C语言程序设计大作业

Codeforces Rating Clawer 2023

单 位：

班 级：

学 号：

姓 名：

任课教师：

大 学

**2023**年**5**月

## 文档信息

|  |  |  |  |
| --- | --- | --- | --- |
| 源代码 | 可执行文件 | 自评成绩 | 答辩PPT |
| 有 | 有 | 优 | 有 |

## 学术诚信承诺书

## 题目概述

1. 概述

内容：Codeforces Rating Clawer 2023通过调用Codeforces官方API读取用户做题数据，并进行统计，通过可视化图表显示统计结果。

目标：实现从输入用户名到查看可视化统计数据的简易流程。

1. 所要完成的任务说明

|  |  |  |
| --- | --- | --- |
| 序号 | 功能分类 | 任务 |
| 1 | 必做 | 读取用户名单，并进行用户做题情况进行抓取 |
| 2 | 必做 | 对数据进行分析，统计  a) 统计用户的做题总数  b) 用户做题数，可以按年份、月份、周统计  c) 统计用户题目的知识点标签的分布  d) 统计用户题目难度的分布 |
| 3 | 必做 | 将结果变成对应的表格，输出为HTML文件 |
| 4 | 选作 | 绘制个人做题数的热力图 |
| 5 | 选作 | 绘制用户题目知识点标签的分布比例饼图 |
| 6 | 选作 | 绘制用户题目难度的分布情况的直方图 |

1. 完成情况

|  |  |
| --- | --- |
| 功能序号 | 完成情况说明 |
| 1 | 通过在Codeforces Rating Clawer-2023.exe输入用户名(handle)抓取用户信息 |
| 2 | 完成功能1的操作后,生成response. json和分析数据后的response\_parsed.json,  数据存储于response\_parsed.json:   1. 做题总数:存储于data\_all\_solved 2. 年份做题数:存储于group\_year\_data   月份做题数:按年份存储于group\_month\_data\_per\_year  周做题数:通过功能4热力图间接展示   1. 标签分布:存储于data\_tag 2. 难度分布:存储于data\_level |
| 3 | 利用response\_parsed.json与Codeforces Rating Clawer-2023.html实现 |
| 4 | 利用response\_parsed.json与Codeforces Rating Clawer-2023.html绘制个人做题数的热力图 |
| 5 | 利用response\_parsed.json与Codeforces Rating Clawer-2023.html绘制知识点分布的比例饼图(本作业使用南丁格尔图，为饼图修改版) |
| 6 | 利用response\_parsed.json与Codeforces Rating Clawer-2023.html绘制难度分布直方图 |

1. 开发环境说明

Windows版本:Windows 11

操作系统版本:22621.1413

开发工具:Microsoft Visual Studio Community 2022;

Microsoft Visual Studio Code

## 程序分析与设计

1. 系统分析

函数说明：

|  |  |
| --- | --- |
| 编号：001 | char\* input\_handle(char\* handle) |
| 函数功能 | 把输入的字符串handle转化为URL |
| 参数说明 | handle: char\*型，一个字符串 |
| 函数返回值 | char\*型，转化后的URL |

|  |  |
| --- | --- |
| 编号：002 | void get\_json(char\* url) |
| 函数功能 | 向Codeforces网站发送请求，返回用户的submission数据 |
| 参数说明 | url: char\*型，一个字符串 |
| 函数返回值 | 无 |

|  |  |
| --- | --- |
| 编号：003 | size\_t write\_callback(char\* ptr, size\_t size, size\_t nmemb, void\* userdata) |
| 函数功能 | 应用curl库时必须设置的回调函数 |
| 参数说明 | ptr：char\*型，一个字符串，指向接收到的数据  size：size\_t型，无符号整数型，每个数据项的大小  nmemb：接收到的数据项数量  userdata：一个指针，指向用户自定义数据的指针 |
| 函数返回值 | Size\_t型：返回实际写入的数据字节数 |

|  |  |
| --- | --- |
| 编号：004 | char\* string\_format(char\* root) |
| 函数功能 | 对发送请求后返回的response.json中的题目标签字符串删去两端多余的引号 |
| 参数说明 | root: char\*型，一个字符串 |
| 函数返回值 | 去除题目标签两端多余引号的字符串 |

