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 антивирусом оказалось, что файл «*apu004.dll*» заражен, а остальные файлы не содержат вредоносного кода.

Администратор предположил, что были подменены (заражены) отдельные файлы в пакетах и для других терминалов. С помощью специальной программы администратору удалось получить некоторые фрагменты пакетов обновлений нескольких терминалов.

Проанализируйте эти фрагменты и выясните, какие из файлов в них были заражены. Содержимое архива пакета обновлений с диска администратора:

Имя	Размер	CRC32
..		
apu001.dll	2 993	A36E9BCD
apu002.dll	447	C970284D
apu003.dll	2 815	9EF05D77
apu004.dll	917	BDE99372
apu005.dll	1 423	9735DEE7

Комментарий. К задаче прилагается: исходный архив с диска администратора (*apu_test.rar*), фрагменты пакетов обновлений для четырех терминалов (*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 это, 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. 10).

```
00000000: 52 61 72 21 1A 07 00 CF|90 73 00 00 0D 00 00 00  | Rar!...Пђs......
00000010: 00 00 00 00 36 BD 74 20|90 2F 00 EA 0A 00 00 B1  | ....6St ђ/.к...±
00000020: 0B 00 00 02 CD 9B 6E A3|4A 88 53 49 1D 33 0A 00  | ....H>nJJ■SI.3..
00000030: 20 00 00 00 61 70 75 30|30 31 2E 64 6C 6C 00 B0  |  ...apu001.dll.°
00000040: DF 14 80 11 DD 54 14 C8|8D BC 94 19 B1 EB 04 52  | Я.Ъ.ЭТ.ИЌj".±л.R
00000050: 74 81 44 4B 40 AB 5A F8|AD AB 5A DE 95 6A A3 D1  | tЃDK@«Zш-«ZЮ•jJC
00000060: 71 6D F2 A8 F4 47 A6 B5|A8 F4 78 98 78 F4 19 6B  | qmтЁфG¦µЁфx.xф.k
00000070: D5 6A B6 49 C9 D0 80 D5|1E 81 41 24 88 14 6A 30  | Xj¶IЙРЂX.ЃA$■.j0
00000080: 46 8F 45 54 06 96 43 90|91 80 2B CE 71 93 92 4E  | FЏET.-Cђ`Ъ+Oq"'N
00000090: 4E B3 A1 24 84 12 42 1D|1C 56 04 F3 2E 5C C5 24  | Ni¦Ў$„.B..V.у.\E$
000000A0: E4 93 E1 7E 7B 37 BF 46|B5 BC DE B3 37 AD 66 FE  | д"б~{7їFµjЮі7-fю
000000B0: 3B FF 7A CD E6 F5 AC CD|EB D1 FB F7 C3 F5 7A 9D  | ;яzНжх¬НлСычГхzќ
```

Рис. 10. 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.

Исключение составляет файл «*apu004.dll*», который изменен в исходном архиве. Если во фрагментах обнаружится этот файл с другой контрольной суммой, это означает, что файл не изменен.

Рассмотрим фрагменты архива для разных терминалов.

Терминал 1

Анализируя фрагмент «apu_term1.004» для терминала 1 было обнаружено имя файла «*apu004.dll*» и контрольная сумма **BD E9 93 72**, что соответствует исходному архиву, в котором этот файл заражен. А значит и в терминале 1 этот файл был изменен.

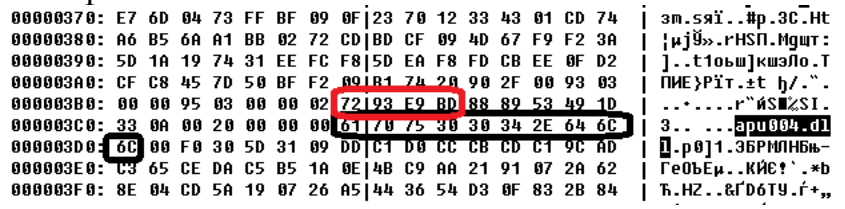


Рис. 11. Анализ фрагмента «apu_term1.004» для терминала 1

Для остальных файлов во фрагментах терминала 1 контрольные суммы совпадают с исходным архивом.

Терминал 2

Анализируя фрагмент «apu_term2.003» для терминала 2 было обнаружено имя файла «*apu002.dll*», однако контрольная сумма в файле содержится не полностью – только первый байт 88. Значит оставшаяся часть контрольной суммы находится в предыдущем фрагменте «apu_term2.002» – последние 3 байта.

