

**Материалы заданий Межрегиональной олимпиады школьников имени
И.Я. Верченко по информатике и компьютерной безопасности
(2019/2020 учебный год)**

Содержание

8-10 КЛАССЫ	2
Условия задач отборочного этапа	2
Задача 1. Заражение вирусом.....	2
Задача 2. Хэш пароля.....	2
Задача 3. Пароль.....	3
Задача 4. Скрытое сообщение	3
Задача 5. Блок-схема.....	3
Условия задач заключительного этапа	4
Задача 1. Парольная комбинация	4
Задача 2. Сетевой трафик.....	9
Задача 3. Шифрование.....	10
Задача 4. Стеганография	13
Задача 5. Web-сайт.....	14
11 КЛАСС	17
Условия задач отборочного этапа	17
Задача 1. Удалённый файл	17
Задача 2. Гамма	19
Задача 3. Ключи	19
Задача 4. Шифр	20
Задача 5. Блок-схема.....	21
Условия задач заключительного этапа	22
Задача 1. Парольная комбинация	22
Задача 2. Сетевой трафик.....	27
Задача 3. Вирус.....	30
Задача 4. Стеганография	31
Задача 5. Разработка процессора.....	32
Критерии оценивания заданий Межрегиональной олимпиады школьников имени И.Я. Верченко по информатике и компьютерной безопасности (2018/2019 учебный год)	35

8-10 КЛАССЫ

Условия задач отборочного этапа

Задача 1. Заражение вирусом

Пользователь в начале рабочего дня скачал из сети Интернет программу, в которой содержится вирус. Вирус способен делать свою копию в другом исполняемом файле при каждом запуске программы-носителя, но только раз в сутки, начиная со следующего после заражения дня. Сколько программ в системе будет заражено через 11 суток работы компьютера, если предположить, что каждая имеющаяся на компьютере программа запускается 1 раз в сутки, а всего в компьютере 10000 программ.

Ответ: 2048.

Задача 2. Хэш пароля

Пользователь использует компьютерную систему, в которой по введенному паролю вычисляется и хранится хеш-значение. Хэш-значение пароля вычисляется функцией *GetHash* (см. листинг).

Политика безопасности системы требует, чтобы пароль содержал *не более пяти символов* и состоял только из *букв латинского алфавита в нижнем регистре в кодировке ASCII*.

```

/* Входные параметры:
   PWD - строка пароля
   HASH - переменная, в которой будет содержаться хеш
Возвращаемое значение:
   false - пароль не соответствует правилам системы;
   true - пароль соответствует правилам системы;
   int HASH - хеш пароля (целое число)
*/
bool GetHash( char const *Pwd, int *hash )
{
    // Длина пароля
    int n = strlen(Pwd);
    // Проверка пароля на длину
    if(n > 5)
        return false;
    // Проверка пароля на содержание только латинских букв нижнего
    регистра
    for(int i = 0; i < n; i++)
        if( Pwd[i] < 'a' || Pwd[i] > 'z')
            return false;
    //Вычисление хеша
    *hash = 0;
    for(int i = 0; i < n; i++)
        *hash += (Pwd[i] - 'a' + 1);

    return true;
}

```

У какого пароля хеш-значение будет меньше 30?

Варианты:

- а) asdfg
- б) qwер
- в) sabbe
- г) basic

Ответ: **в) sabbe.**

Задача 3. Пароль

У пользователя в качестве пароля компьютерной системы используется такое *простое четырехзначное число*, для которого *первая цифра равна сумме трех последующих*.

Какое максимальное количество паролей подходит под указанные условия?

Ответ: **14.**

Задача 4. Скрытое сообщение

Взаимодействие между двумя пользователями осуществляется по каналу связи (их роли в сеансе связи: *отправитель* и *приемник*), позволяющему передавать несколько сообщений в произвольном порядке. Размеры сообщений кратны 1 Кб. В ходе осуществления очередного сеанса связи было отправлено 3 *сообщения общим размером 13 Кб*. Для однозначного определения размеров сообщений в сеансе связи *приемнику* потребовалось дополнительно сообщить следующее:

- а) величину, равную произведению их размеров,
 - б) сообщение наибольшего размера было отправлено первым.
- Сообщения каких размеров были переданы?

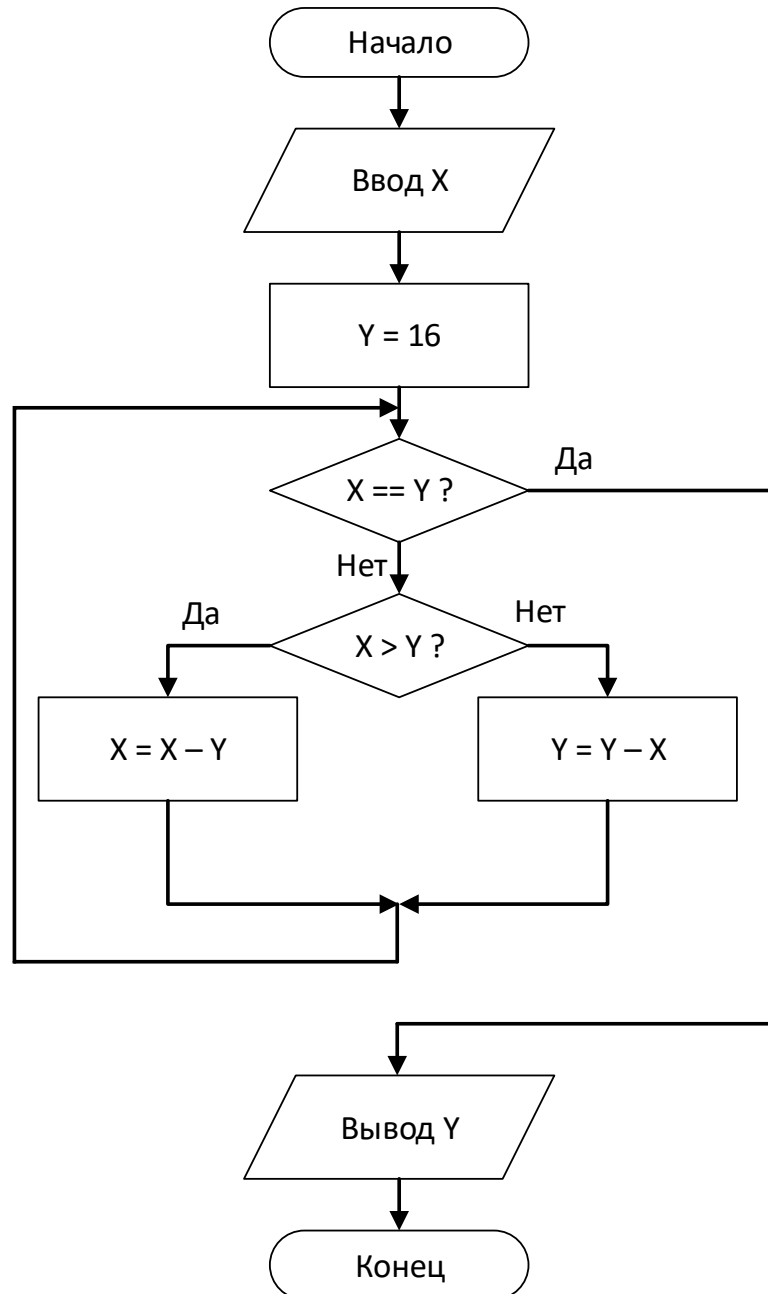
Варианты:

- а) (9,2,2)
- б) (6,5,2)
- в) (5,4,3)
- г) (11,1,1)

Ответ: **а) (9,2,2).**

Задача 5. Блок-схема

На рисунке представлена блок-схема алгоритма обработки целого числа X , вводимого пользователем.



Что выведет программа, если на вход подать 66?

Ответ: **2.**

Условия задач заключительного этапа

Задача 1. Парольная комбинация

Для входа в систему используется пароль, состоящий из трёх двузначных чисел, расположенных следующим образом:

$xx - xx - xx$

Известно, что пароль состоит из 3-х неповторяющихся простых чисел. При этом, последняя цифра первого числа равна первой цифре второго числа, а последняя цифра второго числа равна первой цифре третьего числа.

Пример:

$$x1-17-7y$$

Задержка между попытками входа в систему равна 1 секунде. За какое *минимальное* время (в секундах) можно *гарантированно* получить пароль, если на ввод пароля время не тратится, и количество попыток ввода пароля не ограничено?

Решение

Для начала необходимо найти все простые числа в диапазоне от 11 до 99 (двузначные числа). *Простым* является такое число, которое делится только на себя и на 1. Сделать это можно перебором, проверяя, не делится ли проверяемое число на какое-либо от 2-х до самого числа (либо половины числа).

Для ускорения напишем программу, которая выводит на экран все простые числа в указанном диапазоне.

