МИНИСТЕРСТВО ПУТЕЙ СООБЩЕНИЯ СССР
МОСКОВСКИЙ ОРДЕНА ЛЕНИНА
И ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
ИНСТИТУТ ИНЖЕНЕРОВ ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Кафедра математического обеспечения
автоматизированных систем управления

ОСНОВЫ ЯЗЫКА БЭЙСИК ЭВМ ИСКРА-226

Методические указания

по дисциплине

«ПРОГРАММИРОВАНИЕ НА ЭВМ ИСКРА-226»

Часть II

Москва—1987

Утверждено
редакционно-издательским
советом института

ОСНОВЫ ЯЗЫКА БЭЙСИК ЭВМ ИСКРА-226

Методические указания
по дисциплине
"Программирование на ЭВМ ИСКРА-226"
для слушателей факультета повышения квалификации,
преподавателей и студентов

Часть II

Москва - 1987

Методические указания составил
преподаватель МИИТа, канд.техн.наук
Г.М. К у р б а т о в

Р е ц е н з е н т ы: канд.физ.-мат. наук
А.Р.Ротенберг (НИИИС);
канд.техн.наук
Г.А.Шейкина (МИИТ)

ВВЕДЕНИЕ

Во второй части методических указаний рассматриваются матричные операции языка БЭЙСИК, функции символьных и логических переменных, операции преобразования информации.

1. МАТРИЧНЫЕ ОПЕРАЦИИ

Матричные операции языка БЭЙСИК позволяют выполнять действия над цифровыми и символьными массивами (матрицами и векторами). Их использование упрощает запись программ и сокращает время выполнения действий над матрицами по сравнению с программами, записанными без матричных операций [1] .

Признаком матричной операции является оператор МАТ (от *matrix* - матрица) , который интерпретирует операции в теле оператора как действие над матрицами или векторами.

Размерность исходных матриц (векторов) можно задавать в явном виде с помощью операторов DIM или COM. Если размерность исходных массивов в программе не указана, то по умолчанию она принимается для цифровых операндов равной (10,10) , для символьных операндов - (10,10)16 .

Размерность массивов при выполнении матричных операторов может переопределяться явным или неявным образом. В табл.1 приложения приведены 14 матричных операций языка БЭЙСИК. У операций, помеченных знаком "X", текущая размерность массивов переопределяется явным образом. Для операций, помеченных знаками"XXX", текущая размерность операнда, стоящего слева от знака равно, приводится в соответствие с текущей размерностью операнда(ов), стоящего(их) справа от знака равно. Операции, не отмеченные в табл.1 приложения знаками"XX", применимы только для цифровых массивов.

I.I. Операция переопределения размерностей матриц (MAT REDIM)

О п е р а т о р MAT REDIM предназначен для объявления новых текущих размерностей цифровых и символьных матриц (векторов).

Форма записи оператора:

MAT REDIM <список переопределяемых массивов> .

При переопределении размерности массива новая текущая размерность не должна превышать ранее заданной максимальной размерности. Матрица не может быть переопределена вектором, а вектор - матрицей.

Пример:

```
10 DIM A(3,4),B(6)
20 MAT INPUT A,B:MAT PRINT A;B
30 MAT REDIM A(3,3),B(4)
40 MAT PRINT A;B
 1  2  3  4          }
 5  6  7  8          }   матрица A(3,4) до переопределения;
 9  10  11  12       }
 6                   }
 5                   }
 4                   }   вектор B(6) до переопределения;
 3                   }
 2                   }
 1                   }

 1  2  3  }
 4  5  6  }   матрица A(3,3) после переопределения;
 7  8  9  }
 6         }
 5         }   вектор B(4) после переопределения .
 4         }
 3         }
```

При использовании оператора MAT REDIM значения элементов матрицы сохраняются, возможно перераспределение индексов элементов в соответствии с новыми заданными параметрами матрицы.

Пример:

```
10 DIM A(3,4)
20 MAT INPUT A:MAT PRINT A;
30 MAT REDIM A(4,3)
40 MAT PRINT A;
 1   2   3   4
 5   6   7   8
 9   10   11   12
```

матрица A(3,4) до переопределения;

```
 1   2   3
 4   5   6
 7   8   9
 10   11   12
```

матрица A(4,3) после переопределения.

I.2. Операция ввода элементов матриц (MAT INPUT)

О п е р а т о р MAT INPUT предназначен для ввода элементов матриц или векторов с клавишного устройства в процессе исполнения программы.

Форма записи оператора:

MAT INPUT <список вводимых массивов> .

Пример:

```
10 DIM A(10,10),B(8),C$(4,5)32
20 MAT INPUT A,B,C$
```

Оператор MAT INPUT в 20-й строке программы предусматривает ввод с клавиатуры 100 элементов цифровой матрицы A, 8 элементов цифрового вектора B и 20 элементов символьной матрицы C$ с длиной каждого элемента, равной 32 символам. В процессе ввода цифровым массивам должны соответствовать числовые данные, а символьным массивам - символьные данные.

Оператор MAT INPUT позволяет в процессе ввода переопределять в явном виде текущие размерности массивов.

Пример:

```
10 DIM A(10,10),B(8),C$(4,5)32
20 MAT INPUT A(5,5),B,C$(3,3)16
```

При вводе данных цифровая матрица A(10,10) и символьная матрица C$(4,5)32 получают новые текущие размерности A(5,5) и C$(3,3)16 .

I.3. Операция чтения элементов матриц (MAT READ)

О п е р а т о р MAT READ предназначен для выборки констант из блоков DATA и присвоения их значений соответствующим матрицам или векторам, указанным в этом операторе.

Форма записи оператора:

MAT READ <список вводимых массивов> .

Между значениями переменных, перечисленных в операторе MAT READ , и их числовыми значениями в блоке DATA должно быть установлено соответствие: цифровым матрицам или векторам должны соответствовать числовые данные, а символьным матрицам или векторам - символьные данные.

Пример:

```
10 DIM A(10,10),B(30),C$(5)2
20 MAT READ A(3,4),B(6),C$
30 MAT PRINT A;
40 MAT PRINT B;
50 MAT PRINT C$;
60 DATA 1,2,3,4,5,6,7,8,9
70 DATA 10,11,12,8,7,6,5,4
80 DATA 3,"A","B","C","D","E"
 1  2  3  4          }
 5  6  7  8          }   цифровая матрица A(3,4) ;
 9  10  11  12       }

 8                   }
 7                   }
 6                   }   цифровой вектор B(6);
 5                   }
 4                   }
 3                   }

A                    }
B                    }
C                    }   символьный вектор C$(5). .
D                    }
E                    }
```

В 20 операторе программы происходит переопределение размерностей матрицы A(I0,I0) на A(3,4) и вектора B(30) на B(6), выборка из блоков DATA цифровых и символьных элементов и заполнение ими массивов A, B и C$. Действие этого оператора эквивалентно выполнению следующей группы из 7 обычных операторов:

```
20 FOR I=1TO3
21 FOR J=1TO4
22 READ A(I,J):NEXT J:NEXT I
23 FOR K=1TO6
24 READ B(K):NEXT K
25 FOR I=1TO5
26 READ C$(I):NEXT I
```

Если в операторе DATA задано значений меньше, чем требуется для ввода в зарезервированные ячейки памяти, то возникает сообщение об ошибке.

I.4. Операция вывода элементов матриц (MAT PRINT)

О п е р а т о р MAT PRINT предназначен для вывода цифровых или символьных массивов в виде матриц или векторов на заданное устройство.

Форма записи оператора:

MAT PRINT[<номер устройства>] <список выводимых массивов> .

Матрицы и векторы печатаются в том порядке, в каком они перечислены в операторе MAT PRINT . Печать матрицы осуществляется по строкам, вектора – в виде столбца. Если в операторе MAT PRINT после идентификатора матрицы стоит точка с запятой, то матрица печатается в уплотненном формате. Если стоит запятая или нет знака, то печать матрицы производится в зонном формате.