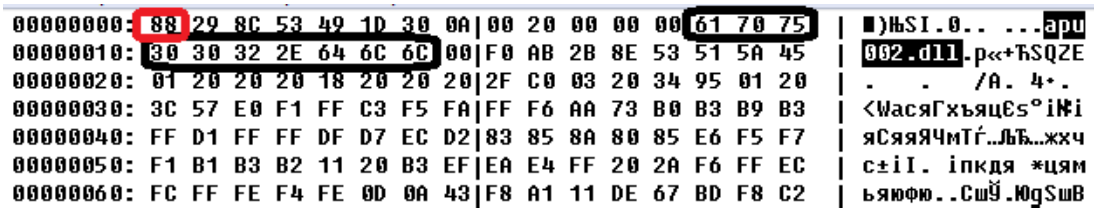


Рис. 12. Анализ фрагмента «apu_term2.003» для терминала 2

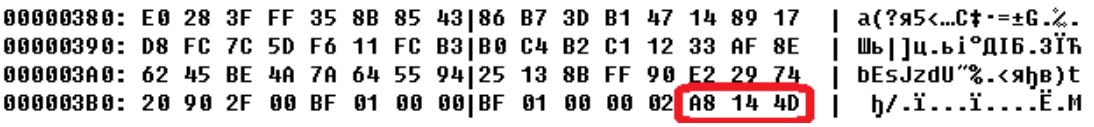


Рис. 13. Анализ фрагмента «apu_term2.002» для терминала 2

В результате для файла «*apu002.dll*» контрольная сумма будет **88 4D 14 A8**, что не соответствует исходному архиву, значит в терминале 2 этот файл был изменен.

Для остальных файлов во фрагментах терминала 2 контрольные суммы совпадают с исходным архивом, включая файл «*apu004.dll*», который в терминале 2 тоже был изменен.

Терминал 3

Анализируя фрагмент «apu_term3.003» для терминала 3 было обнаружено имя файла «*apu003.dll*» и его контрольная сумма **43 FA BC 74**, которая не соответствует исходному архиву, значит в терминале 3 этот файл был изменен.

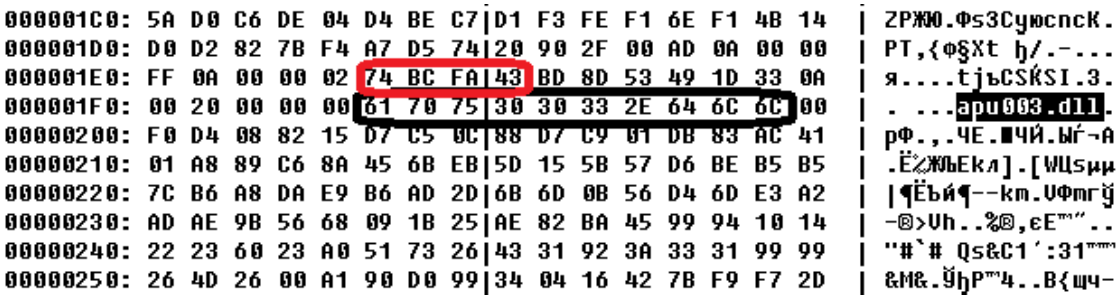


Рис. 14. Анализ фрагмента «apu_term3.003» для терминала 3

Анализируя фрагмент «apu_term3.005» для терминала 3 было обнаружено имя файла «*apu004.dll*» и его контрольная сумма **04 25 84 D7**, которая не соответствует исходному архиву. Поскольку в исходном архиве этот файл был заражен, то можно сделать предположение, что в этом терминале файл «apu004.dll» является исходным, то есть без изменений.

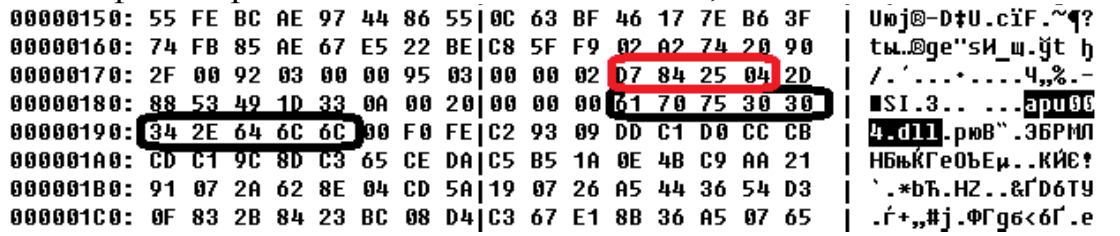


Рис. 15. Анализ фрагмента «apu_term3.005» для терминала 3

Для остальных файлов во фрагментах терминала 3 контрольные суммы совпадают с исходным архивом.

Терминал 4

Анализируя фрагмент «apu_term4.005» для терминала 4 было обнаружено имя файла «*apu004.dll*» и его контрольная сумма **04 25 84 D7**, которая не соответствует исходному архиву. Поскольку в исходном архиве этот файл был заражен, то можно сделать предположение, что в этом терминале файл «apu004.dll» является исходным, то есть без изменений.

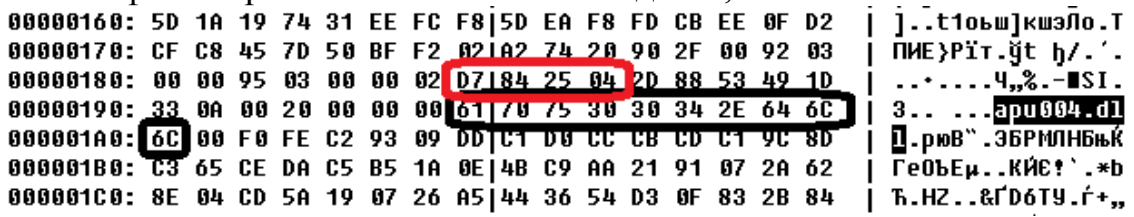


Рис. 16. Анализ фрагмента «apu_term4.005» для терминала 4

Для остальных файлов во фрагментах терминала 4 контрольные суммы совпадают с исходным архивом.

