МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет автоматики и вычислительной техники
Кафедра автоматики

Двоично-десятичный код

Выполнил:
Студент гр.АА-96
Жижин В.Е.
15 Октября 2021 года

Проверил:
Катасонов Д.Н.
_____
(подпись)

Новосибирск
2021

Введение.

Кодирование.

Под кодированием понимается взаимно однозначное преобразование множества дискретных сообщений в множество сигналов в виде кодовых комбинаций. Каждый код строится по определенным правилам и представляет собой совокупность кодовых комбинаций (условных сигналов или символов).

Постановка задачи.

Задачей является релизация двоично-десятичного кодирования, в котором каждый разряд десятичного числа кодируется 4 битами. Каждый бит имеет собственный вес, сума произведения всех битов на их вес дает закодированную цифру. В задаче дан только один известный параметр - мощность кода. Остальные нужно вычислить в ходе работы программы. Программа должна иметь функционал кодирования и декодирования сообщения и нахождения неизвестных параметров.

Обзорная часть.

Теоритическая информация.

Как было сказано ранее двоично-десятичное кодирование подразумевает под собой кодирование каждого разряда десятично числа при помощи четырех взвешеных битов. M[1] будет составлять $10^{n/4}$, так как каждый разряд кодируется четырмя битами. Обратно же n[2] будет равно $log_{10}(M)$ округленному в большую сторону. Так как данный код не подразумевает нахождения ошибок, то k[3] будет равно n. Опять же в силу того, что у кода нет возможности нахождения ошибок, d[4] будет всегда равно единице.

[1] Мощность кодировки.
[2] Общее количество бит.
[3] Количество полезных бит.
[4] Кодовое расстояние

Реализацния.

Для решения данной задачи будет использоваться язык программирования Си.

Нахождение неизвестных параметров.

main.c

```c
int find_k(double M) {
  return (int)ceil(log10(M)) * 4;
}

int hamming_dist(char num1, char num2) {
  int dist = 0;
  for (int i = 0; i < 4; i++) {
    if (num1 % 2 != num2 % 2) {
      dist++;
    }
    num1 >>= 1;
    num2 >>= 1;
  }
  return dist;
}

int find_d(BCDString s) {
  int d = 4;
  for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
      if (j != i) {
        int dist=hamming_dist(s.allowed[i].nibble, s.allowed[j].nibble);
        if (d > dist) {
          d = dist;
        }
      }
    }
  }
  return d;
}

double find_D(int k) {
  return 1.0 - ((double)k/log2f((double)pow(10, k)));
}
```

Формулы:

$$k = n = log_{10}(M) * 4, \quad D = 1 - \frac{log_{10}(10^k)}{log_2(10^k)}$$

Работа с весами, нахождение разрешенных комбинаций и взаимодействие с ними.

Основные структуры:

main.c

```c
typedef struct {
  char nibble : 4;
  int real;
} BCDChar;

typedef struct {
  BCDChar* string;
  unsigned int length;
  int weights[4];
  BCDChar allowed[10];
} BCDString;
```

BCDChar отвечает за представление одного кода/разряда десятичного числа. Хранит в себе сразу оба представления: в виде кода и в виде десятичного числа.

BCDString отвечает за представление всего числа. Хранит в себе массив **BCDChar**, количество элементов в массиве, весы битов и массив разрешенных комбинаций [5].

Ввод весов пользователем:

main.c

```c
void parse_weights(int* weights) {

  for (int i = 0; i < 4; i++) {
    printf("Enter %d weight: ", i+1);
    scanf("%d", weights+i);
  }

}
```

Взвешивание числа:

main.c

```c
int weight(int* weights, int num) {
  unsigned int real = 0;
  for (int i = 3; i >= 0; i--) {
    unsigned int temp = num >> i;
    if (temp & 1)
      real += weights[3-i];
  }
  return real;
}
```

[5] В моей реализации все же существуют запрещенные комбинации, чтобы привнести хоть какую-то детекцию ошибок.

Функция возвращает десятичное число полученное из закодированной формы. Для этого мы берем значение последнего бита, если оно равно единице, прибавляем значение соответствующего веса к результату, после чего сдвигаем число на один разряд вправо и продолжаем так до конца закодированного числа.

Вычисление разрешенных комбинаций:

main.c

```
void get_allowed(int* weights, BCDChar* b) {

  int a[10] = {0};

  int i = 0;

  for (int j = 0; j < 16; j++) {
    int temp = weight(weights, j);
    if (temp < 10 && !a[temp]) {
      BCDChar num = { (char)j, temp };
      b[temp] = num;
      a[temp] = 1;
      i++;
    }
  }
}
```

Функция генерирует массив разрешенных комбинаций. Для этого она перебирает все доступные комбинации, подсчитывает их значение в десятичной форме и, если такое значение уже не было найдено, записывает в массив разрешенных комбинаций.