Листинг программы на языке Python.

```
# Функция, определяющая, является ли число простым
# ПАРАМЕТР:
# n - проверяемое число
# ВОЗВРАЩАЕТ:
# True - число n простое
# False - число n непростое
def isprime(n):
    if n == 1:
        return True
    for d in range(2, n//2):
        if n % d == 0:
            return False
    return True

# Цикл перебора чисел от 11 до 99
# Для каждого числа вызывается функция isprime()
# Если функция вернула True - число добавляется
# в массив list_input
# Счетчик Count считает количество простых чисел
Count = 0
list_input = []
for x in range(11,100):
    if isprime(x) == True:
        list_input.append(x)
        Count += 1
print(list_input)
print("Total:", Count)
```

Листинг программы на языке C.

```
// Функция, определяющая, является ли число простым
// ПАРАМЕТР:
// n - проверяемое число
// ВОЗВРАЩАЕТ:
// true - число n простое
```

```

// false - число n непростое
bool isprime(int n)
{
    if (n == 1)
        return true;
    for (int d = 2; d < n / 2; d++)
        if (n % d == 0)
            return false;
    return true;
}

int main()
{
    // Цикл перебора чисел от 11 до 99
    // Для каждого числа вызывается функция isprime()
    // Если функция вернула true - число добавляется
    // в массив
    // Счетчик Count считает количество простых чисел
    int Count = 0;
    int list_input[100] = { 0 };
    // Формирование массива простых чисел
    for (int x = 11; x < 100; x++)
    {
        if (isprime(x) == true)
        {
            list_input[Count] = x;
            Count += 1;
        }
    }
    // Вывод на экран
    for (int i = 0; i < Count; i++)
    {
        printf("%d ", list_input[i]);
    }
    printf("\nTotal: %d\n", Count);
    return 0;
}

```

Результат работы программы:

```

11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
Total: 21

```

Всего 21 простых чисел в диапазоне от 11 до 99. Теперь необходимо выделить все возможные комбинации, удовлетворяющие условию *«последняя цифра первого числа равна первой цифре второго числа, а последняя цифра второго числа равна первой цифре третьего числа»*.

Сделать это можно перебором всех комбинаций. В цикле перебираем всевозможные числа и проверяем два условия:

- все числа разные;
- последняя цифра первого числа равна первой цифре второго числа, а последняя цифра второго числа равна первой цифре третьего числа.

Если все условия выполнены, то увеличиваем значение счетчика. В результате счетчик будет содержать количество комбинаций, удовлетворяющих заданию. Поскольку задержка между вводом паролей равна 1 секунде, а после ввода

последней комбинации задержки нет, то ответ – количество найденных комбинаций минус 1.

Пример программы представлен в листингах ниже.

Листинг программы на языке Python.

```
# Функция возвращает список всех комбинаций чисел,
# удовлетворяющих заданному условию паролей
# ПАРАМЕТР:
#     list_input - список чисел, из которых будет
#                 сформироваться комбинация
# ВОЗВРАЩАЕТ:
#     list_combination - список комбинаций
#                 из 3-х чисел, удовлетворяющих условию
def combination(list_input):
    list_combination = []
    for a in list_input:
        for b in list_input:
            for c in list_input:
                if a != b and b != c and a != c:
                    if str(a)[-1] == str(b)[0] and
                       str(b)[-1] == str(c)[0]:
                        list_combination.append(str(a) +
                                                "-" + str(b) + "-" + str(c))
    return list_combination

# Заполнение списка всеми простыми числами
# в заданном диапазоне
list_input = []
for x in range(11,100):
    if isprime(x) == True:
        list_input.append(x)
# Список, содержащий все комбинации,
# удовлетворяющие заданному условию
result_list = combination(list_input)
# Вывод всех комбинаций
for i in result_list:
    print(i)
# Вывод количества комбинаций
print("Total:", len(result_list))
```

Листинг программы на языке C.

```
// Функция выводит на экран все комбинации,
// удовлетворяющие условию
// ПАРАМЕТРЫ:
//     list_input - массив чисел, из которых будут
//                 строиться комбинации
//     count - количество чисел в массиве list_input
// ВОЗВРАЩАЕТ:
//     выводит на экран комбинации, удовлетворяющие
//     условию
//     выводит на экран количество комбинаций
void combination(int list_input[], int count)
{
    int last_digit1;
```

```

int first_digit2;
int last_digit2;
int first_digit3;
int total = 0;
for (int i = 0; i < count; i++)
    for (int j = 0; j < count; j++)
        for (int k = 0; k < count; k++)
            {
                if ( list_input[i] != list_input[j] &&
                    list_input[j] != list_input[k] &&
                    list_input[i] != list_input[k] )
                    {
                        // последняя цифра числа
                        last_digit1 = list_input[i] % 10;
                        // первая цифра числа
                        first_digit2 = list_input[j] / 10;
                        last_digit2 = list_input[j] % 10;
                        first_digit3 = list_input[k] / 10;
                        if ( last_digit1 == first_digit2 &&
                            last_digit2 == first_digit3 )
                            {
                                printf("%d-%d-%d\n",
                                    list_input[i],
list_input[j],
                                    list_input[k]);
                                total++;
                            }
                    }
            }
// Вывод общего количества комбинаций
printf("Total: %d\n", total);
}

```

В результате выполнения программы получим следующие данные:

```

11-13-31, 11-13-37, 11-17-71, 11-17-73, 11-17-79, 11-19-97, 13-
31-11, 13-31-17, 13-31-19, 13-37-71, 13-37-73, 13-37-79, 17-71-11,
17-71-13, 17-71-19, 17-73-31, 17-73-37, 17-79-97, 19-97-71, 19-97-
73, 19-97-79, 23-31-11, 23-31-13, 23-31-17, 23-31-19, 23-37-71, 23-
37-73, 23-37-79, 29-97-71, 29-97-73, 29-97-79, 31-11-13, 31-11-17,
31-11-19, 31-13-37, 31-17-71, 31-17-73, 31-17-79, 31-19-97, 37-71-
11, 37-71-13, 37-71-17, 37-71-19, 37-73-31, 37-79-97, 41-11-13, 41-
11-17, 41-11-19, 41-13-31, 41-13-37, 41-17-71, 41-17-73, 41-17-79,
41-19-97, 43-31-11, 43-31-13, 43-31-17, 43-31-19, 43-37-71, 43-37-
73, 43-37-79, 47-71-11, 47-71-13, 47-71-17, 47-71-19, 47-73-31, 47-
73-37, 47-79-97, 53-31-11, 53-31-13, 53-31-17, 53-31-19, 53-37-71,
53-37-73, 53-37-79, 59-97-71, 59-97-73, 59-97-79, 61-11-13, 61-11-
17, 61-11-19, 61-13-31, 61-13-37, 61-17-71, 61-17-73, 61-17-79, 61-
19-97, 67-71-11, 67-71-13, 67-71-17, 67-71-19, 67-73-31, 67-73-37,
67-79-97, 71-11-13, 71-11-17, 71-11-19, 71-13-31, 71-13-37, 71-17-
73, 71-17-79, 71-19-97, 73-31-11, 73-31-13, 73-31-17, 73-31-19, 73-
37-71, 73-37-79, 79-97-71, 79-97-73, 83-31-11, 83-31-13, 83-31-17,
83-31-19, 83-37-71, 83-37-73, 83-37-79, 89-97-71, 89-97-73, 89-97-
79, 97-71-11, 97-71-13, 97-71-17, 97-71-19, 97-73-31, 97-73-37
Total: 126

```


Всего таких комбинаций будет 126, следовательно, максимальное затраченное время будет равно

$$(126 - 1) * 1 \text{ сек} = 125 \text{ сек.}$$

Ответ: минимальное время, за которое можно гарантированно получить пароль, равно **125 секунд**.

Задача 2. Сетевой трафик

Был получен фрагмент сетевого трафика пользователя при взаимодействии с игровым сервером. Известно, что сервер работает по протоколу UDP и его порт назначения равен 8229. Структура UDP-дейтаграммы представлена ниже:

2 байта	2 байта	2 байта	2 байта	...
UDP-порт отправителя	UDP-порт получателя	Длина UDP-дейтаграммы	Контрольная сумма	Данные

Длина UDP-дейтаграммы включает в себя размер заголовка и размер данных в байтах.

Дамп трафика:

```
0B D7 20 25 00 16 1D DC 47 45 54 20 43 4F 4D 4D 41 4E 44 3A 20 25
20 25 0B D7 00 12 69 AF 53 45 54 20 43 4F 4F 52 44 3A 20 25 0B D7
00 12 25 C0 20 25 28 33 34 2C 35 34 29 00 0B D7 20 25 00 14 2C 8F
43 4F 4D 4D 41 4E 44 20 2D 20 4F 4B
```

Определите, какие данные *сервер отправил клиенту*.

Решение

Из условия задачи известно, что взаимодействие происходит между клиентом и сервером. Для адресации используется поле UDP-порт отправителя и UDP-порт получателя (на каждый из них выделяется по 2 байта). При этом, если порт сервера равен 8229, то в дейтаграммах, отправленных клиентом, поле «UDP-порт получателя» будет равно 8229. В дейтаграммах, отправленных сервером, поле «UDP-порт отправителя» будет равно 8229.

Число $8229_{10} = 2025_{16}$.

