МАТЕРИАЛЫ ЗАДАНИЙ МЕЖРЕГИОНАЛЬНОЙ ОЛИМПИАДЫ ШКОЛЬНИКОВ ИМЕНИ И.Я. ВЕРЧЕНКО ПО ПРОФИЛЮ «КОМПЬЮТЕРНАЯ БЕЗОПАСНОСТЬ» (2023/2024 УЧЕБНЫЙ ГОД)

ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП
11 КЛАСС

СОДЕРЖАНИЕ

Задача 1. Компьютерная игра

Вариант 1

В компьютерной игре общий рейтинг команды определяется набором персонажей и их влиянием друг на друга. Каждый персонаж может влиять на рейтинг команды как положительно, так и отрицательно. В команды можно включать несколько персонажей одного типа.

Для вычисления общекомандного рейтинга необходимо учесть взаимные веса всех возможных пар персонажей внутри команды, включая влияние одинаковых персонажей друг на друга.

Например, рассмотрим таблицу взаимных весов персонажей.

	Персонаж 1	Персонаж 2	Персонаж 3	Персонаж 4
Персонаж 1	0	-3	+5	+2
Персонаж 2	-3	0	+1	-4
Персонаж 3	+5	+1	0	-6
Персонаж 4	+2	-4	-6	0

Если команда состоит из двух персонажей, то наибольший рейтинг будет равен 5 у команды, состоящей из Персонажа 1 и Персонажа 3.

Если в команду будут входить Персонаж 1 (П1), Персонаж 2 (П2) и Персонаж 3 (П3), то рейтинг команды будет равен сумме рейтингов всех пар персонажей команды: (П1-П2), (П1-П3), (П2-П3):

$$-3 + 5 + 1 = 3.$$

Если в команду будет входить Персонаж 1 (П1) и два персонажа 3 (П3), то рейтинг команды будет равен сумме рейтингов всех пар персонажей команды: (П1-П3), (П1-П3), (П3-П3):

$$5 + 5 + 0 = 10.$$

На основе таблицы взаимных весов персонажей определите команду из *пяти* персонажей, которая будет иметь НАИБОЛЬШИЙ общекомандный рейтинг. Допускается брать *не более двух* персонажей одного типа в команду. В ответе укажите имена, количество персонажей и значение общекомандного рейтинга.

	Странник	Рыцарь	Эльф	Титан	Воин	Король	Друид	Мудрец	Волшебница	Дракон
Странник	0	-7	+2	-6	+3	-4	-5	-4	+6	-1
Рыцарь	-7	0	+1	-5	+3	+5	+7	-2	-5	+8
Эльф	+2	+1	0	-4	-9	+5	-1	+1	+4	-2
Титан	-6	-5	-4	0	+4	-7	+5	-3	-5	-2
Воин	+3	+3	-9	+4	0	+3	+3	+6	-9	-4
Король	-4	+5	+5	-7	+3	0	-5	-2	+3	-7
Друид	-5	+7	-1	+5	+3	-5	0	+3	+1	-4
Мудрец	-4	-2	+1	-3	+6	-2	+3	0	-1	-8
Волшебница	+6	-5	+4	-5	-9	+3	+1	-1	0	+6
Дракон	-1	+8	-2	-2	-4	-7	-4	-8	+6	0

Решение

Необходимо перебрать все возможные комбинации из трех персонажей (их всего 120). Для каждой посчитать общий рейтинг и найти команду с максимальным рейтингом.

Для упрощения поиска можно написать программу, перебирающую все возможные комбинации и выводящую максимальное значение рейтинга на экран.

Листинг на языке C++	*Листинг на языке Python*
`#include <iostream>` `using namespace std;`	

```
int main()
{
   intA[10][10]={
      {0,-7,2,-6,3,-4,-5,-4,6,-1},
      {-7,0,1,-5,3,5,7,-2,-5,8},
      {2,1,0,-4,-9,5,-1,1,4,-2},
      {-6,-5,-4,0,4,-7,5,-3,-5,-2},
      {3,3,-9,4,0,3,3,6,-9,-4},
      {-4,5,5,-7,3,0,-5,-2,3,-7},
      {-5,7,-1,5,3,-5,0,3,1,-4},
      {-4,-2,1,-3,6,-2,3,0,-1,-8},
      {6,-5,4,-5,-9,3,1,-1,0,6},
      {-1,8,-2,-2,-4,-7,-4,-8,6,0}
   };
   int sumMax=-100, sum;
   int p1,p2,p3,p4,p5;
   int counter;
   for (int i1 = 0; i1 < 10; i1++)
   for (int i2 = 0; i2 < 10; i2++)
   for (int i3 = 0; i3 < 10; i3++)
   for (int i4 = 0; i4 < 10; i4++)
   for (int i5 = 0; i5 < 10; i5++)
   {
      // считаем число повторов
      // персонажа 1
      counter = 1;
      if (i1 == i2)
        counter++;
      if (i1 == i3)
        counter++;
      if (i1 == i4)
        counter++;
      if (i1 == i5)
        counter++;
      // если персонажей больше 2
      // переходим на след итерацию
      if (counter > 2)
        continue;
      // считаем число повторов
      // персонажа 2
      counter = 1;
      if (i2 == i3)
        counter++;
      if (i2 == i4)
        counter++;
      if (i2 == i5)
        counter++;
      if (counter > 2)
        continue;
      // считаем число повторов
      // персонажа 3
      counter = 1;
      if (i3 == i4)
        counter++;
      if (i3 == i5)
        counter++;
      if (counter > 2)
        continue;
      sum = A[i1][i2] + A[i1][i3] +
A[i1][i4] + A[i1][i5] + A[i2][i3] +
A[i2][i4] + A[i2][i5] + A[i3][i4] +
A[i3][i5] + A[i4][i5];
```

```
      if (sumMax <= sum)
      {
        sumMax = sum;
        p1 = i1 + 1;
        p2 = i2 + 1;
        p3= i3 + 1;
        p4= i4 + 1;
        p5= i5 + 1;
      }
   }
   cout << "Сумма " << sumMax <<
"\nПерсонажи " << p1 << "," << p2 <<
"," << p3 << "," << p4 << "," << p5;
}
```

Результат выполнения программы:

```
Сумма 40
Персонажи 7,7,5,2,2 (2 друида, 2 война, рыцарь)
```

Ответ: **1. Максимальная сумма – 40**
2. Персонажи: 2 Рыцаря, 1 Воин, 2 Друида

Вариант 2

В компьютерной игре общий рейтинг команды определяется набором персонажей и их влиянием друг на друга. Каждый персонаж может влиять на рейтинг команды как положительно, так и отрицательно. В команды можно включать несколько персонажей одного типа.

Для вычисления общекомандного рейтинга необходимо учесть взаимные веса всех возможных пар персонажей внутри команды, включая влияние одинаковых персонажей друг на друга.

Например, рассмотрим таблицу взаимных весов персонажей.

	Персонаж 1	Персонаж 2	Персонаж 3	Персонаж 4
Персонаж 1	0	-3	+5	+2
Персонаж 2	-3	0	+1	-4
Персонаж 3	+5	+1	0	-6
Персонаж 4	+2	-4	-6	0

Если команда состоит из двух персонажей, то наибольший рейтинг будет равен 5 у команды, состоящей из Персонажа 1 и Персонажа 3.

Если в команду будут входить Персонаж 1 (П1), Персонаж 2 (П2) и Персонаж 3 (П3), то рейтинг команды будет равен сумме рейтингов всех пар персонажей команды: (П1-П2), (П1-П3), (П2-П3):

$$-3 + 5 + 1 = 3.$$

Если в команду будет входить Персонаж 1 (П1) и два персонажа 3 (П3), то рейтинг команды будет равен сумме рейтингов всех пар персонажей команды: (П1-П3), (П1-П3), (П3-П3):

$$5 + 5 + 0 = 10.$$

На основе таблицы взаимных весов персонажей определите команду из *пяти* персонажей, которая будет иметь НАИМЕНЬШИЙ общекомандный рейтинг. Допускается брать *не более двух* персонажей одного типа в команду. В ответе укажите имена, количество персонажей и значение общекомандного рейтинга.

	Странник	Рыцарь	Эльф	Титан	Воин	Король	Друид	Мудрец	Волшебница	Дракон
Странник	0	3	4	-4	-6	1	3	-1	-1	5
Рыцарь	3	0	1	8	7	4	1	-9	2	-8
Эльф	4	1	0	-6	2	2	4	9	-8	-3

Титан	-4	8	-6	0	8	-8	2	4	4	-6
Воин	-6	7	2	8	0	7	-6	-7	5	4
Король	1	4	2	-8	7	0	-5	-3	2	-2
Друид	3	1	4	2	-6	-5	0	6	-6	-1
Мудрец	-1	-9	9	4	-7	-3	6	0	8	6
Волшебница	-1	2	-8	4	5	2	-6	8	0	5
Дракон	5	-8	-3	-6	4	-2	-1	6	5	0

Решение

Необходимо перебрать все возможные комбинации из трех персонажей (их всего 120). Для каждой посчитать общий рейтинг и найти команду с максимальным рейтингом.

Для упрощения поиска можно написать программу, перебирающую все возможные комбинации и выводящую максимальное значение рейтинга на экран.