Пример:

```
10 DIM A(3,3),B(5)
20 MAT READ A,B
30 MAT PRINT A;B
40 DATA 1,2,3,4,5,6,7
50 DATA 8,9,10,11,12,13,14
 1   2   3
 4   5   6      печать матрицы A(3,3) ;
 7   8   9
 10
 11
 12             печать вектора B(5) .
 13
 14
```

В операторе MAT PRINT можно задавать устройство, на которое необходимо вывести результаты вычислений.

Пример:

```
10 DIM A(3,3),B(5)
20 MAT READ A,B
30 MAT PRINT /05,A
40 MAT PRINT /0C,B
50 DATA 1,2,3,4,5,6,7
60 DATA 8,9,10,11,12,13,14
 10
 11
 12             печать вектора B(5) .
 13
 14
```

При выполнении программы значения матрицы A(3,3) индицируются на дисплее (ФАУ - 05) , а значения вектора B(5) печатаются на принтере (ФАУ - 0C) .

I.5. Операция заполнения матрицы нулями (MAT ZER)

О п е р а т о р MAT ZER предназначен для присвоения всем элементам матрицы или вектора значений, равных нулю.

Форма записи оператора:

MAT<имя цифрового массива>= ZER[(<новая размерность массива>)] .

Оператор MAT ZER позволяет переопределять текущую размерность массива.

Пример:

```
10 DIM A(3,3),B(4)
20 MAT READ A,B
30 MAT PRINT A;B
40 MAT A=ZER(2,2)
50 MAT B=ZER(3)
60 MAT PRINT A;B
70 DATA 1,2,3,4,5,6,7
80 DATA 8,9,10,11,12,13
```

```
 1  2  3
 4  5  6        } исходная матрица A(3,3) ;
 7  8  9
 10
 11             } исходный вектор B(4) ;
 12
 13

 0  0
 0  0           } нулевая матрица A(2,2) ;
 0
 0
 0              } нулевой вектор B(3) .
```

Выполнение матричных операторов, записанных в 40 и 50 строках программы, эквивалентно выполнению следующей группы обычных операторов:

```
40 FOR I=1TO2
42 FOR J=1TO2
44 A(I,J)=0
46 NEXT J:NEXT I
48 FOR K=1TO3
50 B(K)=0:NEXT K
```

I.6. Операция заполнения матрицы единицами (MAT CON)

О п е р а т о р MAT CON предназначен для присвоения всем элементам матрицы или вектора значений, равных единице.

Форма записи оператора:

MAT<имя цифрового массива>=CON[(<новая размерность массива>)] .

Оператор MAT CON отвечает тем же требованиям, что и оператор MAT ZER .

Пример:

```
10 DIM A(3,3),B(4)
20 MAT READ A,B
30 MAT PRINT A,B
40 MAT A=CON
50 MAT B=CON
60 MAT PRINT A,B
70 DATA 1,2,3,4,5,6,7
80 DATA 8,9,10,11,12,13
 1  2  3        }
 4  5  6        }  исходная матрица A(3,3) ;
 7  8  9        }
 10              }
 11              }
 12              }  исходный вектор B(4) ;
 13              }

 1  1  1        }
 1  1  1        }  результирующая матрица A(3,3) ;
 1  1  1        }
 1               }
 1               }
 1               }  результирующий вектор B(4) .
 1               }
```

I.7. Операция приведения матрицы к единичной (MAT IDN)

О п е р а т о р MAT IDN предназначен для приведения цифровой матрицы к единичной. При этом диагональные элементы исходной матрицы принимают значения, равные единице, а остальные - равные нулю.

Исходная матрица должна быть квадратной матрицей.

Форма записи оператора:

MAT<имя цифровой матрицы>= IDN [(<новая размерность матрицы>)].

Оператор MAT IDN позволяет переопределять текущую размерность матрицы. Новая текущая размерность должна также задавать квадратную матрицу.

Пример:

```
10 DIM A(2,5)
20 MAT READ A
30 MAT PRINT A;
40 MAT A=IDN(3,3)
50 MAT PRINT A;
60 DATA 1,2,3,4,5,6,7,8,9,10
```

```
 1  2  3  4  5
 6  7  8  9  10
```

исходная матрица A(2,5) ;

```
 1  0  0
 0  1  0
 0  0  1
```

единичная матрица A(3,3) .

Матричный оператор в 40 строке программы заменяет 7 обычных операторов:

```
40 FOR I=1TO3
42 FOR J=1TO3
44 A(I,J)=0
46 IF I<>JTHEN50
48 A(I,J)=1
50 NEXT I:NEXT J
```

I.8. Операция переноса значений из одной матрицы в другую (MAT=)

О п е р а т о р MAT= предназначен для присвоения значений одной матрицы или вектора другой матрице или вектору.

Форма записи оператора:

MAT<имя цифрового массива>=<имя цифрового массива> .

При выполнении оператора MAT= текущая размерность массива, стоящего слева от знака равно, приводится в соответствие с текущей размерностью массива, стоящего справа от знака равно (см.табл.I.1). При этом максимальное количество элементов массива, стоящего слева от знака равно, должно быть не меньше текущего количества элементов массива, стоящего справа от знака равно.

Пример:

```
10 DIM A(3,3),B(3,3)
20 MAT READ A
30 MAT PRINT A;
40 MAT B=A
50 MAT PRINT B;
60 DATA 1,2,3,4,5,6,7,8,9
```

```
 1  2  3
 4  5  6
 7  8  9
```

} исходная матрица A(3,3) ;

```
 1  2  3
 4  5  6
 7  8  9
```

} значения, присвоенные матрице B(3,3) .

40-й оператор программы заменяет группу из пяти обычных операторов:

```
40 FOR I=1TO3
42 FOR J=1TO3
44 B(I,J)=A(I,J)
46 NEXT J:NEXT I
```

Таблица I.I

Правила переопределения размерностей оператора MAT=

Левый операнд	Правый операнд	Операция	Новая размерность левого операнда	Примечание
вектор B(M)	вектор C(*N*)	MAT B=C	вектор B(*N*)	M ≥ *N*
матрица A(*N*,M)	матрица B(K,*L*)	MAT A=B	матрица A(K,*L*)	*N* x M ≥ K x *L*
вектор B(M)	матрица A(K,I)	MAT B=A	вектор B(K)	M ≥ K
матрица A(*N*,M)	вектор B(K)	MAT A=B	матрица A(K,I)	*N* x M ≥ K

1.9. Операция сложения матриц (MAT+)

О п е р а т о р MAT+ предназначен для сложения цифровых матриц или векторов.

Форма записи оператора:

MAT<имя цифрового массива> = <имя цифрового массива> + <имя цифрового массива> .

В результате сложения элементам массива, указанного слева от знака равно, присваивается сумма соответствующих элементов массивов, указанных справа от знака равно.

Пример:

```
10 DIM A(2,3),B(2,3),C(2,3)
20 MAT READ A:MAT PRINT A;
30 MAT READ B:MAT PRINT B;
40 MAT C=A+B
50 MAT PRINT C;
60 DATA 1,2,3,4,5,6,7,8,9,10,11,12
```

```
 1  2  3          }  матрица-слагаемое A(2,3) ;
 4  5  6          }

 7  8  9          }  матрица-слагаемое B(2,3) ;
10  11  12        }

 8  10  12        }
14  16  18        }  результирующая матрица C(2,3) .
```

40-й оператор программы эквивалентен записи пяти обычных операторов:

```
40 FOR I=1TO2
42 FOR J=1TO3
44 C(I,J)=A(I,J)+B(I,J)
46 NEXT J:NEXT I
```

При сложении текущая размерность массива, стоящего слева, приводится в соответствие с текущими размерностями массивов слагаемых (см.табл.1.2). Операнды, стоящие справа от знака равенства, должны иметь одинаковую размерность. Если один из операндов, стоящих справа, вектор, а другой -

матрица, то размерность матрицы должна быть равной $(N,1)$, где N – размерность вектора. Если операнд, стоящий слева, вектор, то операнды, стоящие справа, должны быть или векторами или матрицами с размерностью $(N,1)$.

