

```
print #win.st1, "Текст"  
print #win.st2, "!font courier_new 10 30"  
print #win.st2, "Текст"  
wait
```

Какие дополнительно операции с объектами интерфейса можно произвести будет рассмотрено позже.

б) Реализовать программу вывода одного четверостишья:

У Лукоморья дуб зеленый.

Златая цепь на дубе том.

И днем и ночью кот ученый,

Все ходит по цепи кругом.

Каждую строчку четверостишья вывести своим шрифтом.

Кнопка, оператор «BUTTON»

Вывод кнопки

Параметры кнопки

Для вывода кнопки используют оператор «button #win.btt, «Вкл», , UL, 7, 10, 30, 15», где «button» - оператор вывода кнопки, «#win.btt» - на кнопку с именем «btt», ««Вкл»» - текст кнопки, «» - лейбл, куда перейти в коде программы при нажатии кнопки, «UL» - команда привязки кнопки к левому верхнему углу основного окна

для отсчета координат кнопки и при изменении его размеров («UR»- верхний правый, «LL»-нижний левый, «LR»-нижний правый), «7, 10» - координаты кнопки, «30, 15» - ширина и высота кнопки.

Пример:

```
nomainwin
```

```
WindowWidth = 230
```

```
WindowHeight = 100
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
statictext #win.ttt, "Текст", 150, 25, 130, 30
```

```
button #win.btt1, "Вкл", , UL, 10, 10, 75, 20
```

```
button #win.btt2, "Выкл", , UL, 10, 30, 75, 20
```

```
open "Кнопки" for dialog as #win
```

```
wait
```

```
print #win.ttt, "Горит" 'меняет текст у надписи
```

```
wait
```

```
print #win.ttt, "Негорит" 'меняет текст у надписи
```

```
wait
```

Реализовать:

программу вывода четверостишья. Первые две строчки появляются при нажатии одной кнопки, третья и четвертая строчки заменяют их при нажатии другой кнопки;

аналогичное задание, но переключение строчек происходит при нажатии одной кнопки.

Текстовое окно, оператор «TEXTBOX», команды «CONTENTS?», «CLS», знак « \ »

Вывод текстового окна

Чтение текста из текстового окна

Для вывода текстового окна используют оператор «textbox #win.ttb, 20, 10, 260, 25», где «textbox» - непосредственно сам оператор, «#win.ttb» - указатель на текстовое окно с именем «ttb», «20, 10» - координаты текстового окна, «260, 25» - ширина и высота окна.

Пример:

```
nomainwin
```

```
WindowWidth = 230
```

```
WindowHeight = 130
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
statictext #win.ttt, "Введите свое имя:", 10,10,96,15
```

```
textbox #win.ttb, 10, 30, 100, 20
```

```
button #win.btt, "OK", , UL, 10, 50, 75, 20
```

```
open "Текстовое окно" for dialog as #win
```

```
wait
```

```
print #win.ttb, "!contents?"
```

```
'читает информацию из текстового окна
```

```
input #win.ttb, Name$
```

'записывает прочтенную информацию

```
print #win.ttb, "Привет ";Name$
```

'Выводит текст в текстовое окно

```
wait
```

То же самое можно написать следующим образом.

Пример:

```
nomainwin
```

```
WindowWidth = 230
```

```
WindowHeight = 130
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
statictext #win.ttt, "Введите свое имя:", 10,10,96,15
```

```
textbox #win.ttb, 10, 30, 100, 20
```

```
button #win.btt, "OK", , UL, 10, 55, 75, 20
```

```
open "Текстовое окно" for window_nf as #win
```

'откроет окно другого типа

```
wait
```

#win.ttb, "!contents? Name\$" 'читает информацию из текстового  
окна в переменную Name\$

```
#win.ttb, "Привет ";Name$
```

'Выводит текст в текстовое окно

```
wait
```

В приведенном примере в последних строчках операторы «print» и «input» не используются.

Вариант текстового окна в графическом окне.

Пример:

```
nomainwin
```

```
WindowWidth = 230
```

```
WindowHeight = 130
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
statictext #win.ttt, "Введите свое имя:", 10,10,120,15
```

```
textbox #win.ttb, 10, 30, 100, 25
```

```
button #win.btt, "OK", , UL, 10, 55, 75, 20
```

```
open "Текстовое окно" for graphics_nf_nsb as #win
```

```
wait
```

```
#win.ttb, "!contents? Name$"
```

```
#win.ttb, "Привет ";Name$
```

```
wait
```

В графическом окне оператор «statictext» в Liberty Basic версии 4.3 жестко не отображается («глючит»), текст виден после обновления окна. В Liberty Basic версии 4.5.1 данная ошибка устранена.

Текст в окне графического типа можно вывести и так.

Пример:

```
nomainwin
```



```

WindowWidth = 230
WindowHeight = 130
UpperLeftX = DisplayWidth/2-150
UpperLeftY = DisplayHeight/2-75
textbox #win.ttb, 10, 30, 100, 25
button #win.btt, "OK", , UL, 10, 55, 75, 20
open "Текстовое окно" for graphics_nf_nsb as #win
#win "place 10 15"
'определяем место, где вывести текст
#win "\Введите свое имя:" 'Сам текст
'Обязательно перед текстом поставить знак «\»
wait

#win.ttb, "!contents? Name$"
#win.ttb, "Привет ";Name$
#win "CLS"
wait

```

Так как в приведенном примере текст является рисунком, то при использовании команды «#win "CLS"» (команда стирает все нарисованные объекты в графических окнах или подокнах) все, что было нарисовано, будет стерто. В графических окнах лучше использовать текстовые окна для вывода текста и надписей.

Реализовать программу:

компьютер спрашивает имя;

после ввода имени и нажатия кнопки компьютер приветствует по имени, заменяя текст первой надписи, и спрашивает возраст;

после ввода возраста компьютер прибавляет к нему случайную величину и выводит текст на место первой надписи «А мне 00 лет.», вместо «00» указывает соответственно полученную величину.

Группировка элементов, «GROUPBOX», команда «SETFOCUS»

В качестве дополнительного оформления для группировки элементов интерфейса используют оператор «groupbox #win.gbox, “name”, 10, 15, 100, 100», где «“name”» заголовок группировки.

Пример:

```
nomainwin
```

```
WindowWidth = 230: WindowHeight = 130
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
groupbox #win.gbox, “Введите данные”, 5, 5, 110, 90
```

```
statictext #win.ttt, “Введите свое имя:”, 10, 30, 96, 15
```

```
statictext #win.ttt, “”, 120, 55, 96, 15
```

```
textbox #win.ttb, 10, 50, 100, 20
```

```
button #win.btt, “OK”, , UL, 10, 70, 75, 20
```

```
open “Текстовое окно” for dialog as #win
```

```
wait
```

```
#win.ttb, “!contents? Name$”
```

```
#win.ttt, “Привет ”;Name$;!”
```

wait

Отметка параметров, оператор «CHECKBOX», команды «SET», «RESET», «VALUE?», «TRAPCLOSE» и «CLOSE»

Для установления каких либо параметров используют оператор «checkbox #win.cbox, “- Текст”, , , 10, 10, 130, 20», где «checkbox» - непосредственно сам оператор, «#win.cbox» - указатель на чекбокс с именем «сbox», «“- Текст”» - надпись чекбокса, «» - лейбл места, куда перейти, если чекбокс отмечен, «» - лейбл места, куда перейти, если у чекбокса отметка убрана, «10, 10» - координаты чекбокса, «130, 20» - ширина и высота.

Пример:

```
nomainwin: WindowWidth = 260: WindowHeight = 160
zncb$="" 'Переменная для хранения значения чекбокса
checkbox #win.cbox, “- Отметьте чекбокс.”, , , 10, 10, 130, 20
button #win, “Поставить отметку”, , UL, 10 , 30, 120, 25
button #win, “Убрать отметку”, , UL, 10, 60, 120, 25
textbox #win.ttb, 10, 90, 180, 24
button #win, “&Ok”, , UL, 200, 90, 40, 25
open “Текстовое окно” for dialog as #win
print #win, "trapclose "
'Куда перейти при нажатии кнопки закрытия окна
wait

print #win.cbox, “set”
'Устанавливает значение чекбокса как отмеченное
```

```
goto
```

```
print #win.cbox, "reset"
```

```
'Устанавливает значение чекбокса как убранное
```

```
goto
```

```
end
```

```
print #win.cbox, "value? zncb$"
```

```
'Читает значение чекбокса
```

```
print #win.ttb, "Чек бокс имеет значение - ";zncb$
```

```
wait
```

```
'Действия по нажатию кнопки закрытия окна
```

```
close #win 'Освобождение памяти от нашего окна
```

```
end
```

В приведенном примере для чтения значения чекбокса используется команда «"value? zncb\$"», значение чекбокса будет храниться в переменной «zncb\$» равное «set» или «reset». Для корректного выхода из программы и высвобождения памяти использована команда «close #win», закрывающая открытое окно «#win».

Реализовать программу:

есть чекбокс установки параметра вывода текста заглавными или прописными буквами;

компьютер запрашивает имя и приветствует по имени;

все надписи должны выводиться в соответствии с параметром чекбокса или заглавными или прописными буквами.

Переключатель «радиокнопка», оператор «RADIOBUTTON»

Для организации переключателей или так называемых радиокнопок используют оператор «radiobutton #cfg.Rus, “Русский”, , , 20, 30, 130, 20», где «radiobutton» - непосредственно сам оператор, «#cfg.Rus» - указатель на радиокнопку с именем «Rus», «“Русский”» - надпись радиокнопки, «» -лейбл места, куда перейти если радиокнопка включена, «» лейбл места, куда перейти, если радиокнопка выключена, «20, 30» - координаты чекбокса, «130, 20» - ширина и высота. Следует отметить, что «» лейбл места, куда перейти, если радиокнопка выключена, не имеет значения, так как радиокнопка при нажатии всегда включена. Все радиокнопки на интерфейсе взаимосвязаны и если одна из них включается, остальные выключаются.

Пример:

```
nomainwin
```

```
WindowWidth = 230
```

```
WindowHeight = 150
```

```
groupbox #cfg.gb1, “Язык:”, 10, 10, 90, 100
```

```
radiobutton #cfg.Rus, “Русский”, , , 20, 30, 130, 20
```

```
radiobutton #cfg.Eng, “English”, , , 20, 60, 130, 20
```

```
open “Радиокнопка” for dialog as #cfg
```

```
print #cfg, "trapclose "
```

```
print #cfg.Rus, “set”
```

```
wait
```

```
'-----
```

```
print #cfg.gb1, "Язык:"
```

```
wait
```

```
'-----
```

```
print #cfg.gb1, "Language:"
```

```
wait
```

```
'-----
```

```
close #cfg
```

```
end
```

```
wait
```

Если необходимо организовать отдельные группы радиокнопок, то такие группы объединяют оператором «groupbox».

Пример:

```
nomainwin
```

```
WindowWidth = 350
```

```
WindowHeight = 150
```

```
'-----
```

```
groupbox #cfg.gb1, "Язык:", 10, 10, 120, 100
```

```
radiobutton #cfg.Rus, "Русский", , , 20, 30, 95, 20
```

```
radiobutton #cfg.Eng, "English", , , 20, 60, 95, 20
```

```
'-----
```

```
groupbox #cfg.gb2, "Шрифт:", 125, 10, 190, 100
```

```
radiobutton #cfg.TNR, "Times New Roman", , , 130, 30, 180, 20
radiobutton #cfg.CN, "Courier New", , , 130, 60, 150, 20
'-----

open "Радиокнопка" for window as #cfg
print #cfg, "trapclose "
wait
print #cfg.gb1, "Язык:"
print #cfg.gb2, "Шрифт:"
wait

print #cfg.gb1, "Language:"
print #cfg.gb2, "Font:"
wait

'-----

print #cfg, "font Times_New_Roman"
wait
'-----

print #cfg, "font Courier_New"
wait
'-----

close #cfg

end
```

Реализовать программу, используя вышеуказанный пример. Компьютер запрашивает имя и приветствует по имени. Все вопросы и ответы выводятся в соответствии с параметрами радиокнопок.

Выбор из списка параметров, операторы «LISTBOX», команда «SELECTION?»

Для вывода вариантов выбора используют оператор «listbox #win.lbox, level\$, , 5, 35, 250, 65», где «listbox» - непосредственно сам оператор, «#win.lbox» - указатель на листбокс с именем «lbox», «level\$()» - массив данных, который выводится в листбоксе, «» - лейбл места, куда перейти при выборе элемента списка, «5, 35» - координаты листбокса, «250, 65» - ширина и высота.

Пример:

```
nomainwin
```

```
WindowWidth = 270
```

```
WindowHeight = 180
```

```
Sel$=""
```

```
level$(0) = "Незаметный"
```

```
level$(1) = "Легкий"
```

```
level$(2) = "Средний"
```

```
level$(3) = "Тяжелый"
```



```
level$(4) = "Невыносимый"
```

```
listbox #win.lbox, level$, , 5, 35, 250, 65
```

```
button #win, "Выбрать", , UL, 5, 5
```

```
statictext #win.tt, "", 10, 110, 270, 15
```

```
open "Работа Листбокса" for window as #win
```

```
wait
```

```
'-----
```

```
#win.lbox, "selection? Sel$"
```

```
#win.tt, "Был выбран " + Sel$ + " уровень."
```

```
wait
```

Следует заметить, что выбор можно подтвердить нажатием кнопки «Выбрать» или двойным кликом «мышки» по выбранному пункту списка. Если весь список не помещается в отведенное пространство под листбокс, то у окна листбокса справа автоматически появляется элемент вертикальной прокрутки списка.

Нарисоватьслучайный лабиринт в графическом подокне. Для рисования лабиринта необходимо:

вывести список уровней сложности лабиринта;

кнопку обновления или перерисовки лабиринта;

определить область рисования лабиринта;

весь лабиринт условно разбить на квадраты;

в каждом квадрате случайно рисовать или левую сторону условного квадрата или нижнюю сторону или левую и нижнюю сторону условного квадрата вместе или оставлять пустым;

вероятность рисования пустого квадрата, квадрата с одной линией или квадрата с двумя линиями определить исходя из уровня сложности выбранного в списке.

Выпадающий список параметров, оператор «COMBOBOX», команда «SELECTINDEX»

Если интерфейс окна не позволяет использовать листбокс, то тогда для выбора параметров используют выпадающий список, так называемый комбобокс, вызываемый оператором «combobox #win.cbox, level\$, , 5, 35, 100, 65», где «combobox» - непосредственно сам оператор, «#win.cbox» - указатель на комбобокс с именем «cbox», «level\$()» - массив данных, который выводится в комбобоксе, «» - лейбл места, куда перейти при выборе элемента списка, «5, 35» - координаты комбобокса, «100, 65» - ширина и высота. Начальный выбор определяется оператором «#win.cbox, "selectindex 2"», где «"selectindex 2"» дает команду поставить начальным параметром второй элемент списка.