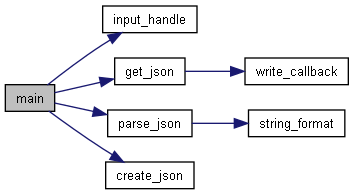
|  |  |
| --- | --- |
| 编号：005 | void parse\_json(char\* handle) |
| 函数功能 | 对发送请求后返回的response.json中的数据进行遍历分析 |
| 参数说明 | handle: char\*型，一个字符串 |
| 函数返回值 | 无 |

|  |  |
| --- | --- |
| 编号：006 | void create\_json(char\* handle) |
| 函数功能 | 把遍历分析后的做题数据包装成json |
| 参数说明 | handle: char\*型，一个字符串 |
| 函数返回值 | 无 |

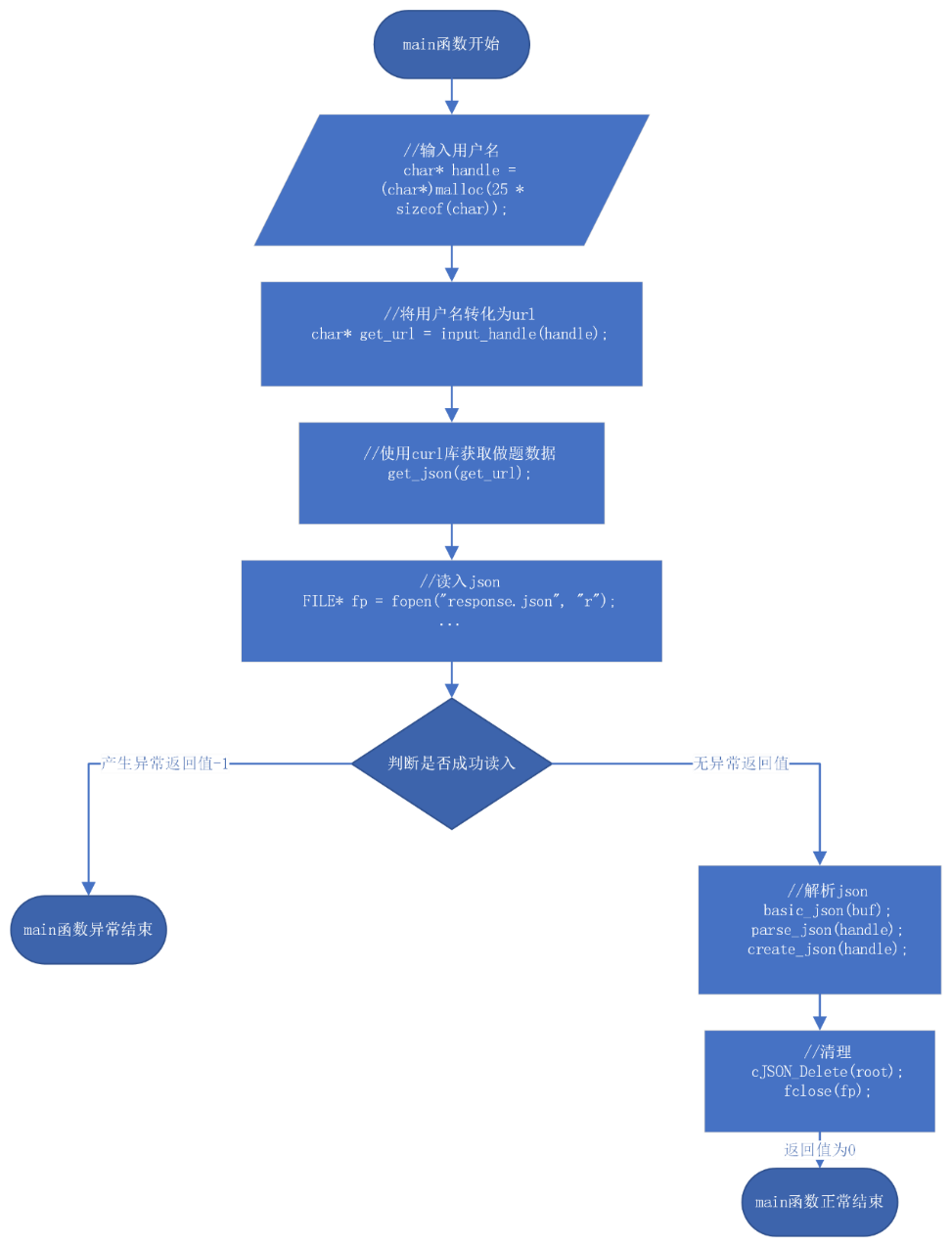
函数之间的调用关系图

图示

描述已自动生成



1. 函数设计



int main()

{

*//输入用户名*

    char\* handle = (char\*)malloc(25 \* sizeof(char));

*//将用户名转化为url*

    char\* get\_url = input\_handle(handle);

*//printf("test url : %s\n", get\_url);*

*//使用curl库获取做题数据*

    get\_json(get\_url);

*//读入json*

    FILE\* fp = fopen("response.json", "r");

    if (fp == nullptr) return -1;

    fseek(fp, 0, SEEK\_END);

    int filesize = ftell(fp);

    fseek(fp, 0, SEEK\_SET);

    char\* buf = (char\*)malloc(filesize \* sizeof(char));

    memset(buf, 0, filesize \* sizeof(char));

    if (int ret = fread(buf, filesize, sizeof(char), fp) == -1)

    {

        fclose(fp);

        return -1;

    }

*//解析json*

    basic\_json(buf);

    parse\_json(handle);

    create\_json(handle);