Таблица 1.2

Правила переопределения размерностей оператора МАТ+

Левый операнд	Правый операнд I	Правый операнд 2	Операция	Новая размерность левого операнда	Примечание
матрица C(N,M)	матрица A(K,L)	матрица B(K,L)	МАТ. C=A+B	матрица C(K,L)	$N \times M \geq K \times L$
вектор A(M)	вектор B(N)	вектор C(N)	МАТ A=B+C	вектор A(N)	$M \geq N$
матрица C(N,M)	вектор A(L)	вектор B(L)	МАТ C=A+B	матрица C(L,I)	$N \times M \geq L$
вектор A(M)	матрица B(N,I)	вектор C(N)	МАТ A=B+C	вектор A(N)	$M \geq N$
вектор A(M)	матрица B(L,K)	матрица C(L,K)	МАТ A=B+C	вектор A(L)	$M \geq L$

I.10. Операция вычитания матриц (МАТ–)

О п е р а т о р МАТ – предназначен для вычитания цифровых матриц или векторов.

Форма записи оператора:

МАТ<имя цифрового массива> = <имя цифрового массива>

– <имя цифрового массива>

В результате вычитания элементам массива, указанного слева от знака равно, присваивается разность между соответствующими элементами массивов, указанных справа от знака равно.

Пример:

```
10 DIM A(2,3),B(2,3),C(2,3)
20 MAT READ A:MAT PRINT A;
30 MAT READ B:MAT PRINT B;
40 MAT C=A-B:MAT PRINT C;
50 DATA 1,8,3,-5,4,-2,-1
60 DATA 7,-6,4,-9,5
```

```
 1  8  3
-5  4 -2
```
} матрица A(2,3) ;

```
-1  7 -6
 4 -9  5
```
} матрица B(2,3) ;

```
 2  1  9
-9 13 -7
```
} результирующая матрица C(2,3) .

Оператор MAT- отвечает тем же требованиям, что и оператор MAT+

1.11. Операция умножения матрицы на числовое выражение (MAT()*)

О п е р а т о р MAT()* предназначен для умножения цифровых матриц или векторов на скалярную величину, заданную арифметическим выражением.

Форма записи оператора:

MAT <имя цифрового массива> = (<арифметическое выражение>) * <имя цифрового массива> .

В результате выполнения операции элементам массива, указанного слева от знака равно, присваиваются значения соответствующих элементов массива, стоящего справа от знака

равно, и умноженных на значение арифметического выражения.

Пример:

```
10 DIM A(2,3),B(2,3)
20 X=2
30 MAT READ A
40 MAT B=(X↑2)*A
50 MAT PRINT B;
60 DATA 2,1,4,3,6,5
 8   4  16
 12  24  20
```

} результат умножения матрицы A(2,3) на X^2 .

40-й оператор программы заменяет пять обычных операторов:

```
10 FOR I=1TO2
20 FOR J=1TO3
30 B(I,J)=(X↑2)*A(I,J)
40 NEXT J:NEXT I
```

При выполнении оператора MAT()* текущая размерность массива, стоящего слева от знака равно, приводится в соответствие с текущей размерностью массива, стоящего справа от знака равно. Правила переопределения размерностей оператора MAT()* те же, что и оператора MAT= (см.табл.I.I).

I.12. Операция умножения матриц (MAT*)

О п е р а т о р MAT* предназначен для умножения двух цифровых операндов: матрицы на матрицу или матрицы и вектора.

Форма записи оператора:

MAT <имя цифрового массива> = <имя цифрового массива> * <имя цифрового массива> .

В результате умножения элементам массива, указанного слева от знака равно, присваивается сумма произведений элементов массивов, указанных справа от знака равно, под-

считанных по формуле:

$$C_{ij} = \sum_{K=1}^{M} A_{iK} B_{Kj} \quad ,$$

где M – количество столбцов первого сомножителя и строк второго сомножителя.

Пример:

```
10 DIM A(2,2),B(2,2),C(2,2)
20 MAT READ A
30 MAT READ B
40 MAT C=A*B
50 MAT PRINT C;
60 DATA 2,-1,3,-4
70 DATA -2,1,-3,4
```

```
-1 -2
 6 -13
```

} результат умножения матрицы A(2,2) на матрицу B(2,2) .

40-й оператор заменяет программу, состоящую из трех вложенных друг в друга циклов:

```
10 FOR I=1TO2
20 FOR J=1TO2
30 C(I,J)=0
40 FOR K=1TO2
50 C(I,J)=C(I,J)+A(I,K)*B(K,J)
60 NEXT K:NEXT J:NEXT I
```

При записи оператора не допускается использование одного и того же массива в левой и правой частях. При умножении текущая размерность массива, стоящего слева, приводится в соответствие с текущими размерностями массивов – сомножителей (см.табл.I.3).

Если сомножители являются матрицами, то их текущие размерности должны быть такими, чтобы количество столбцов первого сомножителя равнялось числу строк второго сомножителя. Если множимое является матрицей, а множитель – вектор, тогда количество столбцов матрицы должно быть равно раз-

мерности вектора. Если множимое - вектор, а множитель - матрица, то текущая размерность матрицы должна быть (1,*N*), где *N* - текущая размерность вектора. Если операнд, стоящий в левой части, является вектором, то операнд - множитель должен быть либо вектором, либо матрицей с размерностью (*N*,1) .

Таблица 1.3

Правила переопределения размерностей оператора МАТ*

Левый операнд	Правый операнд 1	Правый операнд 2	Операция	Новая размерность левого операнда	Примечание
матрица C(*N*,M)	матрица A(K,L)	матрица B(L,P)	МАТ C=A*B	матрица C(K,P)	*N* x M ≥ K * P
матрица C(*N*,M)	матрица A(K,L)	вектор B(L)	МАТ C=A*B	матрица C(K,1)	*N*xM ≥ K
матрица C(*N*,M)	вектор B(L)	матрица A(1,K)	МАТ C=B*A	матрица C(L,K)	*N* x M ≥ L x K
вектор B(M)	матрица A(*N*,K)	вектор C(K)	МАТ B=A*C	вектор B(*N*)	M ≥ *N*

Оператор МАТ* не позволяет умножать вектор на вектор кроме случая, когда векторы содержат по одному элементу.

1.13. Операция транспонирования матрицы (MAT TRN)

О п е р а т о р MAT TRN предназначен для транспонирования цифровых матриц или векторов.

Форма записи оператора:

МАТ <имя цифрового массива> = TRN(<имя цифрового массива>) .

В результате выполнения оператора происходит транспонирование элементов массива, указанного в правой части, и присвоение результата массиву, указанному в левой части.

Пример:

```
10 DIM A(3,2),B(2,3)
20 MAT READ A:MAT PRINT A;
30 MAT B=TRN(A):MAT PRINT B;
40 DATA 3,1,4,5,0
50 DATA 1,2,3,4,3,-2,1,-1
 3  1          }
 4  5          }  исходная матрица A(3,2);
 0  1          }

 3  4  0       }
 1  5  1       }  транспонированная матрица B(2,3).
```

Оператор MAT TRN в программе вызывает такое же действие, как группа из пяти обычных операторов:

```
10 FOR I=1TO3
20 FOR J=1TO2
30 B(I,J)=A(J,I)
40 NEXT J:NEXT I
```

При записи оператора MAT TRN не допускается использование одного и того же массива в его левой и правой частях. При транспонировании текущая размерность результирующей матрицы или вектора переопределяется в соответствии с текущей размерностью транспонируемой матрицы или вектора (см. табл.I.4).

Если в левой части оператора стоит вектор, то в правой его части должна стоять матрица с текущей размерностью (I,*N*). Если вектор стоит в правой части, то в левой части должна стоять матрица с текущей размерностью (M,*N*) .

Таблица I.4

Правила переопределения размерностей оператора MAT TRN

Левый операнд	Правый операнд	Операция	Новая размерность левого операнда	Примечание
матрица A(*N*,M)	матрица B(K,L)	MAT A=TR*N*(B)	матрица A(*L*,K)	*N* x M ≥ K x *L*
вектор B(M)	матрица A(I,*N*)	MAT B=TR*N*(A)	вектор B(*N*)	M ≥ *N*
матрица A(*N*,M)	вектор B(*L*)	MAT A=TR*N*(B)	матрица A(I,*L*)	N x M ≥ *L*

I.I4. Операция обращения матрицы и вычисление ее определителя (MAT INV)

О п е р а т о р MAT INV предназначен для вычисления матрицы, обратной данной, и определителя исходной матрицы.