Проверка разрешена ли комбинация:

main.c

```
int is_in_allowed(BCDChar* b, char a) {

  for (int i = 0; i < 10; i++) {

    if ((a & 15) == (b[i].nibble & 15)) {
      return i;
    }

  }

  return -1;
}
```

В случае если комбинация разрешена, функция вернет неотрицательное число равное значению комбинации в десятичной форме. В случае запрещенной комбинации функция вернет отрицательное число.

Кодирование и декодирование сообщения.

Кодирование:

main.c

```
void encode(BCDString string) {

  printf("Enter number to encode: ");
  int number;
  scanf("%d", &number);
  if (number > pow(10, string.length)-1) {
    printf("Error");
  } else {

    for (int i = string.length-1; i >= 0; i--) {

      int remainder = number % 10;
      number /= 10;
      BCDChar new_bcd;
      new_bcd.nibble = (char)string.allowed[remainder].nibble;
      new_bcd.real = number;
      string.string[i] = new_bcd;

    }

    printf("Encoded number: ");
    for (int i = 0; i < string.length; i++) {
      print_binary((int)string.string[i].nibble);
      printf(" ");
    }

  }
}
```

Функция переводит введенное число в закодированное. После ввода числа пользователем программа проверяет возможно ли закодировать его[6]. Далее берется остаток от деления числа на десять[7] и вычисляется его закодированная форма[8]. Далее число делится на 10 и алгоритм повторяется дальше. После чего в окно терминала выводится результат.

[6]В данной задаче число должно быть меньше чем мощность кодировки.

[7]Для получения последнего разряда числа.

[8]Так как массив разрешенных комбинаций явлется отсортированным по возрастанию, то достаточно просто взять необходимый элемент по индексу.

Декодирование:

main.c

```
void decode(BCDString string) {

  int result = 0;
  printf("Enter %d binary numbers: ", string.length);
  for (int i = 0; i < string.length; i++) {
    char nibble = 0;
    for (int j = 0; j < 4; j++) {
      char input;
      scanf(" %c", &input);
      if (input == '1') {
        nibble += (1 << (3-j));
      }

    }

    int check = is_in_allowed(string.allowed, nibble);
    if ( check >= 0) {
      result += check * pow(10, string.length-i-1);
    } else {
      printf("Unallowed sequence!\nExiting.");
      break;
    }
  }
  printf("Result is %d", result);

}
```

Функция производит декодирование сообщения в десятичное число. Программа посимвольно считывает введенное значение, в случае когда введеный символ является единицой, к значению текущего прибавляется единица, сдвинутая влево на необходимое значение, соответствующее позиции данной единицы. После того, как весь код будет обработан, производится проверка на то, что данная комбинация разрешена, если это не так, то появляется сообщение об ошибке и программа завершается.

Тесты.

Проверка корректности вычисления неизвестных параметров:

Входные данные:

```
M = 1100
```

Результат:

```
n = 16
k = 16
d = 1
D = 0.698970
```

Ожидаемый результат:

```
n = 16
k = 16
d = 1
D = 0.698970
```

Проверка корректности генерации разрешенных комбинаций:

Входные данные:

```
Weights = [8 4 2 1]
```

Результат:

```
Allowed combinations =
  [0 0 0 0] [0 0 0 1] [0 0 1 0] [0 0 1 1]
  [0 1 0 0] [0 1 0 1] [0 1 1 0] [0 1 1 1]
  [1 0 0 0] [1 0 0 1]
```

Ожидаемый результат:

```
Allowed combinations =
  [0 0 0 0] [0 0 0 1] [0 0 1 0] [0 0 1 1]
  [0 1 0 0] [0 1 0 1] [0 1 1 0] [0 1 1 1]
  [1 0 0 0] [1 0 0 1]
```

Кодирование корректного числа:

Входные данные:

```
Input = 123
```

Результат:

```
Output = 0000 0001 0010 0011
```

Ожидаемый результат:

```
Output = 0000 0001 0010 0011
```

Кодирование некорректного числа:

Входные данные:

```
Input = 10001
```

Результат:

```
Output = Error
```

Ожидаемый результат:

```
Output = Error
```

Декодирование корректного числа:

Входные данные:

```
Input = 0000 0001 0010 0011
```

Результат:

```
Output = 123
```

Ожидаемый результат:

```
Output = 123
```

Декодирование некорректного числа:

Входные данные:

```
Input = 1111 0001 0010 0011
```

Результат:

```
Output = Unallowed sequence!
  Exiting.
```

Ожидаемый результат:

```
Output = Error
```