# **E卷-TLV解码[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go ）**

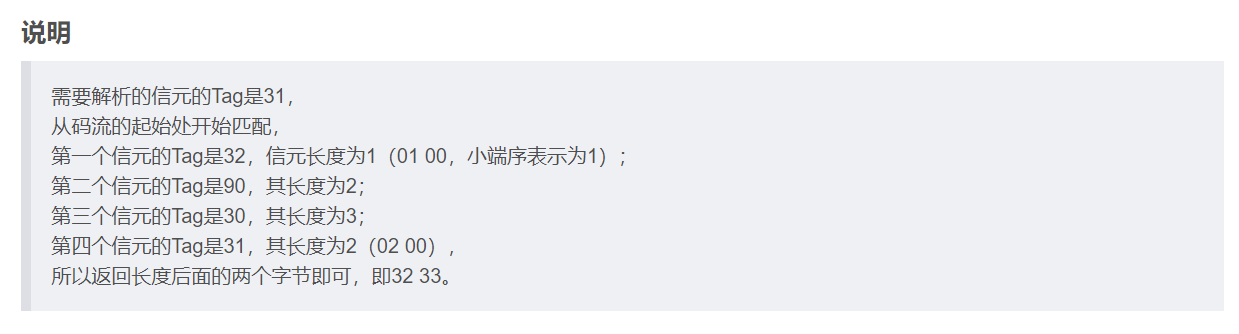


31

32 01 00 AE 90 02 00 01 02 30 03 00 AB 32 31 31 02 00 32 33 33 01 00 CC



32 33





import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String targetTag =

scanner.next(); // 读取第一行输入作为待解码信元的Tag

scanner.nextLine();

String[] arr =

scanner.nextLine().split(" "); // 读取第二行输入并分割成数组

scanner.close();

int i = 0; // 初始化指针i为0，用于遍历码流

StringBuilder result = new

StringBuilder(); // 初始化结果字符串，用于存储待解码信元的Value

// 循环遍历整个码流数组

while (i < arr.length) {

String tag = arr[i]; // 读取当前字节作为Tag

int len1 = Integer.parseInt(arr[i + 1],

16); // 读取下一个字节作为Length的低位字节（小端序）

int len2 = Integer.parseInt(arr[i + 2],

16); // 读取再下一个字节作为Length的高位字节（小端序）

// 计算长度Length，按照小端序对高低位字节进行组合

int length = len2 \* 256 + len1;

i += 3; // 指针i向前移动3位，因为已经读取了Tag和Length字段

// 如果当前Tag等于目标Tag，则需要解码此信元的值

if (tag.equals(targetTag)) {

// 将信元的Value字节逐个添加到结果字符串中

for (int j = 0; j < length; j++) {

result.append(arr[i + j]);

if (j < length - 1) {

result.append(" ");

}

}

}

i += length; // 指针i向前移动Length位，跳过当前信元的Value字节

}

// 输出解码后的值

System.out.println(result.toString());

}

}



n = input() # 读取第一行输入作为待解码信元的Tag

arr = input().split(" ") # 读取第二行输入表示的16进制码流，并以空格分隔存入数组

i = 0 # 初始化指针i为0，用于遍历码流

ans = [] # 初始化结果列表，用于存储待解码信元的Value

# 循环遍历整个码流数组

while i < len(arr):

tag = arr[i] # 读取当前字节作为Tag

Len1 = arr[i + 1] # 读取下一个字节作为Length的低位字节（小端序）

Len2 = arr[i + 2] # 读取再下一个字节作为Length的高位字节（小端序）

# 计算长度Length，按照小端序对高低位字节进行组合

Len = int(Len2, 16) \* 256 + int(Len1, 16)

i += 3 # 指针i向前移动3位，因为已经读取了Tag和Length字段

# 如果当前Tag等于目标Tag，则需要解码此信元的值

if tag == n:

# 将信元的Value字节逐个添加到结果列表中

for j in range(Len):

ans.append(arr[i + j])

i += Len # 指针i向前移动Len位，跳过当前信元的Value字节

# 将结果列表中的字节值拼接成字符串，并以空格分隔后输出

print(" ".join(ans))



#include <iostream>

#include <sstream>

#include <vector>

using namespace std;

int main() {

string targetTag;

getline(cin, targetTag); // 读取第一行输入作为待解码信元的Tag

string line;

getline(cin, line); // 读取第二行输入

istringstream iss(line);

vector<string> arr;

string temp;

// 将输入的16进制码流分割存入数组

while (iss >> temp) {

arr.push\_back(temp);

}

int i = 0; // 初始化指针i为0，用于遍历码流

vector<string>

result; // 初始化结果数组，用于存储待解码信元的Value

// 循环遍历整个码流数组

while (i < arr.size()) {

string tag = arr[i]; // 读取当前字节作为Tag

int len1 = stoi(arr[i + 1], nullptr,

16); // 读取下一个字节作为Length的低位字节（小端序）

int len2 = stoi(arr[i + 2], nullptr,

16); // 读取再下一个字节作为Length的高位字节（小端序）

// 计算长度Length，按照小端序对高低位字节进行组合

int length = len2 \* 256 + len1;

i += 3; // 指针i向前移动3位，因为已经读取了Tag和Length字段

// 如果当前Tag等于目标Tag，则需要解码此信元的值

if (tag == targetTag) {

// 将信元的Value字节逐个添加到结果数组中

for (int j = 0; j < length; j++) {

result.push\_back(arr[i + j]);