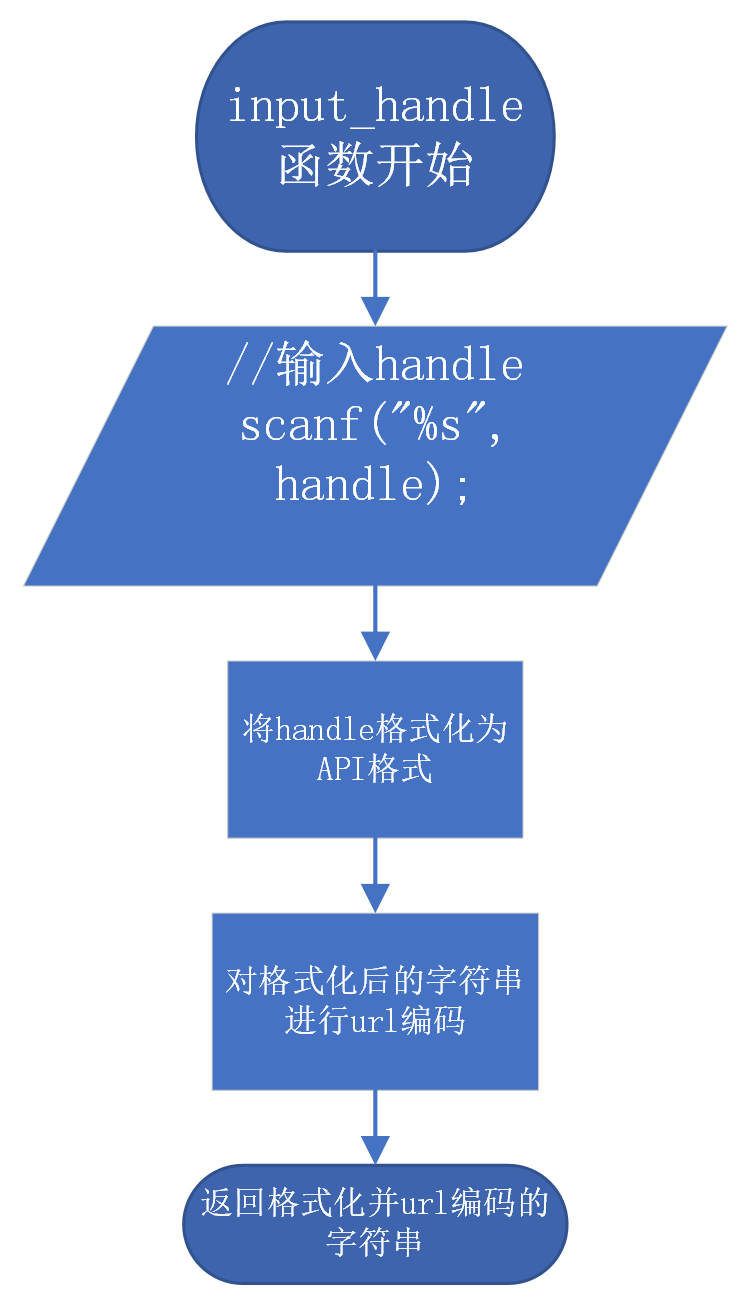
*//清理*

    cJSON\_Delete(root);

    fclose(fp);

    return 0;

}



char\* input\_handle(char\* *handle*)

{

*//使用curl库进行url编码*

    CURL\* curl = curl\_easy\_init();

    char\* url = (char\*)malloc(75 \* sizeof(char));

