Условия задач заключительного этапа

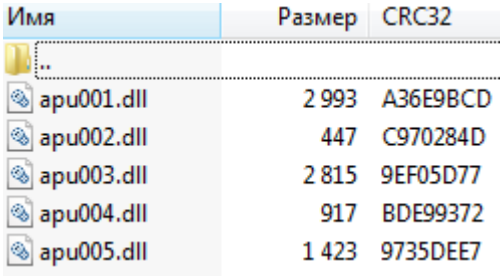
Задача 1. RAR

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:



Комментарий. К задаче прилагается: исходный архив с диска администратора (*apu_test.rar*), фрагменты пакетов обновлений для четырех терминалов (*apu_termX.NNN*), программа-архиватор WinRAR.

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).

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.

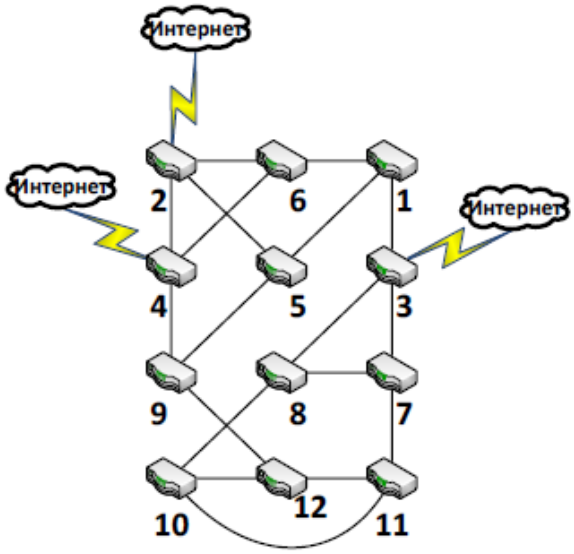
В служебной части сетевых пакетов имеется счетчик S, который увеличивается на 1 при каждой пересылке между маршрутизаторами. Из Интернет пакеты попадают в сети со счетчиком S = 1.

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.



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;
}
```

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.

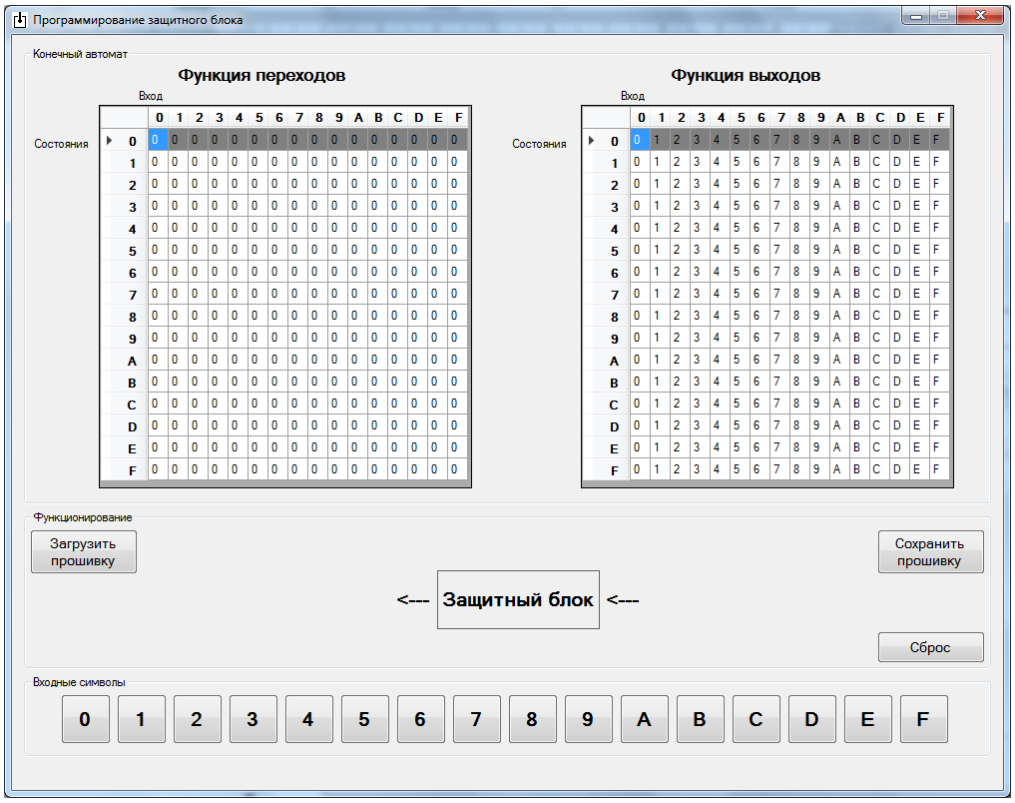


Рис. 23. Программа обучения и тестирования защитного блока