Форма записи оператора:

MAT <имя цифровой матрицы> = INV(<имя цифровой матрицы>) [, <цифровая переменная>] .

Оператор вычисляет матрицу, обратную стоящей в правой части квадратной матрице, и присваивает ее значения матрице, стоящей в левой части. Размерность матрицы, стоящей в левой части, становится равной размерности квадратной матрицы, стоящей в правой части.

Если в записи оператора присутствует цифровая переменная, то вычисляется также определитель обращаемой матрицы, и его значение присваивается цифровой переменной.

Пример:

```
10 DIM C(3,3)
20 MAT READ C
30 MAT PRINT C;
40 MAT B=INV(C),D
50 MAT PRINT B;
60 PRINT D
70 DATA 2,1,3,1,2,4,-2,0,-1
```

```
 2  1  3
 1  2  4
-2  0 -1
```
- исходная матрица C(3,3);

```
-2  1 -2
-7  4 -5
 4 -2  3
```
- матрица B(3,3) , обратная матрице C(3,3),

```
 1
```
- значение определителя матрицы C(3,3) .

2. ФУНКЦИИ СИМВОЛЬНЫХ ПЕРЕМЕННЫХ

Функции символьных переменных предназначены для обработки текстовой информации, заданной в виде строк символов [2]. Любой символ в такой строке представлен 8 битовым двоичным словом - байтом. Все символы языка БЭЙСИК представляются в памяти машины различными комбинациями нулей и единиц в 8 разрядном слове, например: 01101101 ; 10001011 и др. Содержимое байта можно также представить в виде двух цифр шестнадцатеричного кода - HEX-кода, например: 6D ; 8B и др. В табл.2.1 показано соответствие между шестнадцатеричными цифрами, десятичными числами и их двоичными кодами.

Таблица 2.1

Таблица соответствия между шестнадцатеричными цифрами, десятичными числами и их двоичными кодами

ДЕСЯТИЧНАЯ	ДВОИЧНАЯ	ШЕСТНАДЦАТЕРИЧНАЯ
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Таким образом, любой символ, имеющийся на клавиатуре, может быть выражен HEX-кодом (см.табл.2 приложения).

Возможность представления строк символов HEX-кодами, например HEX(I5B38C), позволяет обрабатывать последовательность любых 8 битных кодов любых символов.

Для обработки символьных переменных в языке БЭЙСИК предусмотрены встроенные функции LEN, STR, POS, VAL, NUM (см.табл.3 приложения).

2.I. Функция определения длины символьной переменной LEN

Ф у н к ц и я LEN дает цифровое значение, равное количеству символов в строке.

Форма записи функции:

LEN <символьная переменная> .

Пробелы в конце строки не учитываются. Если строка содержит одни пробелы, то функция LEN принимает значение, равное единице, т.к. первый пробел воспринимается функцией как символ, а второй - как конечный пробел.

Пример:

```
10 A$="ROMB1 RING2 "
20 B=LEN(A$)
30 PRINT B
 11
```

Количество символов в символьной переменной равно II. Пробел в середине переменной учитывается как символ. Пробел в конце переменной не учитывается.

Пример:

```
10 A$="ROMB1 RING2 "
20 B=2+SQR(LEN(A$)-2)
30 PRINT B
 5
```

в 20 строке программы вычисляется квадратный корень из выражения, в состав которого входит функция LEN . Значение этой функции равно 11. Переменная B принимает значение, равное 5.

2.2. Функция выделения строки символов STR

Функция STR позволяет выделять отдельный символ или группу символов из символьной переменной.

Форма записи функции:

STR(<символьная переменная>,<номер символа>
[,<количество символов>]),

где <номер символа> - номер, с которого начинается выборка символов из символьной переменной;

<количество символов> - число выбираемых символов.

Если параметр <количество символов> не задан, то он приравнивается остатку заданной строки.

Пример:

```
10 A$="ROMA"
20 B$=STR(A$,2,2)
30 PRINT B$
OM
```

В символьной переменной "ROMA" со второго символа выбираются два символа . Новое символьное значение "OM" присваивается переменной B$.

Функция STR может быть записана как слева, так и справа от знака равно.

Пример:

```
10 A$="ПРОФЕССИОНАЛ"
20 B$="ДОКТОР"
30 STR(A$,8,5)=STR(B$,5,2)
40 PRINT A$
ПРОФЕССОР
```

В этом случае пять последних символов переменной A$ заменяются двумя последними символами переменной B$. В результате переменная A$ получает новое значение "ПРОФЕССОР".

2.3. Функция определения номера символа в строке POS

Ф у н к ц и я **POS** определяет положение символа в строке, отвечающего заданному условию при сравнении.

Форма записи функции:

POS(<символьная переменная><операция сравнения>
{<"символ алфавита языка">}
{<символьная переменная>} .

Первый же символ, удовлетворяющий заданному условию, завершает сравнение и его номер фиксируется. Если ни один из символов заданной строки не удовлетворяет этому условию, то функция POS принимает значение, равное 0 .

Пример:

```
10 A$="ROMBUS5"
20 B=POS(A$="B")
30 PRINT B
 4
```

Символ "B" последовательно сравнивается с каждой буквой переменной A$. Сравнение завершается на четвертом символе.

Пример:

```
10 A$="ВЕРОНИКА"
20 B$="ВЕРА"
30 FOR I=1TO8
40 S$=STR(B$,I,1)
60 IF POS(A$=S$)<>ITHEN80
65 PRINT S$
70 NEXT I
80 PRINT "I=";I
В
Е
Р
I= 4
```

В программе попарно сравниваются символы двух символьных переменных A$ и B$. Сравнение заканчивается на четвертой позиции при несовпадении символов.

2.4. Функция преобразования кода символа в десятичное число VAL

Ф у н к ц и я VAL предназначена для преобразования HEX-кода первого символа символьной переменной или константы в десятичное число.

Форма записи функции:

VAL { (<символьная переменная>) / (<символьная константа>) } .

Пример:

```
10 A$="FBL"
20 X=VAL(A$)
30 PRINT X
 70
```

Первому символу "F" символьной переменной A$ соответствует HEX-код 46 (см.табл.2 приложения) или двоичный код 0I00 0II0 (см.табл.2.I) . При переводе 8 битного кода 0I00 0II0 из двоичной формы в десятичную имеем:

$$0x2^7+Ix2^6+0x2^5+0x2^4+0x2^3+Ix2^2+Ix2^I+0x2^0 = 70.$$

2.5. Функция определения количества символов числа NUM

Ф у н к ц и я NUM предназначена для определения количества символов, принадлежащих числу, в символьной переменной, начинающейся с символа числа. К символам числа относятся символы, с помощью которых можно записать целое или дробное число: 0 I 2 3 4 5 6 7 8 9 . + - E пробел. .

Форма записи функции:

NUM <символьная переменная> .

Функция NUM принимает значение 16, если символьная переменная начинается с символа числа и содержит только числовые символы.

Пример:

```
10 AX="135.67E-02"
20 B=NUM(AX)
30 PRINT B
 16
```

Функция NUM принимает значение количества символов в символьной переменной, если символьная переменная начинается с символа числа и содержит другие (не числовые) символы.

Пример:

```
5 DIM AX32
10 AX="-135.67E-2              A"
20 B=NUM(AX)
30 PRINT B
 24
```

Функция NUM принимает значение 0, если символьная переменная начинается не с числового символа.

Пример:

```
10 AX="ABCD  -135.67E-02"
20 B=NUM(AX)
30 PRINT B
 0
```

3. ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Операторы ADD, OR, XOR, AND, BOOL выполняют различные логические операции над битовыми структурами символов символьных переменных.

Логическими аргументами этих операций могут быть: символьная переменная и двузначное шестнадцатеричное число или две символьные переменные.