    strcpy(url, "https://codeforces.com/api/user.status?handle=");

*//输入handle*

    scanf("%s", *handle*);

    strcat(url, *handle*);

*//curl库url编码*

    char\* encodedUrl = curl\_easy\_escape(curl, url, 0);

    if (encodedUrl) {

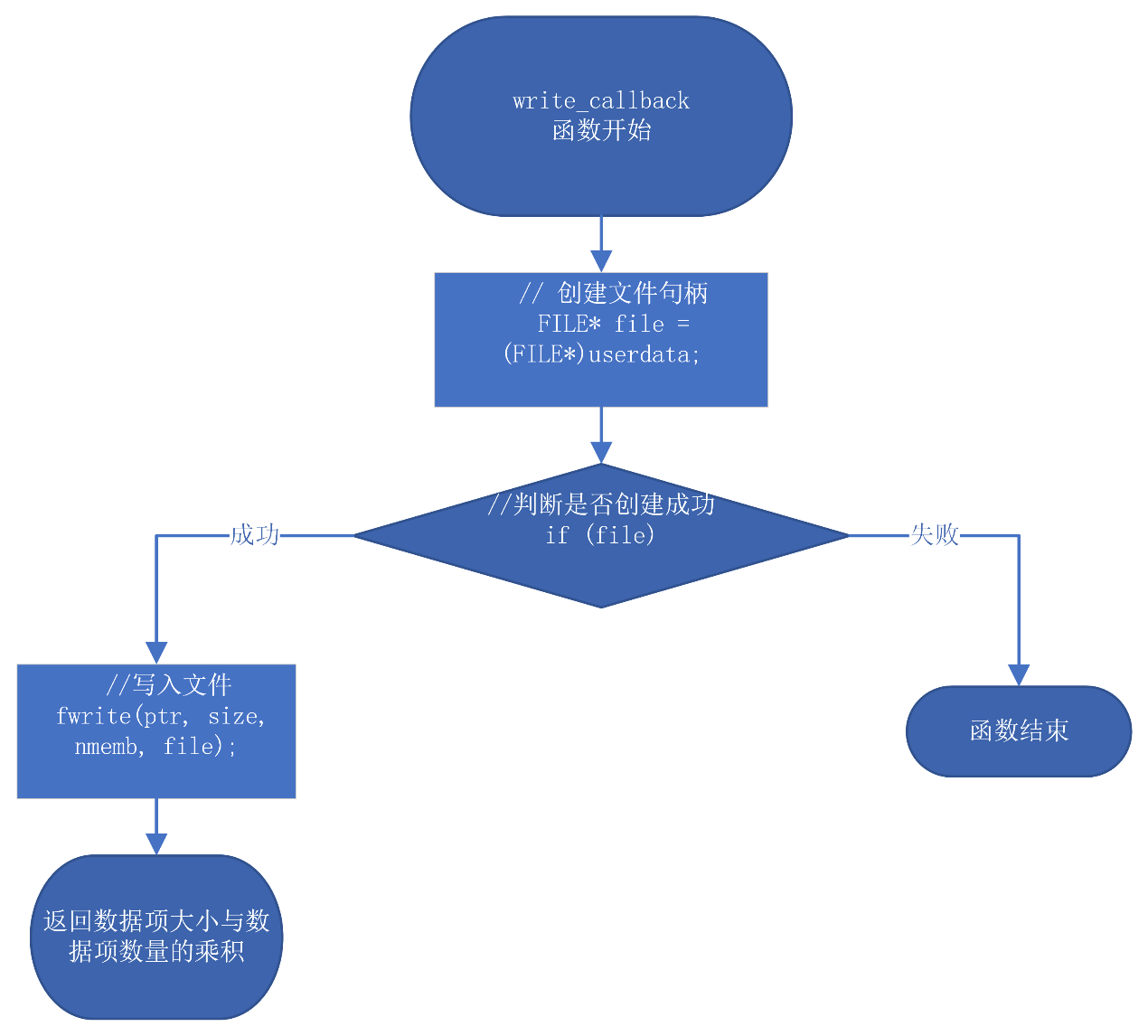
*//printf("Encoded URL: %s\n", encodedUrl);*

