Условия задач заключительного этапа

Задача 1. RAR

Платежные терминалы получают индивидуальные пакеты обновлений для установленного в них программного обеспечения через сеть. При этом в целях безопасности эти пакеты пересылаются в зашифрованных архивах. Пароли шифрования для терминалов разные и администратору не известны.

Администратор на CD-R диске получил очередной незашифрованный пакет обновлений (файл *apu_test.rar*) для отладочного терминала. В ходе проверки данного пакета антивирусом оказалось, что ни один из файлов не содержит вредоносного кода.

Вместе с тем стало известно, что злоумышленники запланировали атаку на один из терминалов и, возможно, подменили некоторые файлы в пакете обновлений. С помощью специальной программы администратору удалось получить некоторые фрагменты пакета обновлений терминалов.

Проанализируйте эти фрагменты и выясните, какие из файлов в пакете были подменены.

Содержимое архива пакета обновлений с диска администратора:

Имя	Размер	CRC32
..		
apu001.dat	3 167	B2F4FB0E
apu002.dat	906	8234B7C1
apu003.dat	3 328	A5508028
apu004.dat	1 431	7BDBC313
apu005.dat	451	519817B0

Комментарий. К задаче прилагается: исходный архив без вирусов (*apu_test.rar*), фрагменты пакета обновлений (*apu_termX.NNN*), программа-архиватор WinRAR.

Решение

Из условия можно извлечь информацию о содержимом архива, а именно – имена файлов, их размер и контрольная сумма (CRC32). При внесении изменений в файл его имя и размер могут не измениться, однако контрольная сумма при этом будет другой. Учитывая этот факт необходимо в перехваченных фрагментах определить контрольные суммы файлов и сравнить их с исходными.

Чтобы определить структуру информации, хранимую в архиве, необходимо открыть файл с помощью редактора и посмотреть его содержимое в шестнадцатеричном формате.

В этом режиме просмотра необходимо найти имя первого файла – «*apu001.dat*». Нетрудно заметить, что со смещением 16 байт влево от имени файла хранится контрольная сумма в обратном порядке байтов (рис. 1).

```
00000000: 52 61 72 21 1A 07 00 CF|90 73 00 00 0D 00 00 00  | Rar!...Пђs......
00000010: 00 00 00 00 F8 C8 74 20|90 2F 00 01 0C 00 00 5F  | ....иИt ђ/....._
00000020: 0C 00 00 02 0E FB F4 B2|73 BA 4F 49 1D 33 0A 00  | .....ыФІsєOI.3..
00000030: 20 00 00 00 61 70 75 30|30 31 2E 64 61 74 00 B0  |  ...apu001.dat.°
00000040: 93 26 02 15 D7 C5 0C 88|D3 C9 41 DB 82 B5 52 85  | "&..ЧЕ.■УЙАЫ‚µR…
00000050: 20 D0 4A 6B 4A 91 6E BB|B5 5B 6A DF 0D DA ED AD  |  PJkJ`n»µ[jЯ.Ън-
00000060: AB A3 6D 47 6D AA DB A8|9B 5A DB A1 75 BB 52 B7  | «JmGmЄЫЁ>ZЫЎu»R·
00000070: 5A BA BB 6A A5 56 7A 2C|96 AD 0D 7C 40 99 94 88  | Zє»jҐVz,--.|@™"■
00000080: F8 04 44 6C 04 6A 8A 2E|64 C8 66 32 08 B9 98 49  | ш.Dl.jљ.dИf2.№.I
00000090: 99 26 4F 13 00 50 C8 78|33 27 80 41 64 26 FF 3B  | ™&O..PИx3'ЂAd&я;
000000A0: CB 6A 78 5F 89 F7 E0 7D|E7 79 DF 8D 77 9D F8 0F  | Лjx_‰ча}зуЯЌwќш.
000000B0: EE 7E FD F0 FF EB EE EB|BD 88 E4 3F 91 2F FA 91  | о~эрялолS■д?`/ъ`
000000C0: 24 F2 48 67 79 3F A5 D1|FA 5F 1F 0A 30 90 BB D1  | $тHgy?ҐСъ_..0ђ»С
000000D0: B0 27 2A 3F 1A EE 95 74|62 23 46 78 5D E9 29 54  | °'*?.о•tb#Fx]й)Т
000000E0: 2A 46 97 B9 0B 04 F5 F5|D5 BE 76 42 BB BA F7 F0  | *F-№..ххХѕvB»єчр
000000F0: 89 81 EB F2 63 5E C4 8D|87 07 0E 1C FF 1B E9 DA  | ‰Ґлтс^ДЌ‡...я.йЬ
00000100: 3B F5 E8 A1 F1 7E A9 1B|82 A1 C4 FB F9 D5 DC 08  | ;хиЎс~©.‚ЎДыщХЬ.
00000110: 37 12 72 54 30 7C 7C 50|FD AD CE 46 2C 3F 87 02  | 7.rT0||Pэ-ОF,?‡.
00000120: 1B 6A DA D2 D2 23 C5 34|B3 07 29 D9 01 94 03 90  | .jЪТТ#Е4і.)Щ.”.ђ
00000130: DA 2C 29 D2 C9 9E 83 11|01 19 C0 B7 E1 2C EE 61  | Ъ,)ТЙћѓ...А·б,оа
00000140: AC D2 68 DC 43 8F 58 63|A2 CA B3 EF 3B 25 A1 52  | ¬ThЬCЏXcўKin;%ЎR
```