Пример:

```
nomainwin
```

```
WindowWidth = 270
```

```
WindowHeight = 180
```

```
Sel$=""
```

```
level$(0) = "Незаметный"  
level$(1) = "Легкий"  
level$(2) = "Средний"  
level$(3) = "Тяжелый"  
level$(4) = "Невыносимый"
```

```
combobox #win.cbox, level$, , 5, 35, 100, 65  
statictext #win.tt, "", 10, 110, 270, 15  
open "Работа Комбобокса" for window as #win  
#win.cbox, "selectindex 2"  
'выбор начального значения комбобокса  
wait  
'-----  
  
#win.cbox, "selection? Sel$"  
#win.tt, "Был выбран " + Sel$ + " уровень."  
wait
```

Выпадающее меню, оператор «POPUPMENU», команда «WHEN RIGHTBUTTONUP», знаки «&», «\_» и «;»

Выпадающее меню

Переход по клику правой кнопки манипулятора «мышь»

Объединение нескольких команд и операторов

Разбитие одной строки на несколько строк

Для вывода выпадающего меню используют оператор «`popupmenu "&Пятиконечная спираль", , "&Квадратная спираль", , "&Треугольная спираль", »`, где через запятую перечисляются названия пунктов меню и лейблы мест, куда необходимо перейти при выборе пункта меню. Названия пунктов меню заключаются в кавычки. Знак «&» говорит программе, что следующая за этим знаком буква является буквой быстрого выбора, то есть можно, к примеру, не выбирать первый пункт меню «“&Пятиконечная спираль”», а нажать клавишу с буквой «п», что равносильно выбору этого пункта.

Для вызова выпадающего меню добавлена команда «`#geo, "when rightButtonUp "`», которая при нажатии правой кнопки манипулятора «мышь» переходит по лейбле «»

Пример:

```
nomainwin
```

```
WindowWidth = 270
```

```
WindowHeight = 270
```

```
open "Гео - узор" for graphics_nsb as #geo
```

```
#geo, "when rightButtonUp "
```

```
wait
```

```
'-----
```

```
popupmenu "&Пятиконечная спираль", ,_
```

```
"&Квадратная спираль", ,_
```

```
"&Треугольная спираль",
```

```
wait
```

```
'-----
```

```
#geo, "cls; home; down; color darkgreen"  
for x = 1 to 120  
#geo, "go ";x;"; turn 69"  
next x  
wait  
'-----
```

```
#geo, "cls; home; down; color red"  
for x = 1 to 120  
#geo, "go ";x;"; turn 87"  
next x  
wait  
'-----
```

```
#geo, "cls; home; down; color blue"  
for x = 1 to 120  
#geo, "go ";x;"; turn 117"  
next x  
wait
```

В приведенном примере использован особый стиль написания строки:

```
поруптмену "&Пятиконечная спираль", ,_  
"&Квадратная спираль", ,_  
"&Треугольная спираль",
```

— данная строка разбита на три части, знак «\_» в конце незаконченных строк говорит программе, что продолжение в

следующей строке. Данный стиль применен для лучшего восприятия программного кода, а также для того, чтобы видеть всю строку.

Следует также обратить внимание на строку:

```
#geo, "cls; home; down; color darkgreen"
```

— в данной строке три графические команды и один оператор объединены в одну строку и разделены знаком «;», что также сделано для компактного написания программного кода.

Реализовать задание из 7 параграфа III главы при этом обновление лабиринта и выбор уровня сложности вынести в выпадающее меню.

Меню в заголовке, оператор «MENU», знак «|»

Для вывода выпадающего меню используют оператор «menu #geo, "&Заголовок", "&Подменю 1", ,|, "П&одменю 2", », где «menu» сам оператор, «#geo» указатель на окно, «"&Заголовок"» название раздела меню, «"&Подменю 1"» название подраздела меню, «» лейбл куда необходимо перейти при выборе данного подраздела, знак «|» создает горизонтальную линию для отделения различных подразделов меню.

Пример:

```
nomainwin
```

```
WindowWidth = 270
```

```
WindowHeight = 270
```

```
menu #geo, "Тип &спирали", _  
"&Пятиконечная спираль", , _  
"&Квадратная спираль", ,|, _  
"&Треугольная спираль",
```

```
open "Гео - узор" for graphics_nsb as #geo  
wait
```

```
#geo, "cls; home; down; color darkgreen"  
for x = 1 to 120  
#geo, "go ";x;"; turn 69"  
next x  
wait  
'-----
```

```
#geo, "cls; home; down; color red"  
for x = 1 to 120  
#geo, "go ";x;"; turn 87"  
next x  
wait  
'-----
```

```
#geo, "cls; home; down; color blue"  
for x = 1 to 120  
#geo, "go ";x;"; turn 117"  
next x  
wait
```

Реализовать игру «Хоккей»:

имеется игровое поле размером с графическое подокно;

игровое поле ограничено сверху и снизу бортами;

правую сторону поля защищает хоккеист, управляемый компьютером в виде прямоугольника;

левую сторону поля защищает пользователь;

по полю движется шайба в виде закрашенного круга;

шайба отскакивает от бортов и от хоккеистов;

если шайба проходит в левую сторону за игровое поле, то засчитывается балл компьютеру, если шайба проходит в правую сторону за игровое поле, то засчитывается балл пользователю;

после гола игра начинается снова;

набранные балы должны быть отображены на экране.

Управление вратарем происходит нажатием выбранных клавиш на клавиатуре. Для проверки нажатия клавиш по мимо команды «when characterInput» использовать команду «scan» в коде анимации движения шайбы. Подробнее о команде «scan» изложено в 10 параграфе V Главы.

Работа с файлами

Введение



Работа с файлами в Liberty Basic подразделяется на несколько способов. Первый способ - это непосредственно работа с графическими файлами для оформления кнопок, отображения рисунков и проигрывания звуковых файлов, второй способ - это непосредственно создание и редактирование файлов, а также доступ к оборудованию.

Отображение нарисованной кнопки, операторы «BMPBUTTON» и «LOADBMP», команда «BITMAP»

Для отображения рисунка вместо кнопки используют оператор «bmpbutton #win.btt1, "bmp\on.bmp", , UL, 10, 20», где «bmpbutton» - сам оператор, «#win.btt1» - указатель на нарисованную кнопку, «"bmp\on.bmp"» - путь к файлу рисунка, «» лейбл места, куда перейти при нажатии кнопки.

Следует убедиться, что файл рисунка существует по указанному пути. Путь можно указывать полный или как в нижеприведенном примере от места запуска программы. Программа, без дополнительных команд, обрабатывает файлы рисунков в формате «.bmp».

Пример:

```
nomainwin
```

```
WindowWidth = 230: WindowHeight = 100
```

```
statictext #win.ttt, "Текст", 150, 25, 130, 30
```

```
bmpbutton #win.btt1, "bmp\on.bmp", , UL, 10, 20
```

```
bmpbutton #win.btt2, "bmp\off.bmp", , UL, 70, 20
```

```
open "Рисованные Кнопки" for dialog as #win
```

```
wait
```

```
print #win.ttt, "Горит"
```

```
wait
```

```
print #win.ttt, "Не горит"
```

```
wait
```

Для смены рисунка кнопки, необходимо этот рисунок сперва загрузить оператором «loadbmp "Vkl", "bmp\light.bmp"», где «loadbmp» -сам оператор, «Vkl» -указатель рисунка, «"bmp\light.bmp"» -путь к файлу рисунка. Для смены или перерисовки рисунка используют команду «print #win.btt1, "bitmap Vkl"», где «#win.btt1» - имя объекта, который будет перерисовываться, «Vkl» - имя рисунка, который будет перерисовывать объект.

Пример:

```
nomainwin
```

```
WindowWidth = 230
```

```
WindowHeight = 100
```

```
VV=0 'Метка включения
```

```
loadbmp "Vkl", "bmp\light.bmp"
```

```
loadbmp "Vikl", "bmp\dark.bmp"
```

```
bmpbutton #win.btt1, "bmp\on.bmp", , UL, 10, 20
```

```
open "Кнопки" for window as #win
```

```
wait
```

```
if VV=00 Then
```

```
print #win.btt1, "bitmap Vkl"
```

```
VV=1
Else
print #win.btt1, "bitmap Vikl"
VV=0
End If
wait
```

Для перехода к открывшемуся окну или тому или иному активному элементу интерфейса (кнопка, текстовое окно, радиопереключатели и т.д.) без участия манипулятора «мышь», используют команду «print #main, "setfocus"» - если необходимо перейти к окну, или «print #main.button1, "!setfocus"» - если необходимо перейти к элементу окна. После перехода на активный элемент интерфейса с ним можно взаимодействовать при помощи клавиатуры нажатием клавиш «Enter», «Пробел» и других.

Пример:

```
nomainwin
WindowWidth = 248
WindowHeight = 175
loadbmp "arrow", "bmp\arrwbbtn.bmp"
loadbmp "blue", "bmp\bluebbtn.bmp"
bmpbutton #main.button1, "bmp\blank4.bmp", , UL, 22, 11
bmpbutton #main.button2, "bmp\blank4.bmp", , UL, 22, 46
open "Нарисованные кнопки" for window as #main
```

```
print #main, "trapclose "
```

```
wait
```

```
'Переключение на вторую кнопку
```

```
print #main.button2, "setfocus"
```

```
print #main.button2, "bitmap arrow"
```

```
print #main.button1, "bitmap blue"
```

```
wait
```

```
'Переключение на первую кнопку
```

```
print #main.button1, "setfocus"
```

```
print #main.button1, "bitmap arrow"
```

```
print #main.button2, "bitmap blue"
```

```
wait
```

```
close #main
```

```
end
```

Нарисовать кнопку в виде круга, при нажатии которой кнопка меняется на треугольник и обратно.

Отображение рисунка в формате «.bmp», оператор «DRAWBMP», команда «REDRAW»

Для простого вывода рисунка в формате «.bmp» используют оператор «print #main, "drawbmp Cr 15 35"», где «#main» указатель

на объект, на котором будет нарисован рисунок, «drawbmp» сам оператор, «Cr» имя предварительно загруженного рисунка, «15 35» координаты места вывода рисунка.

Пример:

```
nomainwin
```

```
WindowWidth = 248
```

```
WindowHeight = 175
```

```
loadbmp "Cr", "bmp\CHERRY.bmp"
```

```
open "Отображение рисунка" for graphics_nf_nsb as #main
```

```
print #main, "trapclose "
```

```
print #main, "drawbmp Cr 15 35"
```

```
print #main, "flush"
```

```
wait
```

```
close #main
```

```
end
```

!Оператор «drawbmp» работает только в графическом окне или подокне.

Вот примеры вывода рисунка в графических подокнах.

Пример:

```
nomainwin
```

```
WindowWidth = 248
```

```
WindowHeight = 175
```

```
loadbmp "Cr", "bmp\CHERRY.bmp"
```

```
graphicbox #main.gb, 0, 0, 230, 165
```

```
open "Отображение рисунка" for window as #main
```

```
print #main, "trapclose "  
print #main.gb, "drawbmp Cr 15 35"  
print #main.gb, "flush"  
print #main.gb, " redraw" 'для появления рисунка  
wait  
  
close #main  
end
```

Пример:

```
nomainwin  
WindowWidth = 248  
WindowHeight = 175  
loadbmp "Cr", "bmp\CHERRY.bmp"  
graphicbox #main.gb, 0, 0, 230, 165  
open "Отображение рисунка" for dialog as #main  
print #main, "trapclose "  
print #main.gb, "drawbmp Cr 15 35"  
print #main.gb, "flush"  
print #main.gb, " redraw" 'для появления рисунка  
wait  
  
close #main  
end
```

Следует заметить, что в связи с «глюком» компилятора Liberty Basic 4.03 рисунок в графическом подокне сразу не виден, поэтому

необходимо обновить окно или использовать команду «redraw» для перерисовки графических элементов закрепленных до этого командой «flush». В Liberty Basic 4.5.1 такого «глюка» не наблюдалось.

Нарисовать шагающего человечка, который будет шагать при смене картинок. Ограничиться 3-5 периодически повторяющимися картинками шагающего человечка.

Проигрывание звукового файла в формате «.wav», оператор «PLAYWAVE»

Для проигрывания звукового файла в формате «.wav» используют оператор «playwave "media\beer.wav"», где «playwave» сам оператор, «"media\beer.wav"» путь к звуковому файлу от нашей программы. Может проигрываться только один звуковой файл. Имеется четыре режима проигрывания звукового файла:

«playwave "media\beer.wav"» - режим когда: запускается звуковой файл, выполнение программы приостанавливается, файл проигрывается до тех пор когда не закончится;

«playwave "media\beer.wav", async» - режим когда: запускается звуковой файл, продолжается выполнение программы, файл проигрывается до тех пор когда не закончится или не запустится другой звуковой файл;

«playwave "media\beer.wav", loop» - режим когда: запускается звуковой файл, продолжается выполнение программы, файл проигрывается повторяясь бесконечно до тех пор пока не запустится другой звуковой файл;

«playwave ""» - режим остановки проигрывания звукового файла.

Пример:

```
nomainwin
```

```
WindowWidth = 200
```

```
WindowHeight = 150
```

```
button #win, "Играть звук 1", , UL, 5, 5
```

```
button #win, "Играть звук 2", , UL, 5, 35
```

```
button #win, "Стоп звук", , UL, 5, 65
```

```
open "Звук" for dialog as #win
```

```
print #win, "trapclose "
```

```
wait
```

```
playwave "media\beep.wav", loop
```

```
wait
```

```
playwave "media\bump.wav", loop
```

```
wait
```

```
playwave ""
```

```
wait
```

```
playwave ""
```

```
close #win
```

```
end
```

Убедительно рекомендуется использовать команды «trapclose» и «close» упомянутыми в 5 параграфе III Главы.



Реализовать маленькую программу плеер. Программа должна выводить имя открытого «.wav» файла, запускать проигрывание и останавливать проигрывание.

Проигрывание звукового файла в формате «.mid», операторы «PLAYMIDI» И «TIMER», команды «MIDIPOS()» и «STOPMIDI»

Для проигрывания звукового файла в формате «.mid» используют оператор «playmidi “media\theme.mid”, length», где «playmidi» сам оператор, «“media\theme.mid”» путь к звуковому файлу, «length» имя переменной в которую считывается специфическая длина звукового файла. Требуется периодически проверять достигнут ли конец миди файла командой «midipos()», и если конец файла достигнут то остановить проигрывание командой «stopmidi».

Пример:

```
nomainwin: WindowWidth = 200: WindowHeight = 150
```

```
button #win, "Играть миди", , UL, 5, 5
```

```
open "Звук" for dialog as #win
```

```
print #win, "trapclose "
```

```
wait
```

```
playmidi "media\theme.mid", length
```

```
timer 1000,
```

```
wait
```

```
if length = midipos() then stopmidi: timer 0
```

```
wait
```

```
stopmidi
```

```
timer 0 'Оператор «timer 0» можно не использовать если  
программу завершить командой «end»
```

```
close #win
```

```
wait
```

В данном примере периодическая проверка конца миди файла реализована через оператор «timer 1000, », где «timer» сам оператор, который периодически переходит по лейблу, «1000» время периодичности перехода в миллисекундах, «» лейбл, куда необходимо периодически переходить. Если таймер необходимо отключить, то используют команду «timer 0».

!Команды «stopmidi» и «midipos()» вызывают ошибку, если были вызваны до запуска миди файла.

Реализовать маленькую программу плеер. Программа должна выводить имя открытого «.wav» файла, запускать проигрывание и останавливать проигрывание, иметь счетчик в секундах, который запускается и останавливается при запуске и остановке проигрывания файла пользователем.

Переименование файла, оператор «NAME..AS..»

Для переименования файла используют оператор «name “file1.txt” as “text2.bas”», где «name» сам оператор, «“file1.txt”» имя исходного файла для переименования, «“text2.bas”» новое имя файла.