Анализируя трафик, можно заметить, что 3-й и 4-й байты равны «2025» – это поле «UDP-порт получателя». Значит, первая дейтаграмма отправлена клиентом серверу. Необходимо найти размер этой дейтаграммы (5-6 байты):

$$0016_{16} = 22_{10}.$$

Следующая дейтаграмма начинается с 23-го байта:

```
2025 0BD7 0012 69AF ...
```

Как видно, 1-2 байты дейтаграммы равны 20 25 – это поле «UDP-порт отправителя». Это означает, что данная дейтаграмма была отправлена сервером клиенту. Размер данной дейтаграммы: $0012_{16} = 18_{10}$.

Вся дейтаграмма выглядит следующим образом:

2025 0BD7 0012 69AF 53 45 54 20 43 4F 4F 52 44 3A
 Поле данных (10 байт): 53 45 54 20 43 4F 4F 52 44 3A.

В соответствии с ASCII-таблицей, каждому байту соответствует символ. Получаем следующую строку:

SET COORD:

Анализируя дальнейший дамп трафика, можно заметить, что следующая дейтаграмма также начинается с байтов 2025, это означает, что следующая дейтаграмма также отправлена сервером клиенту:

2025 0BD7 0012 25C0 20 25 28 33 34 2C 35 34 29 00

Размер дейтаграммы равен $0012_{16} = 18_{10}$.

Поле данных (10 байт): 20 25 28 33 34 2C 35 34 29 00

В соответствии с ASCII-таблицей, каждому байту соответствует символ. Получаем следующую строку:

_(34,54)

Следующая дейтаграмма:

0BD7 2025 0014 2C8F 43 4F 4D 4D 41 4E 44 20 2D 20 4F 4B

Поле «UDP-порт назначения» равно 2025 – эта дейтаграмма отправлена клиентом серверу. Размер дейтаграммы $0014_{16} = 20_{10}$.

Больше дейтаграмм в дампе нет.

Таким образом, от сервера клиенту было отправлено две дейтаграммы со следующим содержимым:

SET COORD:_(34,54)

Ответ: SET COORD:_(34,54).

Задача 3. Шифрование

В системе используется следующий алгоритм шифрования текстовых сообщений: значение каждого следующего байта циклически сдвигается побитно влево N раз, где N – значение предыдущего зашифрованного байта. Первый байт сообщения не шифруется.

Расшифруйте предоставленный зашифрованный фрагмент текста:

53 2B 73 23 01 C2 D5 8E 1A 80 72 95 2E 5D AC 37 3A

Решение

Приведенный в задании алгоритм подразумевает, что первый байт не шифруется – $53_{16} = 83_{10}$, что соответствует символу 'S'.

Следующий символ был зашифрован путем циклического сдвига влево 83 раза. Стоит отметить, что если значение байта циклически сдвинуть влево 8 раз, получится исходное значение байта. Таким образом, сдвиг влево 83 раза равен сдвигу влево 3 раза ($83 \% 8 = 3$, где $\%$ – остаток от деления).

Для того, чтобы получить исходное значение байта, сдвинутого влево 3 раза, необходимо его еще сдвинуть влево $8 - 3 = 5$ раз (см. таблицу).

Сдвиг	Значение байта в 2-ом формате	Значение байта в 16-ом формате

0	00101011	2В
1	01010110	56
2	10101100	АС
3	01011001	59
4	10110010	В2
5	01100101	65

Сверившись с ASCII-таблицей, коду 65_{16} соответствует символ 'e'.

Следующий байт т был сдвинут влево $2B_{16} = 43_{10}$ раз, что эквивалентно сдвигу влево 3 раза ($43 \% 8 = 3$).

Чтобы восстановить значение байта, необходимо сдвинуть его влево 5 раз.

Сдвиг	Значение байта в 2-ом формате	Значение байта в 16-ом формате
0	01110011	73
1	11100110	E6
2	11001101	CD
3	10011011	9B
4	00110111	37
5	01101110	6E

Сверившись с ASCII-таблицей, коду $6E_{16}$ соответствует символ 'n'.

Для ускорения процесса расшифрования можно написать программу. Листинги программы представлены ниже.

Листинг программы на языке Python.

```
# Функция циклического сдвига влево
# ПАРАМЕТРЫ:
#   s - число (байт)
#   n - сколько раз сдвигать число s
# РЕЗУЛЬТАТ:
#   res - циклически сдвинутое влево число s n раз
def shift(s, n):
    res = int(s)
    for i in range(0, n):
        t = res
        # сдвиг влево на 1 бит
        res = (t << 1) & 0xFF
        # на место младшего бита записываем старший,
        # который был до сдвига
        res = res | (t >> 7)
    return res

# Функция расшифрования массива чисел
# ПАРАМЕТР:
#   smass - массив чисел
# ВОЗВРАЩАЕТ:
#   расшифрованную строку
def decipher(smass):
    # Результирующий текст
    text = ""
    # Копируем первый символ без изменений
```

```

text += chr(smass[0])
# Цикл по оставшимся числам
for i in range(1, len(smass)):
    # Сдвигаем очередной символ столько раз,
    # чтобы суммарный сдвиг с учетом шифрования
    # был кратен 8
    n = 8 - (smass[i-1] % 8)
    s = shift(smass[i], n)
    # s преобразуем в символ
    # и добавляем в конец строки результата
    text += chr(s)
return text

mass = [0x53, 0x2B, 0x73, 0x23, 0x01, 0xC2, 0xD5, 0x8E, 0x1A, 0x80,
0x72, 0x95, 0x2E, 0x5D, 0xAC, 0x37, 0x3A]
text = decipher(mass)
print(text)

```

Листинг программы на языке C.

```

// Функция циклического сдвига влево
// ПАРАМЕТРЫ:
// s - число (байт)
// n - сколько раз сдвигать число s
// РЕЗУЛЬТАТ:
// res - циклически сдвинутое влево число s n раз
char shift(unsigned char a, int n)
{
    char res = a;
    for (int i = 0; i < n; i++)
    {
        res = a << 1;
        res |= a >> 7;
        a = res;
    }
    return res;
}

// Функция расшифрования массива чисел
// ПАРАМЕТР:
// smass - массив чисел
// len - размер массива
// ВОЗВРАЩАЕТ:
// расшифрованную строку
char* decipher(unsigned char* smass, int len)
{
    int n;
    // выделение памяти
    char* text = new char[len+1];
    // первый элемент без изменений
    text[0] = smass[0];
    // цикл по остальным числам из smass
    for (int i = 1; i < len; i++)
    {
        // Сдвигаем очередной символ столько раз,
        // чтобы суммарный сдвиг с учетом шифрования
        // был кратен 8
        n = 8 - (smass[i-1] % 8);
        text[i] = shift(smass[i], n);
    }
}

```

```

// добавляем к text ноль-символ конца строки
text[len] = '\0';
return text;
}

```

При подаче на вход программы последовательности из задания, получится следующий результат:

Send auth request

Данная фраза и есть ответ.

Ответ: Send auth request.

Задача 4. Стеганография

Аналитику удалось обнаружить папку с графическими изображениями, в которой скрыто осмысленное кодовое слово. Помогите определить кодовое слово, если известно, что для его сокрытия содержимое файлов не менялось.

К задаче прилагается: [папка с файлами](#) (1.jpg, 2.jpg, ... , 32.jpg) (см. рисунок).

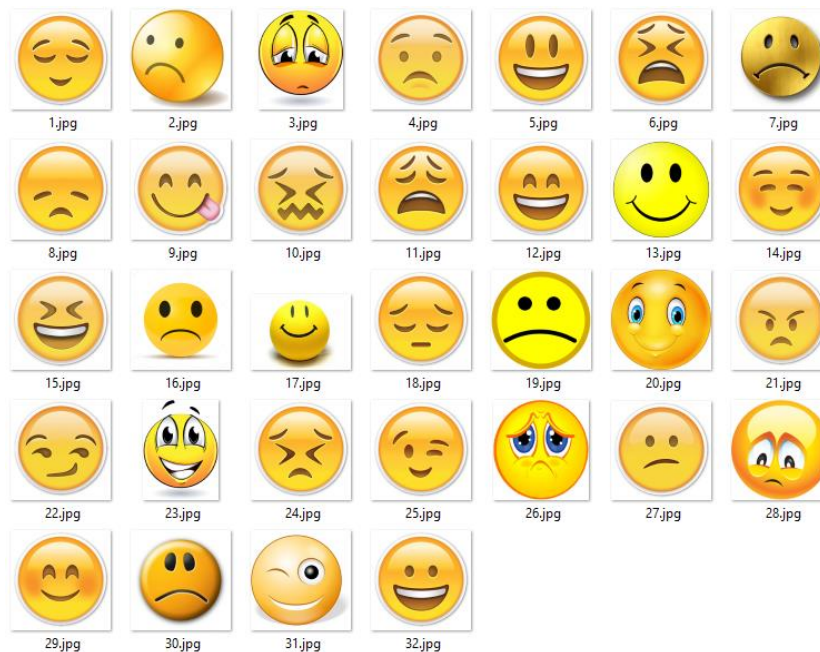


Рисунок. Прилагаемые изображения

Решение

Содержимое изображений не изменялось. Следовательно, содержимое файлов детально изучать нецелесообразно.