В первом случае шестнадцатеричные цифры добавляются к каждому байту символьной переменной. Исключение составляет оператор ADD C. Во втором случае, если символьные переменные имеют разную длину, то недостающие позиции более короткой переменной дополняются слева нулями. Результат выполнения логической операции присваивается первому операнду. Если результат длиннее первого операнда, то старшие разряды результата отбрасываются.

3.1. Операция сложения по модулю 2 (ADD).

О п е р а т о р ADD предназначен для сложения логических аргументов.

Форма записи оператора:

ADD C(<ЛОГИЧЕСКИЕ АРГУМЕНТЫ>) .

При исполнении оператора ADD побитно сравниваются два аргумента. Если один из сравниваемых битов равен 1, а другой 0, то результирующему биту присваивается значение 1, в противном случае - 0.

Параметр C в операторе ADD не является обязательным. Он указывает на поразрядный перенос между складываемыми байтами.

Пример:

```
10 DIM B%2                      B% = 00010010 10110010  (12B2)
20 B%=HEX(12B2)                      10101000 10101000  (A8A8)
30 ADD (B%,A8)
40 HEXPRINT B%                       10111010 01011010  (BA5A)

BA5A
```

Выполняется сложение символьной переменной "12B2" с двузначным шестнадцатеричным числом A8. При сложении шестнадцатеричные цифры A8 добавляются к каждому байту символьной переменной B$. Перенос между байтами отсутствует.

Пример:

```
10 DIM B%2                      B% = 00010010 10110010  (12B2)
20 B%=HEX(12B2)                      00000000 10101000    (A8)
30 ADD C (B%,A8)
40 HEXPRINT B%                       00010011 01011010  (135A).

135A
```

Осуществляется сложение символьной переменной "I2B2" с двузначным шестнадцатеричным числом A8 при наличии переноса между байтами. Второй байт шестнадцатеричного числа A8 слева дополняется нулями.

Пример:

```
10 DIM A%1,B%1                  A%=  1110 1100   (EC)
20 A%=HEX(EC):B%=HEX(39)        B%=  0011 1001   (39)
30 ADD (A%,B%)
40 HEXPRINT A%                       0010 0101   (25)
25
```

Складываются символьные переменные A$="EC" и B$="39" одинаковой длины. Символьная переменная A$ получает новое значение "25".

3.2. Операция логического сложения (OR)

О п е р а т о р OR выполняет логическую операцию "ИЛИ".

Форма записи оператора:

OR(<ЛОГИЧЕСКИЕ АРГУМЕНТЫ>) .

При исполнении оператора OR побитно сравниваются два аргумента. Если хотя бы один из сравниваемых битов равен I, результирующий бит получает значение I, в противном случае - 0 .

Пример:

```
10 DIM B$2                          B$  00010010 10110010  (12B2)
20 B$=HEX(12B2)                         10101000 10101000  (A8A8)
30 OR(B$,A8)
40 HEXPRINT B$                          10111010 10111010  (BABA) .
BABA
```

Складывается символьная переменная "I2B2" с шестнадцатеричным числом A8 . Символьная переменная B$ принимает новое значение "BABA".

Пример:

```
10 DIM A$1,B$1
20 A$=HEX(EC):B$=HEX(39)          A$=  1110 1100   (EC)
30 OR(A$,B$)                      B$=  0011 1001   (39)
40 HEXPRINT A$
FD                                     1111 1101   (FD) .
```

Складываются две символьные переменные A$="EC" и B$="39" . В результате сложения символьная переменная A$ принимает новое значение "FD" .

3.3. Операция равнозначности (XOR)

О п е р а т о р XOR выполняет логическую операцию равнозначности или эквивалентности.

Форма записи оператора:

XOR (<логические аргументы>) .

При исполнении оператора XOR побитно сравниваются два аргумента. Если сравниваемые биты одинаковы, то результирующему биту присваивается значение 0. В противном случае - значение I.

Пример:

```
10 DIM B$2                    B$  00010010 10110010  (12B2)
20 B$=HEX(12B2)                   10101000 10101000  (A8A8)
30 XOR(B$,A8)
40 HEXPRINT B$                    10111010 00011010  (BA1A)
BA1A
```

Операция XOR выполняется над символьной переменной "I2B2" и шестнадцатеричным числом A8. Символьная перемекная B$ принимает новое значение "BAIA" .

Пример:

```
10 DIM A$1,B$1                A$=  1110 1100   (EC)
20 A$=HEX(EC):B$=HEX(39)      B$=  0011 1001   (39)
30 XOR(A$,B$)
40 HEXPRINT A$                     1101 0101   (D5)
D5
```

Операция XOR выполняется над двумя символьными переменными A$="EC" и B$="39" . Переменная A$ принимает новое значение "D5" .

3.4. Операция логического умножения (AND)

О п е р а т о р AND выполняет операцию умножения логических аргументов.

Форма записи оператора:

AND(<ЛОГИЧЕСКИЕ АРГУМЕНТЫ>)

При исполнении оператора AND побитно сравниваются два аргумента. Если соответствующие биты сравниваемых аргументов равны I, то результирующий бит принимает значение I. В противном случае - 0.

Пример:

```
10 DIM B$2                    B$  00010010 10110010  (12B2)
20 B$=HEX(12B2)                   10101000 10101000  (A8A8)
30 AND(B$,A8)
40 HEXPRINT B$                    00000000 10100000  (00A0)
00A0
```

Операция AND выполняется над символьной переменной "I2B2" и шестнадцатеричным числом A8. Символьная переменная B$ принимает новое значение "00A0" .

Пример:

```
10 DIM A$1,B$1                A$=  1110 1100   (EC)
20 A$=HEX(EC):B$=HEX(39)      B$=  0011 1001   (39)
30 AND(A$,B$)
40 HEXPRINT A$                     0010 1000   (28)
28
```

Операция AND выполняется над двумя символьными переменными A$="EC" и B$="39" . Переменная A$ принимает новое значение "28" .

3.5. Операции булевых функций (BOOL)

О п е р а т о р BOOL предназначен для выполнения любой из 16 логических операций, приведенных в табл.4 приложения. Операции применены к двум логическим аргументам: X - 1100 и Y - 1010. Результат всегда присваивается первому аргументу.

Форма записи оператора:

BOOL<номер функции>(<логические аргументы>) ,

где <номер функции> - шестнадцатеричная цифра .

Выполнение каждой из операций оператора BOOL проследим над символьной переменной A$="2841" и шестнадцатеричным числом "FA" .

О п е р а ц и я 0 - константа нуль .

При исполнении операции все биты первого аргумента принимают значение нуль независимо от битов второго аргумента.

Пример:

```
10 DIM A$2
20 A$=HEX(2841)
30 BOOL 0(A$,FA)
40 HEXPRINT A$
0000
```

```
A$=  00101000 01000001  (2841)
     11111010 11111010  (FAFA)

     00000000 00000000  (0000)
```

О п е р а ц и я 1 - отрицание дизъюнкции, стрелка Пирса. При исполнении операции результирующий бит принимает значение, равное 1, если оба сравниваемых бита равны 0. В противном случае его значение становится равным 0.

Пример:

```
10 DIM A$2
20 A$=HEX(2841)
30 BOOL 1(A$,FA)
40 HEXPRINT A$
0504
```

```
A$=  00101000 01000001  (2841)
     11111010 11111010  (FAFA)

     00000101 00000100  (0504)
```

О п е р а ц и я 2 - запрет по X, отрицание импликации. При исполнении операции результирующий бит принимает значение I, если бит первой переменной равен 0, а бит второй переменной равен I. В противном случае его значение становится равным 0 .

Пример:

```
10 DIM A%2                    A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                    11111010 11111010  (FAFA)
30 BOOL 2(A%,FA)
40 HEXPRINT A%                     11010010 10111010  (D2BA)
D2BA
```

О п е р а ц и я 3 - инверсия X, функция НЕ .

При исполнении операции все биты первого аргумента меняют свое значение на противоположное, нули становятся единицами, а единицы - нулями.