Рис. 1. Имя файла и его контрольная сумма в архиве

Проверив это и для остальных файлов архива, можно убедиться, что для всех файлов структура информации следующая:

[00 02] [xx xx xx xx] [другие данные 12 байт] [имя файла],

где *xx xx xx xx* – 4 байта контрольной суммы.

Для нахождения измененного файла необходимо в представленных фрагментах последовательно осуществлять поиск по имени файла и определять контрольную сумму. При несовпадении контрольной суммы с исходной можно однозначно сделать вывод, что файл был изменен.

Анализируя фрагмент «apu_1.004» можно заметить имя файла «*apu002.dat*» и контрольную сумму **78 95 D4 75**, не соответствущей исходной.

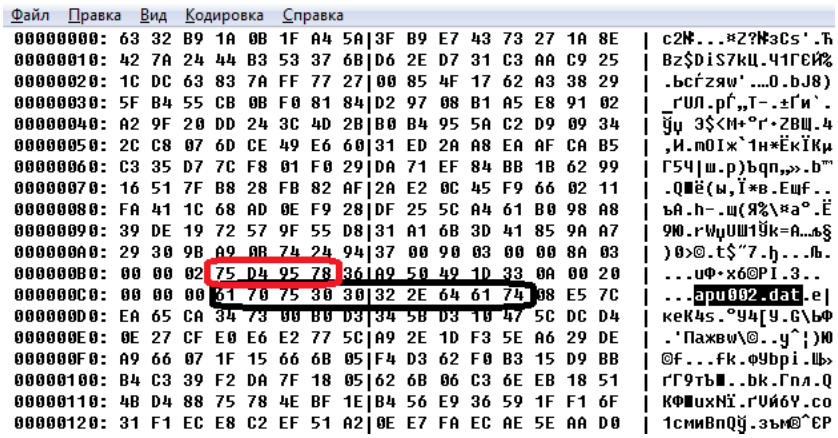

Рис. 2. Анализ фрагмента «apu_1.004»

Аналогично для файла «apu_1.008» было обнаружено имя файла «*apu004.dat*» и контрольная сумма **C0 E2 36 A8**, что не соответствует исходной.

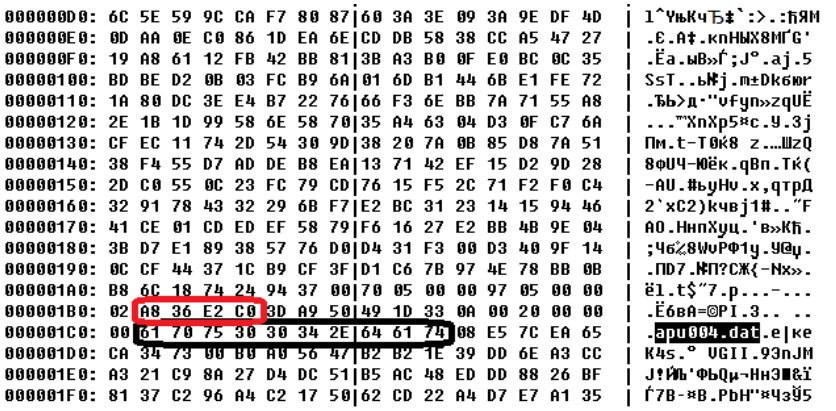

Рис. 3. Анализ фрагмента «apu_1.008»

Для остальных файлов контрольные суммы совпадают с исходной.