Листинг на языке C++

```
#include <iostream>
using namespace std;
int main()
{
   int A[10][10]={
      {0,3,4,-4,-6,1,3,-1,-1,5},
      {3,0,1,8,7,4,1,-9,2,-8},
      {4,1,0,-6,2,2,4,9,-8,-3},
      {-4,8,-6,0,8,-8,2,4,4,-6},
      {-6,7,2,8,0,7,-6,-7,5,4},
      {1,4,2,-8,7,0,-5,-3,2,-2},
      {3,1,4,2,-6,-5,0,6,-6,-1},
      {-1,-9,9,4,-7,-3,6,0,8,6},
      {-1,2,-8,4,5,2,-6,8,0,5},
      {5,-8,-3,-6,4,-2,-1,6,5,0}
   };
   int sumMin=100, sum;
   int p1, p2, p3, p4, p5;
   int counter = 0;
   for (int i1 = 0; i1 < 10; i1++)
   for (int i2 = 0; i2 < 10; i2++)
   for (int i3 = 0; i3 < 10; i3++)
   for (int i4 = 0; i4 < 10; i4++)
   for (int i5 = 0; i5 < 10; i5++)
   {
      // считаем число повторов
      // персонажа 1
      counter = 1;
      if (i1 == i2)
        counter++;
      if (i1 == i3)
        counter++;
      if (i1 == i4)
        counter++;
      if (i1 == i5)
        counter++;
      // если персонажей больше 2
      // переходим на след итерацию
      if (counter > 2)
        continue;
      // считаем число повторов
      // персонажа 2
      counter = 1;
      if (i2 == i3)
```

Листинг на языке Python

```
        counter++;
      if (i2 == i4)
        counter++;
      if (i2 == i5)
        counter++;
      if (counter > 2)
        continue;
      // считаем число повторов
      // персонажа 3
      counter = 1;
      if (i3 == i4)
        counter++;
      if (i3 == i5)
        counter++;
      if (counter > 2)
        continue;
      sum  =  A[i1][i2]  +  A[i1][i3]  +
A[i1][i4]  +  A[i1][i5]  +  A[i2][i3]  +
A[i2][i4]  +  A[i2][i5]  +  A[i3][i4]  +
A[i3][i5] + A[i4][i5];
      if (sumMin >= sum)
      {
        sumMin = sum;
        p1 = i1 + 1;
        p2 = i2 + 1;
        p3 = i3 + 1;
        p4 = i4 + 1;
        p5 = i5 + 1;
      }
   }
   cout  <<  "Сумма  "  <<  sumMin  <<
"\nПерсонажи " << p1 << "," << p2 << ","
<< p3 << "," << p4 << "," << p5;
}
```

Результат выполнения программы:

```
Сумма -48
Персонажи 10,6,6,4,4 (2 титана, 2 короля, 1 дракон)
```

Ответ: **1. Минимальная сумма – -48**
2. Персонажи: 2 Титана, 2 Короля, Дракон

Задача 2. Секретный архив

Вариант 1

Мария спрятала IP-адрес с тестовым сайтом организации в формате `xxx.xxx.xxx.xxx` (`xxx` – число от 0 до 255) внутри текстового документа с названием «*Секретный архив*». Изучите документ и найдите спрятанный IP-адрес.

К задаче прилагается:

«*Секретный архив_в1.docx*» – секретный архив.

Решение

Прилагаемый текстовый документ является пустым, поэтому извлечь информацию из его содержимого невозможно.

Однако документ имеет название «*Архив*», что является прямым намеком на то, что документ необходимо открыть с помощью архиватора.

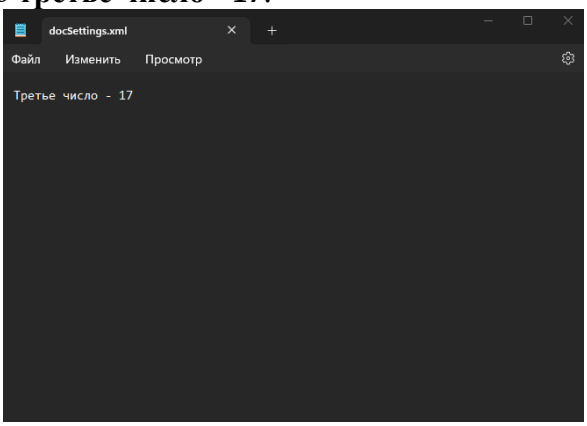
Открываем его с помощью архиватора и изучаем содержимое. Первая папка имеет название **192.** Это первое число из адреса.

Имя	Размер	Сжатый	Изменен	Создан	Открыт	Атрибуты	Зашифр...	Коммент...	CRC	Метод	Характер...	Система	Версия	Номер Т...	Смещение	Папок	Файлов
192.	1 506	765							09C06389							0	2
word	47 107	7 618							82B5692C							2	8
_rels	590	239							B71A911E							0	1
[Content_Type...	1 312	338	2024-02-...	2024-02-...	2024-02-...	A	-		6CD2A4DF	Deflate	NTFS	FAT	20	0	0		

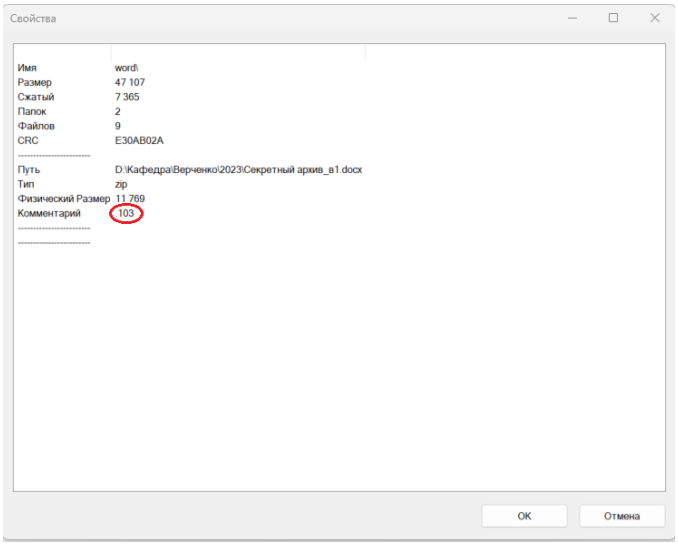
Далее просмотрим содержимое папок. В папке world содержится файл **.70..xml**. Можно предположить, что это число также относится к IP-адресу и находится в середине, но пока неясно, является это число вторым или третьим.

theme	8 444	1 786	
_rels	817	244	
.70..xml	2	2	2024-02-...
docSettings.xml	28	28	2024-02-...
document.xml	2 808	705	2024-02-...
fontTable.xml	1 658	486	2024-02-...
settings.xml	2 973	1 009	2024-02-...
styles.xml	29 483	2 777	2024-02-...
webSettings.x...	894	328	2024-02-...

Больше в названиях файлов нет чисел. Просмотрим содержимое файлов. В файле *docSettings.xml* указано, что **третье число - 17**.



Теперь известно, что IP-адрес имеет вид 192.70.17.xxx. Осталось найти последнюю цифру. В содержимом файлов больше нет подходящих цифр. Но если посмотреть свойства архива, то можно увидеть комментарий «**.103**»



Ответ: 192.70.17.103

Вариант 2

Мария спрятала IP-адрес с тестовым сайтом организации в формате `xxx.xxx.xxx.xxx` (`xxx` – число от 0 до 255) внутри текстового документа с названием «*Секретный архив*». Изучите документ и найдите спрятанный IP-адрес.

К задаче прилагается:

«*Секретный архив_в2.docx*» – секретный архив.

Решение

Прилагаемый текстовый документ является пустым, поэтому извлечь информацию из его содержимого невозможно.

Однако документ имеет название «*Архив*», что является прямым намеком на то, что документ необходимо открыть с помощью архиватора.

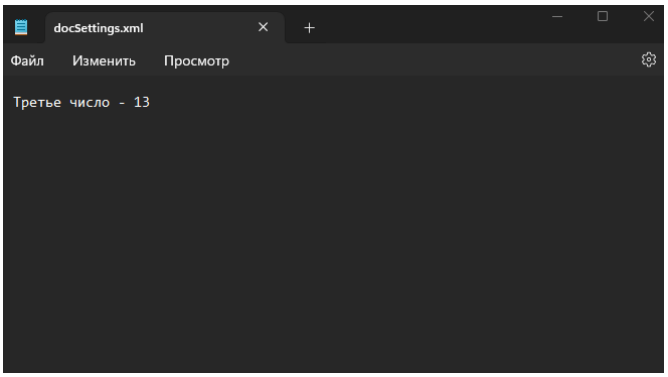
Открываем его с помощью архиватора и изучаем содержимое. Первая папка имеет название **192.** Это первое число из адреса.

Имя	Размер	Сжатый	Изменен	Создан	Открыт	Атрибуты	Зашифр...	Коммент...	CRC	Метод	Характер...	Система	Версия	Номер Т...	Смещение	Папок	Файлов
192	1 506	765							09C06389							0	2
word	47 107	7 618							82B5692C							2	8
_rels	590	239							B71A911E							0	1
[Content_Type...	1 312	338	2024-02-...	2024-02-...	2024-02-...	A	-		6CD2A4DF	Deflate	NTFS	FAT	20	0	0		

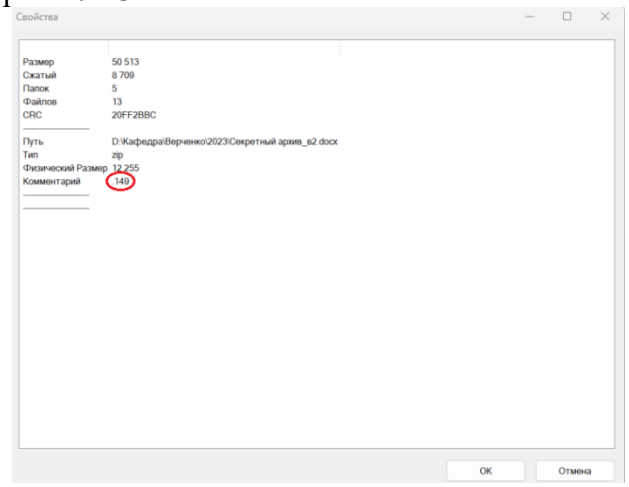
Далее просмотрим содержимое папок. В папке world содержится файл **.24.**.xml. Можно предположить, что это число также относится к IP-адресу и находится в середине, но пока неясно, является это число вторым или третьим.

Имя	Размер	Сжатый	Изменен
theme	8 444	1 786	
_rels	817	244	
.24..txt	0	0	2024-02-...
docSettings.xml	28	28	2024-02-...
document.xml	2 808	705	2024-02-...
fontTable.xml	1 658	486	2024-02-...
settings.xml	2 973	1 005	2024-02-...
styles.xml	29 483	2 777	2024-02-...
webSettings.x...	894	328	2024-02-...

Больше в названиях файлов нет чисел. Просмотрим содержимое файлов. В файле *docSettings.xml* указано, что **третье число - 17**.



Теперь известно, что IP-адрес имеет вид 192.24.13.xxx. Осталось найти последнюю цифру. В содержимом файлов больше нет подходящих цифр. Но если посмотреть свойства архива, то можно увидеть комментарий «**.149**»



Ответ: 192.24.13.149

Задача 3. Хеш-пароль

Вариант 1

Система аутентификации проверяет корректность введенного пароля, сравнивая вычисленное хеш-значение. Инженеру Павлу удалось перехватить несколько запросов(паролей) и вычисленных хеш-значений (см. таблицу).