Пример:

```
10 DIM A%2                    A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                    11111010 11111010  (FAFA)
30 BOOL 3(A%,FA)
40 HEXPRINT A%                     11010111 10111110  (D7BE)
D7BE
```

О п е р а ц и я 4 - запрет по У, отрицание импликации. При исполнении операции результирующий бит принимает значение, равное I, если бит первой переменной равен I, а бит второй переменной равен 0. В противном случае его значение становится равным 0 .

Пример:

```
10 DIM A%2                    A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                    11111010 11111010  (FAFA)
30 BOOL 4(A%,FA)
40 HEXPRINT A%                     00000000 00000001  (0001)
0001
```

О п е р а ц и я 5 - инверсия У, функция НЕ .

При исполнении операции все биты второго аргумента меняют свое значение на противоположное, нули становятся единицами, а единицы - нулями. Полученное значение присваивается первому аргументу.

Пример:

```
10 DIM A%2                  A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                  11111010 11111010  (FAFA)
30 BOOL 5(A%,FA)
40 HEXPRINT A%                   00000101 00000101  (0505)
0505
```

О п е р а ц и я 6 - неравнозначность, сложение по модулю 2 .

При исполнении программы результирующий бит принимает значение, равное I, если биты первой и второй переменной не совпадают. В противном случае результирующий бит принимает значение 0 .

Пример:

```
10 DIM A%2                  A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                  11111010 11111010  (FAFA)
30 BOOL 6(A%,FA)
40 HEXPRINT A%                   11010010 10111011  (D2BB)
D2BB
```

О п е р а ц и я 7 - отрицание конъюнкции, штрих Шеффера.

При исполнении операции результирующий бит принимает значение, равное I, если хотя бы один из сравниваемых битов равен 0 . В противном случае результирующий бит принимает значение, равное 0 .

Пример:

```
10 DIM A%2                  A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                  11111010 11111010  (FAFA)
30 BOOL 7(A%,FA)
40 HEXPRINT A%                   11010111 10111111  (D7BF)
D7BF
```

О п е р а ц и я 8 - конъюнкция, логическое умножение. При исполнении операции результирующий бит принимает значение, равное I, если оба сравниваемых бита равны I. В противном случае результирующий бит принимает значение, равное 0.

Пример:

```
10 DIM A%2                    A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                    11111010 11111010  (FAFA)
30 BOOL 8(A%,FA)
40 HEXPRINT A%                     00101000 01000000  (2840)
2840
```

О п е р а ц и я 9 - равнозначность, эквивалентность. При исполнении операции результирующий бит принимает значение, равное I, если оба сравниваемых бита совпадают. В противном случае результирующий бит принимает значение, равное 0 .

Пример:

```
10 DIM A%2                    A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                    11111010 11111010  (FAFA)
30 BOOL 9(A%,FA)
40 HEXPRINT A%                     00101101 01000100  (2D44)
2D44
```

О п е р а ц и я A - повторение по У, импликация. При исполнении этой операции результат всегда принимается равным первому операнду.

Пример:

```
10 DIM A%2                     A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                     11111010 11111010  (FAFA)
30 BOOL A(A%,FA)
40 HEXPRINT A%                      11111010 11111010  (FAFA)
FAFA
```

О п е р а ц и я B – импликация от X к У.

При исполнении операции результирующий бит принимает значение I, если бит первой переменной равен 0 или бит второй переменной равен I. В противном случае он становится равным 0 .

Пример:

```
10 DIM A%2                  A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                  11111010 11111010  (FAFA)
30 BOOL B(A%,FA)
40 HEXPRINT A%                   11111111 11111110  (FFFE)
FFFE
```

О п е р а ц и я C – повторение по X, импликация.

При исполнении этой операции результат всегда принимается равным второму операнду.

Пример:

```
10 DIM A%2                  A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                  11111010 11111010  (FAFA)
30 BOOL C(A%,FA)
40 HEXPRINT A%                   00101000 01000001  (2841)
2841
```

О п е р а ц и я D – импликация от У к X.

При исполнении операции результирующий бит принимает значение I, если бит первой переменной равен I или бит второй переменной равен 0. В противном случае он становится равным 0 .

Пример:

```
10 DIM A%2                  A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                  11111010 11111010  (FAFA)
30 BOOL D(A%,FA)
40 HEXPRINT A%                   00101101 01000101  (2D45)
2D45
```

О п е р а ц и я E - дизъюнкция, логическое сложение. При исполнении операции результирующий бит принимает значение I, если хотя бы один из сравниваемых битов равен I. В противном случае он принимает значение 0 .

Пример:

```
10 DIM A%2                A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                11111010 11111010  (FAFA)
30 BOOL E(A%,FA)
40 HEXPRINT A%                 11111010 11111011  (FAFB)
FAFB
```

О п е р а ц и я F - константа единица.

При исполнении операции все биты первого операнда принимают значение, равное единице, независимо от битов второго аргумента.

Пример:

```
10 DIM A%2                A%=  00101000 01000001  (2841)
20 A%=HEX(2841)                11111010 11111010  (FAFA)
30 BOOL F(A%,FA)
40 HEXPRINT A%                 11111111 11111111  (FFFF)
FFFF
```

4. ОПЕРАЦИИ ПРЕОБРАЗОВАНИЯ ИНФОРМАЦИИ

Операторы BIN, CONVERT, ROTATE, INIT, PACK, UNPACK выполняют различные преобразования над символьными и цифровыми переменными (см. табл. 5 приложения).

4.1. Операция преобразования десятичного числа в двоичное (BIN)

О п е р а т о р **BIN** предназначен для преобразования целой части арифметического выражения в двоичное число (байт) и присвоения полученного значения первому символу символьной переменной.

Форма записи оператора:

BIN (<символьная переменная>) = <арифметическое выражение> .

Целая часть арифметического выражения должна быть не больше 255.

Пример:

```
10 DIM B$1
20 X=8.2:B$="1F3A"
30 BIN(B$)=X*2+3.5
40 HEXPRINT B$
50 PRINT B$

46
F
```

В 30-й строке программы вычисляется целая часть арифметического выражения, равная 70, что соответствует двоичному коду 01000110 . В шестнадцатеричной системе счисления это число 46 или HEX-код символа F (см. табл. 2 приложения). Следовательно, в первом байте символьной переменной B$ будет записан символ F .

4.2. Операции преобразования символьной переменной в цифровое значение или цифровой переменной в символьное значение (CONVERT)

О п е р а т о р CONVERT имеет две формы.

Форма I оператора CONVERT служит для преобразования символьной переменной в цифровое значение и присвоение его цифровой переменной:

CONVERT <символьная переменная> TO <цифровая переменная> .

Пример:
```
10 A$="4231"
20 CONVERT A$TOX
30 PRINT X

 4231
```

В результате выполнения оператора CONVERT цифровая переменная X получает значение 4231.

Форма 2 оператора CONVERT служит для преобразования значения цифровой переменной в символьное значение по формату, заданному в операторе, и присвоения его символьной переменной:

CONVERT <арифметическое выражение> TO
<символьная переменная>,(<формат>) .

Правила задания формата в операторе CONVERT те же, что и в операторе PRINTUSING . Число знаков, указанных в формате, не должно превышать I3.

Пример:
```
10 X=4231
20 CONVERT XTOA$,(####)
30 PRINT A$

4231
```

При выполнении оператора CONVERT символьная переменная A$ принимает значение "4231".

Пример:

```
10 X=32.78
20 B$="ALFACON"
30 CONVERT XTOSTR(B$,2,2),(##)
40 PRINT B$
A32ACON
```

При выполнении оператора CONVERT второй и третий символы символьной переменной B$="ALFACON" подменяются значением целой части цифровой переменной X и символьная переменная B$ получает новое значение "A32ACON".

4.3. Операция сдвига битов символьной переменной (ROTATE)

О п е р а т о р ROTATE предназначен для циклического сдвига влево битов каждого символа в символьной переменной на указанное число позиций.

Форма записи оператора:

ROTATE (<символьная переменная>,<число сдвигов>) ,

где <число сдвигов>- цифра от 1 до 7 .

При сдвиге старшие разряды становятся на место младших.