Все изображения представляют собой различные смайлики. Внимательно просмотрев все изображения, можно заметить, что изображенные смайлики либо грустные, либо веселые. Предположим, что каждая из картинок – бит информации, тогда можно соотнести одну категорию с единицей, а вторую – с нулем.

Всего 32 изображения – получается комбинация из 32-х нулей и единиц (32 бита).

Зная, что в байте 8 бит, а также тот факт, что каждому символу в ASCII-таблице соответствует 1 байт, можно предположить, что в картинках скрыто 4-символьное сообщение.

Предположим, что грустному смайлику соответствует «0», а веселому – «1». Получается следующая комбинация:

```
1000 1000
1001 1110
1001 0110
1000 1011
```

Переведем в 16-й формат: 88 9E 96 8B.

Согласно ASCII-таблице, эти коды не соответствуют ни буквам, ни цифрам. Такое сообщение не несет смысл. Значит, исходное предположение было неверным.

Предположим теперь наоборот: грустному смайлику соответствует «1», а веселому – «0». Получается следующая комбинация:

```
0111 0111
0110 0001
0110 1001
0111 0100
```

Переведем в 16-й формат: 77 61 69 74.

Согласно ASCII-таблице, эти коды соответствуют следующим символам: *wait* – что является осмысленным словом и ответом на задачу.

Ответ: wait.

Задача 5. Web-сайт

На компьютере нарушителя было найдено множество архивов, каждый из которых защищён некоторым паролем. Проведённый анализ показал, что только *три* архива содержат полезную информацию: *один* содержит «скрытое сообщение», а *два других* – фрагменты пароля к этому архиву. Остальные архивы пустые и не содержат важной информации.

Всю необходимую информацию для доступа к архивам и секретному сообщению нарушитель спрятал на Web-странице. Определите «скрытое сообщение».

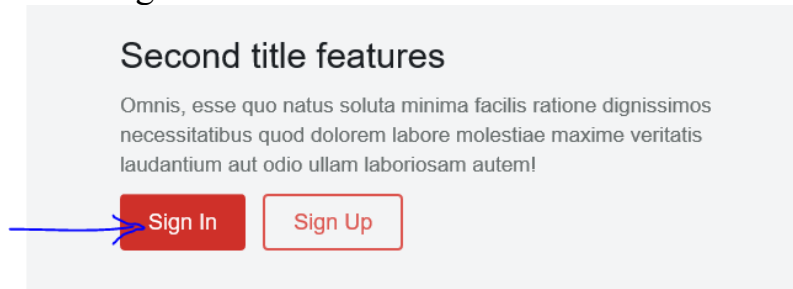
К задаче прилагается:

- 1) [папка с содержимым web-страницы](#),
- 2) [папка с архивами](#) (*архив1.rar, архив2.rar, ... , архив30.rar*).

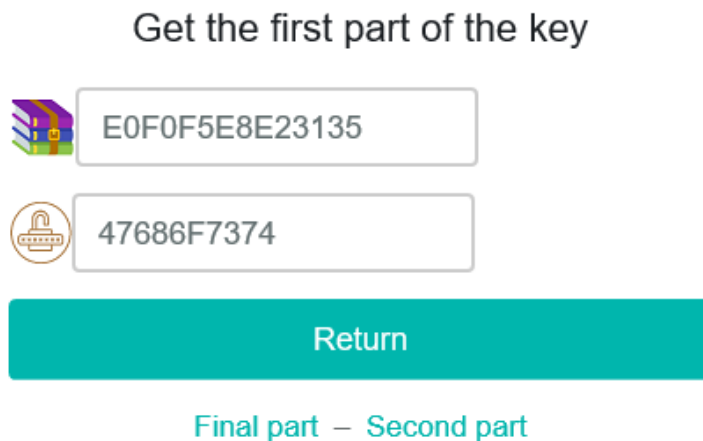
Решение

Обращаем внимание, что среди архивов ровно 3 имеют отличные от остальных дату и время модификации: *архив8.rar, архив15.rar, архив21.rar*.

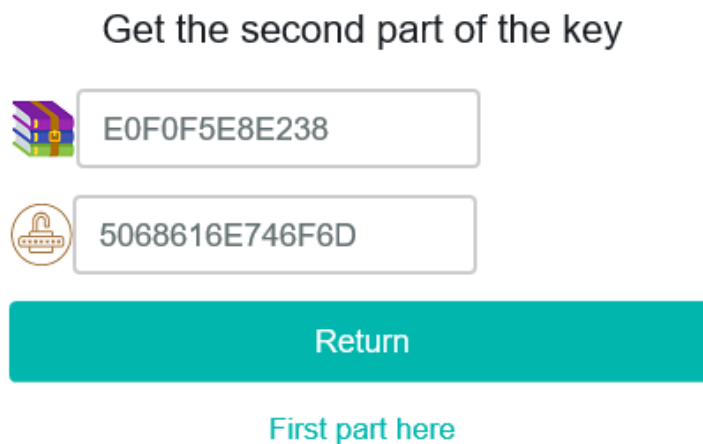
При анализе содержимого web-страницы обращаем внимание, что при переходе по ссылке «Sign-In»



открывается форма, в которой поля, обычно предназначенные для ввода логина и пароля, заполнены какой-то полезной информацией.



Перейдя по ссылкам «Second Part», получаем похожую картину:



Переход по ссылке «Final part» даёт информацию только об имени архива:

Final part

Using the received key, find the password to the archive that contains the answer.



E0F0F5E8E23231

Return

Исходя из условия задачи, можно предположить, что первые две части относятся к архивам, содержащим фрагменты пароля к финальному архиву с искомым «скрытым сообщением».

Проанализируем полученную информацию.

На первой форме (SignIn) содержится 2 сообщения:

- 0xE0F0F5E8E23135 (E0 F0 F5 E8 E2 31 35), что по ASCII-таблице соответствует строке «*архив15*»,
- в поле «пароль» указан пароль 0x47686F7374 (47 68 6F 73 74), то соответствует строке «*Ghost*».

Можно предположить, что к файлу архив15.rar паролем является Ghost (с учетом регистра). В архиве содержится файл с изображением первой части пароля от архива со «скрытым сообщением»: *World*.

На второй форме (Second Part) так же содержатся 2 сообщения:

- 0xE0F0F5E8E238 = «*архив8*»,
- 0x5068616E746F6D = «*Phantom*», что является паролем к файлу архив8.rar.

В архиве8 содержится файл с изображением второй части пароля от архива со «скрытым сообщением»: *Map*.

На третьей форме (Final Part) содержится только одно сообщение: 0xE0F0F5E8E23231 = «*архив21*». Можно сделать вывод, что ответ содержится в файле архив21.rar, пароль к которому «складывается из двух частей» – *WorldMap*.

В архиве содержится файл, внутри которого можно найти «скрытое сообщение» – «*Sense of security*», которое и является ответом.

Ответ: Sense of security

11 КЛАСС

Условия задач отборочного этапа

Задача 1. Удалённый файл

Для организации файловой системы используется структура хранения данных, включающая в себя три основных секции:

Таблица 1	Таблица 2	Данные
-----------	-----------	--------

Все файлы имеют размер кратный *12 байтам*. Каждый файл разбивается на блоки размером по 12 байт, информация о которых хранится в «Таблице 2», имеющей следующую структуру и восстановленный фрагмент заполнения:

Номер записи	Адрес строки блока данных	Номер следующей строки
1	1	2
2	3	0
3	2	0
4	8	9
5	4	0
6	5	5
7	7	20
8	6	0
9	9	7
10	20	11
11	13	0
12	14	13
13	15	16
14	18	15
15	19	0
16	12	0
17	27	18
18	17	0
19	22	0
20	21	0

где «Адрес блока данных» – номер блока данных в секции «Данные»;

«Номер следующей строки» – номер записи в «Таблице 2», содержащей информацию о следующем блоке данных файла, либо 0, если достигнут конец файла.

Для определения начала каждого файла используется «Таблица 1», имеющая следующую структуру и заполнение:

Имя файла	Номер строки в Таблице 2
1.txt	1
2	3
0	10
б.txt	3
virus.exe	17
a.out	0
kernel-5	4
00	8
Deleted	6

где «Имя файла» – полный путь и название файла в системе;

«Номер строки в Таблице 2» – номер строки в «Таблице 2», содержащей информацию о первом блоке данных файла.

Размер «Таблицы 1» фиксирован, все неиспользуемые строки заполнены значением 0. Для повышения скорости работы с файловой системой при удалении файла обнуляется (записывается значение 0) только соответствующая ему строка «Таблицы 1».