Пароль	Хеш-значение (в 16-ом формате)
dragon	6F 0D 14 28
monkey	65 1A 28 27
strongestpassword	6E 36 21 2B
password	77 2E 2C 1E

Павлу удалось частично восстановить программный код генератора хеш-значений (см. листинги 1, 2).

Листинг 1 – Функция нормирования длины пароля

Язык C++

```
// Нормирование длины пароля перед
// вычислением хеш-значения
// DATA  -  строка (пароль)
// SIZE  -  требуемая длина
//          нормированной строки
// RETURN
//       нормированная по длине строка
unsigned char* normalizeDataLen(
                    char* data,
                    int size)
{
   // выделение памяти под новую строку
   // + 1 символ конца строки
   unsigned char* res = new
              unsigned char[size + 1];
   // длина входной строки
   int dataLen = strlen(data);
   int index = 0;
   // копируем символы из data
   // до требуемого размера size.
   // Если длина строки data меньше size
   // то циклически повторяем ее
   // требуемое количество раз
   for (int i = 0; i < size; i++)
      res[i] = data[i % dataLen];

   // нуль-байт - символ конца строки
   res[size] = 0;
   return res;
}
```

Язык Python

```
# Нормирование длины пароля перед
вычислением хеш-значения
# DATA - строка (пароль)
# SIZE - требуемая длина
#        нормированной строки
# RETURN
#    нормированная по длине строка
def normalizeDataLen(
       data: str,
       size: int):
   # будущий результат
   res = ""
   # длина входной строки
   dataLen = len(data)

   # копируем символы из data
   # до требуемого размера size.
   # Если длина строки data меньше size
   # то циклически повторяем ее требуемое
количество раз
   for i in range(0, size):
     res = res + data[i % dataLen]
   return res
```

Листинг 2 – Функция получения одного хеш-значения строки

Язык C++

```
// Вычисление одного хеш-значения
// от строки
// DATA  -  строка
// MASK  -  массив значений коэффициентов
//       (размер совпадает с длиной data)
// SIZE  -  длина строки и маски
// MOD   -  модуль вычисления хеш-значения
//          (по умолчанию, 256)
// RETURN
```

Язык Python

```
# Вычисление одного хеш-значения
# от строки
# DATA -строка
# MASK -массив значений коэффициентов
#       (размер совпадает с длиной data)
# SIZE -длина строки и маски
# MOD  -модуль вычисления хеш-значения
#         (по умолчанию, 256)
# RETURN
```

```cpp
//    вычисленное хеш-значение
unsigned char hashMask(
           unsigned char* data,
           unsigned char* mask,
           int size,
           int mod = 256)
{
  unsigned char sum = 0;
  // в цикле сложение байтов строки,
  // умноженных на коэффициенты
  // из массива MASK
  // (коэффициенты равны 0 или 1)
  for (int i = 0; i < size; i++)
     sum += data[i] * mask[i];

  // вычисление модуля итоговой суммы
  // (остаток от деления на MOD)
  sum = sum % mod;
  return sum;
}
```

```python
#       вычисленное хеш-значение
def hashMask(
           data: str,
           mask: [],
           size: int,
           mod = 256):
  # будущий результат
  sum = 0
  # в цикле сложение байтов строки,
  # умноженных на коэффициенты
  # из массива MASK
  # (коэффициенты равны 0 или 1)
  for i in range(0, size):
    sum += ord(data[i]) * mask[i]

  # вычисление модуля итоговой суммы
  # (остаток от деления на MOD)
  sum = sum % mod
  return sum
```

Павлу удалось определить, что все пароли перед вычислением хеш-значений, нормируются по длине к *9 символам*. Также стало известно, что для каждого байта хеш-значения используется свой массив коэффициентов из 0 и 1 (маска), то есть для вычисления первого байта хеш-значения применяется маска1, для второго байта – маска2 и так далее.

Помогите Павлу и определите значение масок и правильное хеш-значение для пароля:

`superman`

К задаче прилагается:

«listing.cpp» – листинг функция на языке программирования C++,

«listing.py» – листинг функция на языке программирования Python.

Решение

Для решения задачи необходимо перебрать все значения масок и сравнить результат вычисления хеш-значений на известных примерах из таблицы.

Если нормированная строка имеет длину 9 символов, то и размер маски так же равен 9 числам. Числа в маске могут принимать значение 0 или 1.

Алгоритм решения:

1) Нормируем длину известных паролей (используем функцию `normalizeDataLen` с параметром `size=9`).

2) Перебираем значение маски от `{0,0,0,0,0,0,0,0,0}` до `{1,1,1,1,1,1,1,1,1}` и вычисляем первый байт хеш-значения. Находим такую маску, при которой хеш-значение первого байта совпадает во всех четырех примерах. При этом значение маски должно быть одинаково во всех четырех примерах.

3) Выполняем п.2) для 2-го, 3-го и 4-го байта хеш-значений. Формируем 4 маски.

4) Используем сформированные в п.3) маски для вычисления 4-х байт хеш-значения для пароля “*superman*”. Предварительно строку необходимо нормализовать до длины 9 символов (*supermans*).

Ответ: **supermans – 72 2F 21 27**
Маски: **{0,0,0,0,1,0,0,0,0},**
{1,1,1,1,1,0,0,0,0},
{0,0,0,0,1,1,1,1,1},
{0,1,0,1,1,1,0,1,0}

Вариант 2

Система аутентификации проверяет корректность введенного пароля, сравнивая вычисленное хеш-значение. Инженеру Павлу удалось перехватить несколько запросов(паролей) и вычисленных хеш-значений (см. таблицу).

Пароль	Хеш-значение (в 16-ом формате)
batman	61 3C 12 D4
greenlight	6E 3F 0D E7
wonderer	65 3B 0E 0A
terminator	69 44 17 00

Павлу удалось частично восстановить программный код генератора хеш-значений (см. листинги 1, 2).

Листинг 1 – Функция нормирования длины пароля

Язык C++

```
// Нормирование длины пароля перед
// вычислением хеш-значения
// DATA  -  строка (пароль)
// SIZE  -  требуемая длина
//          нормированной строки
// RETURN
//       нормированная по длине строка
unsigned char* normalizeDataLen(
                  char* data,
                  int size)
{
  // выделение памяти под новую строку
  // + 1 символ конца строки
  unsigned char* res = new
            unsigned char[size + 1];
  // длина входной строки
  int dataLen = strlen(data);
  int index = 0;
  // копируем символы из data
  // до требуемого размера size
  // если длина строки data меньше size
  // то циклически повторяем ее
  // требуемое количество раз
  for (int i = 0; i < size; i++)
     res[i] = data[i % dataLen];
  // нуль-байт - символ конца строки
  res[size] = 0;
  return res;
}
```

Язык Python

```
# Нормирование длины пароля перед
вычислением хеш-значения
# DATA - строка (пароль)
# SIZE - требуемая длина
#        нормированной строки
# RETURN
#    нормированная по длине строка
def normalizeDataLen(
       data: str,
       size: int):
   # будущий результат
   res = ""
   # длина входной строки
   dataLen = len(data)

   # копируем символы из data
   # до требуемого размера size
   # если длина строки data меньше size
   # то циклически повторяем ее требуемое
количество раз
   for i in range(0, size):
     res = res + data[i % dataLen]
   return res
```

Листинг 2 – Функция получения одного хеш-значения строки

Язык C++

```
// Вычисление одного хеш-значения
// от строки
// DATA  -  строка
// MASK  -  массив значений коэффициентов
//          (размер совпадает с длиной data)
// SIZE  -  длина строки и маски
// MOD   -  модуль вычисления хеш-значения
//          (по умолчанию, 256)
// RETURN
//    вычисленное хеш-значение
unsigned char hashMask(
           unsigned char* data,
           unsigned char* mask,
           int size,
           int mod = 256)
```

Язык Python

```
# Вычисление одного хеш-значения
# от строки
# DATA -строка
# MASK -массив значений коэффициентов
#       (размер совпадает с длиной data)
# SIZE -длина строки и маски
# MOD  -модуль вычисления хеш-значения
#         (по умолчанию, 256)
# RETURN
#       вычисленное хеш-значение
def hashMask(
          data: str,
          mask: [],
          size: int,
          mod = 256):
```

```
{
  unsigned char sum = 0;
  // в цикле сложение байтов строки,
  // умноженных на коэффициенты
  // из массива MASK
  // (коэффициенты равны 0 или 1)
  for (int i = 0; i < size; i++)
     sum += data[i] * mask[i];
  // вычисление модуля итоговой суммы
  // (остаток от деления на MOD)
  sum = sum % mod;
  return sum;
}
```

```
  # будущий результат
  sum = 0
  # в цикле сложение байтов строки,
  # умноженных на коэффициенты
  # из массива MASK
  # (коэффициенты равны 0 или 1)
  for i in range(0, size):
    sum += ord(data[i]) * mask[i]
  # вычисление модуля итоговой суммы
  # (остаток от деления на MOD)
  sum = sum % mod
  return sum
```

Павлу удалось определить, что все пароли перед вычислением хеш-значений, нормируются по длине к *9 символам*. Также стало известно, что для каждого байта хеш-значения используется свой массив коэффициентов из 0 и 1 (маска), то есть для вычисления первого байта хеш-значения применяется маска1, для второго байта – маска2 и так далее.

Помогите Павлу и определите значение масок и правильное хеш-значение для пароля:

`robocop`

К задаче прилагается:

«listing.cpp» – листинг функция на языке программирования C++,

«listing.py» – листинг функция на языке программирования Python.

Решение

Для решения задачи необходимо перебрать все значения масок и сравнить результат вычисления хеш-значений на известных примерах из таблицы.

Если нормированная строка имеет длину 9 символов, то и размер маски так же равен 9 числам. Числа в маске могут принимать значение 0 или 1.

Алгоритм решения:

1) Нормируем длину известных паролей (используем функцию `normalizeDataLen` с параметром `size=9`).