Пример:

```
name "error.log" as "wrong.txt"  
print "файл error.log переименован в wrong.txt"
```

Удаление файла, оператор «KILL»

Для удаления файла используют оператор «kill “file1.txt”», где «kill» сам оператор, «“file1.txt”» имя удаляемого файла.

Пример:

```
kill "wrong.txt"  
print "файл wrong.txt удален"
```

!Оператором «kill» необходимо пользоваться осторожно, удаленные файлы не попадают в «корзину».

Доступ к файлу только для чтения, режим «INPUT», команды «EOF()», «LINE INPUT», «INPUT» и «INPUT\$()»

Для работы с файлом используют оператор который мы использовали для создания интерфейса, но с другими параметрами

«open filename\$ for input as #text», где «open» - оператор для открытия файла, «filename\$» - переменная в которой находится путь к файлу и имя файла, «input» - режим доступа к файлу последовательный, только для чтения, «#text» - указатель на файл.

Чтение информации из файла производится оператором «line input #text, item\$», где «line input» сам оператор который читает информацию из файла последовательно, построчно, «#text» указатель на файл, «item\$» строковая переменная куда сохраняются прочитанные данные.

Пример:

```
nomainwin
```

```
WindowWidth = 350
```

```
WindowHeight = 360
```

```
filename$="letter.txt"
```

```
open filename$ for input as #text
```

```
texteditor #win.te 15,15,300,300
```

```
open "Чтение файла" for window as #win
```

```
print #win, "trapclose "
```

```
'читает файл построчно пока не достигнет конца
```

```
if eof(#text) <> 0 then
```

```
line input #text, item$ 'читает файл построчно
```

```
print #win.te, item$
```

```
goto
```

```
close #text
print #win.te, "Конец файла."
wait
```

```
close #win
end
```

В данном примере конец файла перед его чтением проверяет команда «eof(#text)», в результате проверки которая возвращает «-1» если был достигнут конец файла или «0» если конец файла еще не достигнут. Следует заметить, что команда ищет специальную метку в конце файла «EOF» - End of File, которая есть у каждого файла. Если попытаться прочесть файл после достижения его конца, то программа выдаст ошибку.

Если использовать оператор «input #text, item\$», то чтение файла происходит до запятой или до конца строки.

Пример:

```
nomainwin: WindowWidth = 350: WindowHeight = 360
filename$="letter.txt"
open filename$ for input as #text
texteditor #win.te 15,15,300,300
open "Чтение файла" for window as #win
print #win, "trapclose "
```

```
'читает файл до запятых построчно пока не достигнет конца
if eof(#text) <> 0 then
```

```
input #text, item$
'читает до запятой или до конца строки
print #win.te, item$
goto

close #text
print #win.te, "Конец файла."
wait

close #win
end
```

При использовании оператора «input\$(#text,1)» чтение файла происходит по одному символу.

Пример:

```
nomainwin
WindowWidth = 350
WindowHeight = 360

filename$="letter.txt"
open filename$ for input as #text

texteditor #win.te 15,15,300,300
open "Чтение файла" for window as #win
print #win, "trapclose "
```

```

'читает файл по 1 символу пока не достигнет конца
if eof(#text) <> 0 then
item$=input$(#text,1) 'читает по 1 символу
print #win.te, item$
goto

close #text
print #win.te, "Конец файла."
wait

close #win
end

```

При использовании оператора «inputto\$(#text, " ")» чтение файла происходит до указанного символа или до конца строки.

Пример:

```
nomainwin
```

```
WindowWidth = 350: WindowHeight = 360
```

```
filename$="letter.txt"
```

```
open filename$ for input as #text
```

```
texteditor #win.te 15,15,300,300
```

```
open "Чтение файла" for window as #win
```

```
print #win, "trapclose "
```

```
'читает файл до пробелов порциями или до конца строки
if eof(#text) <> 0 then
item$=inputto$(#text, " ")
'читает до пробела или до конца строки
print #win.te, item$
goto

close #text
print #win.te, "Конец файла."
wait

close #win
end
```

Реализовать программу чтения текстового файла по символно.  
Все буквы кириллицы заменить буквами латиницы и вывести  
результат на экран.

Разговор двух админов:

— Слушай, я давно не писал ручкой. Ты не знаешь, где у нее  
можно поменять раскладку с русской на английскую?

Разговор двух админов:

— Слушай, я давно не писал ручкой. Ты не знаешь, где у нее  
можно поменять раскладку с русской на английскую?

Разговор двух админов:



— Слушай, я давно не писал ручкой. Ты не знаешь, где у нее можно поменять раскладку с русской на английскую?

Разговор двух админов:

— Слушай, я давно не писал ручкой. Ты не знаешь, где у нее можно поменять раскладку с русской на английскую?

Создание файла и запись данных, режимы «OUTPUT» и «APPEND», команда «LOF()»

Для создания или открытия файла для записи в него данных используют оператор «open “test.txt” for output as #text», где «open» оператор для открытия файла, «“test.txt”» имя создаваемого файла, «output» режим создания (если файл не существует) или открытия для последовательной записи.

Пример:

```
Nomainwin
```

```
WindowWidth = 250
```

```
WindowHeight = 100
```

```
filename$="test.txt"
```

```
open filename$ for output as #text
```

```
textbox #win.tb 15,15,150,25
```

```
button #win.bb, "Записать", , UL,165,15,60,25
```

```
open "Запись в файл" for window as #win
```

```
print #win, "trapclose "
```

```
wait
```

```
#win.tb, "!contents? item$"
```

```
print #text, item$
```

```
wait
```

```
close #text
```

```
close #win
```

```
end
```

!Следует отметить, что данный режим доступа к файлу, если файл существует, полностью переписывает его содержимое и ставит новую отметку окончания файла.

Если файл открыть в режиме «append», то данные будут дописываться в конец файла.

Пример:

```
nomainwin
```

```
WindowWidth = 250
```

```
WindowHeight = 100
```

```
filename$="test.txt"
```

```
open filename$ for append as #text
```

```
textbox #win.tb 15,15,150,25
```

```
button #win.bb, "Записать", , UL,165,15,60,25
```

```
open "Запись в файл" for window as #win
```

```
print #win, "trapclose "
```

```
wait
```

```
#win.tb, "!contents? item$"
```

```
print #text, item$
```

```
wait
```

```
close #text
```

```
close #win
```

```
end
```

На вопрос о родившихся четырех близняшках, программист машинально ответил:

— Это резервные копии!

На вопрос о родившихся четырех близняшках, программист машинально ответил:

— Это резервные копии!

На вопрос о родившихся четырех близняшках, программист машинально ответил:

— Это резервные копии!

На вопрос о родившихся четырех близняшках, программист машинально ответил:

— Это резервные копии!

Команда «lof(#file)» возвращает длину файла в байтах. Это может пригодиться например для копирования файла.

Пример:

```
open "test.txt" for input as #text
open "test2.txt" for output as #text2

length=lof(#text)
item$=input$(#text,length)
print #text2, item$

close #text
close #text2
end
```

Реализовать программу конвертации кодировки кириллицы текстового файла из OEM/DOS в ASCII 256 и обратно. Шрифт для проверки результата использовать «Terminal» и «Arial». Текст выводить в два разных окна до и после конвертации.

Произвольная работа с файлом, режим «BINARY», оператор «SEEK», команда «LOK()»

Для работы с файлом в любом его месте используют режим доступа «open “test.txt” for binary as #text», где «binary» непосредственно сам режим доступа.

Для перехода в определенное место для чтения данных используют оператор «seek #text, mesto», где «seek» сам оператор для перехода в определенное место, «#text» указатель на файл, «mesto» место в файле в байтах после которого будет произведено чтение или запись данных. Если оператор «seek» превышает текущую длину файла, то оператор вызывает ошибку.

Для определения текущего места в файле после чтения или записи данных используют команду «loc(#text)», которая возвращает текущую позицию в файле в байтах.

Чтение и запись данных производятся операторами, описанными в 5 параграфе IV Главы.

Пример:

'создание пустого файла или стирание данных из существующего файла

```
open "myfile.bin" for output as #myfile
```

```
close #myfile
```

'открытие файла в режиме произвольного доступа

```
open "myfile.bin" for binary as #myfile
```

```
txt$ = "Четыре черненьких, чумазеньких, чертенка."
```

```
print "Исходный текст для записи в файл: ";txt$
```

'запись текста в файл

```
print #myfile, txt$
```

'возвращение текущей позиции после записи текста

```
newPos = LOC(#myfile)
```

'перемещение на новую позицию

```
newPos = newPos - 9
```

```
seek #myfile, newPos
```

'чтение данных с текущей позиции

```
line input #myfile, txt$
```

'вывод прочитанных данных

```
print "Данные прочитанные с позиции ";newPos;": ";txt$
```

'перемещение в файле на конкретное место

```
seek #myfile, 7
```

'запись новых данных поверх имеющихся

```
print #myfile, "беленьких, "
```

```
print (loc(#myfile) - 7);" байтов данных записано поверх  
имеющихся."
```

'перемещение в начало файла

```
seek #myfile, 0
```

'чтение данных в строке

```
line input #myfile, txt$
```

'вывод данных

```
print "Новые данные: ";txt$
```

'перемещение на 6 байт от конца файла

```
seek #myfile, lof(#myfile)-6
```

```
'до запись данных в файл
print "Дозапись данных"
print #myfile, "епашки нинзя."
seek #myfile, 0
line input #myfile, txt$
print "Новые данные: ";txt$

'обязательное закрытие файла
close #myfile
end
```

!Следует заметить, что специальные символы «chr\$(13)» - конец строки и «chr\$(10)» - переход на следующую строку, считаются оператором «seek» и командой «loc()» наравне с остальными символами. Данные символы ставятся при нажатии кнопки «Enter» на клавиатуре.

Пример:

```
open "myfile.bin" for output as #myfile
close #myfile
open "myfile.bin" for binary as #myfile
txt$ = "Четыре черненьких, чумазеньких, чертенка."
print "Исходный текст для записи в файл: ";txt$
print #myfile, txt$
seek #myfile, 7
'запись новых данных поверх имеющихся
```

```
print #myfile, "беленьких,"+chr$(13)+chr$(10)
print (loc(#myfile) - 7); " байтов данных записано по верх
имеющихся."
```

```
seek #myfile, 0
line input #myfile, txt$
print "Новые данные: ";txt$
line input #myfile, txt$
print " ";txt$
```

```
seek #myfile, lof(#myfile)-6
```

```
print "Дозапись данных"
print #myfile, "епашки нинзя."
```

```
seek #myfile, 0
line input #myfile, txt$
print "Новые данные: ";txt$
line input #myfile, txt$
print " ";txt$
```

```
close #myfile
end
```

Реализовать программу поиска и замены в текстовом файле пробела « » на подчеркивание «\_» и буквы «а» на символ «@».



## Сохранение рисунка, операторы «BMPSAVE» и «GETBMP»

Для сохранения рисунка в формате «.bmp» используют оператор «bmpsave “pic”, “spiral.bmp”», где «bmpsave» сам оператор, «“pic”» указатель на графический файл или выделеную графическую область, «“spiral.bmp”» имя файла для сохранения, если файл уже существует, то он будет переписан. Если неуказан путь сохранения файла, то графический файл будет сохранен в папку, откуда была запущена программа.

Звонит программист беременной жене:

— Ты где?

— На сохранении.

— Ну ладно, позвонишь, когда сохранишься.

Звонит программист беременной жене:

— Ты где?

— На сохранении.

— Ну ладно, позвонишь, когда сохранишься.

Звонит программист беременной жене:

— Ты где?

— На сохранении.

— Ну ладно, позвонишь, когда сохранишься.

Звонит программист беременной жене:

— Ты где?

— На сохранении.

— Ну ладно, позвонишь, когда сохранишься.

Для выделения графической области используют оператор «print #geo, "getbmp pic -50 -75 350 350"», где «#geo» графическое окно относительно которого будут отсчитываться координаты, «getbmp» непосредственно сам оператор захвата графической области, «pic» указатель на захваченную графическую область, «-50 -75» координаты начала захвата графической области относительно графического окна, «350 350» ширина и высота захватываемой графической области.

Пример:

```
nomainwin
```

```
WindowWidth = 270:WindowHeight = 270
```

```
menu #geo, "Тип &спирали", _
```

```
"&Пятиконечная спираль", , _
```

```
"&Квадратная спираль", ,|, _
```

```
"&Треугольная спираль",
```

```
menu #geo, "&Сохранить", "&Сохранить",
```

```
open "Гео - узор" for graphics_nsb as #geo
```

```
wait
```

```
#geo, "cls; home; down; color darkgreen"
```

```
for x = 1 to 120
```

```
#geo, "go ";x;" ; turn 69"
```

```
next x
```

```
wait
```

```
'-----
```

```
#geo, "cls; home; down; color red"
```

```
for x = 1 to 120
```

```
#geo, "go ";x;" ; turn 87"
```

```
next x
```

```
wait
```

```
'-----
```

```
#geo, "cls; home; down; color blue"
```

```
for x = 1 to 120
```

```
#geo, "go ";x;" ; turn 117"
```

```
next x
```

```
wait
```

```
'-----
```

```
print #geo, "getbmp pic -50 -75 350 350"
```

```
bmpsave "pic", "spiral.bmp"
```

```
notice "Файл spiral.bmp сохранен."
```

```
wait
```

Вот сохраненный файл «spiral.bmp»

Следует обратить внимание, что захват графической области произошел, включая и изображение, не входящее в графическое

окно, то есть произошел захват части общей картинке, отображаемой монитором.

Реализовать программу получения 4-х частей картинке экрана, перестановка их местами и вывод их в графическое подокно диалогового окна типа «dialog\_popup» на весь экран.

Работа со стандартными диалоговыми окнами и с системой

## Введение

В Liberty Basic есть возможность вызывать ряд стандартных диалоговых окон для упрощения взаимодействия программы и пользователя, также есть ряд команд и операторов для обмена информацией с системой такой как системное время, положение манипулятора «мышь», нажатие кнопок на клавиатуры или на манипуляторе «мышь», с различными устройствами к примеру принтером и т.д.

Диалоговое окно выбора цвета, оператор «COLORDIALOG»

Liberty Basic может вызывать стандартные диалоговые окна.

Для вызова диалогового окна выбора цвета используют оператор «colordialog "255 0 0", chosen\$» или «colordialog "red", chosen\$», где «colordialog» - сам оператор, «"255 0 0"» или «"red"» - предварительно выбранный цвет, если предварительно выбранный цвет пустой, то будет выбран черный цвет, «chosen\$» - переменная в

которой будет находиться выбранный цвет в виде строки «255 255 0 yellow» если цвет стандартный, или в виде строки «255 120 13» если выбран оттенок цвета. Переменная «chosen\$» будет равна пустой строке «”» если будет выбрана отмена выбора цвета.

Пример:

```
nomainwin
```

```
WindowWidth = 270
```

```
WindowHeight = 270
```

```
ch$="green"
```

```
menu #geo, "Тип &спирали", _
```

```
"&Пятиконечная спираль", , _
```

```
"&Квадратная спираль", ,|, _
```

```
"&Треугольная спираль",
```

```
menu #geo, "&Цвет", "&Цвет",
```

```
open "Гео - узор" for graphics_nsb as #geo
```

```
wait
```

```
#geo, "cls; home; down; color ";ch$
```

```
for x = 1 to 120
```

```
#geo, "go ";x;"; turn 69"
```

```
next x
```

```
wait
```

```
'-----
```

```
#geo, "cls; home; down; color ";ch$
```

```
for x = 1 to 120
```

```
#geo, "go ";x;"; turn 87"
```

```

next x
wait
'-----

#geo, "cls; home; down; color ";ch$
for x = 1 to 120
#geo, "go ";x; "; turn 117"
next x
wait
'-----

colordialog "red", ch$
if ch$="" then ch$="0 0 0"
wait

```

В указанном примере для чистоты кода необходимо было бы в переменную «ch\$» выделить цвет только в виде чисел «255 255 0» без названия стандартного цвета «255 255 0 yellow», например как в нежеуказанном примере.