Ответ: были изменены файлы «*apu002.dat*» и «*apu004.dat*».

Задача 2. Гамма

В центр обработки информации поступило четыре файла, каждый из которых является зашифрованным представлением изображения формата PNG. Известно, что шифрование осуществлялось методом «двоичного гаммирования», т.е. путем выполнения операции «побитового исключающего ИЛИ» между байтами исходного файла и байтами, полученными циклическим повторением последовательности из 4-х байтов ключа. Сотрудники центра успели расшифровать только три файла с именами *First.png*, *Second.png* и *Third.png*.

Помогите расшифровать оставшийся файл «*Secret.enc*». В ответе укажите слово, изображенное на полученной картинке формата PNG.

Комментарий. К задаче прилагаются файлы с расшифрованными картинками (*First.png*, *Second.png*, *Third.png*), файл с зашифрованной картинкой *Secret.enc*, редактор файлов в шестнадцатеричном формате (HexEditor).

Решение

Рассмотрим первые байты шестнадцатеричного представления всех четырех файлов.

First.png
89 50 4E 47 0D 0A 1A 0A …
Second.png
89 50 4E 47 0D 0A 1A 0A …
Third.png
89 50 4E 47 0D 0A 1A 0A …
Secret.enc
F3 64 5C D7 77 3E 08 9A …

Заметим, что для первых трех изображений формата PNG последовательность начальных байтов совпадает. Следовательно, можно предположить, что и четвертый файл *Secret.enc* после его расшифрования должен содержать аналогичные начальные байты. Так как для шифрования применялся метод «двоичного гаммирования» с длиной ключа 4 байта, то для определения самого ключа достаточно выполнить операцию «побитового исключающего ИЛИ» между первыми 4-мя байтами любого из файлов *First.png*, *Second.png* и *Third.png* и первыми 4-мя байтами файла Secret.enc:

89 50 4E 47 ^ F3 64 5C D7 = 7A 34 12 90

Таким образом, можно предположить, что в качестве ключа использовалась последовательность – **7A 34 12 90**. Далее необходимо применить операцию «побитового исключающего ИЛИ» между всеми байтами файла *Secret.enc* и байтами, полученными циклическим повторением последовательности байтов ключа, то есть **7A 34 12 90 7A 34 12 90 … 7A 34 12 90**. Для этого необходимо написать программное средство, позволяющее выполнить данную операцию в автоматическом режиме. Результат его работы – файл формата PNG, открытие которого с использованием редактора изображений приведет к отображению на экране искомого слова: «**АССЕМБЛЕР**».

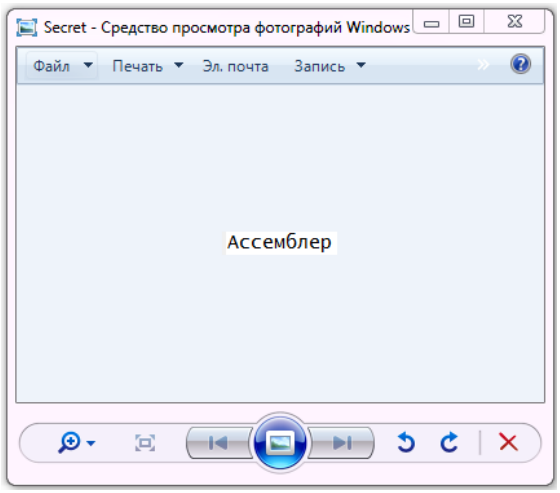

Рис. 4. Результат расшифрования картинки

Ответ: Ассемблер

Задача 3. Маршрутизация

В студенческом городке развернуто 10 локальных вычислительных сетей (ЛВС). В каждой сети есть один маршрутизатор, его номер соответствует номеру сети. Линии связи между маршрутизаторами указаны на рисунке. Соединение с Интернет имеют только маршрутизаторы с номерами 1, 3 и 5.