2) Перебираем значение маски от `{0,0,0,0,0,0,0,0,0}` до `{1,1,1,1,1,1,1,1,1}` и вычисляем первый байт хеш-значения. Находим такую маску, при которой хеш-значение вычисленного байта совпадает во всех четырех примерах. При этом значение маски должно быть одинаково во всех четырех примерах.

3) Выполняем п.2) для 2-го, 3-го и 4-го байта хеш-значений. Формируем 4 маски.

4) Используем сформированные в п.3) маски для вычисления 4-х байт хеш-значения для пароля “*robocop*”. Предварительно строку необходимо нормализовать до длины 9 символов (*robocopro*).

Ответ: **robocopro - 63 41 13 03**
Маски: {0,0,0,0,1,0,0,0,0},
{0,0,0,1,1,1,0,0,0},
{0,0,1,1,1,1,1,0,0},
{1,1,0,1,1,1,0,1,1}

Задача 4. Субтитры

Вариант 1

Секретные агенты использовали файлы с субтитрами к фильмам для скрытия тайных сообщений. Они придумали специальный метод сокрытия данных в субтитрах, который не влияет на их отображение и не вызывает подозрений при просмотре фильма.

В прилагаемом файле скрыто одно слово. Определите метод сокрытия и расшифруйте спрятанное слово.

К задаче прилагается:

«Субтитры_в1.srt» – текстовый документ с субтитрами.

Решение

В прилагаемом файле содержится 80 субтитров. Если предположить, что один субтитр используется для скрытия одного бита, то в данном файле скрыто 80 бит или 10 символов.

Поскольку придуманный метод шифрования не влияет на отображение исходных субтитров, то искать информацию в тексте нет смысла, его точно не меняли. Посмотрим на время появления субтитров и попробуем отыскать закономерность.

Можно заметить, что во всех субтитрах последняя цифра во времени их появления равна либо 5, либо 8. Выпишем последовательность этих цифр, разбив на 10 блоков по 8 цифр.

```
88558585
88558555
88555558
88555858
88585555
88555555
88585585
88555555
88558585
88555555
```

Есть всего два варианта шифрования данных: 5 может быть использована для передачи 0, 8 – для передачи 1 или наоборот.

Если перевести полученные биты в символы с помощью таблицы ASCII, то в первом варианте результатом будет слово КИБЕРАТАКА.

Ответ: КИБЕРАТАКА

Вариант 2

Секретные агенты использовали файлы с субтитрами к фильмам для скрытия тайных сообщений. Они придумали специальный метод сокрытия данных в субтитрах, который не влияет на их отображение и не вызывает подозрений при просмотре фильма.

В прилагаемом файле скрыто одно слово. Определите метод сокрытия и расшифруйте спрятанное слово.

К задаче прилагается:

«Субтитры_в2.srt» – текстовый документ с субтитрами.

Решение

В прилагаемом файле содержится 80 субтитров. Если предположить, что один субтитр используется для скрытия одного бита, то в данном файле скрыто 80 бит или 10 символов.

Поскольку придуманный метод шифрования не влияет на отображение исходных субтитров, то искать информацию в тексте нет смысла, его точно не меняли. Посмотрим на время появления субтитров и попробуем отыскать закономерность.

Можно заметить, что во всех субтитрах последняя цифра во времени их появления равна либо 3, либо 7. Выпишем последовательность этих цифр, разбив на 10 блоков по 8 цифр.

```
77337737
77333333
77373333
77373377
77377333
77337333
77373373
77333737
77337377
77377733
```

Есть всего два варианта шифрования данных: 3 может быть использована для передачи 0, 7 – для передачи 1 или наоборот.

Если перевести полученные биты в символы с помощью таблицы ASCII, то в первом варианте результатом будет слово НАРУШИТЕЛЬ.

Ответ: НАРУШИТЕЛЬ

Задача 5. QR-код

Вариант 1

QR код – это монохромная картинка, в которой закодированы некоторые данные (алфавитно-цифровые символы, побайтовые данные). QR-код состоит из закрашенных и незакрашенных модулей. Закрашенный модуль соответствует «1», незакрашенный – «0».

QR-код типа Micro-QR имеет размер 15x15 модулей и состоит из следующих компонентов (см. рисунок 1):

(1) – поисковый блок (чёрный квадрат размером 3 на 3 модуля, который окружён рамкой из белых модулей, которая окружена рамкой из чёрных модулей, которая окружена рамкой из белых модулей только с внутренних сторон) – расположен в левом верхнем углу;

(2) – горизонтальная синхрополоса (начинаются от самого верхнего правого белого модуля поискового блока и идет, чередуя чёрные и белые модули, по горизонтали);

(3) – вертикальная синхрополоса (начинаются от самого нижнего левого белого модуля поискового блока и идет, чередуя чёрные и белые модули, по вертикали);

(4) – служебная информация, содержащая тип и версию QR-кода (7 бит) – продублирована 2 раза (дубль1 слева-направо, дубль2 снизу-вверх, значение *i-x* битов в дублях совпадают);

(5) – заголовок данных, содержащий формат данных (*1110* – алфавитно-цифровые символы);

(6) – размер данных в символах (8 бит) – число от 0 до 15;

(7) – сами данные и байты заполнителя (при необходимости).

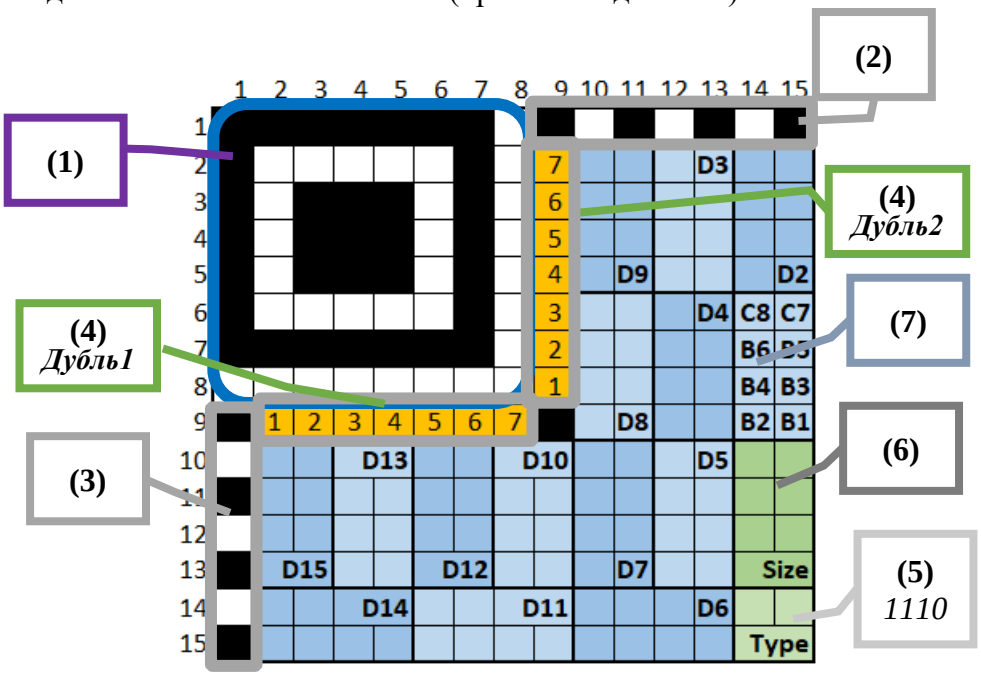


Рисунок 1 – Структура QR-кода

Данные разбиты на блоки по 8 бит. Сами данные закодированы следующим образом: каждому символу ставится в соответствии число из таблицы (см. таблицу 1) размером 6 бит ($B_1, B_2, \dots, B_6$). Оставшиеся 2 бита (C_7, C_8) заполняются значением контрольной суммы по формуле:

$$C_7 = B_1 \oplus B_3 \oplus B_4,$$
$$C_8 = B_2 \oplus B_5 \oplus B_6,$$

где $\oplus$ – сложение по модулю 2 (исключающее ИЛИ, XOR):

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

Таблица 1 – Кодирование/декодирование символов для алфавитно-цифрового режима

Символ	Значение	Дв.значение	Символ	Значение	Дв.значение	Символ	Значение	Дв.значение
0	0	000000	**F**	15	001111	**U**	30	011110
1	1	000001	**G**	16	010000	**V**	31	011111
2	2	000010	**H**	17	010001	**W**	32	100000
3	3	000011	**I**	18	010010	**X**	33	100001
4	4	000100	**J**	19	010011	**Y**	34	100010
5	5	000101	**K**	20	010100	**Z**	35	100011
6	6	000110	**L**	21	010101	**пробел**	36	100100
7	7	000111	**M**	22	010110	**$**	37	100101
8	8	001000	**N**	23	010111	**%**	38	100110
9	9	001001	**O**	24	011000	*****	39	100111
A	10	001010	**P**	25	011001	**+**	40	101000
B	11	001011	**Q**	26	011010	**-**	41	101001
C	12	001100	**R**	27	011011	**.**	42	101010
D	13	001101	**S**	28	011100	**/**	43	101011
E	14	001110	**T**	29	011101	**:**	44	101100

Максимальная длина строки в microQR – 15 символов.

Если данных меньше, то оставшиеся 8-битные блоки заполняются масками 2-х типов, чередуя друг друга: *1101 1100*, *0001 0001*, …, пока не заполнится всё пространство данных QR-кода.

Заполнение начинается с правого нижнего угла, идёт в пределах столбика справа-налево, снизу-вверх. Если текущий модуль занят (например, полосой синхронизации, служебными данными или поисковым блоком), то он пропускается. Если достигнут верх столбика, то движение продолжается с верхнего правого угла столбика, который расположен левее, и идёт сверху вниз. Достигнув низа, движение продолжается от нижнего правого угла столбика, который расположен левее, и идёт снизу-вверх. И так далее, пока всё свободное пространство не будет заполнено. Нумерация и порядок заполнения модулей QR-кода представлен на рис.2.