Ответ:

а) терминал 1 – заражен файл «apu004.dll»,
б) терминал 2 – заражены файлы «apu002.dll» и «apu004.dll»,
в) терминал 3 – заражен файл «apu003.dll»,
г) терминал 4 – нет зараженных файлов.

Задача 2. Зашифрованная картинка

Имеются три файла с картинками в формате Bitmap Picture (.bmp). Структура bmp-файла приведена ниже.

Заголовок (54 байта)	Данные

Два из трех имеющихся файлов зашифрованы. Известно, что для этого использовалась следующая процедура. Файл разбивался на равные блоки, размер которых совпадает с длиной ключа. Далее осуществлялась поразрядная операция сложения по модулю 2 (XOR) каждого блока с ключом.

На одной из картинок изображено текстовое сообщение. Требуется найти ключ и текстовое сообщение на картинке.

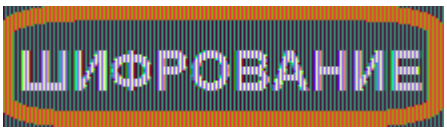
Комментарий. К задаче прилагается: два зашифрованных файла (*picture11.enc*, *picture12.enc*), один открытый файл (*picture13.bmp*), редактор файлов в шестнадцатеричном формате (HexEditor).

Решение

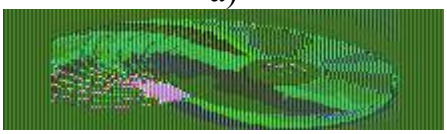
Способ 1

Анализируя файлы, можно заметить, что размер файлов одинаковый, а значит и размеры картинок одинаковые. Отсюда можно сделать предположение, что заголовки всех картинок будут одинаковыми.

Из условий задачи известно, что заголовок занимает первые 54 байта. Можно взять заголовок открытого файла и подменить на зашифрованных файлах. Картинки при этом откроются, но изображение будет искажено, хотя и читаемо.



а)



б)

Рис. 17. Зашифрованные файлы с подмененным заголовком

На изображении а) четко прослеживается надпись «ШИФРОВАНИЕ».

Далее, для нахождения ключа шифрования необходимо выполнить операцию «Поразрядное исключающее ИЛИ» (XOR) между заголовками открытого файла и заголовками зашифрованного файла. Получится периодическая последовательность вида:

FA CE 8D BA 55 F5 FA CE 8D BA 55 F5 FA CE 8D BA 55 F5 …

Можно обнаружить периодичность последовательности и извлечь ключ: **FA CE 8D BA 55 F5**.

Способ 2

Анализируя заголовок открытого файла, можно увидеть, что некоторые поля нулевые.

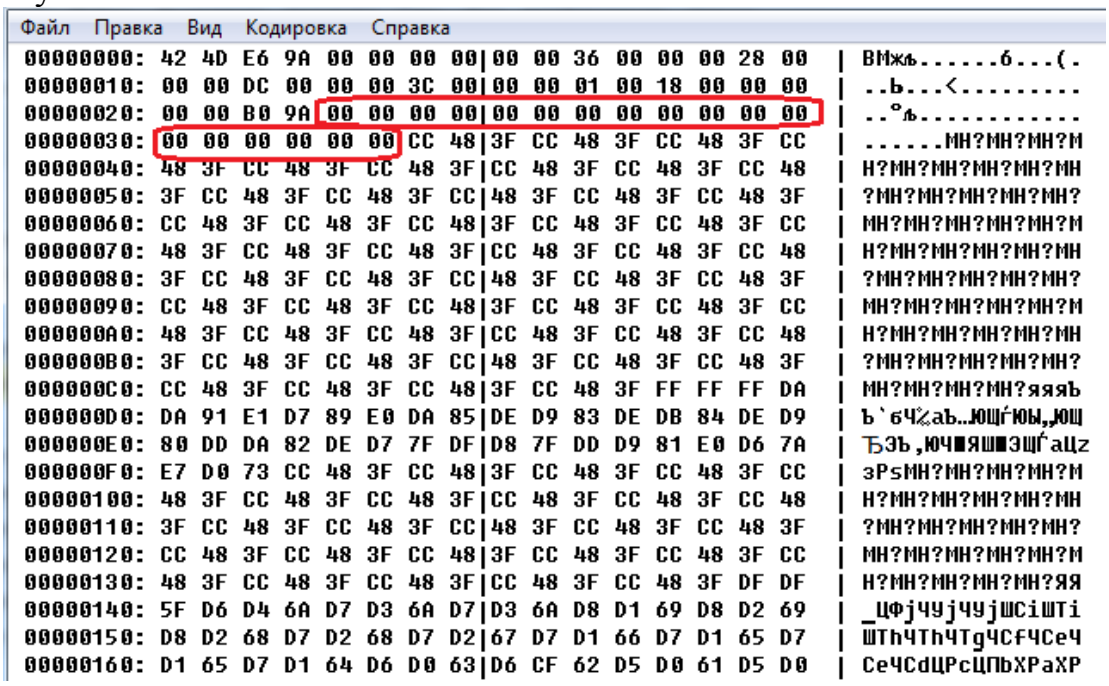


Рис. 18. Заголовок открытого файла

Отметив смещение нулевых байтов и сравнив соответствующие байты в зашифрованных файлах, можно увидеть периодичность в значениях.