Пример:

```

nomainwin
WindowWidth = 270
WindowHeight = 270
ch$="green"
menu #geo, "&Цвет", "&Цвет",
open "Гео - узор" for graphics_nsb as #geo

```

```
#geo, "cls; home; down; color ";ch$
```

```
for x = 1 to 120
```

```
#geo, "go ";x;"; turn 69"
```

```
next x
```

```
wait
```

```
'-----
```

```
colordialog "red", ch$
```

```
if ch$="" then ch$="0 0 0"
```

```
p1=instr(ch$," ")
```

```
'поиск места расположения первого пробела
```

```
p2=instr(ch$," ",p1+1)
```

```
'поиск места расположения второго пробела
```

```
p3=instr(ch$," ",p2+1)-1
```

```
'поиск места расположения до третьего пробела
```

```
if p3<=0 then p3=len(ch$) 'если третьего пробела нет, то  
берется длина всей строки
```

```
ch$=mid$(ch$,1,p3)
```

```
'из исходной строки вырезается цвет выраженный числами
```

```
goto
```

Реализовать программу как указанную выше. В выпадающем меню указать «Цвет рисунка», «Цвет фона», «Толщина линии».

Диалоговое окно выбора шрифта, оператор «FONTDIALOG»

Для вызова диалогового окна выбора шрифта используют оператор «fontdialog “times\_new\_roman 10”, chosenFont\$», где «fontdialog» сам оператор, «“times\_new\_roman 10”» предварительно выбранный шрифт, «chosenFont\$» переменная где будет храниться выбранный шрифт со всеми параметрами такими как размер шрифта и способ начертания.

Пример:

```
nomainwin
```

```
WindowWidth = 300
```

```
WindowHeight = 150
```

```
chosenFont$="times_new_roman 10"
```

```
button #win.bt, "Выбор шрифта", UL, 5, 5, 100, 25
```

```
statictext #win.st1, "", 5, 30, 290, 100
```

```
open "Шрифт" for dialog as #win
```

```
print #win, "trapclose "
```

```
print #win.st1, "!font times_new_roman 10"
```

```
print #win.st1, "Образец ШРИФТА."
```

```
wait
```

```
fontdialog chosenFont$, chosenFont$
```

```
if chosenFont$ <> "" then
```

```
print #win.st1, "!font ";chosenFont$
```

```
print #win.st1, "Образец ШРИФТА."
```

```
end if
```

```
wait
```

```
close #win
```

```
end
```



Диалоговое окно открытия и сохранения файла, оператор «FILEDIALOG», команда «!CLS»

Для вызова диалогового окна открытия или сохранения файла используют оператор «filedialog “Открыть файл...”, “\*.txt;\*.bas”, filename\$», где «filedialog» сам оператор, «“Открыть файл...”» заголовок диалогового окна, «“\*.txt;\*.bas”» перечисление расширения файлов которые будут отображаться, «filename\$» переменная в которой будет храниться полный путь и название выбранного файла.

Пример:

```
nomainwin
```

```
WindowWidth = 375
```

```
WindowHeight = 370
```

```
'-----
```

```
textbox #win.tb 5,5,250,25
```

```
button #win.bb “Выбрать”,,UL,255,5,55,25
```

```
button #win.bb “Открыть”,,UL,5,30,55,25
```

```
texteditor #win.te 5,60,300,260
```

```
open “Чтение файла” for window as #win
```

```
print #win, "trapclose "
```

```
wait
```

```
'-----
```

```
print #win.tb, “!contents? filename$”
```

```

filedialog "Открыть файл...", "*.txt", filename$
print #win.tb, filename$
wait
'-----

if filename$="" Then filedialog "Открыть файл...", "*.txt", filename
$

print #win.tb, filename$
if filename$="" Then wait
open filename$ for input as #text
print #win.te, "!cls" 'очищает окно texteditor
print #win.te, "!contents #text"
close #text
wait
'-----

close #win
end

```

Что бы вызвать диалоговое окно для сохранения файла необходимо наличие слова «save» в заголовке диалогового окна «filedialog "(save)Сохранить как...", "\*.txt;\*.bas", filename\$».

Пример:

```

nomainwin: WindowWidth = 375:WindowHeight = 370
textbox #win.tb 5,5,250,25
button #win.bb "Выбрать",,UL,255,5,55,25

```

```
button #win.bb "Открыть",,UL,5,30,55,25
```

```
button #win.bb "Сохранить",,UL,65,30,65,25
```

```
texteditor #win.te 5,60,300,260
```

```
open "Чтение файла" for window as #win
```

```
print #win, "trapclose "
```

```
wait
```

```
print #win.tb, "!contents? filename$"
```

```
filedialog "Выбрать файл...", "*.txt", filename$
```

```
print #win.tb, filename$
```

```
wait
```

```
print #win.tb, "!contents? filename$"
```

```
if filename$="" Then filedialog "Открыть файл...", "*.txt", filename
```

```
$: print #win.tb, filename$
```

```
print #win.tb, "!contents? filename$"
```

```
if filename$="" Then wait
```

```
open filename$ for input as #text
```

```
print #win.te, "!cls" 'очищает окно texteditor
```

```
print #win.te, "!contents #text"
```

```
close #text
```

```
wait
```

```
print #win.tb, "!contents? filename$"
```

```
filedialog "(save)Сохранить как...", "*.txt", filename$: print
```

```
#win.tb, filename$
```

```
print #win.tb, "!contents? filename$"
```

```
if filename$="" Then wait
```

```
open filename$ for output as #text
print #win.te, "!contents? item$"
print #text, item$
close #text

wait

close #win
end
```

Окно сообщения, оператор «NOTICE»

Для вывода окна простого сообщения используют оператор «notice "Строка сообщения"», где «notice» сам оператор, «"Строка сообщения"» текст самого сообщения.

Пример:

```
nomainwin
notice "Вот в таком виде выходит сообщение."
```

Для вывода своего заголовка и нескольких строк сообщения используют символ переход на следующую строку «CHR\$(13)» в качестве разделителя в виде «notice "Заголовок сообщения" + CHR

$\$(13) + \text{“Первая строка сообщения.”} + \text{CHR}\$(13) + \text{“Вторая строка сообщения.”}$ »

Пример:

```
nomainwin
```

```
notice “Заголовок сообщения”+CHR$(13)+_
```

```
“Первая строка сообщения.”+CHR$(13)+_
```

```
“Вторая строка сообщения.”
```

Реализовать программу открытия текстового файла и вывода названия файла в заголовок сообщения, а первые 30 строк текста файла в строки сообщения.

Окно выбора, оператор «CONFIRM»

Для вывода окна выбора используют оператор «confirm “Вопрос?»; answer\$», где «confirm» сам оператор, «“Вопрос?”» текст окна выбора, «answer\$» переменная в которой будет находиться результат ответа, может принимать значения «yes» или «no».

Пример:

```
nomainwin
```

```
confirm “Ответте на вопрос.”+CHR$(13)+_
```

```
“У вас есть машина?»; answer$
```

```
If answer$=“yes” Then
```

```
notice “Вы водитель.”
```

```
Else
```

```
notice “Вы пешеход.”
```

```
End If
```

## Окно ввода строки, оператор «PROMPT»

Для вывода окна ввода строки используют оператор «prompt "Заголовок" + CHR\$(13) + "Текст"; answer\$», где «prompt» сам оператор, «"Заголовок"» заголовок окна ввода строки, «"Текст"» текст окна ввода строки, «answer\$» переменная значение которой сперва выходит в текстовый блок и в которой будет храниться содержимое текстового блока после закрытия окна ввода строки.

Пример:

```
nomainwin
response$ = "Аноним"
prompt "Запрос имени" + chr$(13) + "Введите свое имя:";
response$
notice "Здравствуйте ";response$
```

Печать текстовых данных через принтер, оператор «LPRINT», команда «DUMP»

Для печати текстовых данных через принтер используют оператор «lprint text\$;"Аноним ";3+2;"=";C», где «lprint» сам оператор отправляющий данные не на монитор, а на принтер, «text \$;"Аноним ";3+2;"=";C» различные текстовые данные. Для завершения печати необходимо применить команду «dump».

Пример:

```
lprint text$;"Аноним ";3+2;"=";C
dump
```

Для определения шрифта печати используют специальную переменную «PrinterFont\$», которая содержит название и параметры шрифта печати.

Пример:

```
PrinterFont$="Times_New_Roman 18 italic"
lprint text$;"Аноним ";3+2;"=";C
dump
```

Окно печати, оператор «PRINTERDIALOG»

Для вывода окна печати используют оператор «printerdialog». Параметры печати хранятся в специальных переменных:

«PrinterName\$» - содержит имя принтера выбранного для печати, если значение не определено, то будет выбран принтер по умолчанию;

«PrintCopies» - содержит количество печатаемых копий, если значение не определено, то будет напечатан один экземпляр. Не все принтеры поддерживают печать копий в этом случае необходимо печать копий вызвать программно;

«PrintCollate» - содержит общее количество экземпляров включая первый, обычно параметр равен «0», а его значения трансформируются в «PrintCopies»;

«PrinterFont\$» - содержит название и параметры шрифта печати.

После закрытия диалогового окна печати для вывода текстовой информации на принтер используют оператор «lprint».

Пример:

```
'=====выбор файла для печати
filedialog "Печать BAS файла", "*.bas", file$
'- - - - -проверка, был ли выбран файл для печати
if file$ = "" then end
printerdialog
'вызов диалогового окна определения параметров печати
'=====параметры печати
print "Имя принтера - ";PrinterName$
print "Кол-во дополнительных копий - ";PrintCopies
print "Общее кол-во экземпляров - ";PrintCollate
PrinterFont$= "courier_new 14"
print "Шрифт печати - ";PrinterFont$
'- - - - -проверка, был ли определен принтер
if PrinterName$ = "" then end
'=====открытие файла для чтения
open file$ for input as #text

if not(eof(#text)) then 'проверка конца файла
line input #text, line$ 'чтение файла по строчно
lprint line$ 'отправление строки на принтер
```



```
goto  
end if  
close #text  
dump  
end
```

### Печать графики, оператор «PRINT»

Для вывода на печать содержимого графических окон или подокон используют оператор «print» или «print », где «» не имеет значения. Печатаемая страница масштабируется в соотношении к разрешению ширины монитора, то есть если разрешение ширины монитора 1024 пикселей, то и на страницу в ширину помещается 1024 пикселя, если разрешение ширины монитора 1366 пикселей, то и ширина страницы будет 1366 пикселе. Если нарисованная картинка в размере больше чем разрешение монитора, то часть картинки не будет напечатана.

При указанном способе печати, команда «dump» не требуется.

Пример:

```
nomainwin: WindowWidth = 800: WindowHeight = 600  
loadbmp "Cr", "bmp\Orchid.bmp"  
open "Отображение рисунка" for graphics_nf_nsb as #main  
print #main, "trapclose "  
print #main, "drawbmp Cr 0 0"
```

```
print #main, "flush"  
print #main, "print" 'печатает содержимое графического окна с  
указателем #main  
wait  
  
close #main  
end
```

Исходная картинка с шириной 1600 пикселей.

Напечатанная страница, равная ширине монитора 1280 пикселей.

Отображенная часть картинки в графическом окне шириной 800 пикселей.

Следует заметить, что на печать выходит даже не отображенная часть графического окна (подокна).

Есть три предустановленных режима ширины печатаемой страницы:

«print vga» - ширина страницы равна 600 пикселей;

«print svga» - ширина страницы равна 800 пикселей;

«print xga» - ширина страницы равна 1024 пикселя.

Пример:

```
nomainwin
```

```
loadbmp "Cr", "bmp\Orchid.bmp"
```

```

open "Отображение рисунка" for graphics_nf_nsb as #main
print #main, "trapclose "
print #main, "drawbmp Cr 0 0"
print #main, "flush"
print #main, "print vga" 'печатает на страницу шириной 600
пикселей
wait

close #main
end

```

Перехват действий пользователя без остановки программы,  
команда «SCAN»

Для перехвата действий пользователя без остановки  
выполнения программы используют команду «scan» которая  
проверяет действия пользователя с манипулятором «мышь» и  
нажатием клавиатуры или кнопок.

Пример:

```

nomainwin
WindowWidth = 300: WindowHeight = 150
statictext #win.st1, "", 5, 10, 135, 65
statictext #win.st2, "", 5, 75, 130, 60
open " Перехват действий пользователя" for graphics as #win
print #win, "setfocus"
print #win, "when characterInput "
print #win, "trapclose "
x=0.5

```

```
'-----
```

```
k$=Inkey$
```

```
print #win.st2, "Клавиша:";k$
```

```
'-----
```

```
x=1+x
```

```
print #win.st1, "Нахождение коэффициента золотого сечения:";x
```

```
x=1/x
```

```
print #win.st1, "Нахождение коэффициента золотого сечения:";x
```

scan 'Здесь программа проверяет не совершил ли пользователь каких бы то ни было действий, в данном случае закрытие программы или нажатие клавиши

```
goto
```

```
'-----
```

```
close #win
```

```
end
```

В вышеуказанном примере, в процессе вычисления коэффициента «золотого сечения», пользователь может нажимать клавиши на клавиатуре и программа будет их считывать не прерывая вычислений.

!Если программу запустить без команды «scan», то она заикнется и не будет реагировать на действия пользователя, тогда ее придется выгружать из «Диспетчера задач».

На основе выше указанного примера реализовать программу расчета числа «Пи» по формуле «», перехват нажатия клавиш не нужен, только клик на закрытие программы.

Дата, оператор «DATE\$()»

Для получения текущей даты используют оператор «date\$()». Данный оператор имеет следующие форматы запросов:

date\$() - возвращает строку с текущей датой в виде «Jul 26, 2017», где «Jul» английское сокращение текущего месяца, «26» текущая дата, «2017» текущий год;

date\$("mm/dd/yyyy") - возвращает строку с текущей датой в виде «07/26/2017», где числовые выражения текущего месяца, дня и года разделены знаком «/»;

date\$("mm/dd/yy") - возвращает строку с текущей датой в виде «07/26/17», где числовые выражения текущего месяца, дня и года разделены знаком «/», год выражен последними двумя цифрами;

date\$("yyyy/mm/dd") - возвращает строку с текущей датой в виде «2017/07/26», где числовые выражения текущего года, месяца и дня разделены знаком «/»;

date\$("days") - возвращает число с количеством дней от 1 января 1901 года до текущей системной даты, если значение будет присвоено строковой переменной «d\$=date\$("days")», то оператор сам конвертирует значение в строковой формат;

date\$("7/30/2017") - возвращает число «42579» с количеством дней от 1 января 1901 года до указанной даты в формате: первое число месяц, второе число день и третье число год, если значение

будет присвоено строковой переменной, то оператор сам конвертирует значение в строковой формат;

`date$("42575")` - возвращает строку с датой «07/26/2017» в формате: первое число месяц, второе число день, третье число год, полученную путем прибавления к 1 января 1901 года указанного количества дней, можно использовать и отрицательное значение количества дней, в этом случае дни будут вычитаться.

