# **E卷-最大值[200分]（ Java | Python3 | C++ | C语言 | JsNode | Go ）**



10 9



910



import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 读取输入的数字组合

String input = scanner.nextLine();

String[] nums = input.split(" "); // 按空格分割输入字符串

// 使用自定义比较器对字符串数组进行排序

Arrays.sort(nums, new Comparator<String>() {

@Override

public int compare(String x, String y) {

// 比较两个字符串拼接后的大小

return (y + x).compareTo(x + y); // 反向排序

}

});

// 如果排序后的第一个元素是 '0'，则结果是 '0'

if (nums[0].equals("0")) {

System.out.println("0");

return;

}

// 拼接排序后的字符串

StringBuilder result = new StringBuilder();

for (String num : nums) {

result.append(num);

}

// 输出结果

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

}

}



from functools import cmp\_to\_key

# 自定义比较函数

def compare(x, y):

# 比较两个字符串拼接后的大小

if x + y > y + x:

return -1 # x 应该排在 y 前面

else:

return 1 # y 应该排在 x 前面

def largestNumber(nums):

# 将整数转换为字符串

nums\_str = list(map(str, nums))

# 使用自定义比较函数对字符串进行排序

nums\_str.sort(key=cmp\_to\_key(compare))

# 如果排序后的第一个元素是 '0'，则结果是 '0'

if nums\_str[0] == '0':

return '0'

# 拼接排序后的字符串

return ''.join(nums\_str)

# 主程序

if \_\_name\_\_ == "\_\_main\_\_":

# 读取输入

input\_nums = input()

nums = list(map(int, input\_nums.split())) # 将输入转换为整数列表

# 获取并输出最大的整数

result = largestNumber(nums)

print(result)



#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

// 自定义比较函数

bool compare(const string& x, const string& y) {

// 比较两个字符串拼接后的大小

return (x + y) > (y + x); // 反向排序

}

int main() {

vector<string> nums; // 存储输入的数字组合

string input;

// 读取输入

getline(cin, input);

size\_t pos = 0;

while ((pos = input.find(' ')) != string::npos) {

nums.push\_back(input.substr(0, pos)); // 提取子字符串

input.erase(0, pos + 1); // 移除已处理的部分

}

nums.push\_back(input); // 添加最后一个数字

// 使用自定义比较函数对字符串数组进行排序

sort(nums.begin(), nums.end(), compare);

// 如果排序后的第一个元素是 '0'，则结果是 '0'

if (nums[0] == "0") {

cout << "0" << endl;

return 0;

}

// 拼接排序后的字符串并输出结果

for (const string& num : nums) {

cout << num;

}

cout << endl;

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_INPUT\_SIZE 1000 // 定义最大输入字符串的长度

// 自定义比较函数，用于 qsort 排序

int compare(const void\* a, const void\* b) {

// 将 void 指针转换为字符串指针

char \*x = \*(char\*\*)a;

char \*y = \*(char\*\*)b;

// 创建两个拼接字符串

char xy[40], yx[40];

strcpy(xy, x);

strcat(xy, y);

strcpy(yx, y);

strcat(yx, x);

// 比较拼接后的字符串

return strcmp(yx, xy); // 如果 yx > xy，返回正数，表示 y 应该排在 x 前面

}

// 函数：将数字数组转换为能组成的最大数字

char\* largestNumber(char\*\* numsStr, int numsSize) {

// 使用 qsort 对字符串数组进行排序，传入自定义的比较函数

qsort(numsStr, numsSize, sizeof(char\*), compare);

// 如果排序后的第一个元素是 "0"，则返回 "0"

if (strcmp(numsStr[0], "0") == 0) {

return "0";

}

// 计算结果字符串的长度

int totalLength = 0;

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

totalLength += strlen(numsStr[i]);

}

// 分配足够的空间来存储最终结果

char\* result = (char\*)malloc((totalLength + 1) \* sizeof(char));

result[0] = '\0'; // 初始化空字符串

// 拼接排序后的字符串

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

strcat(result, numsStr[i]);

}

return result; // 返回最终拼接后的字符串

}

int main() {

// 读取输入

char input[MAX\_INPUT\_SIZE];

fgets(input, MAX\_INPUT\_SIZE, stdin); // 读取一行输入

// 去掉输入末尾的换行符

input[strcspn(input, "\n")] = '\0';

// 统计空格数量，推断数字个数

int numsSize = 1; // 初始值为1，因为最后一个数字后面没有空格

for (int i = 0; input[i] != '\0'; i++) {

if (input[i] == ' ') {

numsSize++;

}

}

// 动态分配字符串数组

char\*\* numsStr = (char\*\*)malloc(numsSize \* sizeof(char\*));

char\* token = strtok(input, " "); // 使用空格分割输入字符串

int i = 0;

while (token != NULL) {

numsStr[i] = (char\*)malloc((strlen(token) + 1) \* sizeof(char));

strcpy(numsStr[i], token); // 将每个分割出的数字存入字符串数组

token = strtok(NULL, " ");

i++;

}

// 调用 largestNumber 函数获取结果

char\* result = largestNumber(numsStr, numsSize);

printf("%s\n", result); // 输出结果

// 释放动态分配的内存

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

free(numsStr[i]);

}

free(numsStr);

free(result);

return 0;

}



// 读取输入并将其转换为字符串数组

const input = require("fs").readFileSync(0, "utf-8").trim();

const nums = input.split(" "); // 按空格分割输入字符串

// 自定义比较函数

function compare(x, y) {

// 比较两个字符串拼接后的大小

return (y + x).localeCompare(x + y); // 反向排序

}

// 使用自定义比较函数对字符串数组进行排序

nums.sort(compare);

// 如果排序后的第一个元素是 '0'，则结果是 '0'

if (nums[0] === "0") {

console.log("0");

} else {

// 拼接排序后的字符串并输出结果

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

}



package main

import (

"bufio"

"fmt"

"os"

"sort"

"strings"

)

// 自定义比较函数，用于字符串拼接后的排序

func compare(x, y string) bool {

// 比较两个字符串拼接后的大小

return x+y > y+x

}

// largestNumber 函数用于将字符串数组转换为能组成的最大数字

func largestNumber(nums []string) string {

// 使用 sort.Slice 函数并传入自定义的比较函数

sort.Slice(nums, func(i, j int) bool {

return compare(nums[i], nums[j])

})

// 如果排序后的第一个元素是 "0"，则结果是 "0"

if nums[0] == "0" {

return "0"

}

// 拼接排序后的字符串

return strings.Join(nums, "")

}

func main() {

// 使用 bufio.NewReader 读取输入

reader := bufio.NewReader(os.Stdin)

input, \_ := reader.ReadString('\n') // 读取一整行输入，包含空格和换行符

// 去掉输入末尾的换行符

input = strings.TrimSpace(input)

// 按空格分割输入，得到字符串数组

nums := strings.Split(input, " ")

// 调用 largestNumber 函数获取结果

result := largestNumber(nums)

// 输出结果

fmt.Println(result)

}