Фрагмент блока данных: номер строки, пробел, данные.

```

1 014025020024
2 022016028014
3 012013028011
4 008010013017
5 014028040028
6 007024014017
7 028008026028
8 013025021012
9 028008026028
10 014012013028
11 026024002017
12 014028013024
13 016012014028
14 021012025002
15 025021014013
16 002017028008
17 012013028020
18 013018043028
19 025028038039
20 032007024014
21 028020013018
22 028024017028
23 008007014025
24 024022013009
25 027028014012
26 024017028038
27 039032007024
28 014028020013
29 018028024017
30 028003017013
31 009028014008

```

32 028013022021
 33 002018004014
 34 004013025014
 35 013009011018
 36 028025028021

Известно, что был удалён 1 файл размером 36 байт (36 символов). Укажите название и содержимое удалённого файла и приведите их в качестве строки ответа без пробелов в формате «Имя Содержимое» (например, «test1 12312320302201602801432»).

Ответ: **«a.out 021012025002025021014013014028013024»**
(Блоки 14 15 12 - потерянная цепочка в три блока).

Задача 2. Гамма

Петр решил зашифровать текстовое сообщение с использованием однобайтового ключа путём выполнения *поразрядной операции «Исключающее ИЛИ»*. После первой операции ключ сдвигается на 1 бит относительно текста, и заново выполняется поразрядная операция «Исключающее ИЛИ», затем опять сдвигается еще на 1 бит и ещё раз применяется и т.д. до тех пор, пока последний бит ключа не дойдет до последнего бита текста. Таким образом, удалось зашифровать всё сообщение целиком и был получен следующий набор байт.

b1 90 90 91 9a df 9c 9e 91 df 8d 9a 9e 9b

Определите исходное сообщение, если известно, что оно может содержать только латинские буквы, цифры и пробелы.

Ответ: **noone can read.**

Задача 3. Ключи

Установлено, что передача информации о ключах шифрования между абонентами осуществлялась по каналу связи, позволяющему последовательно передавать несколько ключей шифрования произвольной длины. Длина каждого ключа кратна байту. В ходе осуществления очередного сеанса связи было отправлено 3 ключа. Промежуточным сетевым оборудованием была зафиксирована передача следующей последовательности байт:

FF 26 24 4C 4E FF 53 44 45 21 4C 56 4D FF FE 54 53 4D 44 4B 45 57
 FE 4D 40 FE 24 27 FF 23 24 50 21 FF 51 4D 53 FE

Дополнительно известно, что:

а) Байты «FF» и «FE» служебные (используются для синхронизации оборудования и «изымаются» на приёмной стороне);

б) Для возможности однозначного определения длин всех ключей шифрования абонентам на приёмной стороне дополнительно (по некоторому другому каналу связи) сообщили значение, равное произведению их длин, и тот факт, что ключ наибольшей длины содержит последовательность байтов 0x242723.

Определите размер всех трех ключей.

Варианты:

а) (8,7,2)

б) (9,4,4)

в) (8,6,3)

г) (9,5,3)

Ответ: в) (8,6,3).

Задача 4. Шифр

Сотруднику для анализа был предоставлен перехваченный фрагмент зашифрованного текста.

```
048041027035020021030041016019041046048042042026047041046020013013
021040048040016032026041016019041020045040026017041046048042042026
047048024026042016048017044041048042013020019040041048042042041027
035020021030019041040037048040041020017026041013026026040019041016
017041046035022030040020027035048030037022041048035026041048024026
042016048017044041019016017046026041040037026041046020013013021040
048040016032026041030035020030026035040022041016019041023037048040
041013048033026019041040037026013044041048041027035020021030041016
019041046048042042026047041046035022030040020027035048030037022044
041016017040026035026019040016017027044041037026017046026036041048
017022041019026040041023016040037041030035020030026035040016026019
041015036041049036041038041048017047041012041048024020032026041016
019041046048042042026047041048041027035020021030044041023037016042
019040041048041019026040041023016040037041030035020030026035040016
026019041015036041049036041038036041012041048017047041025041016019
041046048042042026047041048017041048024026042016048017041027035020
021030044
```

Известно, что:

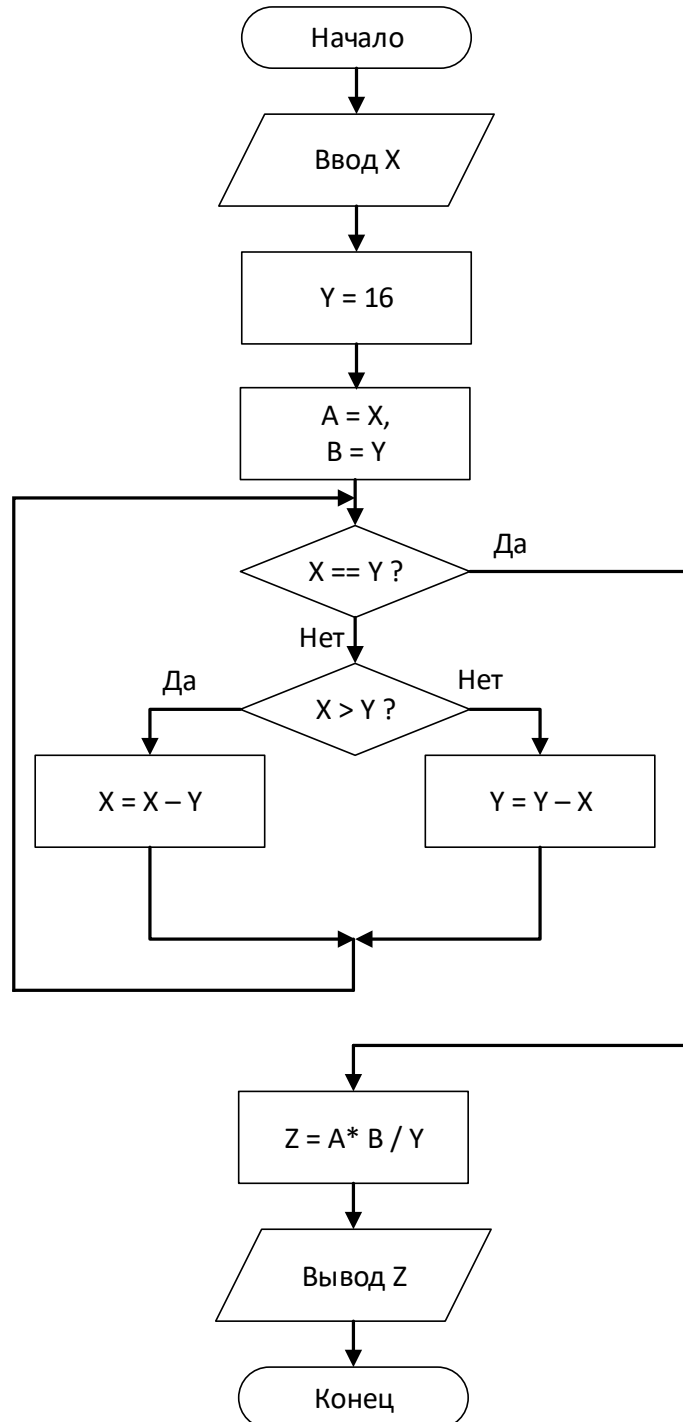
- текст содержит латинские символы нижнего регистра, цифры, пробелы и знаки препинания “.” и “,”;
- некоторые предложения исходного текста начинаются с фразы «*a group is called*» (все символы в нижнем регистре);
- каждый символ исходного текста закодирован тремя цифрами;
- точка имеет код «044», а пробел – «041».

Определите, как будет записано слово «*grade*» с использованием представленного шифра.

Ответ: 027035048047026.

Задача 5. Блок-схема

На рисунке представлена блок-схема алгоритма обработки целого числа X , вводимого пользователем.



Что выведет программа, если на вход подать 66?

Ответ: 528.

Условия задач заключительного этапа

Задача 1. Парольная комбинация

Для входа в систему используется пароль, состоящий из трёх шестизначных чисел, расположенных следующим образом:

xxxxxx-xxxxxx-xxxxxx

Известно, что пароль состоит из 3-х *неповторяющихся простых чисел* в диапазоне от 100 000 до 100 300. При этом *последняя цифра первого числа равна первой цифре второго числа, а последняя цифра второго числа равна первой цифре третьего числа.*

Пример:

xxxxx3-3xxxx7-7xxxxx

Задержка между попытками входа в систему равна 1 секунде. За какое максимальное количество времени (в секундах) можно гарантированно получить пароль, если на ввод пароля время не тратится?

Решение

Для начала необходимо найти все простые числа в диапазоне от 100 000 до 100 300. *Простым* является такое число, которое делится только на себя и на 1. Сделать это можно перебором, проверяя, не делится ли проверяемое число на какое-либо от 2-х до самого числа (либо половины числа).

Для ускорения напишем программу, которая выводит на экран все простые числа в указанном диапазоне.