Пример:

```
print date$() 'Jul 26, 2017
```

```
print date$("mm/dd/yyyy") '07/26/2017
```

```
print date$("mm/dd/yy") '07/26/17
```

```
print date$("yyyy/mm/dd") '2017/07/26
```

```
print date$("days")
```

```
'42575 дня от 1.01.1901 до текущей даты
```

```
print date$("7/30/2017") '42579 дня от 1.07.2017 до 30.07.2017
```

```
print date$(42575) '07/26/2017
```

```
Jul 26, 2017
```

```
07/26/2017
```

```
07/26/17
```

```
2017/07/26
```

```
42575
```

```
42579
```

```
07/26/2017
```

Реализовать расчет дня недели введенной даты.

Время, оператор «`TIME$()`»

Для получения текущего времени используют оператор «time\$()». Данный оператор имеет следующие форматы запросов:

time\$()- возвращает строку с информацией о текущем времени в виде «16:11:44», где часы, минуты и секунды разделены знаком «:»;

time\$("seconds") - возвращает число количества секунд прошедших с полуночи «32314», если переменная является строковой, то оператор автоматически переведет данные в строковой вид;

time\$("milliseconds") или time\$("ms") - возвращает число количества миллисекунд прошедших с полуночи «33221342», если переменная является строковой, то оператор автоматически переведет данные в строковой вид.

Пример:

```
print time$()
print time$("seconds")
print time$("ms")
```

При помощи оператора «time\$()» удобно вставлять паузы в программу.

Пример:

```
print "Старт "; time$() ;":"; time$("ms") - time$("seconds")*1000
a=time$("ms")
```

```
If time$("ms")<a+15 Then goto
```

```
print "Стоп "; time$() ;":"; time$("ms") - time$("seconds")*1000
```

Реализовать программу расчета времени потраченного на вычисление числа «Пи» с точностью до 4-го знака после запятой.

Запуск сторонней программы, оператор «RUN»

Для запуска сторонней программы используют оператор «run "NOTEPAD " + chr\$(34) + "D:\Liberty BASIC v4.5.1\letter.TXT" + chr\$(34), MINIMIZE». Работа оператора аналогична работе окна «Выполнить...», вызываемого нажатием кнопки «Пуск». Формат оператора следующий:

run - непосредственно сам оператор;

"NOTEPAD " - название программы, если это системная программа, то путь до программы и расширение исполняемого файла «.exe» можно не писать;

+ chr\$(34) + "D:\Liberty BASIC v4.5.1\letter.TXT" + chr\$(34) - дополнительная команда для запускаемой программы в формате командной строки. Смысл в том, что необходимо сформировать строку с кавычками в которой указан путь и название открываемой программой файла. Так как кавычки являются служебным символом, то они указываются ANCIИ кодом «chr\$(34)». Если файл расположен там же где и наша программа, то полный путь до файла писать необязательно;

MINIMIZE - дополнительная команда открытия программы может не указываться. В Liberty Basic есть следующие команды открытия программ:

HIDE - программа запустится скрыто, без отображения окна, саму программу можно увидеть в «Диспетчере задач»;



SHOWNORMAL - (режим по умолчанию) программа запустится в обычном режиме само окно будет активным;

SHOWMINIMIZED - программа запустится свернутой;

SHOWMAXIMIZED - программа запустится развернутой на весь экран;

SHOWNOACTIVE - программа запустится обычно, но окно программы активным не будет;

SHOW - программа запустится в обычном режиме само окно будет активным;

MINIMIZE - программа запустится свернутой;

SHOWMINNOACTIVE - программа запустится свернутой и неактивной;

SHOWNA - программа запустится обычно, но окно программы активным не будет;

RESTORE - программа запустится в обычном режиме, само окно будет активным.

Пример:

```
notmainwin
```

```
run "NOTEPAD " + chr$(34) + "letter.TXT" + chr$(34),  
SHOWNOACTIVE
```

Реализовать программу открытия файла «letter.txt» в программе «Winword.exe».

Работа с манипулятором «мышь», команды «WHEN ...» переменные «MouseX» и «MouseY»

Для перехвата нажатия кнопок и движение манипулятора «мышь» используют следующий вид оператора «#w, "when leftButtonDown "», где «#w» указатель графического окна или подокна, «when leftButtonDown» оператор перехвата нажатия или движения манипулятора «мышь», «» лейбл куда необходимо перейти при нажатии кнопок или движении манипулятора «мышь». Для получения текущих координат имеются предопределенные переменные «MouseX» для получения координаты «X» и «MouseY» для получения координаты «Y».

Имеются следующие операторы работы с манипулятором «МЫШЬ»:

«#w, "when leftButtonDown "» - при однократном нажатии левой кнопки манипулятора «мышь», произойдет переход по лейбле «»;

«#w, "when leftButtonMove "» - при движении манипулятора «мышь» с нажатой левой кнопкой, произойдет переход по лейбле «»;

«#w, "when rightButtonDown "» - при однократном нажатии правой кнопки манипулятора «мышь», произойдет переход по лейбле «»;

«#w, "when rightButtonMove "» - при движении манипулятора «мышь» с нажатой правой кнопкой, произойдет переход по лейбле «»;

«#w, "when leftButtonUp "» - при отпускании нажатой левой кнопки манипулятора «мышь», произойдет переход по лейбле «»;

«#w, "when rightButtonUp "» - при отпускании нажатой правой кнопки манипулятора «мышь», произойдет переход по лейбле «»;

«#w, "when leftButtonDouble "» - при двойном нажатии левой кнопки манипулятора «мышь», произойдет переход по лейбле «»;

«#w, "when rightButtonDouble "» - при двойном нажатии правой кнопки манипулятора «мышь», произойдет переход по лейбле «»;

«#w, "when middleButtonDown "» - при однократном нажатии средней кнопки манипулятора «мышь», произойдет переход по лейбле «»;

«#w, "when middleButtonUp "» - при отпускании нажатой средней кнопки манипулятора «мышь», произойдет переход по лейбле «»;

«#w, "when middleButtonDouble "» - при двойном нажатии средней кнопки манипулятора «мышь», произойдет переход по лейбле «»;

«#w, "when mouseMove "» - при движении манипулятора «мышь» без нажатых кнопок, произойдет переход по лейбле «».

Пример:

```
nomainwin
```

```
open "Простое рисование" for graphics as #w
```

```
print #w, "when rightButtonMove "
```

```
print #w, "when leftButtonMove "
```

```
print #w, "when rightButtonDown "
```

```
print #w, "when leftButtonDown "
```

```
print #w, "down; size 3"
```

```
wait
```

```
print #w, "color red"
```

```
print #w, "set ";MouseX;" ";MouseY
```

```
wait
```

```
print #w, "color black"
```

```
print #w, "set ";MouseX;" ";MouseY
```

```
wait
```

```
print #w, "place ";MouseX;" ";MouseY
```

```
print #w, "circle 10"
```

```
wait
```

```
print #w, "cls"
```

wait

Перехват сообщений об ошибке, оператор «ON ERROR», команда «RESUME», переменные «Err\$» и «Err»

Для перехвата ошибок которые могут произойти в программе без прерывания выполнения самой программы используют оператор «on error goto », где «on error» сам оператор, «» лейбл места куда перейти при возникновении ошибки, команда «resume» возвращает в то место где возникла ошибка для продолжения выполнения программы. Переменная «Err» содержит номер, а переменная «Err\$» содержит текст ошибки, некоторые ошибки не содержат номера и описания, тогда «Err» и «Err\$» остаются пустыми.

!В программе может быть только один оператор «on error goto ». Некоторые ошибки оператором не перехватываются, например операторы и команды работы с миди файлами.

Пример:

```
on error goto
```

```
input "Введите А и В для C=SQR(A)/B :";A,B
```

```
C=SQR(A)
```

```
C=C/B
```

```
print "C=";C
```

```
goto
```

```
If A<0 Then print Err$;" , ошибка №";Err:input "Введите A>=0 :";A
If B=0 Then print Err$;" , ошибка №";Err:input "Введите B<>0 :";B
resume
Введите A и B для C=SQR(A)/B :-1
??0
Float invalid operation, ошибка №0
Введите A>=0 :2
Float invalid operation, ошибка №0
Введите B<>0 :0
Division by zero, ошибка №11
Введите B<>0 :-3
C=-0.47140452
```

Реализовать программу последовательного чтения файла без проверки на наличие метки «EOF». При достижении конца файла программа должна перехватить сообщение об ошибке и корректно завершить чтение файла.

Создание исполняемого автономного файла написанной программы и иконки

Для создания автономного исполняемого файла написанной программы необходимо выполнить следующий порядок действий:

Предположим есть текст программы.

Пример:

```
notmainwin
```

```
WindowWidth = 248
```

```
WindowHeight = 175
loadbmp "Cr", "bmp\CHERRY.bmp"
open "Отображение рисунка" for graphics_nf_nsb as #main
print #main, "trapclose "
print #main, "drawbmp Cr 15 35"
print #main, "flush"
wait

close #main
end
```

В любом месте создать папку, где будет находиться наша программа со всеми необходимыми дополнительными файлами, имя папки желательно написать латинскими буквами, для этого:

в меню редактора нажимаем «File», в выпавшем меню выбираем «Save As...»;

в выбранной папке на свободном месте кликаем правой кнопкой «мышки», в выпавшем меню выбираем «Создать» и «Папку»;

У созданной папки пишем имя.

Открываем созданную папку и сохраняем написанный код программы, имя программы желательно писать латинскими буквами.

Для создания исполняемого файла необходимо выбрать «Run», затем «Create Application».

В появившемся окне нажать кнопку «Browse».

Открываем ранее созданную папку, пишем имя программы, желательно латинскими буквами и нажимаем «Сохранить».



Далее нажимаем кнопку «Create».

В результате в папке должны появиться нижеуказанные файлы, где:

Show pic.bas - текст программы;

Show pic.exe - интерпретатор (данный файл на основании своего имени ищет файл, в данном случае «Show pic.tkn», и запускает его) (интерпретатор - программа которая запускает текст программы и выполняет её, читая построчно);

Show pic.tkn - предкомпилированный код программы для интерпретатора (компилятор - программа создающая из текста написанной программы полноценный исполняемый файл, предкомпилятор - программа создающая из текста написанной программы специальный файл для интерпретатора для ускорения его чтения);

vbas31w.sll, vgui31w.sll, voflr31w.sll, vthk31w.dll, vtk1631w.dll, vtk3231w.dll, vvm31w.dll, vvmt31w.dll - файлы необходимые для работы интерпретатора.

Если программа использует дополнительные файлы для своей работы, то их нужно скопировать в соответствующие папки. В примере, программа обращается к файлу «CHERRY.bmp», находящемуся в папке «bmp».

В папке с программой создаем подпапку, для этого на свободном месте кликаем правой кнопкой «мышки», в выпавшем меню выбираем «Создать» и «Папку».

Переименовываем появившуюся папку в «bmp».

Копируем в вышеуказанную папку требуемый файл.

В результате в папке с программой имеются все необходимые файлы для ее работы. Данную папку можно переносить в желаемое место или на другой компьютер и работоспособность программы не будет нарушена.

Для придания программе индивидуальности можно создать оригинальную «иконку» - миниатюрную картинку для исполняемого файла.

Для создания иконки необходимо в меню выбрать «Setup», «Icon Editor».

Откроется окно создания иконки и предупреждающая надпись. На предупреждающей надписи нажать кнопку «ОК».

Инструмент «Mask» определяет прозрачный цвет изображения. После подготовки иконки сохраняем ее. Для этого выбираем в меню «File», «Save Icon ...».

Тов. Сталин: Хочу что бы на моём рабочем столе были иконки, несмотря на советскую власть. Хочу иконки! На рабочем столе. И мышью хочу открывать окна и не только открывать, но ещё и закрывать.

Тов. Берия: -Товарищ Сталин. Что вы курите? Как можно мышью открывать окна — это не возможно...

Тов. Сталин: -Расстрелять!

Тов. Сталин: Хочу что бы на моём рабочем столе были иконки, несмотря на советскую власть. Хочу иконки! На рабочем столе. И мышью хочу открывать окна и не только открывать, но ещё и закрывать.

Тов. Берия: -Товарищ Сталин. Что вы курите? Как можно мышью открывать окна — это не возможно...

Тов. Сталин: -Расстрелять!

Тов. Сталин: Хочу что бы на моём рабочем столе были иконки, несмотря на советскую власть. Хочу иконки! На рабочем столе. И мышью хочу открывать окна и не только открывать, но ещё и закрывать.

Тов. Берия: -Товарищ Сталин. Что вы курите? Как можно мышью открывать окна — это не возможно...

Тов. Сталин: -Расстрелять!

Тов. Сталин: Хочу что бы на моём рабочем столе были иконки, несмотря на советскую власть. Хочу иконки! На рабочем столе. И мышью хочу открывать окна и не только открывать, но ещё и закрывать.

Тов. Берия: -Товарищ Сталин. Что вы курите? Как можно мышью открывать окна — это не возможно...

Тов. Сталин: -Расстрелять!

Открываем папку с программой, указываем имя иконки, желательно латинскими буквами и нажимаем кнопку «Сохранить».

Если имеется готовая иконка, то можно использовать ее. Для этого в меню выбрать «File», «Open Icon ...».

Выбрать необходимую иконку и нажать кнопку «Открыть».

Для внедрения иконки в исполняемый файл необходимо в меню выбрать «File», «Save To Runtime EXE»

Выбрать необходимый исполняемый файл и нажать кнопку «Открыть».

После этого исполняемый файл будет иметь индивидуальное изображение.

Создать автономный исполняемый файл и оригинальную «иконку» для игры «Хоккей» из 10 параграфа III Главы.

Создание исполняемого автономного файла в конвертере «LBB» - Liberty Basic Booster версии 3.10

Расселлом Р.Т. (англ. Russell R.T.) был создан конвертер кода программ Liberty Basic в чистый исполняемый файл, который не требует для своей работы дополнительных библиотек, при этом скорость выполнения программ возрастает в 10-16 раз.

Для создания исполняемого файла в конвертере «LBB» необходимо:

К примеру, есть код программы расчета числа «Фибоначи».

Пример:

```
input "n=";n
tst=time$("ms")
print fib(n)
print (time$("ms")-tst)/1000;" c"
goto
end
'-----
function fib(n)
if n<2 Then fib=1 else fib=fib(n-1)+fib(n-2)
end function
n=29
832040
20.093 c
```

Сохраняем код программы в отдельную папку.

Открываем конвертер «LBB».

Открываем сохраненный код программы в Liberty Basic.

Выбираем ранее сохраненный код программы.

Нажимаем кнопку «Make Executable», для создания исполняемого файла.

Указываем имя создаваемой программы и при необходимости меняем иконку.

Указываем файл с необходимым значком.

Сохраняем программу с выбранной иконкой.

Создается единственный исполняемый файл.

В результате выполнения созданной программы затраченное время равно 1.625 с., что в 12 раз быстрее чем до конвертации.

!Следует учитывать, что при конвертации поддержка больших чисел и длинной арифметики теряется. Могут возникнуть и другие незначительные вопросы совместимости.

По заявлениям создателя конвертера «LVB», включена поддержка практических всех команд и операторов в плодь до версии 4.5.0.

.

## Приложение

### Варианты решения задач

#### Глава I

3.