        curl\_free(encodedUrl);

    }

    return url;

}



size\_t write\_callback(char\* *ptr*, size\_t *size*, size\_t *nmemb*, void\* *userdata*) {

*// 创建文件句柄*

    FILE\* file = (FILE\*)*userdata*;

*//判断是否创建成功*

    if (file) {

*//写入文件*

        fwrite(*ptr*, *size*, *nmemb*, file);

    }

    return *size* \* *nmemb*;

}

图示

描述已自动生成

void get\_json(char\* *url*) *//存储response.json*

{

*// 初始化CURL库*

    curl\_global\_init(CURL\_GLOBAL\_DEFAULT);

*// 创建CURL句柄*

    CURL\* curl = curl\_easy\_init();

    if (!curl) {

        fprintf(stderr, "Failed to initialize CURL\n");

        return;

    }

*// 设置请求的URL*

    curl\_easy\_setopt(curl, CURLOPT\_URL, *url*);

*// 创建用于保存响应数据的文件*

    FILE\* file = fopen("response.json", "w");

    if (!file) {

        fprintf(stderr, "Failed to create file\n");

        curl\_easy\_cleanup(curl);

        return;

    }

*// 设置写入响应数据的回调函数*

    curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, write\_callback);

    curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, file);

*// 发送GET请求*

    CURLcode res = curl\_easy\_perform(curl);

    if (res != CURLE\_OK) {

        fprintf(stderr, "Failed to send request: %s\n", curl\_easy\_strerror(res));