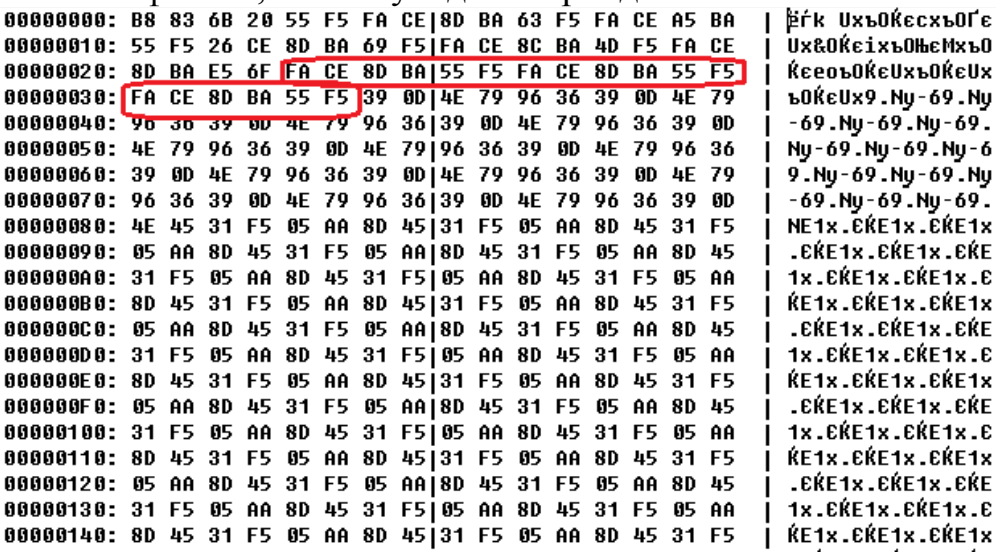


Рис. 19. Заголовок зашифрованного файла

Выделенный фрагмент: **FA CE 8D BA 55 F5 FA CE 8D BA 55 F5 FA CE 8D BA 55 F5**

Можно обнаружить периодичность последовательности и извлечь ключ: **FA CE 8D BA 55 F5.**

Необходимо написать программу, которая выполняет операцию «Поразрядное исключающее ИЛИ» (XOR) над всеми байтами файла с обнаруженным ключом. Результирующий файл сохраняется с расширением

.BMP и открывается средствами просмотра изображения. Если программа написана правильно – на картинке будет слово ШИФРОВАНИЕ.

Ответ: слово – «ШИФРОВАНИЕ», ключ – FA CE 8D BA 55 F5.

Задача 3. DDoS-атака

В студенческом городке развернуто 12 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 2, 3 и 4.

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.

Пакет уничтожается, если он достиг сети назначения или счетчик S > 100.

Определите наибольшее число пересылок пакета, поступившего из Интернет. В ответе укажите через какой маршрутизатор и для какой сети надо отправить соответствующий пакет.

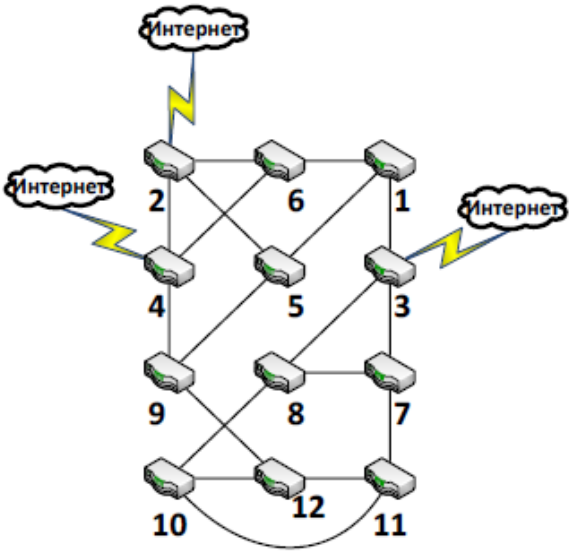


Рис. 20. Схема соединения ЛВС

Решение

Так как в ответе на задачу необходимо указать через какой маршрутизатор и для какой сети надо отправить пакет, а также указать наибольшее число пересылок пакета, то решить задачу теоретически,

получив точный ответ, не представляется возможным. Возможно два варианта поиска ответа: ручной и программный.

Решение задачи вручную и получение полного обоснованного ответа, охватывающего все возможные случаи, когда для каждого из трех вариантов входа может быть одиннадцать вариантов выхода, слишком трудозатратно, так как необходимо проследить пути пакета до всех целевых маршрутизаторов (11 случаев) для всех возможных входов в маршрутизаторы, подключенные к сети Интернет. Но если доказать, что существует путь, для которого пакет «зациклится» и значение счётчика S достигнет 100, то можно будет вручную найти один или более вариант правильного ответа. Зацикливание пакета может произойти в случае, если значение счетчика S станет достаточно большим, чтобы для некоторых двух соседних маршрутизаторов в качестве следующего маршрутизатора для перехода был выбран маршрутизатор с максимальным номером, то есть значение выражения S / R > 2, при этом они являются «соседями» с наибольшим номером друг для друга и, очевидно, что они не должны быть целевыми маршрутизаторами. Такое место в сети есть – это маршрутизаторы с номерами 11 и 12. Далее надо найти такой путь (начальный и конечный маршрутизаторы), для которого пакет на некотором шаге попадёт в маршрутизатор №12 и, в итоге с ростом счётчика S, зациклится.