а) =====

```
Imya$="Алишер"
```

```
DataRojdeniya$="01.01.2017 г."
```

```
Rost=168
```

```
Ves=70
```

```
Print "Имя:";Imya$;", Дата рождения:";DataRojdeniya$; ",  
Рост:";Rost;"см. Вес:";Ves;"кг."
```

```
Print "Имя:";Imya$
```

```
Print "Дата рождения:";DataRojdeniya$
```

```
Print "Рост:";Rost;"см."
```

```
Print "Вес:";Ves;"кг."
```

б) =====

```
print "1-блок,";
```

```
print "2-блок,"
```

```
print "3-блок,";
```

```
print "4-блок,"
```

1-блок,2-блок, 3-блок,4-блок.



Если строка оканчивается на знак «;», то следующая строка печатается сразу же за ней, если строка оканчивается на «,», то следующая строка печатается в этой же строке с отступом в 10 пробелов.

4.

a) =====

m1=70000

m2=5.9726\*10<sup>27</sup>

R=6371000

F=6.74\*10<sup>(-11)</sup>\*m1\*m2/R<sup>2</sup>

print "F=";F;" Ньютон"

б) =====

R1=10

R2=300

R=(R1\*R2)/(R1+R2)

Print "R=";R;" Ом"

в) =====

x=123.321

y=x-int(x)

print "y=";y

г) =====

print "2<sup>2</sup>=";2<sup>2</sup>

print "2<sup>(-2)</sup>=";2<sup>(-2)</sup>

print "2<sup>0</sup>=";2<sup>0</sup>

5. =====

```

input "Введите ваше имя:";Imya$
print "Здравствуйтесь ";Imya$
print "Для нахождения силы притяжения, введите:"
input "вашу массу тела в граммах:";m1
input "массу Земли в граммах:";m2
input "расстояние от центра Земли в метрах:";R
F=6.74*10^(-11)*m1*m2/R^2
print "F=";F;" Ньютон"

```

6.

```

a) =====
input "Введите целое положительное число:";x1
input "Введите дробное положительное число:";x2
a1$=Str$(x1)
a2$=Str$(x2)
a3$=a1$+a2$
al=len(a3$)
y=Val(a3$)
print "y=";y

```

```

б) =====
input "Введите один символ:";A$
B$=Chr$(Asc(A$)-32)
print "B$=";B$

```

```

7. =====
Input "Введите первое число x1=";x1
Input "Введите второе число x2=";x2
If x1>x2 Then Print "x1=";x1;" больше x2=";x2
If x1<x2 Then Print "x1=";x1;" меньше x2=";x2
If x1=x2 Then Print "x1=";x1;" равно x2=";x2

```

8. =====

```
print "Нахождение корней уравнения  $ax^2+bx+c=0$ "
```

```
input "Введите a=";a
```

```
input "Введите b=";b
```

```
input "Введите c=";c
```

```
D=b^2-4*a*c
```

```
if D>0 Then print "x1=";((-1)*b+sqr(D))/(2*a);" x2="; ((-1)*b-sqr  
(D))/(2*a)
```

```
if D=0 Then print "x1=x2=";(-1)*b/(2*a)
```

```
if D<0 Then print "Уравнение корней не имеет"
```

9.

a) =====

```
input "Введите целое, положительное число:";x
```

```
y=1
```

```
For i=1 To x
```

```
y=y*i
```

```
Next i
```

```
print "Факториал ";x;"!=";y
```

б) =====

```
For i=32 To 255
```

```
print i;"-";chr$(i)
```

```
Next i
```

10. =====

```
print "Введите комбинацию спорт лото"
```

```
input "Сколько?";m
```

```
input "Из сколько?";n
if m>n Then goto
x=n
gosub
x1=y
x=m
gosub
x2=y
x=(n-m)
gosub
x3=y
C=x1/(x2*x3)
print m;" из ";n;" имеет ";C;" комбинаций"
end
```

'-----

```
y=1
For i=1 To x
y=y*i
Next i
return
```

12. =====

```
chislo=int(rnd(1)*100+1)
print "Угадайте число от 1 до 100"

input x
if x<chislo Then print "больше":goto
```

```
if x>chislo Then print "меньше":goto
print "Угадал!"
input "Еще раз? (да-1/нет-2)";x
if x=1 Then goto
```

13. =====

```
input "Введите число:";x
Y=(x<0)*(-1)+(x>0)
print "Y=f(";x;")=";Y
```

14. =====

```
input "Введите дробное число:";x
A$=str$(x)
r=instr(A$,".")
if r=0 Then print "Число целое.":goto
B$=mid$(A$,r+1,len(A$)-r)
print B$
```

17.

a) =====

```
input "Введите число от 20 до 30:";x
if x>=20 and x<=30 Then
print "Число в диапазоне"
else
print "Число вне диаппаза"
goto
end if
```

б) =====

```
input "Введите число:";x
```

```
if (x mod 2)=0 Then
```

```
print "Четное"
```

```
else
```

```
print "Не четное"
```

```
end if
```

```
goto
```

```
в) =====
```

```
input "Введите число менее 20 или более 30:";x
```

```
if x<20 or x>30 Then
```

```
print "Верно"
```

```
else
```

```
print "Не верно"
```

```
goto
```

```
end if
```

```
19. =====
```

```
Dim ch(15)
```

```
For i=1 to 15
```

```
ch(i)=Int(Rnd(1)*36+1)
```

```
Print ch(i);"/";
```

```
Next i
```

```
20.
```

```
а) =====
```

```
Dim ch(15)
```

```
For i=1 to 15
```

```

x=Int(Rnd(1)*15+1)
if ch(x)>0 and ch(x)<16 then goto
ch(x)=i
Next i
For i=1 to 15
print "ch(";i;")=";ch(i)
Next i
6)=====
Dim ch(15)
For i=1 to 15
ch(i)=i
Next i
For i=1 to 14
x=Int(Rnd(1)*(16-i)+i)
b=ch(i)
ch(i)=ch(x)
ch(x)=b
Next i
For i=1 to 15
print "ch(";i;")=";ch(i)
Next i
For i=1 To 14
For a=i+1 To 15
If ch(i)>ch(a) Then
b=ch(i)
ch(i)=ch(a)
ch(a)=b
End If

```

```
Next a
Next i
print ""
For i=1 to 15
print "ch(";i;")=";ch(i)
Next i
```

21.

a) =====

```
n=5000
```

```
fib=0
```

```
fib1=1
```

```
fib2=1
```

```
For i=1 To n
```

```
fib1=fib
```

```
fib=fib1+fib2
```

```
fib2=fib1
```

```
Next i
```

```
Print fib
```

б) =====

```
print "Нахождение корней уравнения  $ax^2+bx+c=0$ "
```

```
input "Введите a=";a
```

```
input "Введите b=";b
```

```
input "Введите c=";c
```

```
call Uravneniye a, b, c
```

```
goto
```

```
end
```

'=====



```
Sub Uravneniye a, b, c
```

```
D=b^2-4*a*c
```

```
if D>0 Then
```

```
print "x1=";((-1)*b+sqr(D))/(2*a)
```

```
print "x2=";((-1)*b-sqr(D))/(2*a)
```

```
End If
```

```
if D=0 Then print "x1=x2=";(-1)*b/(2*a)
```

```
if D<0 Then print "Уравнение корней не имеет"
```

```
End Sub
```

```
Глава II
```

```
1. =====
```

```
nomainwin
```

```
WindowWidth = DisplayWidth/2
```

```
WindowHeight = DisplayHeight/2
```

```
UpperLeftX = DisplayWidth/2- WindowWidth/2
```

```
UpperLeftY = DisplayHeight/2- WindowHeight/2
```

```
open "Окно" for graphics_nf_nsb as #wingraf
```

```
wait
```

```
2. =====
```

```
nomainwin
```

```
Sh = DisplayWidth
```

```
Vi = DisplayHeight
```

```
graphicbox #win.gbox, 0, 0, Sh, Vi
```

```

open "Drawing" for dialog_fs as #win
wait
3. =====
nomainwin
WindowWidth = 640
WindowHeight = 480
UpperLeftX = DisplayWidth/2- WindowWidth/2
UpperLeftY = DisplayHeight/2- WindowHeight/2
graphicbox #win.gbox, 10, 10, 600, 400
open "Kpyr" for window as #win
print #win.gbox, "down"
print #win.gbox, "size 2"
X1=300
Y1=200
R=150
For a=0 To 6.283 step 0.02
X=X1+R*Cos(a)
Y=Y1+R*Sin(a)
print #win.gbox, "set ";X;" ";Y
Next a
print #win.gbox, "up"
print #win.gbox, "flush"
wait
4. =====
nomainwin
WindowWidth = 640
WindowHeight = 480
UpperLeftX = DisplayWidth/2- WindowWidth/2

```

```

UpperLeftY = DisplayHeight/2- WindowHeight/2
graphicbox #win.gbox, 10, 10, 600, 400
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, "size 3"
print #win.gbox, "fill 0 0 0"
For i=1 to 3000
R=Int(Rnd(1)*256)
G=Int(Rnd(1)*256)
B=Int(Rnd(1)*256)
X=Int(Rnd(1)*600)
Y=Int(Rnd(1)*400)
print #win.gbox, "color ";R;" ";G;" ";B
print #win.gbox, "set ";X;" ";Y
Next i
print #win.gbox, "up"
print #win.gbox, "flush"
wait

```

5.

a) =====

```

nomainwin
WindowWidth = 640
WindowHeight = 480
UpperLeftX = DisplayWidth/2- WindowWidth/2
UpperLeftY = DisplayHeight/2- WindowHeight/2
graphicbox #win.gbox, 10, 10, 600, 400
open "Спираль" for window as #win
print #win.gbox, "home"

```

```

print #win.gbox, "down"
print #win.gbox, "size 6"
t=33:l=1
For i=1 to 100
l=l+1
print #win.gbox, "turn ";t
print #win.gbox, "go ";l
Next i
print #win.gbox, "up"
print #win.gbox, "flush"
wait
б) =====
nomainwin
WindowWidth = 200
WindowHeight = 200
UpperLeftX = DisplayWidth/2- WindowWidth/2
UpperLeftY = DisplayHeight/2- WindowHeight/2
graphicbox #win.gbox, 10, 10, 150, 150
open "Линии" for window as #win
print #win.gbox, "home"
print #win.gbox, "down"
print #win.gbox, "size 2"
For i=1 To 500
X1=Int(Rnd(1)*150)
Y1=Int(Rnd(1)*150)
X2=Int(Rnd(1)*150)
Y2=Int(Rnd(1)*150)
R=Int(Rnd(1)*256)

```

```

G=Int(Rnd(1)*256)
B=Int(Rnd(1)*256)
print #win.gbox, "color ";R;" ";G;" ";B
print #win.gbox, "line ";X1;" ";Y1;" ";X2;" ";Y2
Next i
print #win.gbox, "flush"
wait
6.
a) =====
nomainwin
WindowWidth=630
WindowHeight=460
UpperLeftX = DisplayWidth/2- WindowWidth/2
UpperLeftY = DisplayHeight/2- WindowHeight/2
graphicbox #win.gbox, 10, 10, 600, 400
open "y=x^2" for window as #win
print #win.gbox, "home"
print #win.gbox, "down"
'-----
print #win.gbox, "size 1"
print #win.gbox, "set ";10;" ";10
print #win.gbox, "box ";590;" ";390
For i=10 to 580 Step 20
print #win.gbox, "set ";i;" ";10
print #win.gbox, "box ";i+10;" ";390
next i
For i=10 to 380 Step 20
print #win.gbox, "set ";10;" ";i

```

```

print #win.gbox, "box ";590;" ";i+10
next i
'-----
print #win.gbox, "size 3"
print #win.gbox, "line ";20;" ";370;" ";580;" ";370
print #win.gbox, "line ";300;" ";20;" ";300;" ";380
'-----
print #win.gbox, "size 4"
print #win.gbox, "color ";0;" ";128;" ";0
For i=-6 To 6 step 0.01
x=300+i*10
y=370-10*i^2
print #win.gbox, "set ";x;" ";y
next i
print #win.gbox, "flush"
wait

```

б) =====

```

nomainwin
WindowWidth = 640
WindowHeight = 480
UpperLeftX = DisplayWidth/2- WindowWidth/2
UpperLeftY = DisplayHeight/2- WindowHeight/2
graphicbox #win.gbox, 10, 10, 600, 400
open "Лабиринт" for window as #win
print #win.gbox, "down"
'-----

```

```

print #win.gbox, "size 1"
print #win.gbox, "set ";10;" ";10
print #win.gbox, "box ";591;" ";391
For i=1 to 2900
a=Int(Rnd(1)*3)
x=Int(Rnd(1)*58)*10+10
y=Int(Rnd(1)*38)*10+10
select case a
case 0
print #win.gbox, "line ";x+10;" ";y;" ";x+10;" ";y+10
case 1
print #win.gbox, "line ";x;" ";y+10;" ";x+10;" ";y+10
case else
end select
Next i
'-----
print #win.gbox, "color ";255;" ";255;" ";255
print #win.gbox, "line ";10;" ";10;" ";20;" ";10
print #win.gbox, "line ";580;" ";390;" ";590;" ";390
print #win.gbox, "flush"
wait
8. =====
nomainwin
graphicbox #win.gbox, 10, 10, 150, 150
open "Drawing" for window as #win
print #win.gbox, "down"
print #win.gbox, "size 1"
print #win.gbox, "place 75 75"

```

```
For i=5 To 100 step 10
print #win.gbox, "ellipse ";i;" ";105-i
Next i
print #win.gbox, "flush"
wait
```

9. =====

```
nomainwin
WindowWidth = 600
WindowHeight = 400
UpperLeftX = DisplayWidth/2- WindowWidth/2
UpperLeftY = DisplayHeight/2- WindowHeight/2
open "Рисование" for graphics as #graph
print #graph, "when characterInput "
print #graph, "down"
print #graph, "size 4"
print #graph, "color black"
print #graph, "set ";300;" ";200
x=300
y=200
wait

key$ = Inkey$
if key$="w" or key$="ц" Then y=y-4
if key$="s" or key$="ы" Then y=y+4
if key$="a" or key$="ф" Then x=x-4
if key$="d" or key$="в" Then x=x+4
```



```
if key$="r" or key$="к" Then print #graph, "color white"
if key$="f" or key$="а" Then print #graph, "color black"
if key$="v" or key$="м" Then print #graph, "color red"
print #graph, "set ";x;" ";y
wait if key$=" " Then end
print "Нажата клавиша: "; key$
wait
```

### Глава III

1.

а) =====

```
nomainwin
```

```
WindowWidth = 300
```

```
WindowHeight = 150
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
statictext #win.ttt, "", 5, 10, 250, 90
```

```
open "Интерфейс" for dialog as #win
```

```
print #win, "font times_new_roman 14"
```

```
A$="У Лукоморья дуб зеленый."+chr$(10)
```

```
A$=A$+"Златая цепь на дубе том."+chr$(10)
```

```
A$=A$+"И днем и ночью кот ученый,"+chr$(10)
```

```
A$=A$+"Все ходит по цепи кругом."
```

```
print #win.ttt, A$
```

```
wait
```

б) =====

```
nomainwin
```

```
WindowWidth = 300
```

```
WindowHeight = 150
```

```

UpperLeftX = DisplayWidth/2-150
UpperLeftY = DisplayHeight/2-75
statictext #win.tt1, "У Лукоморья дуб зеленый.", 5, 10, 250, 20
statictext #win.tt2, "Златая цепь на дубе том.", 5, 30, 250, 20
statictext #win.tt3, "И днем и ночью кот ученый,", 5, 50, 250, 20
statictext #win.tt4, "Все ходит по цепи кругом.", 5, 70, 250, 30
open "Текст" for dialog as #win
print #win.tt1, "!font times_new_roman bold 14"
print #win.tt2, "!font A431 14"
print #win.tt3, "!font Century_Gothic 9 20"
print #win.tt4, "!font Mistral 18"
wait
2.
a) =====
nomainwin
WindowWidth = 310
WindowHeight = 150
UpperLeftX = DisplayWidth/2-150
UpperLeftY = DisplayHeight/2-75
statictext #win.tt1, "", 5, 10, 250, 20
statictext #win.tt2, "", 5, 30, 250, 20
button #win.btt1, "1-2", , UL, 20, 70, 100, 40
button #win.btt1, "3-4", , UL, 120, 70, 100, 40
open "Текст" for dialog as #win
print #win, "font times_new_roman 14"
wait

print #win.tt1, "У Лукоморья дуб зеленый."