    }

*// 关闭文件和CURL句柄*

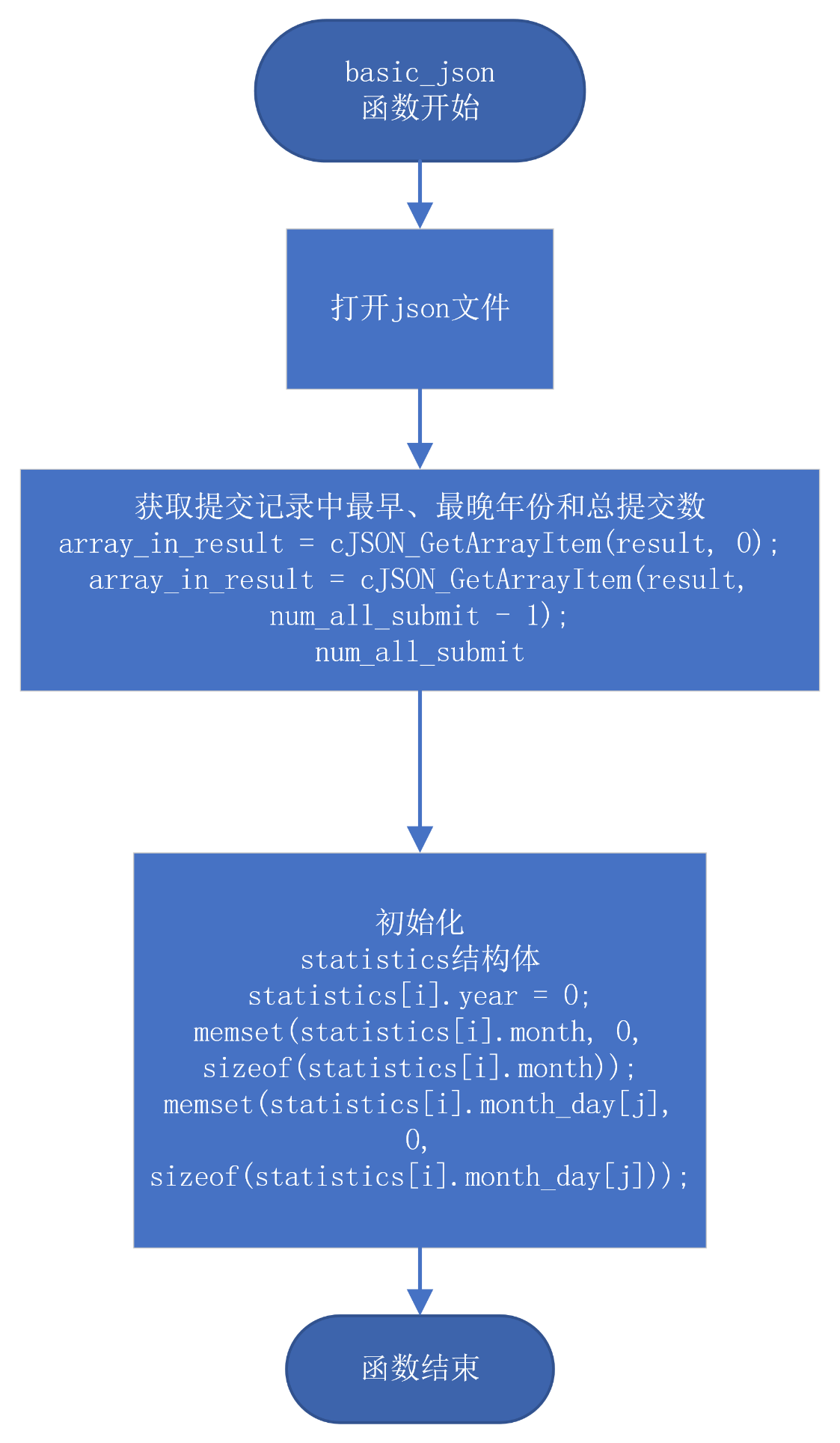
    fclose(file);

    curl\_easy\_cleanup(curl);

*// 清理CURL库*

    curl\_global\_cleanup();

}



void basic\_json(char\* *buf*)

{

    root = cJSON\_Parse(*buf*); *//读入json*

    status = cJSON\_GetObjectItem(root, "status"); *//指向json-status*

    result = cJSON\_GetObjectItem(root, "result"); *//指向json-result*

*//总提交数*

    num\_all\_submit = cJSON\_GetArraySize(result);

*//不同标签数*

    int tag\_size = 0;

*//提交记录中最晚年份*

    array\_in\_result = cJSON\_GetArrayItem(result, 0);

    time\_in\_array = cJSON\_GetObjectItem(array\_in\_result, "creationTimeSeconds");

    time\_temp = (unsigned long)cJSON\_GetNumberValue(time\_in\_array);

    biggest\_time\_info = \*localtime(&time\_temp);

    biggest\_year = biggest\_time\_info.tm\_year;