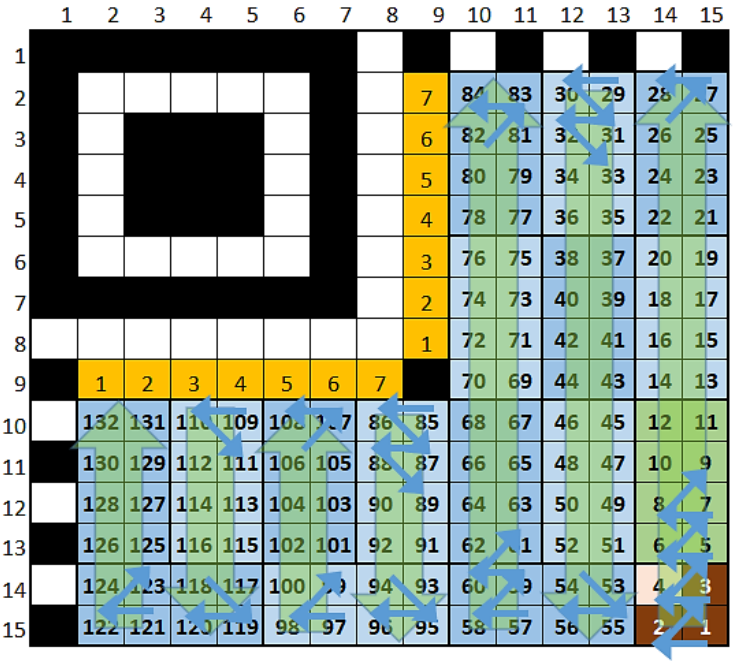
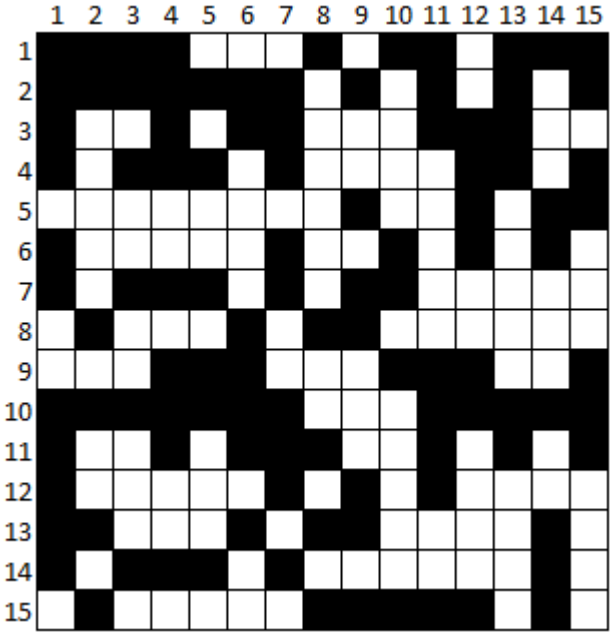


Рисунок 2 – Последовательность битов в блоках QR-кода

По сети передавался QR-код, содержащий некоторое сообщение. Однако в ходе передачи строки QR-кода перепутались. Восстановите исходный QR-код и извлеките сообщение.



Решение

Необходимо определить место каждой строки исходя из структуры QR-кода. Пронумеруем имеющиеся строки и представим их в двоичном формате.

Номер строки	**Двоичный формат**
1.	1 1 1 1 0 0 0 1 0 1 1 0 1 1 1
2.	1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
3.	1 0 0 1 0 1 1 0 0 0 1 1 1 0 0
4.	1 0 1 1 1 0 1 0 0 0 0 1 1 0 1
5.	0 0 0 0 0 0 0 0 1 0 0 1 0 1 1
6.	1 0 0 0 0 0 1 0 0 1 0 1 0 1 0
7.	1 0 1 1 1 0 1 0 1 1 0 0 0 0 0
8.	0 1 0 0 0 1 0 1 1 0 0 0 0 0 0
9.	0 0 0 1 1 1 0 0 0 1 1 1 0 0 1
10.	1 1 1 1 1 1 1 0 0 0 1 1 1 1 1
11.	1 0 0 1 0 1 1 1 0 0 1 0 1 0 1
12.	1 0 0 0 0 0 1 0 1 0 1 0 0 0 0
13.	1 1 0 0 0 1 0 1 1 0 0 0 0 1 0
14.	1 0 1 1 1 0 1 0 0 0 0 0 0 1 0
15.	0 1 0 0 0 0 0 1 1 1 1 1 0 1 0

Исходя из структуры QR-кода, можно выделить обязательные фиксированные блоки – поисковый блок.

На 1-ю позицию(строку) подходит только строка 2, поскольку содержит верхнюю часть квадрата («`111111101010101`»), а также горизонтальную синхрополосу.

На 2-ю позицию подходят только строки 6 или 12 (последовательность «`10000010…`»).

На 3-5-ю позиции подходят строки 4, 7, 14 – они содержат «центральный» квадрат из трех «`1`» и обрамление большого квадрата поискового блока («`10111010…`»).

На 6-ю позицию подходит только строка 12 или 6 (обратная от строки 2).

На 7-ю позицию подходит одна строка из оставшихся – строка 10 («11111110…» – низ квадрата).

На 8-ю позицию подходит только строка 5 («00000000…»).

Теперь разберем нижнюю часть QR-кода.

Известно, что в правом нижнем углу содержится формат данных – «1110». При этом в первом столбце идет вертикальная синхрополоса. Следовательно, необходимо найти строки:

– начинается с «1» и заканчивается на «11» (15-я позиция QR-кода) – подходят строки 1 и 10, но строка 10 уже стоит на 7-й позиции – остается строка 1;

– начинается с «0» и заканчивается на «01» (14-я позиция QR-кода) – подходит только строка 9.

Поле «размер данных» содержит информацию о количестве символов. Из условия известно, что максимальная длина строки – 15 бит. Поскольку на поле длины выделяется 1 байт, то значение 15 будет записано как «00001111». Следовательно, первые 4 бита поля длины всегда будут равны «0000». Осталось найти строки, начинающиеся на «1» и «0» (вертикальная синхрополоса) и заканчивающиеся на «00». Подходят следующие строки:

– на 13-ю позицию – начинается с «1» и заканчивается «00» – подходят строки 3, 7, 12. Однако строки 7 и 12 уже задействованы, значит остается только строка 3;

– на 12-ю позицию – начинается с «0» и заканчивается «00» – строка 8.

Зафиксируем промежуточный результат.

Позиция (номер строки)	**Номер строки из полученного QR-кода**	**Значение строки**
1.	2	1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
2.	6, 12	1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 0
3.	4,	1 0 1 1 1 0 1 0 0 0 0 1 1 0 1
4.	7,	1 0 1 1 1 0 1 0 1 1 0 0 0 0 0
5.	14	1 0 1 1 1 0 1 0 0 0 0 0 0 1 0
6.	12, 6	1 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0
7.	10	1 1 1 1 1 1 1 0 0 0 1 1 1 1 1
8.	5	0 0 0 0 0 0 0 0 1 0 0 1 0 1 1
9.		
10.		
11.		
12.	8	0 1 0 0 0 1 0 1 1 0 0 0 0 0 0
13.	3	1 0 0 1 0 1 1 0 0 0 1 1 1 0 0
14.	9	0 0 0 1 1 1 0 0 0 1 1 1 0 0 1
15.	1	1 1 1 1 0 0 0 1 0 1 1 0 1 1 1

Остались не распределенными строки 11, 13, 15. Из них только строка 15 начинается с «0» – значит в соответствии с вертикальной синхрополосой она встанет на позицию 10.

Известно также, что в 9-й строке в 9-м столбце всегда стоит «1». Из оставшихся двух строк (11, 13) только в строке 13 на 9-м месте стоит «1». Следовательно, строка 13 идет на 9-ю позицию, строка 11 – на 11-ю позицию.

Итоговое распределение строк представлено ниже. Разберем служебную информацию.

Номер строки	Номер строки из полученного QR-кода	Значение строки
1.	2	1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
2.	6, 12	1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 0
3.	4, 7, 14	1 0 1 1 1 0 1 0 0 0 0 1 1 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 0 0 0 1 0
4.		
5.		
6.	12, 6	1 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0
7.	10	1 1 1 1 1 1 1 0 0 0 1 1 1 1 1
8.	5	0 0 0 0 0 0 0 0 1 0 0 1 0 1 1
9.	13	1 1 0 0 0 1 0 1 1 0 0 0 0 1 0
10.	15	0 1 0 0 0 0 0 1 1 1 1 1 0 1 0
11.	11	1 0 0 1 0 1 1 1 0 0 1 0 1 0 1
12.	8	0 1 0 0 0 1 0 1 1 0 0 0 0 0 0
13.	3	1 0 0 1 0 1 1 0 0 0 1 1 1 0 0
14.	9	0 0 0 1 1 1 0 0 0 1 1 1 0 0 1
15.	1	1 1 1 1 0 0 0 1 0 1 1 0 1 1 1

Известно, что служебная информация дублируется в двух местах. Одна копия нам известна (строка 9 биты 2-8) – «1000101».

Вертикальный дубль служебной информации должен ее повторять (столбец 9, строки с 9-й по 2-ю снизу-вверх). Следовательно:

– на 6-й позиции должна находится строка 6;
– на 5-й позиции должна находится строка 14 или 4;
– на 4-й позиции должна находится строка 7;
– на 3-й позиции должна находится строка 4 или 14 (обратное от 5-й позиции);
– на 2-й позиции должна находится строка 12.

Итоговое распределение строк представлено ниже. Разберем получившийся QR-код на блоки.

Номер строки	Номер строки из полученного QR-кода	Значение строки
1.	2	1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
2.	12	1 0 0 0 0 0 1 0 1 0 1 0 0 0 0
3.	4, 14	1 0 1 1 1 0 1 0 0 0 0 1 1 0 1 1 0 1 1 1 0 1 0 0 0 0 0 0 1 0
4.	7	1 0 1 1 1 0 1 0 1 1 0 0 0 0 0
5.	14, 4	1 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 0 0 0 0 1 1 0 1
6.	6	1 0 0 0 0 0 1 0 0 1 0 1 0 1 0
7.	10	1 1 1 1 1 1 1 0 0 0 1 1 1 1 1
8.	5	0 0 0 0 0 0 0 0 1 0 0 1 0 1 1
9.	13	1 1 0 0 0 1 0 1 1 0 0 0 0 1 0
10.	15	0 1 0 0 0 0 0 1 1 1 1 1 0 1 0
11.	11	1 0 0 1 0 1 1 1 0 0 1 0 1 0 1
12.	8	0 1 0 0 0 1 0 1 1 0 0 0 0 0 0
13.	3	1 0 0 1 0 1 1 0 0 0 1 1 1 0 0
14.	9	0 0 0 1 1 1 0 0 0 1 1 1 0 0 1
15.	1	1 1 1 1 0 0 0 1 0 1 1 0 1 1 1

Байт2

Байт1
01111101

Размер данных
$00001001_2 = 9_{10}$

Тип данных
1110

Из восстановленной части QR-кода можно определить, что длина сообщения составляет 9 символов. Известен 1-й символ, который определяется по таблице – *01111101 = 011111 01 = V*.