Листинг программы на языке Python.

```
# Функция, определяющая, является ли число простым
# ПАРАМЕТР:
# n - проверяемое число
# ВОЗВРАЩАЕТ:
# True - число n простое
# False - число n непростое
def isprime(n):
    if n == 1:
        return True
    for d in range(2, n//2):
        if n % d == 0:
            return False
    return True

# Цикл перебора чисел от 100 000 до 100 300
# Для каждого числа вызывается функция isprime()
# Если функция вернула True - число добавляется
# в массив list_input
# Счетчик Count считает количество простых чисел
Count = 0
list_input = []
for x in range(100000,100300):
```

```

    if isprime(x) == True:
        list_input.append(x)
        Count += 1
print(list_input)
print("Total:", Count)

```

Листинг программы на языке C.

```

// Функция, определяющая, является ли число простым
// ПАРАМЕТР:
// n - проверяемое число
// ВОЗВРАЩАЕТ:
// true - число n простое
// false - число n непростое
bool isprime(int n)
{
    if (n == 1)
        return true;
    for (int d = 2; d < n / 2; d++)
        if (n % d == 0)
            return false;
    return true;
}

int main()
{
    // Цикл перебора чисел от 100000 до 100300
    // Для каждого числа вызывается функция isprime()
    // Если функция вернула true - число добавляется
    // в массив
    // Счетчик Count считает количество простых чисел
    int Count = 0;
    int list_input[100] = { 0 };
    // Формирование массива простых чисел
    for (int x = 100000; x < 100300; x++)
    {
        if (isprime(x) == true)
        {
            list_input[Count] = x;
            Count += 1;
        }
    }
    // Вывод на экран
    for (int i = 0; i < Count; i++)
    {
        printf("%d ", list_input[i]);
    }
    printf("\nTotal: %d\n", Count);
    return 0;
}

```

Результат работы программы:

```

100003 100019 100043 100049 100057 100069 100103 100109 100129
100151 100153 100169 100183 100189 100193 100207 100213 100237 100267
100271 100279 100291 100297

```

Total: 23

Всего 23 простых чисел в диапазоне от 11 до 99. Теперь необходимо выделить все возможные комбинации, удовлетворяющие условию «*последняя цифра первого числа равна первой цифре второго числа, а последняя цифра второго числа равна первой цифре третьего числа*».

Сделать это можно двумя способами.

Способ 1. Перебор.

В цикле перебираем всевозможные числа и проверяем два условия:

- все числа разные;
- последняя цифра первого числа равна первой цифре второго числа, а последняя цифра второго числа равна первой цифре третьего числа.

Если все условия выполнены, то увеличиваем значение счетчика. В результате счетчик будет содержать количество комбинаций, удовлетворяющих заданию. Поскольку задержка между вводом паролей равна 1 секунде, а после ввода последней комбинации задержки нет, то ответ – количество найденных комбинаций минус 1.

Пример программы представлен в листингах ниже.

Листинг программы на языке Python.

```
# Функция возвращает список всех комбинаций чисел,
# удовлетворяющих заданному условию паролей
# ПАРАМЕТР:
#     list_input - список чисел, из которых будет
#                 формироваться комбинация
# ВОЗВРАЩАЕТ:
#     list_combination - список комбинаций
#                 из 3-х чисел, удовлетворяющих условию
def combination(list_input):
    list_combination = []
    for a in list_input:
        for b in list_input:
            for c in list_input:
                if a != b and b != c and a != c:
                    if str(a)[-1] == str(b)[0] and
                       str(b)[-1] == str(c)[0]:
                        list_combination.append(str(a) +
                                                "-" + str(b) + "-" + str(c))
    return list_combination

# Заполнение списка всеми простыми числами
# в заданном диапазоне
list_input = []
for x in range(100000,100300):
    if isprime(x) == True:
        list_input.append(x)
# Список, содержащий все комбинации,
# удовлетворяющие заданному условию
result_list = combination(list_input)
# Вывод всех комбинаций
for i in result_list:
```



```

    print(i, end=", ")
# Вывод количества комбинаций
print("Total:", len(result_list))

```

Листинг программы на языке С.

```

// Функция выводит на экран все комбинации,
// удовлетворяющие условию
// ПАРАМЕТРЫ:
//     list_input - массив чисел, из которых будут
//                 строится комбинации
//     count - количество чисел в массиве list_input
// ВОЗВРАЩАЕТ:
//     выводит на экран комбинации, удовлетворяющие
//     условию
//     выводит на экран количество комбинаций
void combination(int list_input[], int count)
{
    int last_digit1;
    int first_digit2;
    int last_digit2;
    int first_digit3;
    int total = 0;
    for (int i = 0; i < count; i++)
        for (int j = 0; j < count; j++)
            for (int k = 0; k < count; k++)
                {
                    if ( list_input[i] != list_input[j] &&
                        list_input[j] != list_input[k] &&
                        list_input[i] != list_input[k] )
                        {
                            // последняя цифра числа
                            last_digit1 = list_input[i] % 10;
                            // первая цифра числа
                            first_digit2 = list_input[j]/100000;
                            last_digit2 = list_input[j] % 10;
                            first_digit3 = list_input[k]/100000;
                            if ( last_digit1 == first_digit2 &&
                                last_digit2 == first_digit3 )
                                {
                                    printf("%d-%d-%d\n",
                                        list_input[i],
list_input[j],
                                        list_input[k]);
                                    total++;
                                }
                        }
                }
    // Вывод общего количества комбинаций
    printf("Total: %d\n", total);
}

```

В результате выполнения программы получим следующие данные:

```

100151-100271-100003, 100151-100271-100019,
100151-100271-100043, 100151-100271-100049,
100151-100271-100057, 100151-100271-100069,

```

100151-100271-100103, 100151-100271-100109,
100151-100271-100129, 100151-100271-100153,
100151-100271-100169, 100151-100271-100183,
100151-100271-100189, 100151-100271-100193,
100151-100271-100207, 100151-100271-100213,
100151-100271-100237, 100151-100271-100267,
100151-100271-100279, 100151-100271-100291,
100151-100271-100297, 100151-100291-100003,
100151-100291-100019, 100151-100291-100043,
100151-100291-100049, 100151-100291-100057,
100151-100291-100069, 100151-100291-100103,
100151-100291-100109, 100151-100291-100129,
100151-100291-100153, 100151-100291-100169,
100151-100291-100183, 100151-100291-100189,
100151-100291-100193, 100151-100291-100207,
100151-100291-100213, 100151-100291-100237,
100151-100291-100267, 100151-100291-100271,
100151-100291-100279, 100151-100291-100297,
100271-100151-100003, 100271-100151-100019,
100271-100151-100043, 100271-100151-100049,
100271-100151-100057, 100271-100151-100069,
100271-100151-100103, 100271-100151-100109,
100271-100151-100129, 100271-100151-100153,
100271-100151-100169, 100271-100151-100183,
100271-100151-100189, 100271-100151-100193,
100271-100151-100207, 100271-100151-100213,
100271-100151-100237, 100271-100151-100267,
100271-100151-100279, 100271-100151-100291,
100271-100151-100297, 100271-100291-100003,
100271-100291-100019, 100271-100291-100043,
100271-100291-100049, 100271-100291-100057,
100271-100291-100069, 100271-100291-100103,
100271-100291-100109, 100271-100291-100129,
100271-100291-100151, 100271-100291-100153,
100271-100291-100169, 100271-100291-100183,
100271-100291-100189, 100271-100291-100193,
100271-100291-100207, 100271-100291-100213,
100271-100291-100237, 100271-100291-100267,
100271-100291-100279, 100271-100291-100297,
100291-100151-100003, 100291-100151-100019,
100291-100151-100043, 100291-100151-100049,
100291-100151-100057, 100291-100151-100069,
100291-100151-100103, 100291-100151-100109,
100291-100151-100129, 100291-100151-100153,
100291-100151-100169, 100291-100151-100183,
100291-100151-100189, 100291-100151-100193,
100291-100151-100207, 100291-100151-100213,
100291-100151-100237, 100291-100151-100267,
100291-100151-100271, 100291-100151-100279,
100291-100151-100297, 100291-100271-100003,
100291-100271-100019, 100291-100271-100043,
100291-100271-100049, 100291-100271-100057,
100291-100271-100069, 100291-100271-100103,
100291-100271-100109, 100291-100271-100129,
100291-100271-100151, 100291-100271-100153,

100291-100271-100169, 100291-100271-100183,
 100291-100271-100189, 100291-100271-100193,
 100291-100271-100207, 100291-100271-100213,
 100291-100271-100237, 100291-100271-100267,
 100291-100271-100279, 100291-100271-100297
 Total: 126

Всего таких комбинаций будет 126, следовательно, максимальное затраченное время будет равно

$$(126 - 1) * 1 \text{ сек} = 125 \text{ сек.}$$

Способ 2. Аналитический.

Можно заметить, что все числа из диапазона 100 000–100 300 начинаются на цифру «1». Это означает, что на место первого и второго числа подойдут только те числа, которые заканчиваются на «1». Шаблон комбинаций будет следующий:

$$1xxxx1-1xxxx1-1xxxxx$$

Анализируя полученные простые числа, всего 3 числа подходят на первую и вторую позицию: 100151, 100271, 100291. Учитывая невозможность повторения чисел возможны следующие варианты:

- на 1-ю позицию возможно 3 варианта,
- на вторую – 2 варианта,
- на третью – оставшиеся простые числа за вычетом использованных на первой и второй позиции (23–2=21).