```

```
print #win.tt2, "Златая цепь на дубе том."
```

```
wait
```

```
print #win.tt1, "И днем и ночью кот ученый,"
```

```
print #win.tt2, "Все ходит по цепи кругом."
```

```
wait
```

```
б) =====
```

```
nomainwin
```

```
WindowWidth = 310
```

```
WindowHeight = 150
```

```
UpperLeftX = DisplayWidth/2-150
```

```
UpperLeftY = DisplayHeight/2-75
```

```
statictext #win.tt1, "", 5, 10, 250, 20
```

```
statictext #win.tt2, "", 5, 30, 250, 20
```

```
button #win.btt1, "(1-2)/(3-4)", , UL, 20, 70, 150, 40
```

```
open "Текст" for dialog as #win
```

```
print #win, "font times_new_roman 14"
```

```
st=-1
```

```
wait
```

```
st=st*(-1)
```

```
If st=1 Then
```

```
print #win.tt1, "У Лукоморья дуб зеленый."
```

```
print #win.tt2, "Златая цепь на дубе том."
```

```
else
```

```

print #win.tt1, "И днем и ночью кот ученый,"
print #win.tt2, "Все ходит по цепи кругом."
end If
wait

3. =====
nomainwin
WindowWidth = 230
WindowHeight = 130
UpperLeftX = DisplayWidth/2-150
UpperLeftY = DisplayHeight/2-75
statictext #win.ttt, "Как Вас Зовут?", 10, 10, 250, 15
textbox #win.ttb, 10, 30, 100, 20
button #win.btt, "OK", , UL, 10, 55, 75, 20
open "Текстовое окно" for window_nf as #win
v=1
wait

#win.ttb, "!contents? O$"
select case v
case 0
#win.ttb, ""
#win.ttt, "Как Вас Зовут?"
v=1
case 1
#win.ttb, ""
#win.ttt, "Привет ";O$;". Сколько вам лет?"
v=2
case 2

```

```

#win.ttb, ""
vv=val(O$)+Int(Rnd(1)*10+1)
#win.ttt, "А мне ";vv;" лет. Намите кнопку ОК."
v=0
end select
wait

5. =====
nomainwin: WindowWidth = 300: WindowHeight = 160
checkbox #win.cbox, "", , , 10, 10, 10, 20
statictext #win.tt1, "", 22, 12, 270, 15
statictext #win.tt2, "", 10, 70, 270, 15
textbox #win.ttb, 10, 90, 180, 24
button #win.bt, "&ok", , UL, 200, 90, 40, 25
open "Текстовое окно" for dialog as #win
#win, "trapclose "
#win.tt1, "- все буквы прописные или заглавные"
#win.tt2, "как вас зовут?"
ch=0
wait

#win.tt1, "- ВСЕ БУКВЫ ПРОПИСНЫЕ ИЛИ ЗАГЛАВНЫ"
#win.tt2, "КАК ВАС ЗОВУТ?"
#win.bt, "ОК"
ch=1
wait

#win.tt1, "- все буквы прописные или заглавные"
#win.tt2, "как вас зовут?"

```

```
#win.bt, "ок"
```

```
ch=0
```

```
wait
```

```
#win.ttb, "!contents? O$"
```

```
if ch=0 Then
```

```
#win.tt2, "привет ";O$;"."
```

```
else
```

```
#win.tt2, "ПРИВЕТ ";O$;"."
```

```
end if
```

```
wait
```

```
close #win
```

```
end
```

```
6. =====
```

```
nomainwin
```

```
WindowWidth = 350
```

```
WindowHeight = 200
```

```
'-----
```

```
groupbox #cfg.gb1, "Язык:", 10, 10, 120, 80
```

```
radiobutton #cfg.Rus, "", , , 20, 30, 20, 20
```

```
statictext #cfg.tt1, "Русский", 40, 32, 80, 15
```

```
radiobutton #cfg.Eng, "", , , 20, 60, 20, 20
```

```
statictext #cfg.tt2, "Английский", 40, 62, 80, 15
```

```
'-----
```

```
groupbox #cfg.gb2, "Шрифт:", 125, 10, 190, 80
```

```
radiobutton #cfg.TNR, "Times New Roman", , , 130, 30, 180, 20
```

radiobutton #cfg.CN, "Courier New", , , 130, 60, 150, 20

'-----

statictext #cfg.tt3, "", 10, 95, 270, 15

textbox #cfg.ttb, 10, 120, 180, 24

button #cfg.bt, "&ok", , UL, 200, 120, 40, 25

open "Радиокнопка" for window as #cfg

print #cfg, "trapclose "

#cfg, "font Times\_New\_Roman 11"

#cfg.Rus, "set"

#cfg.TNR, "set"

#cfg.tt3, "Как вас зовут?"

ch=0

wait

'-----

#cfg.gb1, "Язык:"

#cfg.gb2, "Шрифт:"

#cfg.tt1, "Русский"

#cfg.tt2, "Английский"

#cfg.tt3, "Как вас зовут?"

ch=0

wait

#cfg.gb1, "Language:"

#cfg.gb2, "Font:"

#cfg.tt1, "Russian"

#cfg.tt2, "English"

#cfg.tt3, "What is you name?"

ch=1

wait

'-----

#cfg, "font Times\_New\_Roman 11"

wait

'-----

#cfg, "font Courier\_New 10"

wait

'-----

#cfg.ttb, "!contents? O\$"

if ch=0 Then

#cfg.ttb, "Привет ";O\$;"."

else

#cfg.ttb, "Hello ";O\$;"."

end if

wait

close #cfg

end

7. =====

nomainwin

WindowWidth = 640

WindowHeight = 480

UpperLeftX = DisplayWidth/2- WindowWidth/2



UpperLeftY = DisplayHeight/2- WindowHeight/2

Sel\$=""

tr=1900

level\$(0) = "Незаметный"

level\$(1) = "Легкий"

level\$(2) = "Средний"

level\$(3) = "Тяжелый"

level\$(4) = "Невыносимый"

listbox #win.lbox, level\$, , 10, 410, 120, 35

button #win, "Перерисовать", , UL, 140, 415

graphicbox #win.gbox, 10, 10, 600, 400

open "Лабиринт" for window as #win

#win.lbox, "selection? Sel\$"

select case Sel\$

case "Незаметный"

tr=2300

case "Легкий"

tr=2600

case "Средний"

tr=2900

case "Тяжелый"

tr=3200

case "Невыносимый"

tr=3500

end select

#win.gbox, "down"

'-----

```

#win.gbox, "size 1"
#win.gbox, "set ";10;" ";10
#win.gbox, "color ";0;" ";0;" ";0
#win.gbox, "boxfilled ";591;" ";391
For i=1 to tr
a=Int(Rnd(1)*3)
x=Int(Rnd(1)*58)*10+10
y=Int(Rnd(1)*38)*10+10
select case a
case 0
#win.gbox, "line ";x+10;" ";y;" ";x+10;" ";y+10
case 1
#win.gbox, "line ";x;" ";y+10;" ";x+10;" ";y+10
case else
end select
Next i
'-----
#win.gbox, "color ";255;" ";255;" ";255
#win.gbox, "line ";10;" ";10;" ";20;" ";10
#win.gbox, "line ";580;" ";390;" ";590;" ";390
#win.gbox, "flush"
wait
9. =====
nomainwin
WindowWidth = 640
WindowHeight = 480
UpperLeftX = DisplayWidth/2- WindowWidth/2
UpperLeftY = DisplayHeight/2- WindowHeight/2

```

```

tr=1900
graphicbox #win.gbox, 10, 10, 600, 400
open "Лабиринт" for window as #win
#win.gbox, "when rightButtonUp "

tr=2300:goto

tr=2600:goto

tr=2900:goto

tr=3200:goto

tr=3500

#win.gbox, "down"
'-----
#win.gbox, "size 1"
#win.gbox, "set ";10;" ";10
#win.gbox, "color ";0;" ";0;" ";0
#win.gbox, "boxfilled ";591;" ";391
For i=1 to tr
a=Int(Rnd(1)*3)
x=Int(Rnd(1)*58)*10+10
y=Int(Rnd(1)*38)*10+10
select case a
case 0
#win.gbox, "line ";x+10;" ";y;" ";x+10;" ";y+10

```

```
case 1
#win.gbox, "line ";x;" ";y+10;" ";x+10;" ";y+10
```

```
case else
```

```
end select
```

```
Next i
```

```
'-----
```

```
#win.gbox, "color ";255;" ";255;" ";255
```

```
#win.gbox, "line ";10;" ";10;" ";20;" ";10
```

```
#win.gbox, "line ";580;" ";390;" ";590;" ";390
```

```
#win.gbox, "flush"
```

```
wait
```

```
popupmenu "&Уровень:Незаметный", ,_
```

```
" &Легкий", ,_
```

```
" &Средний", ,_
```

```
" &Тяжелый", ,_
```

```
" &Невыносимый",
```

```
wait
```

```
10. =====
```

```
nomainwin
```

```
UpperLeftX = DisplayWidth/2-640/2
```

```
UpperLeftY = DisplayHeight/2-480/2
```

```
WindowWidth = 640
```

```
WindowHeight = 480
```

```
ok=0
```

```
oi=0
```

```
open "Хоккей (в начале кликните по полю)(вверх-ф, вниз-я)" for
```

```
graphics_nf_nsb as #gr
```

```
#gr, "when characterInput "
```

```
#gr, "trapclose "
```

```
y1=235
```

```
y2=235
```

```
#gr, "font Arial 40"
```

```
x3=320
```

```
y3=240
```

```
dx=4
```

```
dy=4
```

```
a$=Inkey$
```

```
if (a$="a" or a$="A" or a$="ф" or a$="Ф") And y1>0 Then  
y1=y1-10
```

```
if (a$="z" or a$="Z" or a$="я" or a$="Я") and y1<405 Then  
y1=y1+10
```

```
#gr, "cls"
```

```
#gr, "down; color cyan; size 4; bgcolor white"
```

```
#gr, "place 318 0"
```

```
#gr, "goto 318 480"
```

```
#gr, "place 318 227"
```

```
#gr, "circle 120"
```

```
#gr, "circlefilled 7"
```

```
#gr, "place 245 245"
```

```
#gr, "\";oi
#gr, "place 365 245"
#gr, "\";ok
#gr, "color darkred; size 1; bgcolor darkred"
#gr, "place 0 ";y1
#gr, "boxfilled 10 ";y1+50
#gr, "place 620 ";y2
#gr, "boxfilled 630 ";y2+50
#gr, "color black; bgcolor black"
#gr, "place ";x3;" ";y3
#gr, "circlefilled 10"
t=time$("milliseconds")
```

```
if time$("milliseconds")<t+30 Then goto
```

```
y11=y1
```

```
y22=y2
```

```
y33=y3
```

```
x33=x3
```

```
if x3>320 and dx>0 then
```

```
if y3<y2+5 then y2=y2-8
```

```
if y3>y2+45 then y2=y2+8
```

```
end if
```

```
if x3+dx<20 and (y3>y1 and y3<y1+50) then dx=dx*(-1)*(1+rnd
(1)/5)
```

```
if x3+dx>610 and (y3>y2 and y3<y2+50) then dx=dx*(-1)*(1+rnd
(1)/5)
```

if  $y_3+dy < 10$  or  $y_3+dy > 445$  then  $dy=dy*(-1)$

if  $x_3 < -10$  then notice "Комп выиграл":  $ok=ok+1$ :goto

if  $x_3 > 640$  then notice "Игрок выиграл":  $oi=oi+1$ :goto

$x_3=x_3+dx$

$y_3=y_3+dy$

scan

goto

close #gr

end

Глава IV

1. =====

Первоначально подготовил в графическом редакторе два рисунка.

nomainwin

WindowWidth = 248

WindowHeight = 175

VV=0

loadbmp "circle", "bmp\circle.bmp"

loadbmp "triangl", "bmp\triangl.bmp"

bmpbutton #main.button1, "bmp\circle.bmp", , UL, 22, 11

open "Нарисованные кнопки" for window as #main

```
print #main, "trapclose "  
wait  
  
if VV=0 Then  
print #main.button1, "bitmap triangl"  
VV=1  
else  
print #main.button1, "bitmap circle"  
VV=0  
end if  
wait  
  
close #main  
end
```

2. =====

Подготовил в графическом редакторе рисунки шагающего  
человечка.

```
nomainwin  
WindowWidth = 135  
WindowHeight = 240  
loadbmp "1m", "bmp\1man.bmp"  
loadbmp "2m", "bmp\2man.bmp"  
loadbmp "3m", "bmp\3man.bmp"  
loadbmp "4m", "bmp\4man.bmp"
```



```
graphicbox #main.gb, 5, 5, 120, 200
open "Человечек" for window_nf as #main
#main, "trapclose "
#main.gb, "drawbmp 4m 15 35"
#main.gb, "flush"
#main.gb, " redraw"
```

```
#main.gb, "drawbmp 1m 15 35"
call zaderjka
#main.gb, "drawbmp 2m 15 35"
call zaderjka
#main.gb, "drawbmp 3m 15 35"
call zaderjka
#main.gb, "drawbmp 4m 15 35"
call zaderjka
scan
goto
wait
sub zaderjka
for a=1 to 100000
next a
end sub
```

```
close #main
end
```

3. =====

```
nomainwin
```

```
WindowWidth = 200
```

```

WindowHeight = 150
button #win, "Запустить файл", , UL, 5, 5
button #win, "Стоп", , UL, 5, 65
statictext #win.st, "", ,5,45,100,20
open "Звук" for dialog as #win
print #win, "trapclose "
wait

#win.st, "media\beep.wav"
playwave "media\beep.wav", loop
wait

#win.st, ""
playwave ""
wait

playwave ""
close #win
end

4. =====
nomainwin: WindowWidth = 200: WindowHeight = 150
pp=0
button #win.bt1, "Играть миди", , UL, 5, 5
button #win.bt2, "Остановить миди", , UL, 5, 60
statictext #win.st, "", , 5, 40, 100, 20
open "Звук" for dialog as #win
print #win, "trapclose "
wait