Второй символ может принять следующие значения исходя из комбинации строк 4 и 14 на позициях 3 и 5:

10000100 – на 3-й позиции строка 14, на 5-й позиции строка 4,

01001000 – на 3-й позиции строка 4, на 5-й позиции строка 14.

Посчитаем контрольную сумму по формуле из условия:

10000100 – 100001 00:

$C_7 = B_1 \oplus B_3 \oplus B_4$: $0 = 1 \oplus 0 \oplus 0$ – неверно

$C_8 = B_2 \oplus B_5 \oplus B_6$: $0 = 0 \oplus 0 \oplus 1$ – неверно

01001000 – 010010 00:

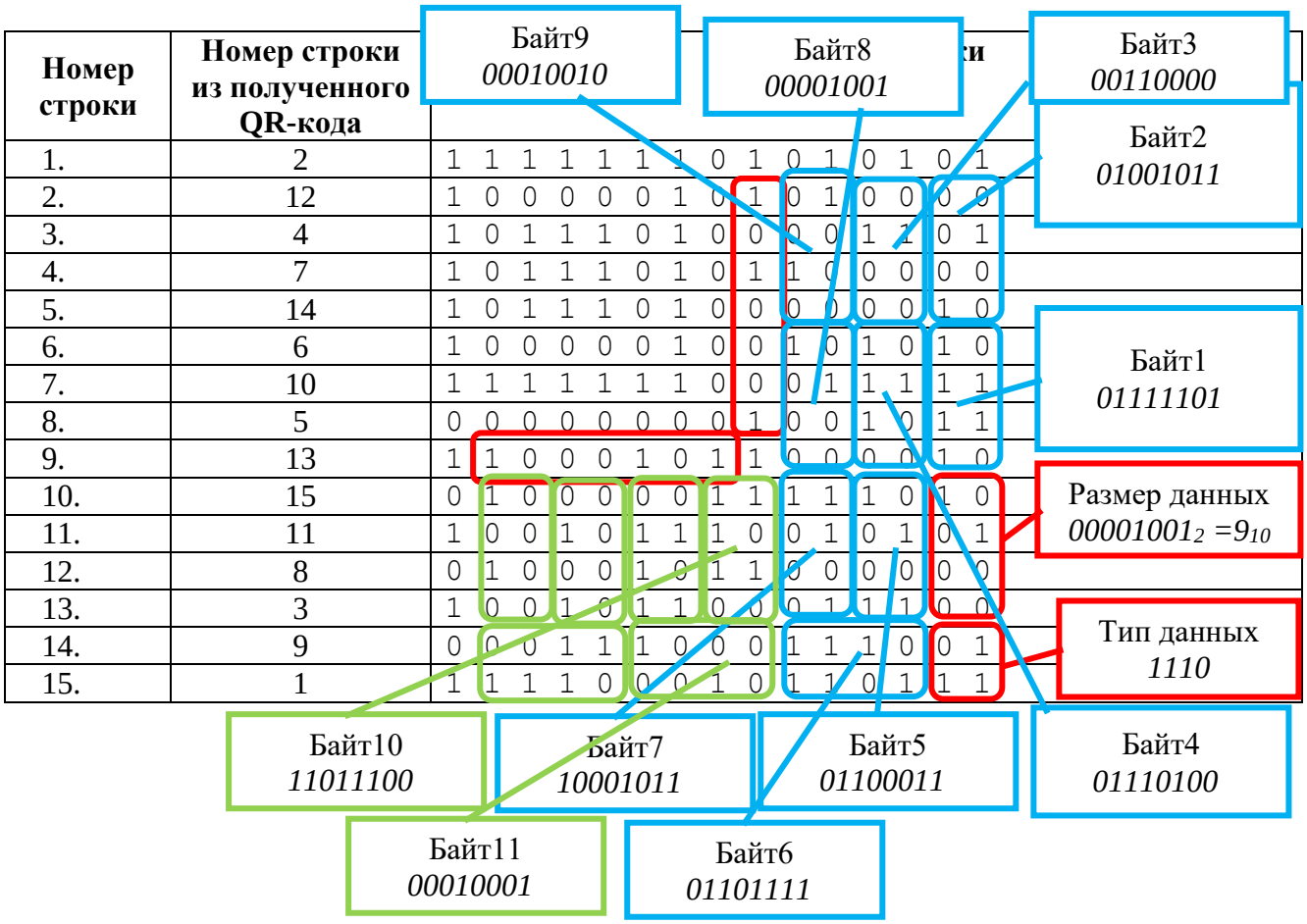
$C_7 = B_1 \oplus B_3 \oplus B_4$: $0 = 0 \oplus 0 \oplus 0$ – верно

$C_8 = B_2 \oplus B_5 \oplus B_6$: $0 = 1 \oplus 1 \oplus 0$ – верно

Следовательно, на 3-й позиции будет строка 4, на 5-й позиции – строка 14.

Построим окончательный QR-код и разобьем на блоки данных.

Номер строки	Номер строки из полученного QR-кода	
1.	2	1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
2.	12	1 0 0 0 0 0 1 0 1 0 1 0 0 0 0
3.	4	1 0 1 1 1 0 1 0 0 0 0 1 1 0 1
4.	7	1 0 1 1 1 0 1 0 1 1 0 0 0 0 0
5.	14	1 0 1 1 1 0 1 0 0 0 0 0 0 1 0
6.	6	1 0 0 0 0 0 1 0 0 1 0 1 0 1 0
7.	10	1 1 1 1 1 1 1 0 0 0 1 1 1 1 1
8.	5	0 0 0 0 0 0 0 0 1 0 0 1 0 1 1
9.	13	1 1 0 0 0 1 0 1 1 0 0 0 0 1 0
10.	15	0 1 0 0 0 0 0 1 1 1 1 1 0 1 0
11.	11	1 0 0 1 0 1 1 1 0 0 1 0 1 0 1
12.	8	0 1 0 0 0 1 0 1 1 0 0 0 0 0 0
13.	3	1 0 0 1 0 1 1 0 0 0 1 1 1 0 0
14.	9	0 0 0 1 1 1 0 0 0 1 1 1 0 0 1
15.	1	1 1 1 1 0 0 0 1 0 1 1 0 1 1 1



Байт 1 – *01111101 = 011111 01* – V
Байт 2 – *01001011 = 010010 11* – I
Байт 3 – *00110000 = 001100 00* – C
Байт 4 – *01110100 = 011101 00* – T
Байт 5 – *01100011 = 011000 11* – O
Байт 6 – *01101111 = 011011 11* – R
Байт 7 – *10001011 = 100010 11* – Y
Байт 8 – *00001001 = 000010 01* – 2
Байт 9 – *00010010 = 000100 10* – 4

Байты 10,11 и дальше повторяют маску: «*11011100 00010001*», что подтверждает корректность полученного QR-кода.

Ответ: VICTORY24

Вариант 2

QR код – это монохромная картинка, в которой закодированы некоторые данные (алфавитно-цифровые символы, побайтовые данные). QR-код состоит из закрашенных и незакрашенных модулей. Закрашенный модуль соответствует «1», незакрашенный – «0».

QR-код типа Micro-QR имеет размер 15x15 модулей и состоит из следующих компонентов (см. рисунок 1):

(1) – поисковый блок (чёрный квадрат размером 3 на 3 модуля, который окружён рамкой из белых модулей, которая окружена рамкой из чёрных модулей, которая окружена рамкой из белых модулей только с внутренних сторон) – расположен в левом верхнем углу;

(2) – горизонтальная синхрополоса (начинаются от самого верхнего правого белого модуля поискового блока и идет, чередуя чёрные и белые модули, по горизонтали);

(3) – вертикальная синхрополоса (начинаются от самого нижнего левого белого модуля поискового блока и идет, чередуя чёрные и белые модули, по вертикали);

(4) – служебная информация, содержащая тип и версию QR-кода (7 бит) – продублирована 2 раза (дубль1 слева-направо, дубль2 снизу-вверх, значение *i-x* битов в дублях совпадают);

(5) – заголовок данных, содержащий формат данных (*1110* – алфавитно-цифровые символы);

(6) – размер данных в символах (8 бит) – число от 0 до 15;

(7) – сами данные и байты заполнителя (при необходимости).

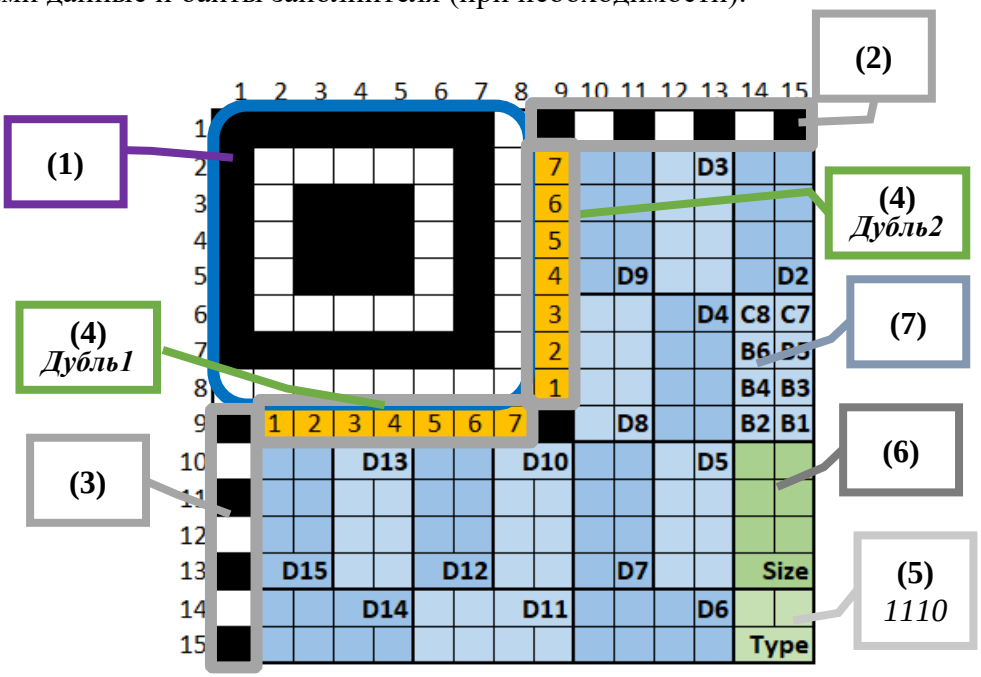


Рисунок 1 – Структура QR-кода

Данные разбиты на блоки по 8 бит. Сами данные закодированы следующим образом: каждому символу ставится в соответствии число из таблицы (см. таблицу 1) размером 6 бит (B_1, B_2, … , B_6). Оставшиеся 2 бита (C_7, C_8) заполняются значением контрольной суммы по формуле:

$$C_7 = B_1 \oplus B_3 \oplus B_4,$$
$$C_8 = B_2 \oplus B_5 \oplus B_6,$$

где $\oplus$ – сложение по модулю 2 (исключающее ИЛИ, XOR):