}

}

i += length; // 指针i向前移动Length位，跳过当前信元的Value字节

}

// 输出解码后的值

for (size\_t j = 0; j < result.size(); j++) {

if (j > 0) cout << " ";

cout << result[j];

}

cout << endl;

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main() {

char targetTag[3];

scanf("%2s", targetTag); // 读取第一行输入作为待解码信元的Tag

getchar(); // 处理换行符

char line[50000 \* 3];

fgets(line, sizeof(line), stdin); // 读取第二行输入

char arr[50000][3];

int arrSize = 0;

char\* token = strtok(line, " ");

// 将输入的16进制码流分割存入数组

while (token != NULL) {

strcpy(arr[arrSize++], token);

token = strtok(NULL, " ");

}

int i = 0; // 初始化指针i为0，用于遍历码流

char result[50000 \* 3] =

""; // 初始化结果字符串，用于存储待解码信元的Value

// 循环遍历整个码流数组

while (i < arrSize) {

char\* tag = arr[i]; // 读取当前字节作为Tag

int len1 = strtol(arr[i + 1], NULL,

16); // 读取下一个字节作为Length的低位字节（小端序）

int len2 = strtol(arr[i + 2], NULL,

16); // 读取再下一个字节作为Length的高位字节（小端序）

// 计算长度Length，按照小端序对高低位字节进行组合

int length = len2 \* 256 + len1;

i += 3; // 指针i向前移动3位，因为已经读取了Tag和Length字段

// 如果当前Tag等于目标Tag，则需要解码此信元的值

if (strcmp(tag, targetTag) == 0) {

// 将信元的Value字节逐个添加到结果字符串中

for (int j = 0; j < length; j++) {

strcat(result, arr[i + j]);

if (j < length - 1) {

strcat(result, " ");

}

}

}

i += length; // 指针i向前移动Length位，跳过当前信元的Value字节

}

// 输出解码后的值

printf("%s\n", result);

return 0;

}



const readline = require('readline');

// 创建接口来读取标准输入

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

let input = [];

rl.on('line', (line) => {

input.push(line); // 读取输入并存储到数组中

if (input.length === 2) {

rl.close();

}

});

rl.on('close', () => {

const targetTag = input[0].trim(); // 读取第一行输入作为待解码信元的Tag

const arr = input[1].trim().split(" "); // 读取第二行输入并分割成数组

let i = 0; // 初始化指针i为0，用于遍历码流

let result = []; // 初始化结果数组，用于存储待解码信元的Value

// 循环遍历整个码流数组

while (i < arr.length) {

const tag = arr[i]; // 读取当前字节作为Tag

const len1 = parseInt(arr[i + 1], 16); // 读取下一个字节作为Length的低位字节（小端序）

const len2 = parseInt(arr[i + 2], 16); // 读取再下一个字节作为Length的高位字节（小端序）

// 计算长度Length，按照小端序对高低位字节进行组合

const length = len2 \* 256 + len1;

i += 3; // 指针i向前移动3位，因为已经读取了Tag和Length字段

// 如果当前Tag等于目标Tag，则需要解码此信元的值

if (tag === targetTag) {

// 将信元的Value字节逐个添加到结果数组中

for (let j = 0; j < length; j++) {

result.push(arr[i + j]);

}

}

i += length; // 指针i向前移动Length位，跳过当前信元的Value字节

}

// 输出解码后的值

console.log(result.join(" "));

});



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

func main() {

scanner := bufio.NewScanner(os.Stdin)

scanner.Scan()

targetTag := scanner.Text() // 读取第一行输入作为待解码信元的Tag

scanner.Scan()

line := scanner.Text() // 读取第二行输入

arr := strings.Split(line, " ") // 将输入的16进制码流分割存入数组

i := 0 // 初始化指针i为0，用于遍历码流

result := []string{} // 初始化结果数组，用于存储待解码信元的Value

// 循环遍历整个码流数组

for i < len(arr) {

tag := arr[i] // 读取当前字节作为Tag

len1, \_ := strconv.ParseInt(arr[i+1], 16, 32) // 读取下一个字节作为Length的低位字节（小端序）

len2, \_ := strconv.ParseInt(arr[i+2], 16, 32) // 读取再下一个字节作为Length的高位字节（小端序）

// 计算长度Length，按照小端序对高低位字节进行组合

length := int(len2\*256 + len1)

i += 3 // 指针i向前移动3位，因为已经读取了Tag和Length字段

// 如果当前Tag等于目标Tag，则需要解码此信元的值

if tag == targetTag {

// 将信元的Value字节逐个添加到结果数组中

for j := 0; j < length; j++ {

result = append(result, arr[i+j])

}

}

i += length // 指针i向前移动Length位，跳过当前信元的Value字节

}

// 输出解码后的值

fmt.Println(strings.Join(result, " "))

}