В качестве сети назначения подойдут сети с номерами 3, 7 и 8, так как можно заметить, что они не достижимы при отправке пакета через заданные условием задачи маршрутизаторы. В сеть 3 можно попасть из сети 1 при условии, что S < 2, но это не возможно потому, что сеть 1 не является входной. А в сети 7 и 8 можно попасть только после того, как пакет пройдёт из сети 9 в сеть 12 со значением счетчика S ≥ 27. При таком значении счетчика из сетей 10 и 11 не перейти в сети 8 и 7 соответственно, так как значение выражения S / R должно быть меньше 2, а оно будет заведомо больше или равно двум.

Рассмотрим пример, когда пакет, переданный через маршрутизатор №2 для сети №3 на шаге №39 зациклится между маршрутизаторами с номерами 11 и 12.

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	2	4	2	4	2	5	1	6	1	6	1	6	2	6	2	6	2	6	4	9	5	9	5	9	5	9

S	27	28	29	30	31	32	33	34	35	36	37	38	39	40	…	100
R	5	9	12	10	12	10	12	10	12	10	12	11	12	11	…	11

Таким образом, получаем, что пакет надо отправить через маршрутизаторы №2 или №4 для сети №3, или через любой из входных маршрутизаторов для сетей №7 или №8. При этом значение счетчика S достигнет 100, то есть между маршрутизаторами будет сделано 99 шагов.

Таким будет ответ, если результат операции деления S на R будем считать целочисленным.

Если решать задачу, считая, что результат операции деления вещественный, то ответ будет следующим: пакет надо отправить через

маршрутизатор №3 для сети №2, через маршрутизаторы №2 и №4 для сети №3, через маршрутизатор №2 для сети №10 или через любой из входных маршрутизаторов для сетей №7 или №8. Между маршрутизаторами будет сделано 99 пересылок.

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 12 на 12 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, как представлено на рисунке. Вместо матрицы смежности можно создать двумерный массив размером 12 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 любой вершине. Получается 36 возможных вариантов с учетом того, что входная и выходная вершины могут совпадать.

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

Рис. 21. Матрица смежности

Далее приведен пример исходных кодов на языке программирования «C» для программного решения задачи при представлении 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[12][3] = { {3,5,6},{4,5,6},{1,7,8},
```

```
            {2,6,9},{1,2,9},{1,2,4},{3,8,11},
            {3,7,10},{4,5,12},{8,11,12},
            {7,10,12},{9,10,11} };
   // для всех входных маршрутизаторов перебор
   // всех выходных сетей
   for (int R_begin=2; R_begin<=4; R_begin++)
   {
      // Вывод на экран номера начального
      // маршрутизатора
      printf("START ROUTER = %d\n",R_begin);
      for (int R_end=1; R_end<=12; R_end++)
      {
         int S = 1; // счетчик шагов (по условию
                    // начинается с 1)
         int R = R_begin; // номер текущего
                          // маршрутизатора
         while (R != R_end && S < 100)
         {
            // Вывод на экран номера текущего
            // маршрутизатора
            printf("%d ", R);
            // 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 значения счетчика S и
         // номера сети назначения
         printf(", S = %d, target net = %d\n", S,
                 R_end);
      }
   }
}
```

Результат работы программы для начального маршрутизатора №2 представлен на рисунке.

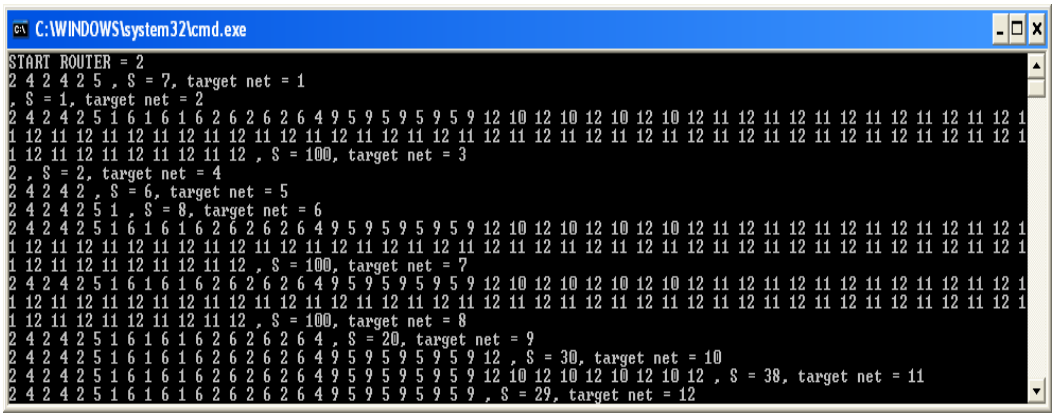


Рис. 22. Программное решение задачи

Ответ:

а) целочисленное деление: пакет надо отправить:

- через маршрутизаторы №2 или 4 для сети №3 или
- через любой из входных маршрутизаторов для сетей №7 или №8;

б) вещественное деление: пакет надо отправить

- через маршрутизатор №3 для сети №2,
- через маршрутизаторы №2 и №4 для сети №3,
- через маршрутизатор №2 для сети №10 или
- через любой из входных маршрутизаторов для сетей №7 или №8.

Задача 4. Восстановление кода

При копировании исходного кода программы произошла ошибка. Помогите определить, какие символы могут быть на месте ▲ и ▼, чтобы функция *function()* всегда корректно выполнялась, и в результате ее выполнения на экран выводилось слово «Yes».

Листинг программы приведен ниже.

Паскаль	Си

procedure func();
var i,size:integer;
r:array [0..ord('-')-
ord(▲)] of char;
begin
i:=ord('M')–ord("");
for i:=ord('#') –
ord('#') to
ord('&')-ord(▼) do
r[i]:=chr((ord('0')-
ord('(')) *
(ord('1')-ord(')'))-
ord(#9) - (ord('?')-
ord('=')) * i);
	void function()
	{
	int i = 'M' - "";
	char r['-'-▲];
	for (i='#'-'#';
	i<('&'-▼); i++)
	{
	*(r+i)=(char)(('0'-
	'(')*('1'-')') -
	'\t'-(('?'-'=')*i));
	}
	*(r+i)='!'-'!';
	
	if ((*(r+('-'-'*')) +