В служебной части сетевых пакетов имеется счетчик S, который увеличивается на 1 при каждой пересылке между маршрутизаторами. Из Интернет пакеты попадают в сети со счетчиком S = 1.

При поступлении пакета в очередной маршрутизатор с номером R осуществляется анализ его адреса назначения. Если сетевой пакет не предназначен какому-либо узлу из сети маршрутизатора, то он отправляется одному из соседних маршрутизаторов по правилу:

- если S / R < 2, то соседу с минимальным номером;
- если S / R == 2, то соседу со средним значением номера;
- если S / R > 2, то соседу с максимальным номером.

В какую сеть надо отправить пакет из Интернет, чтобы он дошел до сети с номером 10 за минимальное число шагов? Найдите это число шагов.

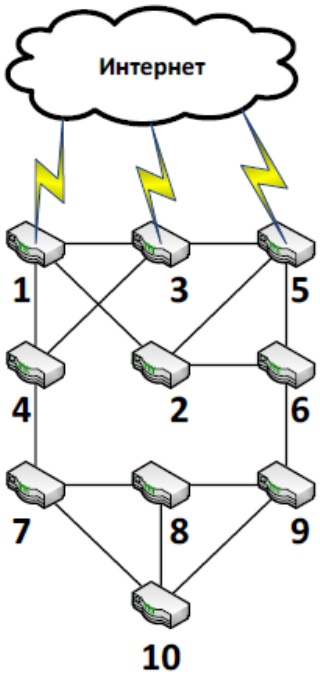

Рис. 5. Схема соединения ЛВС

Решение

Так как в ответе на задачу необходимо указать количество шагов, то заметить какую-либо закономерность (или особенность) и решить задачу теоретически, получив точный ответ, не представляется возможным. Возможно два варианта поиска ответа: ручной и программный.

Для решения задачи вручную и получения обоснованного ответа необходимо проследить пути пакета до целевого (десятого) маршрутизатора для всех возможных входов в маршрутизаторы, подключенные к сети Интернет.

Вход в маршрутизатор №1:

S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
R	1	2	1	4	1	4	1	4	3	5	3	5	3	5	3	5	6	5	6	9	8	9	8	9	8	10

Вход в маршрутизатор №3:

S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
R	3	1	3	1	4	1	4	1	4	3	5	3	5	3	5	6	5	6	9	8	9	8	9	8	10

Вход в маршрутизатор №5:

S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
R	5	2	1	4	1	4	1	4	3	5	3	5	3	5	3	5	6	5	6	9	8	9	8	9	8	10

При вычислении пути пакета, отправленного в маршрутизатор №5, можно заметить, что, начиная с шага 2, этот путь будет совпадать с первым путем и сэкономить время на его вычислении.

Таким образом, получаем, что пакет надо отправить через маршрутизатор №3, и он дойдет до маршрутизатора №10 со значением счетчика S=25, то есть между маршрутизаторами будет сделано 24 шага.

Таким ответ будет, если результат операции деления S на R будем считать целочисленным. А, если решать задачу, считая, что результат операции деления вещественный, то значение счетчика S будет следующим:

19, 16 и 19 шагов соответственно. То есть ответом будет: отправить пакет через маршрутизатор №3, и он дойдет до маршрутизатора №10 за 16 шагов.

Эту задачу удобно решать программным способом, так как ее условие можно формализовать и разработать алгоритм решения. Маршрутизаторы и связи между ними можно программно представить либо в виде матрицы смежности, либо в виде массива «соседей». В матрице смежности размером 10 на 10 ноль в i-ой строке и j-ом столбце будет означать отсутствие связи между маршрутизатором i и маршрутизатором j, а единица – наличие связи. В каждой строке будет по три единицы, означающих наличие переходов в маршрутизаторы с наименьшим, средним и наибольшим номерами (рис.). Вместо матрицы смежности можно создать двумерный массив размером 10 на 3, в каждой строке которого будут последовательно указаны номера маршрутизаторов с наименьшим, средним и наибольшим номерами, соединенных с маршрутизатором, соответствующим номеру этой строки. Далее необходимо реализовать алгоритм обхода графа, представленного одним из описанных выше способов, начиная с каждой из входных вершин и заканчивая в вершине номер 10, соответствующей маршрутизатору №10.