Общее число комбинаций равно:

$$3 * 2 * 21 = 126.$$

Следовательно, максимальное затраченное время будет равно

$$(126 - 1) * 1 \text{ сек} = 125 \text{ сек.}$$

Ответ: минимальное время, за которое можно гарантированно получить пароль, равно **125 секунд.**

Задача 2. Сетевой трафик

Был получен фрагмент сетевого трафика пользователя при взаимодействии с игровым сервером. Известно, что сервер работает по протоколу UDP и его порт назначения равен 3365.

Структура UDP-дейтаграммы представлена ниже:

2 байта	2 байта	2 байта	2 байта	...
UDP-порт отправителя	UDP-порт получателя	Длина UDP-дейтаграммы	Контрольная сумма	Данные

Длина UDP-дейтаграммы включает в себя размер заголовка и размер данных в байтах.

Весь сетевой трафик шифруется методом «двоичного гаммирования», то есть путём выполнения операции «побитового исключающего ИЛИ» между байтами

UDP-дейтаграммы (включая заголовок) и байтами, полученными циклическим повторением последовательности некоторого ключа.

Дамп трафика:

```
A3 67 AA 90 A7 AD 8F 36 E0 F0 F3 95 F4 F0 E4 E7 E2 E1 87 F6 E8 F1
E2 B5 AA 90 A3 67 A7 BB BB 05 EE F1 E3 E4 E3 B5 A3 67 AA 90 A7 A7
C6 1C E4 FA E3 F0 87 98 87 FA EC B5
```

Определите, что сервер ответил на запрос пользователя, если известно, что для шифрования используется 2-байтовый ключ. В ответе укажите только содержимое поля данных UDP-дейтаграммы.

Решение

Из условия задачи известно, что взаимодействие происходит между клиентом и сервером. Можно сделать предположение, что первая дейтаграмма идет в направлении от клиента серверу. Порт клиента неизвестен, однако известен порт сервера. Следовательно, в первой дейтаграмме поле «UDP-порт получателя» (3-й и 4-й байты) будет равно $3365_{10} = 0D25_{16}$.

Весь сетевой трафик зашифрован методом «двоичного гаммирования». В основе «двоичного гаммирования» лежит операция «Исключающее ИЛИ» (XOR, ^). Таблица истинности операции «Исключающее ИЛИ»:

X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Если T – это открытый текст, K – ключ, а C – зашифрованный текст, то верно следующее:

$$T \wedge K = C$$

$$C \wedge K = T$$

При этом, $T \wedge C = K$.

Если известны открытый текст и зашифрованный текст, то можно определить ключ шифрования.

3-й и 4-й байты первой дейтаграммы равны AA 90 – это зашифрованный текст. В этих байтах хранится значение 0D 25 – это открытый текст. Следовательно, ключ можно получить следующим образом:

$$K = AA\ 90 \wedge 0D\ 25.$$

Распишем данную операция поразрядно.

Двоичный код	Шестнадцатеричный код
10101010 10010000	AA 90
00001101 00100101	0D 25
10100111 10110101	A7 B5

Ключ шифрования – A7 B5.

Далее необходимо определить размер первой дейтаграммы. Поле «длина дейтаграммы» – это 5-й и 6-й байты – A7 AD. Однако это поле зашифровано, но нам известен ключ. Результат расшифрования поля:

Двоичный код	Шестнадцатеричный код
10100111 10101101	A7 AD
10100111 10110101	A7 B5
00000000 00011000	00 18

Длина первой дейтаграммы равна $0018_{16} = 24_{10}$.

Выделим и расшифруем поле данных первой дейтаграммы. Если длина всей дейтаграммы 24 байта, а длина заголовка равна $2(\text{порт отправителя}) + 2(\text{порт получателя}) + 2(\text{длина}) + 2(\text{контрольная сумма}) = 8$ байт,

то длина поля данных равна $24 - 8 = 16$ байт.

Зашифрованные данные:

E0 F0 F3 95 F4 F0 E4 E7 E2 E1 87 F6 E8 F1 E2 B5

Используя ключ, расшифруем эти байты:

E0 F0 F3 95 F4 F0 E4 E7 E2 E1 87 F6 E8 F1 E2 B5
 A7 B5 A7 B5 A7 B5 A7 B5 A7 B5 A7 B5 A7 B5 A7 B5
 47 45 54 20 53 45 43 52 45 54 20 43 4F 44 45 00

Каждый полученный байт представляет собой код символа в ASCII-таблице. В результате получаем текстовое сообщение: «GET SECRET CODE».

Следующая дейтаграмма начинается с 25-го байта:

AA90 A367 A7BB BB05 EE F1 ...

По логике взаимодействия вторая дейтаграмма – это ответ сервера. Значит, в поле «UDP-порт отправителя» должно стоять значение 0D25. В зашифрованном трафике в этом поле стоит значение AA90.

Используя вычисленный ранее ключ, можно расшифровать первые 2 байта дейтаграммы:

$$AA90 \wedge A7B5 = 0D25$$

Предположение подтвердилось – данная дейтаграмма отправлена от сервера клиенту. Осталось определить длину дейтаграммы (5-й и 6-й байты).

В зашифрованном виде поля длины дейтаграммы равно A7BB. Используя ключ A7B5 расшифруем это поле.

Двоичный код	Шестнадцатеричный код
10100111 10111011	A7 BB
10100111 10110101	A7 B5
00000000 00001110	00 0E

Длина второй дейтаграммы равна $000E_{16} = 14_{10}$.

Выделим и расшифруем поле данных второй дейтаграммы. Если длина всей дейтаграммы 14 байт, а длина заголовка равна 8 байт, то длина поля данных равна $14 - 8 = 6$ байт.

Зашифрованные данные: EE F1 E3 E4 E3 B5.

Используя ключ, расшифруем эти байты:

EE	F1	E3	E4	E3	B5
A7	B5	A7	B5	A7	B5
49	44	44	51	44	00

Каждый полученный байт представляет собой код символа в ASCII-таблице. В результате получаем текстовое сообщение: «*IDDQD*».

Рассмотрим третью дейтаграмму:

A367 AA90 A7A7 C61C E4 FA E3 F0 87 98 87 FA EC B5

Анализируя поля «UDP-порт источника» и «UDP-порт назначения», можно сделать вывод, что данная дейтаграмма отправлена от клиента серверу (порт назначения равен AA90, что при расшифровании даст 0D25 – порт сервера).

Определим размер третьей дейтаграммы. Поле «длина дейтаграммы» – это 5-й и 6-й байты – A7 A7. Используя известный ключ, расшифруем это поле.

Двоичный код		Шестнадцатеричный код	
10100111	10100111	A7	A7
10100111	10110101	A7	B5
00000000	00010010	00	12

Длина первой дейтаграммы равна $0012_{16} = 18_{10}$.

Выделим и расшифруем поле данных третьей дейтаграммы. Если длина всей дейтаграммы 18 байт, а длина заголовка 8 байт, то длина поля данных равна $28 - 8 = 10$ байт.

Зашифрованные данные:

E4 FA E3 F0 87 98 87 FA EC B5

Используя ключ, расшифруем эти байты:

E4	FA	E3	F0	87	98	87	FA	EC	B5
A7	B5	A7	B5	A7	B5	A7	B5	A7	B5
43	4F	44	45	20	2D	20	4F	4B	00

Каждый полученный байт представляет собой код символа в ASCII-таблице. В результате получаем текстовое сообщение: «*CODE – OK*».

Ответ: сервер отправил клиенту текстовое сообщение **IDDQD**. Ключ шифрования **A7B5**.

Задача 3. Вирус

Имеется система, представляющая собой файл-серверную архитектуру, состоящую из 1 файл-сервера и 6 ПК. На файл-сервере хранится 30 файлов-приложений (*file1.exe, file2.exe, ..., fileN.exe*). Известно, что один из файлов заражён вредоносным кодом, который после попадания на клиентское устройство выводит его из строя через 1 час. Каждый ПК может копировать с файл-сервера любое количество файлов.

За какое *минимальное количество часов* можно точно определить зараженный файл. Ответ обоснуйте.

Решение

Для определения вредоносного файла необходимо каждый из них пронумеровать следующим образом.

1. Номер каждого файла представим в виде двоичного 6-разрядного разложения:
 file1.exe – 000001 (двоичное представление числа 1)
 file2.exe – 000010 (двоичное представление числа 2)
 file3.exe – 000011 (двоичное представление числа 3)
 ...
 file30.exe – 11110 (двоичное представление числа 30)
 Такое разложение позволяет каждому файлу предоставить уникальную последовательность нулей и единиц.
2. Всего 6 ПК. Распределим их номера: 0,1,2,3,4,5.
3. Каждому ПК соответствует разряд в последовательности представления файла:
 ПК0 – младший разряд (0-й),
 ПК1 – второй справа разряд (1-й),
 ...
 ПК5 – старший разряд (5-й)
4. На каждый ПК копируем только те файлы, в соответствующем разряде которого стоит «1»:
 на ПК0 – file1.exe, file3.exe, file5.exe, ... , file29.exe
 на ПК1 – file2.exe, file3.exe, file6.exe, file7.exe, ...
 на ПК2 – file4.exe, file5.exe, file6.exe, file7.exe, ...
 ...
 на ПК5 – file16.exe, file17.exe, file18.exe, ... , file30.exe.
5. Ровно через 1 час после копирования какие-то ПК выйдут из строя. Их номера определяют номер зараженного файла следующим образом: если ПК_i-й не поврежден, то на *i*-м разряде номера файла ставим «0». Если ПК_i-й поврежден, то на *i*-м разряде номера файла ставим «1».
 В результате получится последовательность, переведя которую в десятичный формат можно однозначно определить зараженный файл.