*//提交记录中最早年份*

    array\_in\_result = cJSON\_GetArrayItem(result, num\_all\_submit - 1);

    time\_in\_array = cJSON\_GetObjectItem(array\_in\_result, "creationTimeSeconds");

    time\_temp = (unsigned long)cJSON\_GetNumberValue(time\_in\_array);

    smallest\_time\_info = \*localtime(&time\_temp);

    smallest\_year = smallest\_time\_info.tm\_year;

    statistics = (struct statistic\*)malloc(sizeof(struct statistic) \* (biggest\_year - smallest\_year + 1));

*//初始化存储统计数据的statistics结构体*

    for (int i = 0; i < (biggest\_year - smallest\_year + 1); i++)

    {

*// 设置结构体成员默认值*

        statistics[i].leapyear = false;

        statistics[i].year = 0;

        memset(statistics[i].month, 0, sizeof(statistics[i].month));

        for (int j = 0; j < 12; j++)

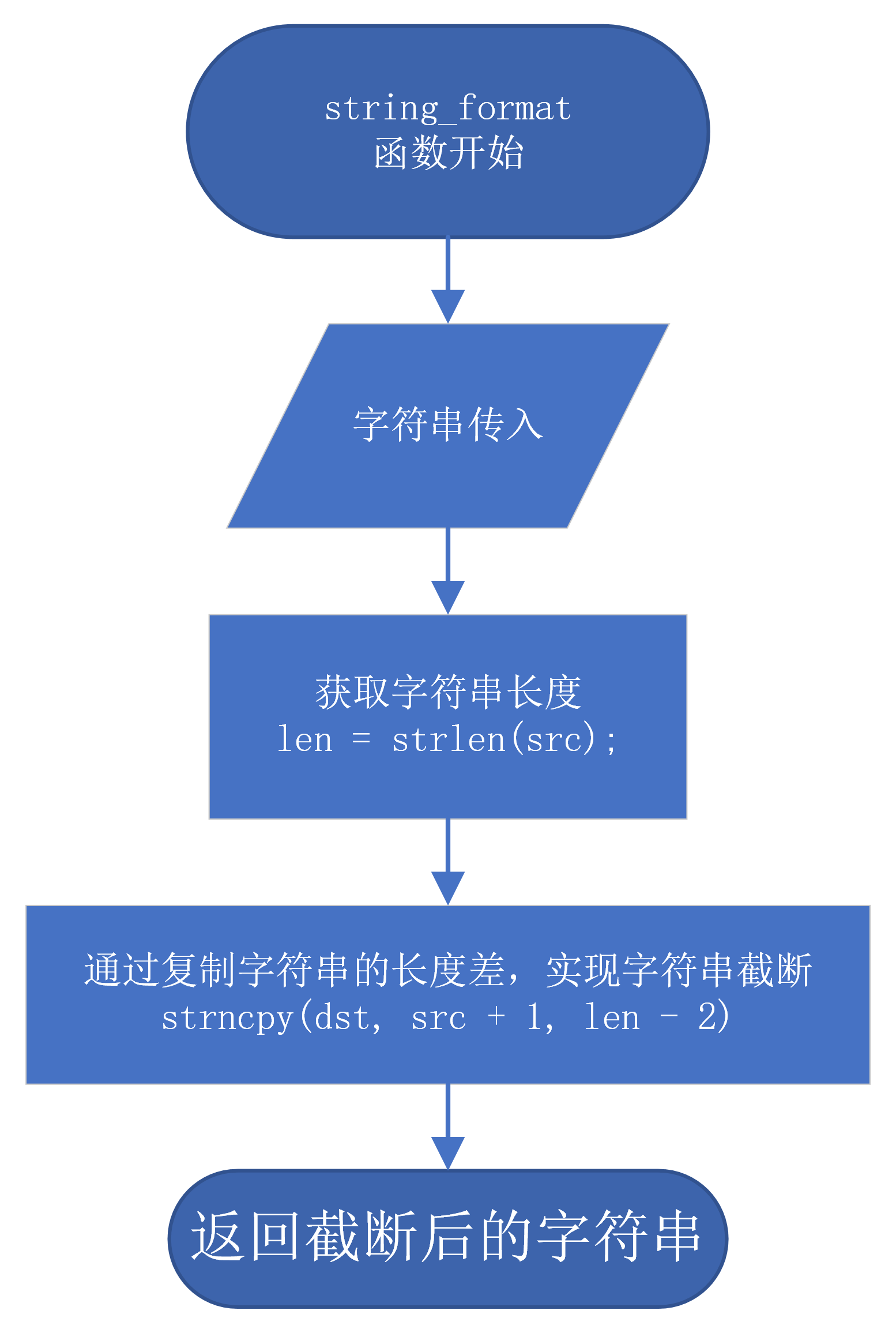
        {

            memset(statistics[i].month\_day[j], 0, sizeof(statistics[i].month\_day[j]));

        }

    }

}



char\* string\_format(char\* *src*){

    int len = strlen(*src*);

    char\* dst = (char\*)malloc(sizeof(char) \* len - 1);

*//通过复制字符串的长度差，实现字符串截断*

    memset(dst, 0, sizeof(char) \* len - 1);

    strncpy(dst, *src* + 1, len - 2);

    return dst;

}

图示

描述已自动生成

void parse\_json(char\* *handle*){

*//遍历提交记录*

    for (int i = 0; i < num\_all\_submit; i++){

        array\_in\_result = cJSON\_GetArrayItem(result, i);

        verdict\_in\_array = cJSON\_GetObjectItem(array\_in\_result, "verdict");

*//若提交状态为'OK'则进入统计*

        if (!strcmp(cJSON\_Print(verdict\_in\_array), "\"OK\"")){

            time\_in\_array = cJSON\_GetObjectItem(array\_in\_result, "creationTimeSeconds");

            time\_temp = (unsigned long)cJSON\_GetNumberValue(time\_in\_array);

            probs[num\_all\_ok].time\_unix = time\_temp;

            probs[num\_all\_ok].time = \*localtime(&time\_temp);

            statistics[(probs[num\_all\_ok].time.tm\_year - smallest\_year)].year++;