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

Рис. 6. Матрица смежности

Далее приведен пример исходных кодов на языке программирования «C» для программного решения задачи при отправке пакета через маршрутизатор №3 и представления связей между маршрутизаторами в виде массива «соседей». Для вычисления номера следующего маршрутизатора используется результат целочисленного деления S на R.

```
#include <stdio.h> // для использования printf
void main()
{
        // Номера соседних маршрутизаторов
        int Links[10][3] = {{2,3,4},{1,5,6},{1,4,5},
                        {1,3,7},{2,3,6},{2,5,9},{4,8,10},
                        {7,9,10},{6,8,10},{7,8,9}};
    int S = 1;       // счетчик шагов (по условию
    // начинается с 1)
    int R = 1;       // номер текущего роутера
                     // (вариант для отправки в
```

```
                    // маршрутизатор №3)
    // Вывод на экран номера начального
    // маршрутизатора
       printf("%d ", R);
       do
       {
            // Вычисление номера следующего
            // маршрутизатора
            // На 1 меньше, т.к. индексация в массиве
            // с 0
            if (S/R < 2)
                 R = Links[R-1][0];
            else if (S/R == 2)
                 R = Links[R-1][1];
            else
                 R = Links[R-1][2];
            // Увеличение счётчика шагов
            S++;
            // Вывод на экран номера текущего
            // маршрутизатора
            printf("%d ", R);
       }
   while (R != 10);
       // Вывод на экран с новой строки значения
       // счетчика S
       printf("\nS = %d\n", S);
}
```

Результат работы программы представлен на рисунке.

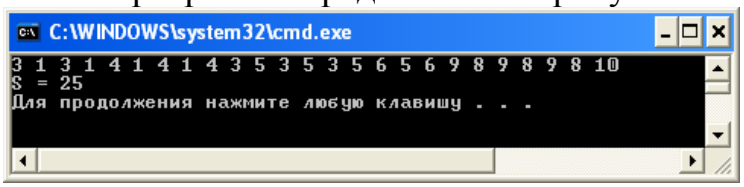

Рис. 7. Программное решение задачи

Ответ:

а) целочисленное деление: пакет надо отправить через маршрутизатор №3, и он дойдет до маршрутизатора №10 со значением счетчика S=25 (24 шага);

б) вещественное деление: пакет надо отправить через маршрутизатор №3, и он дойдет до маршрутизатора №10 за 16 шагов.

Задача 4. Дешифрование

Вася написал свою программу для шифрования сообщений и зашифровал с ее помощью сообщение для своего друга Миши. Чтобы усложнить понимание программы, Вася запутал код и добавил

использование пароля для осуществления операций шифрования и расшифрования.

Вася отправил Мише сообщение:

```
  ="+>j#9j+(%?>j%?8j'//>#$-u
```

и программу, а пароль отправить забыл. Однако Миша легко расшифровал сообщение без пароля.

Помогите Мише расшифровать сообщение и найти ключ.

Листинг программы приведен ниже.

Паскаль

```
Uses crt;
Var a, c, d, e: string; b: integer;
Begin
   Clrscr; c:= '';
   Writeln('Введите текст'); Readln(a);
   Repeat
      Writeln('Введите ключ');
Readln(d);   Until length(d)>=6;
   Repeat
      b:= 1;
      If length(e)<length(a) then
         e:=e+d[b]; b:=b+1;
   Until length(e)>=length(a);
   For b:=1 to length(a) do
      c:=c+chr(byte(a[b]) xor
byte(e[b]));
   For b:=1 to length(a) do write(c[b]);
   Writeln;
   Readln;
End.
```

Си

```
#include <stdio.h>
#include <string.h>
void main()
{
   char a[100]={0}, c[100]={0},
        d[100]={0}, e[100]={0};
   int b;
   printf("Введите текст: ");
   scanf("%s", a);
   do
   {
      printf("Введите ключ: ");
      scanf_s("%s", d, 100);
   }
   while(strlen(d) < 6);
   do
   {
      b = 0;
      if (strlen(e) < strlen(a))
         e[strlen(e)] = d[b];
      b++;
   }
   while(strlen(e) < strlen(a));
   for (b = 0; b < strlen(a); b++)
      c[b] = a[b] ^ e[b];
   for(b = 0; b < strlen(a); b++)
      printf("%c", c[b]);
   printf("\n");
}
```

Решение

Анализируя цикл do .. while():

```
do
   {
      b = 0;
      if (strlen(e) < strlen(a))
```

```
        e[strlen(e)] = d[b];
      b++;
  }
```

можно сделать вывод, что чтрока-ключ (*e*) будет содержать копии символа *d[b]*, при этом b = 0, и количество копий символа равно длине введенного ключа. Длина ключа определяется первым циклом do.. while, и равна 6.