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

Таблица 1 – Кодирование/декодирование символов для алфавитно-цифрового режима

Символ	Значение	Дв.значение	Символ	Значение	Дв.значение	Символ	Значение	Дв.значение
0	0	000000	**F**	15	001111	**U**	30	011110
1	1	000001	**G**	16	010000	**V**	31	011111
2	2	000010	**H**	17	010001	**W**	32	100000
3	3	000011	**I**	18	010010	**X**	33	100001
4	4	000100	**J**	19	010011	**Y**	34	100010
5	5	000101	**K**	20	010100	**Z**	35	100011
6	6	000110	**L**	21	010101	**пробел**	36	100100
7	7	000111	**M**	22	010110	**$**	37	100101
8	8	001000	**N**	23	010111	**%**	38	100110

9	9	001001	**O**	24	011000	*****	39	100111
A	10	001010	**P**	25	011001	**+**	40	101000
B	11	001011	**Q**	26	011010	**-**	41	101001
C	12	001100	**R**	27	011011	**.**	42	101010
D	13	001101	**S**	28	011100	**/**	43	101011
E	14	001110	**T**	29	011101	**:**	44	101100

Максимальная длина строки в microQR – 15 символов.

Если данных меньше, то оставшиеся 8-битные блоки заполняются масками 2-х типов, чередуя друг друга: *1101 1100*, *0001 0001*, …, пока не заполнится всё пространство данных QR-кода.

Заполнение начинается с правого нижнего угла, идёт в пределах столбика справа-налево, снизу-вверх. Если текущий модуль занят (например, полосой синхронизации, служебными данными или поисковым блоком), то он пропускается. Если достигнут верх столбика, то движение продолжается с верхнего правого угла столбика, который расположен левее, и идёт сверху вниз. Достигнув низа, движение продолжается от нижнего правого угла столбика, который расположен левее, и идёт снизу-вверх. И так далее, пока всё свободное пространство не будет заполнено. Нумерация и порядок заполнения модулей QR-кода представлен на рис.2.

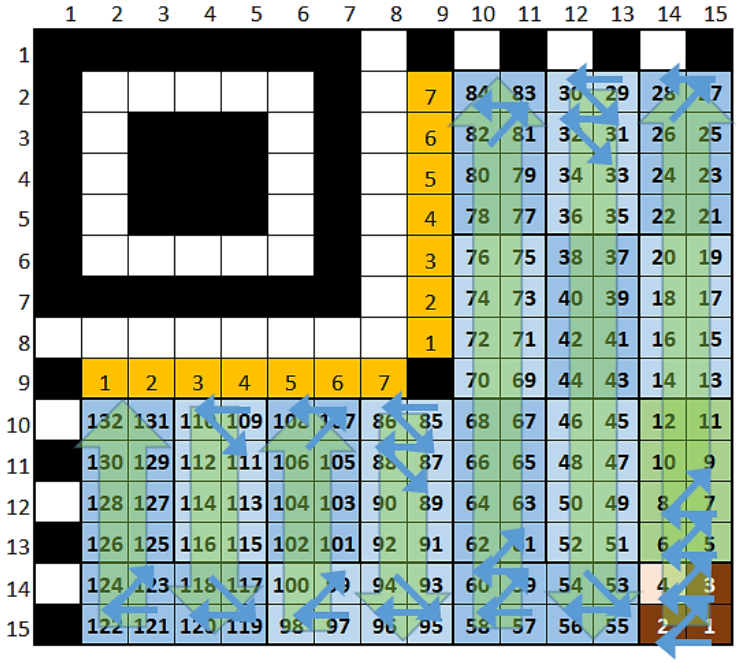
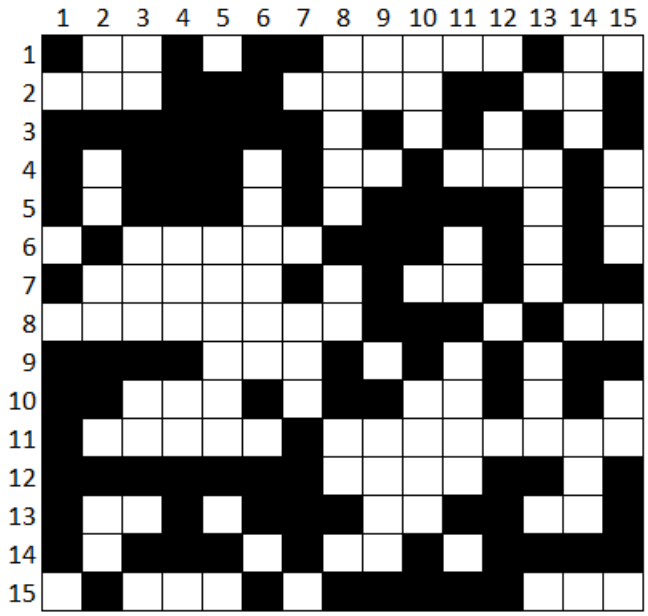


Рисунок 2 – Последовательность битов в блоках QR-кода

По сети передавался QR-код, содержащий некоторое сообщение. Однако в ходе передачи строки QR-кода перепутались. Восстановите исходный QR-код и извлеките сообщение.



Решение

Необходимо определить место каждой строки исходя из структуры QR-кода. Пронумеруем имеющиеся строки и представим их в двоичном формате.

Номер строки	**Двоичный формат**
1.	1 0 0 1 0 1 1 0 0 0 0 0 1 0 0
2.	0 0 0 1 1 1 0 0 0 0 1 1 0 0 1
3.	1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
4.	1 0 1 1 1 0 1 0 0 1 0 0 0 1 0
5.	1 0 1 1 1 0 1 0 1 1 1 1 0 1 0
6.	0 1 0 0 0 0 0 1 1 1 0 1 0 1 0
7.	1 0 0 0 0 0 1 0 1 0 0 1 0 1 1
8.	0 0 0 0 0 0 0 0 1 1 1 0 1 0 0
9.	1 1 1 1 0 0 0 1 0 1 0 1 0 1 1
10.	1 1 0 0 0 1 0 1 1 0 0 1 0 1 0
11.	1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
12.	1 1 1 1 1 1 1 0 0 0 0 1 1 0 1
13.	1 0 0 1 0 1 1 1 0 0 1 1 0 0 1
14.	1 0 1 1 1 0 1 0 0 1 0 1 1 1 1
15.	0 1 0 0 0 1 0 1 1 1 1 1 0 0 0

Исходя из структуры QR-кода, можно выделить обязательные фиксированные блоки – поисковый блок.

На 1-ю позицию(строку) подходит только строка 3, поскольку содержит верхнюю часть квадрата («11111101010101»), а также горизонтальную синхрополосу.

На 2-ю позицию подходят только строки 7 или 11 (последовательность «10000010…»).

На 3-5-ю позиции подходят строки 4, 5, 14 – они содержат «центральный» квадрат из трех «1» и обрамление большого квадрата поискового блока («10111010…»).

На 6-ю позицию подходит только строка 11 или 7 (обратная от строки 2).

На 7-ю позицию подходит одна строка из оставшихся – строка 12 («11111110…» – низ квадрата).

На 8-ю позицию подходит только строка 8 («00000000…»).

Теперь разберем нижнюю часть QR-кода.

Известно, что в правом нижнем углу содержится формат данных – «1110». При этом в первом столбце идет вертикальная синхрополоса. Следовательно, необходимо найти строки:

– начинается с «1» и заканчивается на «11» (15-я позиция QR-кода) – подходят строки 7, 9, 14, но строки 7 и 14 уже стоят на позициях, остается только строка 9;

– начинается с «0» и заканчивается на «01» (14-я позиция QR-кода) – подходит только строка 2.

Поле «размер данных» содержит информацию о количестве символов. Из условия известно, что максимальная длина строки – 15 бит. Поскольку на поле длины выделяется 1 байт, то значение 15 будет записано как «00001111». Следовательно, первые 4 бита поля длины всегда будут равны «0000». Осталось найти строки, начинающиеся на «1» и «0» (вертикальная синхрополоса) и заканчивающиеся на «00». Подходят следующие строки:

– на 13-ю позицию – начинается с «1» и заканчивается «00» – подходят строки 1, 11. Однако строка 11 уже задействована, значит остается только строка 1;

– на 12-ю позицию – начинается с «0» и заканчивается «00» – подходят строки 8, 15. Однако строка 8 уже задействована, значит остается только строка 15.

Зафиксируем промежуточный результат.

Позиция (номер строки)	**Номер строки из полученного QR-кода**	**Значение строки**
1.	3	1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
2.	7, 11	1 0 0 0 0 0 1 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
3.	4,	1 0 1 1 1 0 1 0 0 1 0 0 0 1 0
4.	5,	1 0 1 1 1 0 1 0 1 1 1 1 0 1 0
5.	14	1 0 1 1 1 0 1 0 0 1 0 1 1 1 1
6.	11, 7	1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 0 1 1
7.	12	1 1 1 1 1 1 1 0 0 0 0 1 1 0 1
8.	8	0 0 0 0 0 0 0 0 1 1 1 0 1 0 0
9.		
10.		
11.		
12.	15	0 1 0 0 0 1 0 1 1 1 1 1 0 0 0
13.	1	1 0 0 1 0 1 1 0 0 0 0 0 1 0 0
14.	2	0 0 0 1 1 1 0 0 0 0 1 1 0 0 1
15.	9	1 1 1 1 0 0 0 1 0 1 0 1 0 1 1

Остались не распределенными строки 6, 10, 13. Из них только строка 6 начинается с «0» – значит в соответствии с вертикальной синхрополосой она встанет на позицию 10.