```

```
If pp=1 Then wait
#win.st, "media\theme.mid"
pp=1
playmidi "media\theme.mid", length
timer 500,
wait
```

```
#win.st, ""
if pp=1 Then stopmidi
timer 0
pp=0
wait
```

```
if length = midipos() then stopmidi: timer 0: pp=0: #win.st, ""
wait
```

```
if pp=1 Then stopmidi
timer 0
close #win
wait
```

```
7. =====
```

```
nomainwin
WindowWidth = 350
WindowHeight = 380
filename$ = "TextKir.txt"
open filename$ for input as #text
texteditor #win.te 15,15,300,300
```

```

open "Чтение файла" for window as #win
print #win, "trapclose "
buff$=""

if eof(#text) <> 0 then
item$=input$(#text,1) 'читает по 1 символу
nn$=""
st$=""
k=ASC(item$)
if k>223 And k<256 Then kk=32:k=k-32 Else kk=0
nn$=chr$((k=192)*65+(k=193)*66+(k=194)*86+(k=195)*71
+(k=196)*68+_
(k=197)*89+(k=198)*74+(k=199)*90+(k=200)*73+(k=201)*89+_
(k=202)*75+(k=203)*76+(k=204)*77+(k=205)*78+(k=206)*79+_
(k=207)*80+(k=208)*82+(k=209)*83+(k=210)*84+(k=211)*85+_
(k=212)*70+(k=213)*72+(k=214)*67+(k=215)*67+(k=216)*83+_
(k=217)*83+(k=218)*39+(k=219)*73+(k=221)*69+(k=222)*89+_
(k=223)*89)
if nn$<>chr$(0) then st$=chr$(asc(nn$)+kk*(asc(nn$)<>39))
nn$=chr$((k=197)*69+(k=215)*72+(k=216)*72+(k=217)*67
+(k=219)*39+_
(k=222)*85+(k=223)*65)
if nn$<>chr$(0) then st$=st$+chr$(asc(nn$)+kk*(asc(nn$)<>39))
nn$=chr$((k=217)*72)
if nn$<>chr$(0) then st$=st$+chr$(asc(nn$)+kk)
nn$=chr$((k<192)*k)
if nn$<>chr$(0) then st$=st$+nn$
buff$=buff$+st$

```

goto

print #win.te, buff\$

close #text

wait

close #win

end

8. =====

nomainwin

WindowWidth = 640

WindowHeight = 500

filename\$="TextKir.txt"

texteditor #win.te 0,15,300,400

texteditor #win.te2 335,15,300,400

statictext #win.st1, "ASCII", 100,0,100,15

statictext #win.st2, "OEM/DOS", 425,0,100,15

textbox #win.tb 5,415,150,25

button #win.b1, "ASCII->OEM/DOS", , UL, 220,415,100,25

button #win.b2, "OEM/DOS->ASCII", , UL, 325,415,100,25

button #win.b3, "Открыть", , UL, 155,415,60,25

button #win.b3, "Поменять кодировку", , UL, 250,0,125,17

open "Чтение файла" for window as #win

print #win, "trapclose "

#win.tb "TextKir.txt"

#win.te2 "!font terminal 10"

#win.te "!font arial 10"

buff\$=""

```
ch=0
wait
'-----

print #win.te, "!cls"
print #win.te2, "!cls"
buff$=""
#win.tb "!contents? filename$"
open filename$ for input as #text

if eof(#text) <> 0 then
item$=input$(#text,1) 'читает по 1 символу
buff$=buff$+item$
goto

ch=0
#win.te2 "!font terminal 10"
#win.te "!font arial 10"
#win.st1, "ASCII"
#win.st2, "OEM/DOS"
print #win.te, buff$
buff2$=ToDOS$(buff$)
print #win.te2, buff$
close #text
wait
'-----
```

If ch=1 Then

```
buff2$=ToDOS$(buff$)
#win.te2 "!font terminal 10"
#win.te "!font arial 10"
#win.st1, "ASCII"
#win.st2, "OEM/DOS"
print #win.te, "!cls"
print #win.te, buff$
print #win.te2, "!cls"
print #win.te2, buff$
ch=0
Else
buff2$=ToASCII$(buff$)
#win.te "!font terminal 10"
#win.te2 "!font arial 10"
#win.st2, "ASCII"
#win.st1, "OEM/DOS"
print #win.te2, "!cls"
print #win.te2, buff2$
print #win.te, "!cls"
print #win.te, buff2$
ch=1
End if
wait
'-----

open "DOS_" + filename$ for output as #text2
buff2$=ToDOS$(buff$)
print #text2, buff2$
```

```

print #win.te2, "!cls"
print #win.te2, buff$
close #text2
wait
'-----

Function ToDOS$(buff$)
#win.te2 "!font terminal 10"
#win.te "!font arial 10"
buff2$=""
For i=1 To len(buff$)
k=ASC(mid$(buff$,i,1))
kk=k
if k>191 And k<240 Then kk=k-64
if k>239 And k<256 Then kk=k-16
if k=168 Then kk=240
If k=184 Then kk=241
buff2$=buff2$+chr$(kk)
Next i
ToDOS$=buff2$
End Function
'-----

open "ASCII_" + filename$ for output as #text2
buff2$=ToASCII$(buff$)
print #text2, buff2$
print #win.te2, "!cls"
print #win.te2, buff2$
close #text2

```



wait

'-----

Function ToASCII\$(buff\$)

#win.te “!font terminal 10”

#win.te2 “!font arial 10”

buff2\$=""

For i=1 To len(buff\$)

k=ASC(mid\$(buff\$,i,1))

kk=k

if k>127 And k<176 Then kk=k+64

if k>223 And k<240 Then kk=k+16

if k=240 Then kk=168

If k=241 Then kk=184

buff2\$=buff2\$+chr\$(kk)

Next i

ToASCII\$=buff2\$

End Function

close #win

end

9. =====

open “myfile.bin” for output as #myfile

close #myfile

open “myfile.bin” for binary as #myfile

txt\$ = “Четыре черненьких, чумазеньких, чертенка.”

print “Исходный текст: ”;txt\$

print #myfile, txt\$

For i=0 To lof(#myfile)-1

seek #myfile, i

```
s$=input$(#myfile,1)
if s$=" " Then
s$="_"
seek #myfile, i
print #myfile, s$
end if
if s$="a" Then
s$="@ "
seek #myfile, i
print #myfile, s$
end if
next
seek #myfile,0
line input #myfile, txt$
print "Результат: ";txt$
close #myfile
end
```

10. =====

```
nomainwin
WindowWidth = 10
WindowHeight = 10
UpperLeftX = 1
UpperLeftY = 1
dl=DisplayWidth
sh=DisplayHeight
open "Экран1" for graphics_nf_nsb as #geo1
#geo1, "getbmp pic1 1 -30 ";dl/2;" ";sh/2
#geo1, "getbmp pic2 ";dl/2;" -30 ";dl/2;" ";sh/2
```

```

#geo1, "getbmp pic3 1 ";sh/2-30;" ";dl/2;" ";sh/2
#geo1, "getbmp pic4 ";dl/2;" ";sh/2-30;" ";dl/2;" ";sh/2
close #geo1
WindowWidth = dl
WindowHeight = sh
UpperLeftX = 1
UpperLeftY = 1
graphicbox #geo2.gb, 0, 0, dl, sh
open "Экран2" for dialog_popup as #geo2
#geo2.gb, "down"
#geo2.gb, "drawbmp pic3 ";dl/2;" 1"
#geo2.gb, "drawbmp pic4 1 1"
#geo2.gb, "drawbmp pic1 ";dl/2;" ";sh/2
#geo2.gb, "drawbmp pic2 1 ";sh/2
#geo2.gb, "flush"
#geo2.gb, "when characterInput "
#geo2.gb, "when leftButtonDown "
#geo2.gb, "setfocus"
wait

close #geo2
end
Глава V
1. =====
nomainwin
WindowWidth = 270
WindowHeight = 270
ch$="green"

```

```

chF$="240 240 240"
tl=1
menu #geo, "&Цвет", "&Цвет",
menu #geo, "Цвет &Фона", "Цвет &Фона",
menu #geo, "&Толщина линии", "&Толщина линии(+)", ,
"&Толщина линии(-)",
open "Гео - узор" for graphics_nsb as #geo

#geo, "cls; home; down; color ";ch$
#geo, "fill ";chF$
#geo, "size ";tl
for x = 1 to 120
#geo, "go ";x;"; turn 69"
next x
wait

colordialog "red", ch$
if ch$="" then ch$="0 0 0"
p1=instr(ch$," ")
p2=instr(ch$," ",p1+1)
p3=instr(ch$," ",p2+1)-1
if p3<=0 then p3=len(ch$)
ch$=mid$(ch$,1,p3)
goto

colordialog "red", chF$
if ch$="" then chF$="0 0 0"
p1=instr(chF$," ")

```

```
p2=instr(chF$," ",p1+1)
p3=instr(chF$," ",p2+1)-1
if p3<=0 then p3=len(chF$)
chF$=mid$(chF$,1,p3)
goto
```

```
If tl<10 Then tl=tl+1
goto
```

```
If tl>1 Then tl=tl-1
goto
```

```
4. =====
nomainwin: WindowWidth = 375:WindowHeight = 100
button #win.bb "Открыть файл",,UL,5,15,150,25
open "Чтение файла" for window as #win
print #win, "trapclose "
wait
```

```
buff$=""
stroki$=""
filedialog "Открыть файл...", "*.txt", filename$
if filename$="" Then wait
open filename$ for input as #text
for i=1 to 30
if eof(#text) <> 0 then
LINE INPUT #text, buff$
stroki$=stroki$+buff$+CHR$(13)
next i
```

```
close #text
notice filename$+CHR$(13)+stroki$
wait

close #win
end
10. =====
nomainwin
WindowWidth = 300: WindowHeight = 150
statictext #win.st1, "", 5, 10, 160, 65
statictext #win.st2, "", 5, 25, 130, 60
open "Число Пи" for graphics as #win
print #win, "setfocus"
print #win, "when characterInput "
print #win, "trapclose "
print #win.st1, "Нахождение числа Пи:"
i=1:s=0

s=s+((-1)^(i+1))/(2*i-1)
Pi=4*s
i=i+1
print #win.st2, Pi
scan
goto

close #win
end
```

11. =====

```
input "Введите дату в формате ММ/ДД/ГГГГ:";d$
```

```
kd=date$(d$)
```

```
dn=kd mod 7
```

```
select case dn
```

```
case 0
```

```
print "Эта дата - Вторник"
```

```
case 1
```

```
print "Эта дата - Среда"
```

```
case 2
```

```
print "Эта дата - Четверг"
```

```
case 3
```

```
print "Эта дата - Пятница"
```

```
case 4
```

```
print "Эта дата - Суббота"
```

```
case 5
```

```
print "Эта дата - Воскресенье"
```

```
case 6
```

```
print "Эта дата - Понедельник"
```

```
end select
```

```
goto
```

12. =====

```
i=1
```

```
s=0
```

```
t=time$("ms")
```

```
s=s+((-1)^(i+1))/(2*i-1)
```

```
Pi=4*s
i=i+1
cls
print "Нахождение числа Пи: ";Pi
scan
if Pi>3.1415 And Pi<3.1416 Then goto
goto
```

```
print time$("ms")-t;" миллисекунд"
end
```

13. =====

```
nomainwin
run "C:\Program Files\Microsoft Office\Office14\
winword.exe " + chr$(34) + "D:\PRG\Liberty BASIC _
v4.03\letter.TXT" + chr$(34)
```

15. =====

```
nomainwin
WindowWidth = 350
WindowHeight = 380
filename$="letter.txt"
open filename$ for input as #text
texteditor #win.te 15,15,300,300
open "Чтение файла" for window as #win
print #win, "trapclose "
on error goto
buff$=""
```



'-----

```
item$=input$(#text,1)
```

```
buff$=buff$+item$
```

```
goto
```

'-----

```
print #win.te, buff$
```

```
close #text
```

```
wait
```

'-----

```
close #win
```

```
end
```

16. =====

Иконка к игре «Хоккей».

## Предметный указатель

!

!cls117

A

ABS(x)18

AND34

append104

ASC(x\$)23

ASCII13, 15

async95

B

backcolor62

binary106

bitmap91

bmpbutton90

bmpsave109

box61

boxfilled62

button73

C

Call40

case35

case else35

checkbox78

CHR\$(x)23

circle64

circlefilled64

close79

CLS77

color56

colordialog112

combobox84

confirm120

contents?74, 75, 76

COS(x)18

D

date\$()128  
dialog50  
dialog\_fs52  
dialog\_nf51  
dialog\_popup52  
DIM A(x)37  
Disable71  
DisplayHeight49  
DisplayWidth49  
down53  
drawbmp92  
dump122  
E  
ellipse65  
ellipsefilled66  
end17  
End Function39  
end select35  
End Sub40  
eof99  
EOF99  
Err135  
Err\$135  
EXP(x)18  
F  
false31  
filedialog116  
fill57

flush55

font70

fontdialog114

for..to..step..26

Function39

G

getbmp109

Global40

go58

gosub 27

goto60

goto 25

graphicbox49

graphics45

graphics\_fs46

graphics\_fs\_nsb46

graphics\_nf\_nsb47

graphics\_nsb45

groupbox77

H

HIDE131

home59

I

if...then...23, 33

if...then...else...24

Inkey\$32, 67

input20, 98, 100

input\$101

inputto\$102

INSTR(A\$,B\$,n)32

INT(x)19

K

kill98

L

LEN(x\$)22

let14

line60

line input98

listbox82

LL73

loadbmp91

LOC()106

LOF()105

LOG(x)19

loop95

lprint122, 123

LR73

M

MID\$(A\$,n,m)32

midipos()96

MINIMIZE131

N

name..as..97

next26

nomainwin45

north59

NOT35

notice119

O

on error135

open44

OR34

output103

P

place62

playmidi96

playwave94

popupmenu85

print16, 124

print svga126

print vga126

print xga126

PrintCollate123

PrintCopies123

printerdialog123

PrinterFont\$122, 123

PrinterName\$123

prompt121

R

radiobutton80

redraw94

rem29

reset79

RESTORE131

resume135

return28

RND(1)30

run131

S

save118

scan127

seek106

select case35

selectindex84

selection?83

set53, 79, 80

setfocus91

SHOW131

SHOWMAXIMIZED131

SHOWMINIMIZED131

SHOWMINNOACTIVE131

SHOWNA131

SHOWNOACTIVE131

SHOWNORMAL131

SIN(x)18

size54

SQR(x)19

statictext69

stopmidi96

STR\$(x)22

Sub40

T

TAN(x)18

textbox74

time\$( )130

timer97

trapclose79

TRIM\$(x\$)22

true31

turn58

U

UL73

up53

UpperLeftX48

UpperLeftY48

UR73

V

VAL(x\$)22

value?79

W

wait45

when characterInput67

when leftButtonDouble133

when leftButtonDown133

when leftButtonMove133

when leftButtonUp133

when middleButtonDouble133

when middleButtonDown133

when middleButtonUp133

when mouseMove133



when rightButtonDouble133

when rightButtonDown133

when rightButtonMove133

when rightButtonUp86, 133

window49

window\_nf50

window\_popup50

WindowHeight48

WindowWidth47

X

x MOD y19

x\$+y\$23

x^y19

XOR35

Б

Базовые логические операции24

В

вещественные числа13

вложенные циклы38

Д

дебагер12

З

знак

« \ »76

«!»71

«#»45

«\$»13

«&»85

«^»18

«\_»87

«|»87

«'»29

И

иконка143

интерпретатор141

интерфейс8

К

компилятор141

Л

лейбл25

М

Массивы

многомерные38

одномерные38

О

Основные арифметические операции18

П

пиксель70

предкомпилятор141

Р

Разделитель

«,»17

«:»28

«;»17, 87

Рекурсия42

С

Система координат44

строковые (переменные)13

У

указатель45

Ц

целые числа13

Ч

Число Фибоначи43