В качестве ответа подойдет ключ из одинаковых символов, равных первому введенному символу, и длиной, равной длине введенного ключа.

Осталось перебрать всевозможные символы по ASCII-таблице и получить ответ.

Ответ: Любой пароль длиннее 6 символов, который начинается на J.

Задача 5. Защитный блок

Промышленная установка управляется по 4-разрядной шине данных. Команды по ней передаются последовательно. Для удобства записи будем интерпретировать их как символы в алфавите 0,1,2,..,9,A,B,C,D,E,F.

Известно, что некоторые цепочки команд приводят к поломке установки. Поэтому на шине планируется установить защитный блок, исправляющий такие цепочки на безопасные. Логика работы защитного блока определяется двумя таблицами. Первая из них определяет следующую активную строку в зависимости от входного символа и текущей активной строки (функция переходов). Вторая таблица определяет, что появится на выходе защитного блока в зависимости от входного символа и текущей активной строки (функция выходов). В начальный момент времени активна строка с номером 0. Фрагмент кода функции работы защитного блока приведен ниже.

Паскаль	**Си**

type matrix=
  array[1..n,1..m] of
  integer;
function GetOutput(
  StateMas : matrix;
  OutMas : matrix;
  InSymb : integer;
  var CurState:integer):
integer;
var
  NewState:integer;
  OutSymb:integer;
begin
  NewState :=
  StateMas[CurState]
  [InSymb];
	int GetOutput(
	  int **StateMas,
	  int **OutMas,
	  int InSymb,
	  int& CurState)
	// StateMas –таблица(матрица)
	переходов
	// OutMas – таблица(матрица) выходов
	// InSymb – входной символ
	// CurState – текущее состояние
	(меняется в результате выполнения
	функции)
	// RETURN – выходной символ
	{
	  int NewState;
	  int OutSymb;

```
 OutSymb :=
   OutMas[CurState]
         [InSymb];
 CurState := NewState;
 result := OutSymb;
end;
```

```
 NewState =
   StateMas[CurState]
           [InSymb];
 OutSymb =
   OutMas[CurState]
         [InSymb];
 CurState = NewState;
 return OutSymb;
}
```

Настройте защитный блок таким образом, чтобы он пропускал все команды, кроме запрещенных, вместо которых на выходе должна появиться безопасная выходная последовательность (см. таблицу).

Запрещенная входная последовательность	**Выходная последовательность**
1F10AE	**1F10A1**

Результат выполнения задачи – файл с прошивкой защитного блока.

Комментарий. К задаче прилагается: программа обучения и тестирования защитного блока.