Известно также, что в 9-й строке в 9-м столбце всегда стоит «1». Из оставшихся двух строк (10, 13) только в строке 10 на 9-м месте стоит «1». Следовательно, строка 10 идет на 9-ю позицию, строка 13 – на 11-ю позицию.

Итоговое распределение строк представлено ниже. Разберем служебную информацию.

Номер строки	Номер строки из полученного QR-кода	Значение строки
1.	3	1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
2.	7, 11	1 0 0 0 0 0 1 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
3.	4, 5, 14	1 0 1 1 1 0 1 0 0 1 0 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1 1 1 1
4.		
5.		
6.	11, 7	1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 0 1 1
7.	12	1 1 1 1 1 1 1 0 0 0 0 1 1 0 1
8.	8	0 0 0 0 0 0 0 0 1 1 1 0 1 0 0
9.	10	1 1 0 0 0 1 0 1 1 0 0 1 0 1 0
10.	6	0 1 0 0 0 0 0 1 1 1 0 1 0 1 0
11.	13	1 0 0 1 0 1 1 1 0 0 1 1 0 0 1
12.	15	0 1 0 0 0 1 0 1 1 1 1 1 0 0 0
13.	1	1 0 0 1 0 1 1 0 0 0 0 0 1 0 0
14.	2	0 0 0 1 1 1 0 0 0 0 1 1 0 0 1
15.	9	1 1 1 1 0 0 0 1 0 1 0 1 0 1 1

Известно, что служебная информация дублируется в двух местах. Одна копия нам известна (строка 9 биты 2-8) – «1000101».

Вертикальный дубль служебной информации должен ее повторять (столбец 9, строки с 9-й по 2-ю снизу-вверх). Следовательно:

– на 6-й позиции должна находится строка 11;
– на 5-й позиции должна находится строка 14 или 4;
– на 4-й позиции должна находится строка 5;
– на 3-й позиции должна находится строка 4 или 14 (обратное от 5-й позиции);
– на 2-й позиции должна находится строка 7.

Итоговое распределение строк представлено ниже. Разберем получившийся QR-код на блоки.

Номер строки	Номер строки из полученного QR-кода	Значение строки
1.	3	1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
2.	7	1 0 0 0 0 0 1 0 1 0 0 1 0 1 1
3.	4, 14	1 0 1 1 1 0 1 0 0 1 0 0 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1 1 1 1
4.	5	1 0 1 1 1 0 1 0 1 1 1 1 0 1 0
5.	14, 4	1 0 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 0
6.	11	1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
7.	12	1 1 1 1 1 1 1 0 0 0 0 1 1 0 1
8.	8	0 0 0 0 0 0 0 0 1 1 1 0 1 0 0
9.	10	1 1 0 0 0 1 0 1 1 0 0 1 0 1 0
10.	6	0 1 0 0 0 0 0 1 1 1 0 1 0 1 0
11.	13	1 0 0 1 0 1 1 1 0 0 1 1 0 0 1
12.	15	0 1 0 0 0 1 0 1 1 1 1 1 0 0 0
13.	1	1 0 0 1 0 1 1 0 0 0 0 0 1 0 0
14.	2	0 0 0 1 1 1 0 0 0 0 1 1 0 0 1
15.	9	1 1 1 1 0 0 0 1 0 1 0 1 0 1 1

Байт2

Байт1
01001000

Размер данных
$00001001_2 = 9_{10}$

Тип данных
1110

Из восстановленной части QR-кода можно определить, что длина сообщения составляет 9 символов. Известен 1-й символ, который определяется по таблице – *01001000 = 010010 00 = I.*

Второй символ может принять следующие значения исходя из комбинации строк 4 и 14 на позициях 3 и 5:

01011111 – на 3-й позиции строка 14, на 5-й позиции строка 4,

11010111 – на 3-й позиции строка 4, на 5-й позиции строка 14.

Посчитаем контрольную сумму по формуле из условия:

01011111 – 010111 11:

$C_7 = B_1 \oplus B_3 \oplus B_4$: 1 = 0 ⊕ 0 ⊕ 1 – верно

$C_8 = B_2 \oplus B_5 \oplus B_6$: 1 = 1 ⊕ 1 ⊕ 1 – верно

11010111 – 110101 11:

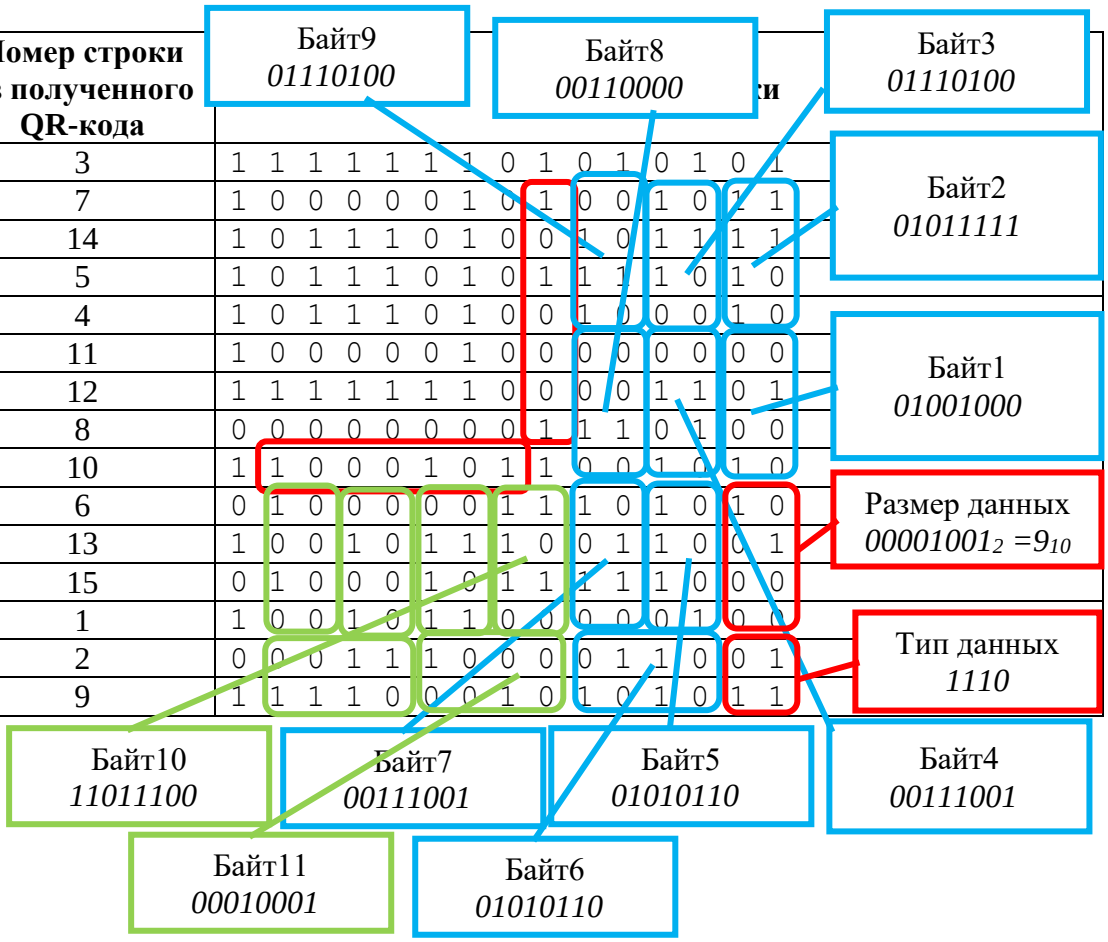
$C_7 = B_1 \oplus B_3 \oplus B_4$: 1 = 1 ⊕ 0 ⊕ 1 – неверно

$C_8 = B_2 \oplus B_5 \oplus B_6$: 1 = 1 ⊕ 0 ⊕ 1 – неверно

Следовательно, на 3-й позиции будет строка 14, на 5-й позиции – строка 4.

Построим окончательный QR-код и разобьем на блоки данных.

Номер строки	Номер строки из полученного QR-кода	
1.	3	1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
2.	7	1 0 0 0 0 0 1 0 1 0 0 1 0 1 1
3.	14	1 0 1 1 1 0 1 0 0 1 0 1 1 1 1
4.	5	1 0 1 1 1 0 1 0 1 1 1 1 0 1 0
5.	4	1 0 1 1 1 0 1 0 0 1 0 0 0 1 0
6.	11	1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
7.	12	1 1 1 1 1 1 1 0 0 0 0 1 1 0 1
8.	8	0 0 0 0 0 0 0 0 1 1 1 0 1 0 0
9.	10	1 1 0 0 0 1 0 1 1 0 0 1 0 1 0
10.	6	0 1 0 0 0 0 0 1 1 1 0 1 0 1 0
11.	13	1 0 0 1 0 1 1 1 0 0 1 1 0 0 1
12.	15	0 1 0 0 0 1 0 1 1 1 1 1 0 0 0
13.	1	1 0 0 1 0 1 1 0 0 0 0 0 1 0 0
14.	2	0 0 0 1 1 1 0 0 0 0 1 1 0 0 1
15.	9	1 1 1 1 0 0 0 1 0 1 0 1 0 1 1



Байт 1 – *01001000* = *010010 00* – I
Байт 2 – *01011111* = *010111 11* – N
Байт 3 – *01110100* = *011101 00* – T
Байт 4 – *00111001* = *001110 01* – E
Байт 5 – *01010110* = *010101 10* – L
Байт 6 – *01010110* = *010101 10* – L
Байт 7 – *00111001* = *001110 01* – E
Байт 8 – *00110000* = *001100 00* – C
Байт 9 – *01110100* = *011101 00* – T

Байты 10,11 и дальше повторяют маску: «*11011100 00010001*», что подтверждает корректность полученного QR-кода.

Ответ: INTELLECT