```
  r[i]:=chr(ord('!') –
    ord('!'));

  if ((ord(r[ord('-')-
    ord('*')])                +
    ord(r[ord('-')-
    ord('+')])                *
    (ord('2')-ord('('))+
    ord(r[ord(')')-
    ord('(')])                *
    (ord('2')-ord('('))*
    (ord('2')-ord('('))+
    ord(r[0]) *
    (ord('2')-ord('('))*
    (ord('2')-ord('('))*
    (ord('2')-ord('('))) = 60859) then
     writeln('Yes')
  else
     writeln('No');
end;
```

```
  ('-'-'+')[r]                          *
  ('2'-'(')+(')'-'(')[r]*
  (('2'-'(')) *
  (('2'-'('))                           +
  *(r)*('2'-'(') *
  ('2'-'(') * ('2'-'('))
  == 60859)
  {
    printf("Yes\n");
  }
  else
  {
    printf("No\n");
  }
        return;
}
```

Решение

Проанализируем исходный код программы на языке «Си». Заметим, что в соответствии с таблицей ASCII предложенный код можно преобразовать к следующему виду:

```
void function()                          //1
{                                        //2
 int i = 43;                             //3
 char r[45 - ▲];                         //4
 for (i=0; i<(38- ▼); i++)               //5
 {                                       //6
  r[i]=(char)(55 - 2*i);                 //7
 }                                       //8
 r[i]=0;                                 //9
 if ((r[3]+10*r[2]+100*r[1]+r[0]*1000) ==
    60859)                               //10
 {                                       //11
  printf("Yes\n");                       //12
 }                                       //13
 else                                    //14
 {                                       //15
  printf("No\n");                        //16
 }                                       //17
 return;                                 //18
}                                        //19
```

После преобразования исходного текста можно заметить, что на сообщение, выводимое на экран в результате выполнения функции function(), влияет только значение выражения r[3]+10*r[2]+100*r[1]+r[0]*1000, входящее в условие оператора *if* в строке 10. Следовательно, для того, чтобы на экран выводилось слово «Yes» необходимо инициализировать минимум первые четыре (с индексами 0-3) элемента массива r, обеспечив при этом выполнение равенства

r[3]+10*r[2]+100*r[1]+r[0]*1000 == 60859 (1)

Заметим, что инициализация элементов массива производится в цикле *for* в строках 5-8, где каждому элементу массива, начиная с первого (r[0]), присваивается уникальное значение в зависимости от его порядкового номера. При количестве итераций цикла не менее 4 обеспечивается выполнение равенства (1). Следовательно, значение выражение (38 - ▼) должно быть не менее 4: (38 - ▼) ≥ 4, т.е. код символа ▼ не должен превосходить 34: ▼ ≤ 34.

Далее определим область допустимых значений символа ▲. Для того, чтобы функция *function()* всегда корректно выполнялась, в строке 4 должно быть выделено необходимое количество памяти для массива *r*. Обратим внимание, что в строке 9 используется максимальное значение индекса массива *r*. Следовательно, в строке 4 необходимо задать размерность массива *r* не меньше ((38- ▼) + 1) , т.е. (45 - ▲) ≥ (39- ▼), а значит ▲ ≤ ▼ + 6.

Ответ: $\begin{cases} \mathbf{0 \le x \le 34} \\ \mathbf{y \le x + 6} \end{cases}$, где x – код символа ▼, y – код символа ▲.

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;	int GetOutput(int **StateMas, int **OutMas, int InSymb, int& CurState)

```
  OutMas : matrix;
  InSymb : integer;
  var CurState:integer):
integer;
var
 NewState:integer;
 OutSymb:integer;
begin
 NewState :=
  StateMas[CurState]
        [InSymb];
 OutSymb :=
  OutMas[CurState]
       [InSymb];
 CurState := NewState;
 result := OutSymb;
end;
```

```
// 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;

 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**
FA0B	FA01
10F1	10FA

Результат выполнения задачи – файл с прошивкой защитного блока.