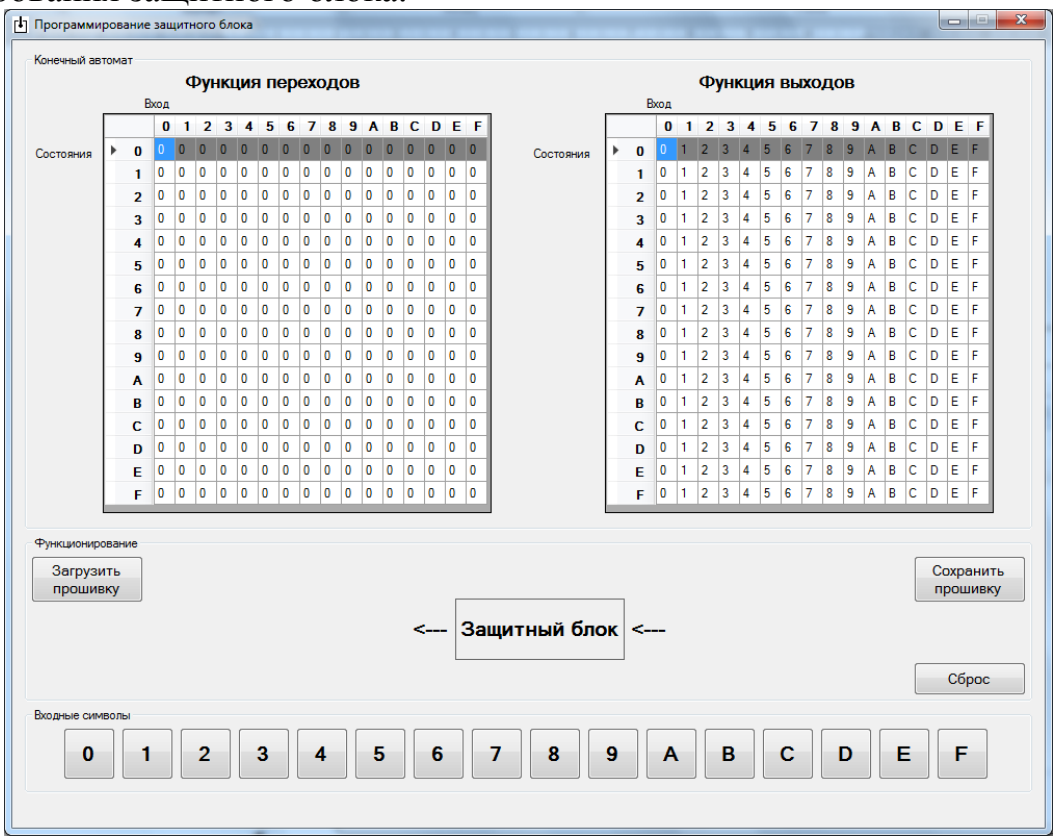

Рис. 8. Программа обучения и тестирования защитного блока

Решение

Защитный блок устроен следующим образом: при подаче на вход символа выполняется две операции:

1) по функции выходов определяется, какой символ будет на выходе. Он зависит от текущего активного состояния (выделенная строка в таблице

выходов) и подданного на вход символа (столбец в таблице выходов). Значение ячейки на пересечении строки и столбца – это и есть выходной символ;

2) по функции переходов определяется новое активное состояние. Оно зависит от текущего активного состояния (выделенная строка в таблице переходов) и подданного на вход символа (столбец в таблице переходов). Значение ячейки на пересечении строки и столбца – это и есть номер нового активного состояния.

По таблицам переходов и выходов видно, что алфавит составляет 16 символов (16 столбцов), и число состояний равно 16 (16 строк в таблице). Изначально активным считается состояние 0.

Чтобы определить последовательность входных символов необходимо следующее:

- при подаче очередного символа последовательности изменять активное состояние;
- при подаче символа не из последовательности сбрасывать активное состояние в начальное.

Чтобы изменить последний символ последовательности необходимо следующее:

- определить, что предыдущие символы принадлежат искомой последовательности;
- при подаче на вход последнего символа последовательности изменить выходной символ в таблице выходов.

Поскольку всего возможно 16 состояний, то максимум такой блок может отслеживать последовательность длиной 16 символов. По условию требуется последовательность длиной 6 символа. Предлагается выделить следующие состояния:

- состояние 0 – не введена никакая последовательность;
- состояние 1 – введена последовательность 1;
- состояние 2 – введена последовательность 1F;
- состояние 3 – введена последовательность 1F1;
- состояние 4 – введена последовательность 1F10;
- состояние 5 – введена последовательность 1F10A;
- состояние 6 – введена последовательность 1F10AE.

Замена последовательности 1F10AE на 1F10A1

1. При подаче на вход первого символа 1 необходимо перейти из любого состояния в состояние 1. Для этого в таблице переходов необходимо заменить ячейки:

StateMas[i][1] = 1, i = 0,1,...,F, кроме i=2.

2. Если на вход подается символом F и активным является состояние 1 (ввод символа 1), значит это продолжение последовательности и необходимо перейти в состояние 2. Для этого в таблице переходов необходимо заменить ячейку:

   StateMas[1][F] = 2.

   Остальные незадействованные ячейки в строке 1 должны быть равны 0.

3. Если на вход подается символом 1 и активным является состояние 2 (ввод символов 1F), значит это продолжение последовательности и необходимо перейти в состояние 3. Для этого в таблице переходов необходимо заменить ячейку:

   StateMas[2][1] = 3.