Пример:

```
10 DIM A$2
20 A$=HEX(1A2B):HEXPRINT A$
30 ROTATE (A$,3)
40 HEXPRINT A$
50 PRINT A$
1A2B
D059
ПУ
```

```
A$=   00011010 00101011  (1A2B)
      00110100 01010110  1 СДВИГ
      01101000 10101100  2 СДВИГ
      11010000 01011001  3 СДВИГ.
```

При выполнении оператора ROTATE все биты символьной переменной A$ сдвигаются на три разряда влево. При последнем сдвиге старший разряд байта занимает место младшего разряда. В результате циклического сдвига переменная A$ принимает новое значение "D059".

4.4. Операция присвоения символьным переменным или символьным массивам заданного значения (INIT)

О п е р а т о р INIT присваивает символьным переменным или символьным массивам значение параметра, заданного в операторе в круглых скобках.

Форма записи оператора:

INIT (<параметр>)<аргумент>[, <аргумент>...] ,

где <параметр>- двузначный шестнадцатеричный код, символ алфавита языка, символьная переменная;

<аргумент>- символьная переменная, символьный массив.

Если в качестве аргумента стоит символьная переменная, то в операторе используется ее первый символ.

Пример:

```
10 DIM A$*1
20 INIT (32)A$
30 HEXPRINT A$
40 PRINT A$

32
2
```

В результате выполнения оператора INIT символьной переменной A$ присваивается значение "32".

Пример:

```
10 DIM A$*1
20 INIT ("A")A$
30 HEXPRINT A$
40 PRINT A$

41
A
```

Символьной переменной A$ присваивается значение "A".

4.5. Операция упаковки значений цифровых переменных и массивов в символьные (PACK)

О п е р а т о р PACK служит для упаковки значений цифровых переменных и массивов в символьные по заданному формату.

Форма записи оператора:

PACK (<Формат>) { <символьная переменная> / <символьный массив> } FROM { <цифровой массив> / <арифметическое выражение> } .

Число знаков, используемых в формате, не должно превышать 13.

Пример:

```
10 DIM A%2
20 X=46.54
30 PACK(##.##)A%FROMX
40 HEXPRINT A%
50 PRINT A%

4654
FT
```

При выполнении оператора PACK цифровая переменная X упаковывается в символьную переменную A$ по формату вещественного числа с фиксированной точкой.

4.6. Операция распаковки данных из символьной переменной или массива в цифровые значения (UNPACK)

О п е р а т о р UNPACK предназначен для распаковки данных, упакованных с использованием оператора PACK, из символьной переменной или массива в цифровые значения.

Форма записи оператора:

UNPACK (<Формат>) { <символьная переменная> / <символьный массив> } TO { <цифровая переменная> / <цифровой массив> } .

Формат должен быть тот же, что и при упаковке.

Пример:

```
10 DIM A%2
20 A%="FT"
30 UNPACK(##.##)A%TOX
40 PRINT X

 46.54
```

При выполнении оператора данные, упакованные в символьной переменной A$, распаковываются по тому же формату и запоминаются в цифровой переменной X.

ПРИЛОЖЕНИЕ

Таблица I

Матричные операции языка БЭЙСИК

NN п/п	ОПЕРАТОР		НАЗНАЧЕНИЕ ОПЕРАТОРА	ПРИМЕР ЗАПИСИ
1	MAT REDIM	X,XX	ПЕРЕОПРЕДЕЛЕНИЕ РАЗМЕРНОСТИ МАТРИЦЫ	MATREDIMA(4,6)
2	MAT INPUT	X,XX	ВВОД ЭЛЕМЕНТОВ МАТРИЦЫ	MATINPUT A,B*
3	MAT READ	X,XX	ВЫБОРКА ДАННЫХ ИЗ БЛОКА DATA	MATREAD A,B*
4	MAT PRINT	XX	ПЕЧАТЬ МАТРИЦ	MATPRINT A,B*
5	MAT ZER	X	ЗАПОЛНЕНИЕ МАТРИЦЫ НУЛЯМИ	MAT A=ZER
6	MAT CON	X	ЗАПОЛНЕНИЕ МАТРИЦЫ ЕДИНИЦАМИ	MAT A=CON
7	MAT IDN	X	ПРИВЕДЕНИЕ МАТРИЦЫ К ЕДИНИЧНОЙ	MAT A=IDN
8	MAT =	XXX	ПРИСВОЕНИЕ ЭЛЕМЕНТАМ ОДНОЙ МАТРИЦЫ ЗНАЧЕНИЙ ДРУГОЙ МАТРИЦЫ	MAT A=B
9	MAT +	XXX	СЛОЖЕНИЕ МАТРИЦ	MAT A=B+C
10	MAT -	XXX	ВЫЧИТАНИЕ МАТРИЦ	MAT A=B-C
11	MAT()*	XXX	СКАЛЯРНОЕ УМНОЖЕНИЕ МАТРИЦЫ НА ЧИСЛО	MAT A=(K)*B
12	MAT *	XXX	УМНОЖЕНИЕ МАТРИЦ	MAT C=A*B
13	MAT TRN	XXX	ТРАНСПОНИРОВАНИЕ	MAT A=TRN(B)
14	MAT INV,C	XXX	ОБРАЩЕНИЕ МАТРИЦЫ И ВЫЧИСЛЕНИЕ ОПРЕДЕЛИТЕЛЯ	MAT A=INV(B) MAT A=INV(B),C

Примечание: X – размерность переопределяется явным образом;

XX – применяется для цифровых и символьных матриц и векторов;

XXX – размерность переопределяется неявным образом.

Таблица 2

Шестнадцатеричные коды символов алфавита языка БЭЙСИК
(в порядке убывания "веса" кодов)

СИМВОЛЫ ЯЗЫКА	HEX-КОДЫ СИМВОЛОВ	СИМВОЛЫ ЯЗЫКА	HEX-КОДЫ СИМВОЛОВ	СИМВОЛЫ ЯЗЫКА	HEX-КОДЫ СИМВОЛОВ
Ч	FE	]	5D	=	3D
Щ	FD	\	5C	<	3C
Э	FC	[	5B	;	3B
Ш	FB	Z	5A	:	3A
Э	FA	Y	59	@	40
Ы	F9	X	58	9	39
Ь	F8	W	57	8	38
В	F7	V	56	7	37
Ж	F6	U	55	6	36
У	F5	T	54	5	35
Т	F4	S	53	4	34
С	F3	R	52	3	33
Р	F2	Q	51	2	32
Я	F1	P	50	1	31
П	F0	O	4F	0	30
О	EF	N	4E	/	2F
Н	EE	M	4D	.	2E
М	ED	L	4C	-	2D
Л	EC	K	4B	,	2C
К	EB	J	4A	+	2B
Й	EA	I	49	*	2A
И	E9	H	48	)	29
Х	E8	G	47	(	28
Г	E7	F	46	'	27
Ф	E6	E	45	&	26
Е	E5	D	44	%	25
Д	E4	C	43	$	24
Ц	E3	B	42	#	23
Б	E2	A	41	"	22
А	E1	?	3F	!	21
Ю	E0	>	3E	ПРОБЕЛ	20

Таблица 3

Функции символьных переменных

NN п/п	ФУНКЦИЯ	НАЗНАЧЕНИЕ ФУНКЦИИ	ПРИМЕР ЗАПИСИ
1	LEN	ОПРЕДЕЛЕНИЕ КОЛИЧЕСТВА СИМВОЛОВ В СИМВОЛЬНОЙ ПЕРЕМЕННОЙ	B= LEN(A$)
2	STR	ВЫДЕЛЕНИЕ ОТДЕЛЬНОГО СИМВОЛА ИЛИ ГРУППЫ СИМВОЛОВ ИЗ СИМВОЛЬНОЙ ПЕРЕМЕННОЙ	B$= STR(A$,2,3)
3	VAL	ПРЕОБРАЗОВАНИЕ HEX-КОДА ПЕРВОГО СИМВОЛА СИМВОЛЬНОЙ ПЕРЕМЕННОЙ В ДЕСЯТИЧНОЕ ЧИСЛО	X= VAL(A$)
4	POS	ОПРЕДЕЛЕНИЕ ПЕРВОГО СИМВОЛА В СИМВОЛЬНОЙ ПЕРЕМЕННОЙ, УДОВЛЕТВОРЯЮЩЕГО ЗАДАННОМУ УСЛОВИЮ	B= POS(A$=F)
5	NUM	ОПРЕДЕЛЕНИЕ КОЛИЧЕСТВА ЦИФРОВЫХ СИМВОЛОВ В СИМВОЛЬНОЙ ПЕРЕМЕННОЙ, НАЧИНАЮЩЕЙСЯ С ЦИФРЫ	L= NUM(A$)