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

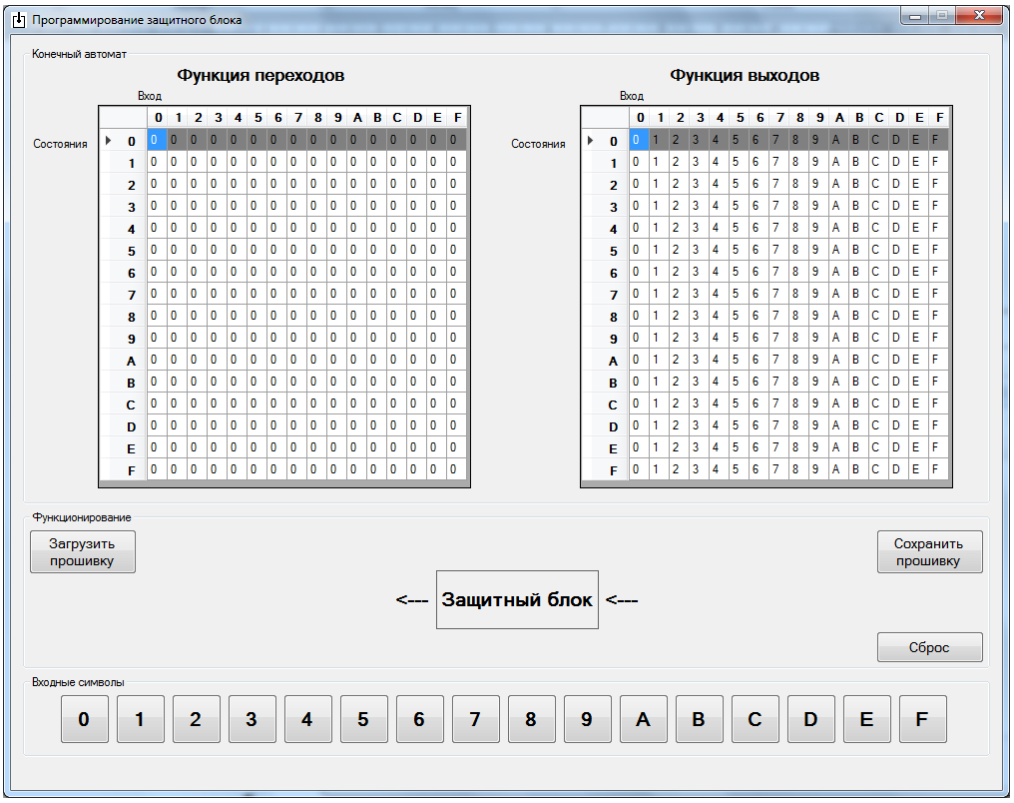


Рис. 23. Программа обучения и тестирования защитного блока

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 две последовательности длиной 4 символа. Предлагается выделить состояния 0-3 для последовательности **FA0B** и состояния 5-8 для последовательности **10F1**:

- состояние 0 – не введена никакая последовательность;
- состояние 1 – введена последовательность F;
- состояние 2 – введена последовательность FA;
- состояние 3 – введена последовательность FA0;
- состояние 4 – введена последовательность FA0B;
- состояние 5 – введена последовательность 1;
- состояние 6 – введена последовательность 10;
- состояние 7 – введена последовательность 10F;
- состояние 8 – введена последовательность 10F1.

Замена последовательности FA0B на FA01

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

   StateMas[i][F] = 1, i = 0,1,...,F, кроме i=6.

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

   StateMas[1][A] = 2.

   Остальные незадействованные ячейки в строке 1 должны быть равны 0.

3. Если на вход подается символом 0 и активным является состояние 2 (ввод символов FA), значит это продолжение последовательности и необходимо перейти в состояние 3. Для этого в таблице переходов необходимо заменить ячейку:

   StateMas[2][0] = 3.

   Остальные незадействованные ячейки в строке 2 должны быть равны 0.

4. Если на вход подается символом B и активным является состояние 3 (ввод символов FA0), значит это окончание последовательности и необходимо перейти в состояние 4 либо в состояние 0. Для этого в таблице переходов необходимо заменить ячейку:

StateMas[3][B] = 4.

Остальные незадействованные ячейки в строке 3 должны быть равны 0.

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

OutMas[3][B] = 1.