   Остальные незадействованные ячейки в строке 2 должны быть равны 0.

4. Если на вход подается символом 0 и активным является состояние 3 (ввод символов 1F1), значит это продолжение последовательности и необходимо перейти в состояние 4. Для этого в таблице переходов необходимо заменить ячейку:

   StateMas[3][0] = 4.

   Остальные незадействованные ячейки в строке 3 должны быть равны 0.

5. Если на вход подается символом A и активным является состояние 4 (ввод символов 1F10), значит это продолжение последовательности и необходимо перейти в состояние 5. Для этого в таблице переходов необходимо заменить ячейку:

   StateMas[4][A] = 5.

   Остальные незадействованные ячейки в строке 4 должны быть равны 0.

6. Если на вход подается символом E и активным является состояние 5 (ввод символов 1F10A), значит это окончание последовательности и необходимо перейти в состояние 6 либо в состояние 0. Для этого в таблице переходов необходимо заменить ячейку:

   StateMas[5][E] = 6.

   Остальные незадействованные ячейки в строке 5 должны быть равны 0.

Кроме того, необходимо изменить выходной символ E→1. Для этого необходимо изменить ячейку в таблице выходов:

OutMas[5][E] = 1.

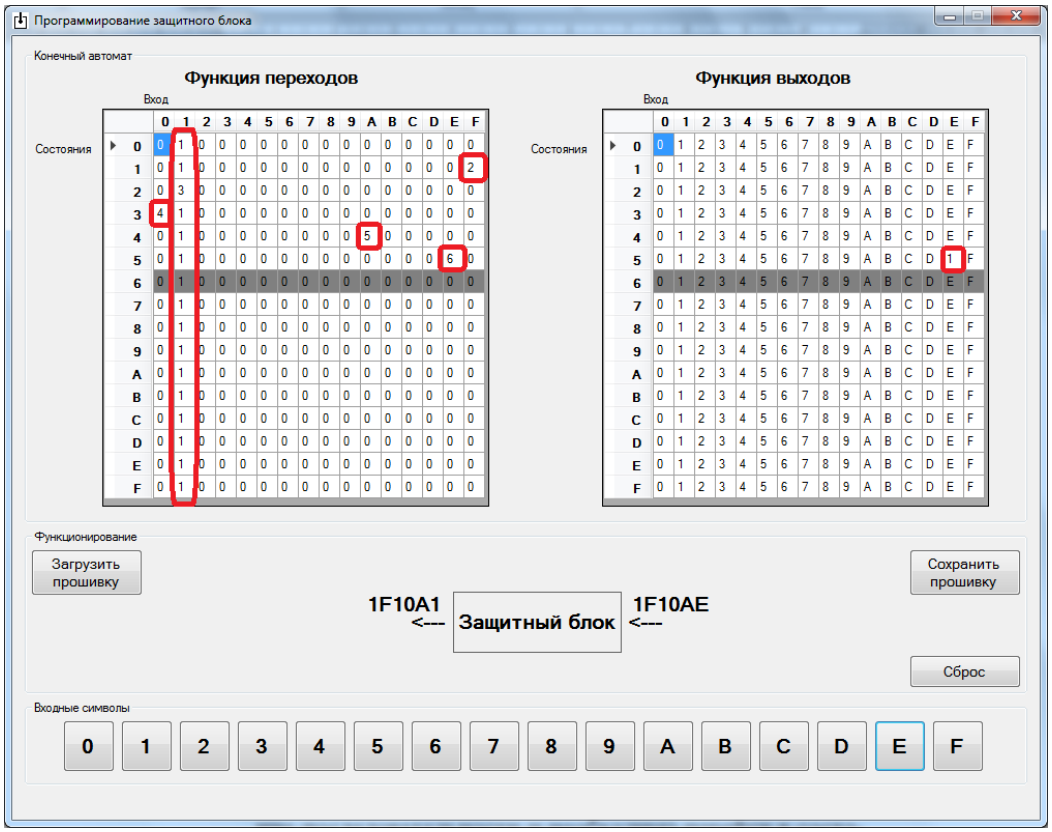

Рис. 9. Замена последовательности 1F10AE на 1F10A1