Таблица 4

Логические операции оператора BOOL

НОМЕР ОПЕРАЦИИ	АРГУМЕНТЫ X	АРГУМЕНТЫ Y	ВЫПОЛНЯЕМАЯ ФУНКЦИЯ	НАЗВАНИЕ ФУНКЦИИ
	1	1		
	1	0		
	0	1		
	0	0		
0	0	0	0	КОНСТАНТА НУЛЬ
1	0	0	$X=\bar{X}\&\bar{Y}$	ОТРИЦАНИЕ ДИЗЪЮНКЦИИ (СТРЕЛКА ПИРСА)
2	0	1	$X=\bar{X}\&Y$	ОТРИЦАНИЕ ИМПЛИКАЦИИ (ЗАПРЕТ ПО X)
3	0	1	$X=\bar{X}\&\bar{Y}\vee\bar{X}\&Y=\bar{X}$	ОТРИЦАНИЕ X (ФУНКЦИЯ "НЕ")
4	1	0	$X=X\&\bar{Y}$	ОТРИЦАНИЕ ИМПЛИКАЦИИ (ЗАПРЕТ ПО Y)
5	1	0	$X=\bar{X}\&\bar{Y}\vee X\&\bar{Y}=\bar{Y}$	ОТРИЦАНИЕ Y (ФУНКЦИЯ "НЕ")
6	1	1	$X=X\&\bar{Y}\vee\bar{X}\&Y$	СУММА ПО MOD 2 (НЕРАВНОЗНАЧНОСТЬ)
7	1	1	$X=\bar{X}\vee\bar{Y}=\overline{X\&Y}$	ОТРИЦАНИЕ КОНЪЮНКЦИИ (ШТРИХ ШЕФФЕРА)
8	0	0	$X=X\&Y$	КОНЪЮНКЦИЯ (ЛОГИЧЕСКОЕ УМНОЖЕНИЕ)
9	0	0	$X=X\&Y\vee\bar{X}\&\bar{Y}$	РАВНОЗНАЧНОСТЬ (ЭКВИВАЛЕНТНОСТЬ)
A	0	1	$X=X\&Y\vee\bar{X}\&Y=Y$	ИМПЛИКАЦИЯ ПО Y
B	0	1	$X=\bar{X}\vee Y$	ИМПЛИКАЦИЯ ОТ X К Y
C	1	0	$X=X\&Y\vee X\&\bar{Y}=X$	ИМПЛИКАЦИЯ ПО X
D	1	0	$X=X\vee\bar{Y}$	ИМПЛИКАЦИЯ ОТ Y К X
E	1	1	$X=X\vee Y=\overline{\bar{X}\&\bar{Y}}$	ДИЗЪЮНКЦИЯ (ЛОГИЧЕСКОЕ СЛОЖЕНИЕ)
F	1	1	1	КОНСТАНТА ЕДИНИЦА

Таблица 5

Функции логических переменных и операторы преобразования информации

NN п/п	ОПЕРАТОР	НАЗНАЧЕНИЕ ОПЕРАТОРА	ПРИМЕР ЗАПИСИ
1	ADD	СЛОЖЕНИЕ ПО МОДУЛЮ 2	ADD(A*,FA) ADD C(A*,B*)
2	OR	ОПЕРАЦИЯ "ИЛИ" (ДИЗЬЮНКЦИЯ)	OR(A*,FA) OR(A*,B*)
3	XOR	ИСКЛЮЧАЮЩЕЕ "ИЛИ"	XOR(A*,FA) XOR(A*,B*)
4	AND	ОПЕРАЦИЯ "И" (КОНЬЮНКЦИЯ)	AND(A*,FA) AND(A*,B*)
5	BOOL	ВЫПОЛНЕНИНЕ ЛЮБОЙ ИЗ 16 ВОЗМОЖНЫХ ЛОГИЧЕСКИХ ОПЕРАЦИЙ	BOOL N(A*,FA)
6	BIN	ПРЕОБРАЗОВАНИЕ ЦИФРОВОЙ ИНФРМАЦИИ В СИМВОЛЬНУЮ ФОРМУ	BIN(A*)=XA2+3.5
7	CONVERT	ПРЕОБРАЗОВАНИЕ СИМВОЛЬНОЙ ПЕРЕМЕННОЙ В ЦИФРОВУЮ ФОРМУ ПРЕОБРАЗОВАНИЕ ЦИФРОВОЙ ПЕРЕМЕННОЙ В СИМВОЛЬНУЮ ФОРМУ	CONVERT A* TO X CONVERT X TO A*, (##.###)
8	ROTATE	ЦИКЛИЧЕСКИЙ СДВИГ КОДА СИМВОЛЬНОЙ ПЕРЕМЕННОЙ ВЛЕВО НА ЗАДАННОЕ КОЛИЧЕСТВО РАЗРЯДОВ	ROTATE(A*,3)
9	INIT	ПРИСВОЕНИЕ СИМВОЛЬНОЙ ПЕРЕМЕННОЙ (МАССИВУ) ЗНАЧЕНИЯ ПАРАМЕТРА, ЗАДАННОГО В СКОБКАХ	INIT(12)A* INIT("A")B*()
10	PACK	УПАКОВКА ЗНАЧЕНИЙ ЦИФРОВЫХ ПЕРЕМЕННЫХ И МАССИВОВ В СИМВОЛЬНЫЕ ПО ЗАДАННОМУ ФОРМАТУ	PACK(##.#)A* FROM X
11	UNPACK	РАСПАКОВКА ДАННЫХ ИЗ СИМВОЛЬНЫХ ПЕРЕМЕННЫХ В ЦИФРОВЫЕ	UNPACK(##.#)A* TO X

СПИСОК ЛИТЕРАТУРЫ

1. Машина вычислительная электронная клавишная программуправляемая "ИСКРА-226". Инструкция по программированию. Матрицы. 1.320.136 Д14-3. Курск.:з-д Счетмаш, 1983, 120 с.

2. Машина вычислительная электронная клавишная программуправляемая "ИСКРА-226". Инструкция по программированию.Базовый объем. 1.320.136 Д14-1. Курск.: з-д Счетмаш, 1983, ч.1, 135 с.

3. Кетков Ю.Л. Программирование на БЭЙСИКЕ. М.: Статистика, 1978, 158 с.

4. Уорт Т. Программирование на языке БЭЙСИК. М.: Машиностроение, 1981, 255 с.

5. Трэктон К. Программы на БЭЙСИКЕ для инженерно-технических расчетов. М.: Радио и связь, 1985, 96 с.

6. Нагинаев В.Н., Чернухин С.И., Шахунянц Т.Г. Программирование на алгоритмическом языке БЭЙСИК ЭВМ "ИСКРА-226". Методические указания к практическим и лабораторным работам. М.: МИИТ, 1985, 52 с.

С О Д Е Р Ж А Н И Е

ОСНОВЫ ЯЗЫКА БЭЙСИК ЭВМ ИСКРА-226

Часть II

Методические указания

по дисциплине

"Программирование на ЭВМ ИСКРА-226"

для слушателей факультета повышения квалификации,

преподавателей и студентов

Редактор М.И. Амелина

Технический редактор О.А. Овечкина

Корректор Е.А.Субботина

Формат 60x84 1/16	Подписано к печати 23.02.87	
Изд. №24-87	Усл.печ.л. 3,25	Уч.-изд.л. 2,0
Тираж 300 экз.	Заказ № 434	Бесплатно

Редакционно-издательский отдел МИИТа
101475, Москва, А-55, ул.Образцова,15
Типография МИИТа