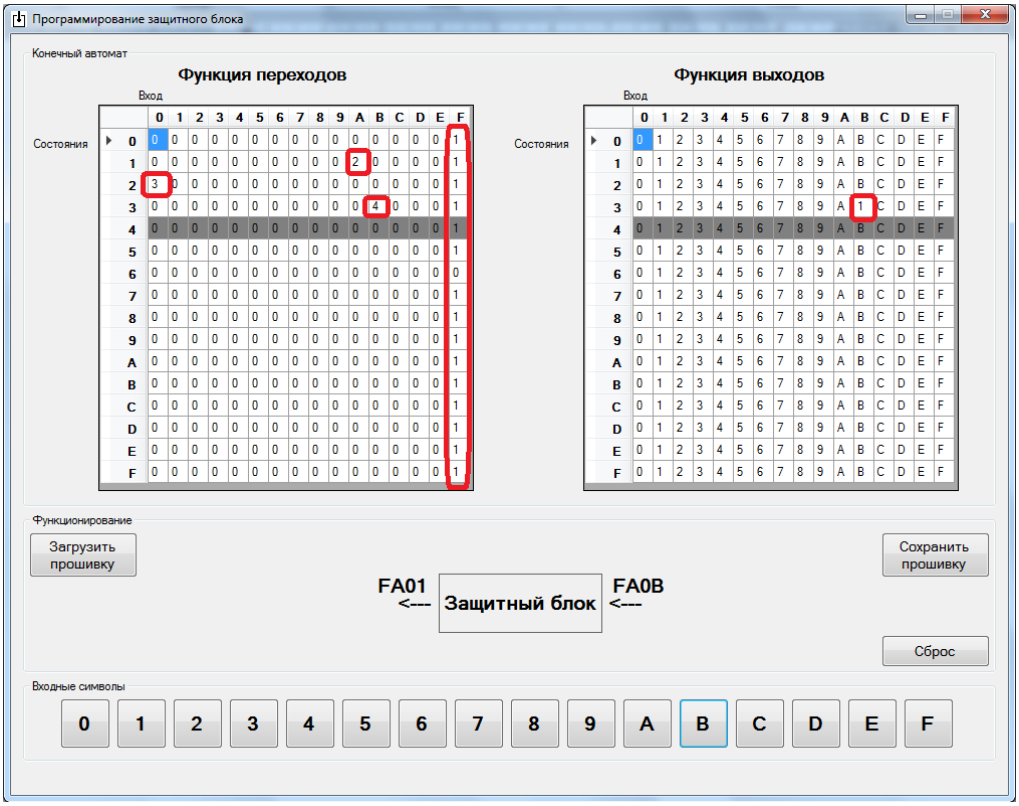


Рис. 24. Замена последовательности FA0B на FA01

Замена последовательности 10F1 на 10FA

1. При подаче на вход первого символа 1 необходимо перейти из любого состояния в состояние 5. Для этого в таблице переходов необходимо заменить ячейки:

   StateMas[i][1] = 5, i = 0,1,...,F, кроме i = 7.

2. Если на вход подается символом 0 и активным является состояние 5 (ввод символа 1), значит это продолжение последовательности и необходимо перейти в состояние 6. Для этого в таблице переходов необходимо заменить ячейку:

   StateMas[5][0] = 6.

   Остальные незадействованные ячейки в строке 5 должны быть равны 0.

3. Если на вход подается символом F и активным является состояние 6 (ввод символов 10), значит это продолжение последовательности и необходимо перейти в состояние 7. Для этого в таблице переходов необходимо заменить ячейку:

   StateMas[6][F] = 7.

   Остальные незадействованные ячейки в строке 6 должны быть равны 0.

4. Если на вход подается символом A и активным является состояние 7 (ввод символов 10F), значит это окончание последовательности и необходимо перейти в состояние 8 либо в состояние 0. Для этого в таблице переходов необходимо заменить ячейку:

StateMas[7][A] = 8.

Остальные незадействованные ячейки в строке 7 должны быть равны 0.

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

OutMas[7][1] = A.

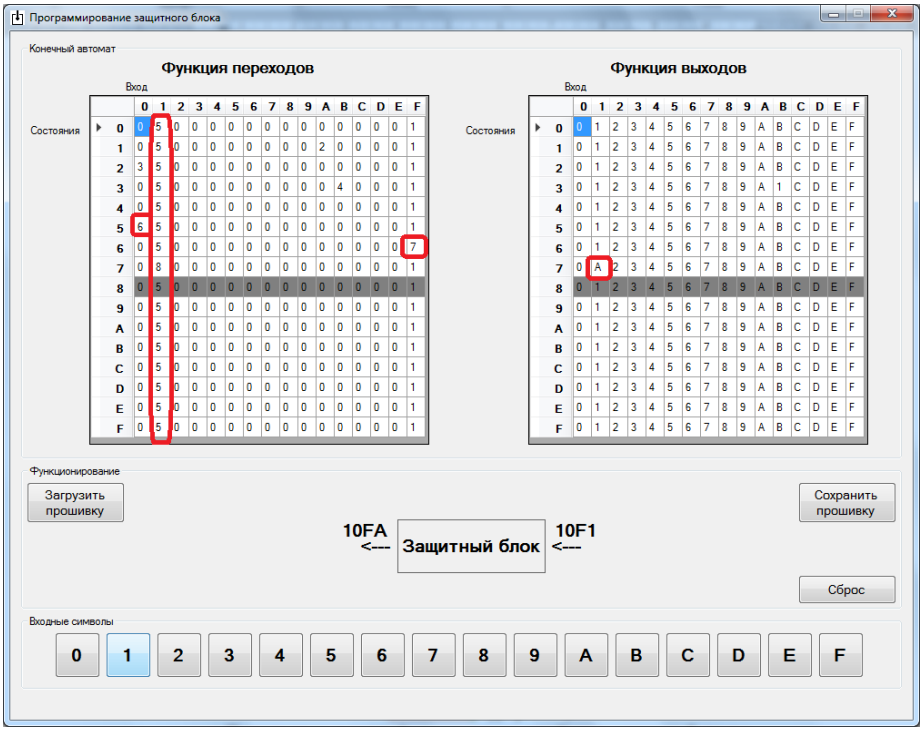


Рис. 25. Замена последовательности 10F1 на 10FA