Ответ: 1 час.

Задача 4. Стеганография

На web-странице содержатся ссылки на скачивание файлов-изображений с указанием контрольных сумм их содержимого. Известно, что в одном из них спрятано секретное текстовое сообщение. Определите это изображение и скрытое в нем секретное текстовое сообщение.

Файлы и контрольные суммы их содержимого (MD5):

d812f179df6b60943dbfd69c4e613aaf	Chrysanthemum.jpg
4e427c78ecb620aecce46bb006d247e5	Desert.jpg
6dbaefbc2e49df3c3cfe7515d5c6cac72	Hydrangeas.jpg
f037c82a48be22696142c59b4eea3298	Jellyfish.jpg
e3614da88d1511dfb8a05dd3ec24999d	Koala.jpg

46b851500907f4ccdfa75c0a29dd8dcf	Lighthouse.jpg
2022b59db2185282fd753f6320e782c4	Penguins.jpg
902b6ea2111efb87b19056655165c72d	Tulips.jpg

К задаче прилагается:

- 1) 8 файлов-изображений (*.jpg),
- 2) программа подсчета контрольной суммы по алгоритму MD5 (fciv.exe),
- 3) файл с описанием формата RAR-архива.

Решение

1. С помощью программы fciv.exe определим MD5-суммы для файлов и сравним их с данными в задаче.
2. Можно заметить, что для файла *Jellyfish.jpg* контрольная сумма не совпадает.
3. Предполагаем, зачем дано описание формата RAR-файла. Возможно, это намек на то, что в файле с изображением содержится скрытый RAR-архив.
4. Из описания RAR-архива понятно, что любой архив начинается с метки 0x5052. Используя HEX-редактор анализируем побайтовое содержимое файла *Jellyfish.jpg* и осуществляем поиск метки 0x5052. Метка найдена.
5. Копируем содержимое файла, начиная с метки, в новый файл, который назовем *архив.rar*

ИЛИ

можно переименовать файл *Jellyfish.jpg* в *архив.rar*.

6. В архиве содержится файл *Сообщение.docx*, открыв который, читаем сообщение – «Система обнаружения вторжений (IDS)».

Ответ: Система обнаружения вторжений (IDS)
(файл – *Jellyfish.jpg*).

Задача 5. Разработка процессора

При разработке процессора инженеры решили реализовать поддержку виртуальной памяти, когда программа оперирует виртуальным адресом, а этот адрес пересчитывается процессором в физический адрес ячейки – байта в оперативной памяти. Инженеры решили использовать структуру виртуального адреса, состоящего из трёх частей:

Индекс-1	Индекс-2	Индекс-3
----------	----------	----------

Вся память делится на непересекающиеся блоки одинаковой длины – страницы.

Индекс-1 содержит номер таблицы, в которой содержится информация о страницах.

Индекс-2 содержит номер страницы из таблицы с номером *Индекс-1*.

Индекс-3 определяет номер байта на странице с номером *Индекс-2* из таблицы с номером *Индекс-1* (см. рисунок).

Каждая таблица занимает ровно 1 страницу, размер таблиц фиксирован. Каждая запись в таблице имеет фиксированный размер и состоит из следующих полей:

- физический адрес страницы в оперативной памяти;
- заполнитель до кратности байту.

Общий диапазон адресов физической памяти, который должен адресоваться процессором – 1 Гб, размер страницы – 1 Кб.

Укажите минимально необходимые размерности полей Индекс-1, Индекс-2, Индекс-3, а также общую размерность виртуального адреса для возможности адресации объёма оперативной памяти в 1 Гб?

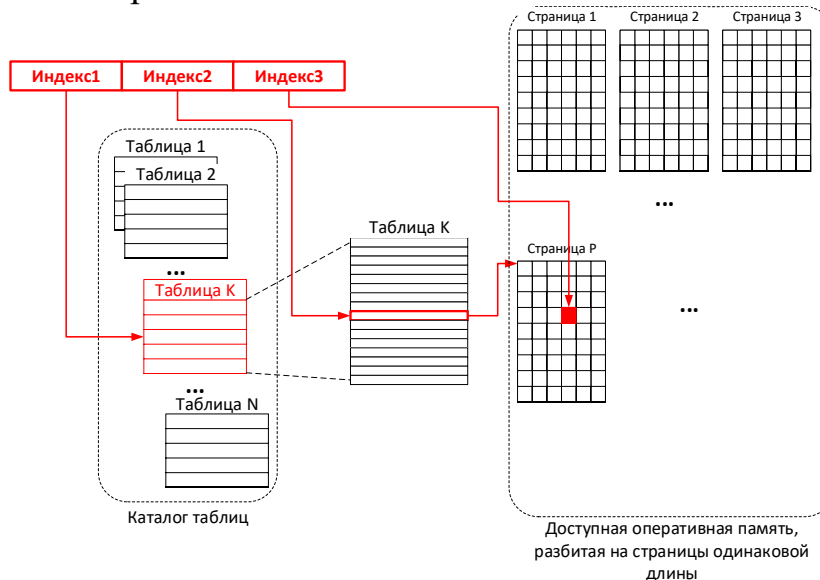


Рисунок. Организация виртуальной памяти

Решение

Вся память разбита на страницы. Страница занимает 1 Кб = 1024 байта. Значит, для адресации байта внутри страницы требуется 10 бит ($2^{10} = 1024$).

Индекс-3 = 10 бит.

В таблице одна запись занимает физический адрес и заполнитель. Для адресации 1 Гб адресного пространства (1 073 741 824 байт) нужен диапазон значений физического адреса $[0:2^{30}-1]$, т.е. 30 двоичных разрядов при линейном строении адреса. Соответственно, одна запись в таблице занимает $30(\text{адрес}) + 2(\text{заполнитель}) = 32$ двоичных разряда (4 байта). Размер таблицы – 1 Кб. Всего в таблице может быть $1024 / 4 = 256$ записей. Для адресации 256 записей необходимо 8 двоичных разрядов ($2^8 = 256$).

Индекс-2 = 8 бит.

В 1 Гб помещается $1\,073\,741\,824 / 1024 = 1\,048\,576$ страниц размером 1 Кб. Учитывая, что в одной таблице 256 записей, всего таблиц должно быть $1\,048\,576 / 256 = 4096$. Для адресации 4096 таблиц понадобится 12 двоичных разрядов ($2^{12} = 4096$).

Индекс-1 = 12 бит.

Итого, общая размерность виртуального адреса будет

$$\text{Индекс-1} + \text{Индекс-2} + \text{Индекс-3} = 12 + 8 + 10 = 30 \text{ бит.}$$

Ответ: **Индекс-1 – 12 бит,**
 Индекс-2 – 8 бит,
 Индекс-3 – 10 бит.
 Всего – 30 бит.

**Критерии оценивания заданий
Межрегиональной олимпиады школьников имени И.Я. Верченко по
информатике и компьютерной безопасности (2018/2019 учебный год)**

Каждая задача заключительного этапа олимпиады при проверке работ оценивалась по системе: «-», «-/+», «+/-», «+»:

«-» — отсутствует решение задачи или решено менее половины задачи;

«-/+» — отсутствует логически законченное решение задачи (решено не менее половины задачи);

«+/-» — логически законченное решение с неточностями (ответ может быть неправильным);

«+» — полностью правильное решение задачи и правильный ответ.

По результатам проверки осуществлялся перевод полученных результатов в баллы с учётом сложности задач. Для каждой параллели классов (8, 9, 10 и 11) разработана своя система баллов (см. Таблицы 1, 2) и критерии определения победителей и призеров (см. Таблица 3), принятые на заседании жюри олимпиады.

Таблица 1

Критерии оценивания задач олимпиады для 8-10 классов

	1 задача	2 задача	3 задача	4 задача	5 задача
-	0	0	0	0	0
-/±	1	2	2	2	1
±	2	3	4	3	2
+	3	4	5	4	3

Максимальное количество баллов за работу – 19.

Таблица 2

Критерии оценивания задач олимпиады для 11 класса

	1 задача	2 задача	3 задача	4 задача	5 задача
-	0	0	0	0	0
-/±	1	2	1	1	2
±	2	4	2	2	5
+	3	6	4	4	6

Максимальное количество баллов за работу – 23.

Критерии определения победителей и призеров олимпиады

Возрастная категория	1 место	2 место	3 место
11 класс	19 баллов и более	16-18 баллов	12-15 баллов
10 класс	16 баллов и более	14-15 баллов	10-13 баллов
9 класс	16 баллов и более	14-15 баллов	10-13 баллов
8 класс	14 баллов и более	10-13 баллов	8-9 баллов