            statistics[(probs[num\_all\_ok].time.tm\_year - smallest\_year)].month[probs[num\_all\_ok].time.tm\_mon] ++;

            statistics[(probs[num\_all\_ok].time.tm\_year - smallest\_year)].month\_day[probs[num\_all\_ok].time.tm\_mon][probs[num\_all\_ok].time.tm\_mday - 1] ++;

            prob\_in\_array = cJSON\_GetObjectItem(array\_in\_result, "problem");

            tag\_in\_array = cJSON\_GetObjectItem(prob\_in\_array, "tags");

            int num\_tags = cJSON\_GetArraySize(tag\_in\_array);

            rating\_in\_array = cJSON\_GetObjectItem(prob\_in\_array, "rating");

            int rating = cJSON\_GetNumberValue(rating\_in\_array);

            if (rating <= 1000){

                probs[i].level = Easy;

                easy++;

            }

            else if (rating > 1000 && rating <= 1500){

                probs[i].level = Intermediate;

                intermediate++;

            }

            else if (rating > 1500 && rating <= 2000){

                probs[i].level = Moderate;

                moderate++;

            }

            else if (rating > 2000 && rating <= 2500){

                probs[i].level = Challenging;

                challenging++;

            }

            else if (rating > 2500 && rating <= 3000){

                probs[i].level = Difficult;

                difficult++;

            }

            else{

                probs[i].level = Expert;

                expert++;

            }

            num\_all\_ok++;

            for (int j = 0; j < num\_tags; j++){

                cJSON\* tag\_cjson = nullptr;

                tag\_cjson = cJSON\_GetArrayItem(tag\_in\_array, j);

*//由于不可操作野指针，决定先写入tag再判断是否存在相同项*

                probs\_cat[tag\_size].tag = string\_format(cJSON\_Print(tag\_cjson));

*//搜索之前储存的所有tag*

                for (int k = 0; k <= tag\_size; k++){

                    if (!strncmp(probs\_cat[tag\_size].tag, probs\_cat[k].tag, sizeof(probs\_cat[k].tag))){

                        probs\_cat[k].num\_tag++;

*//搜索到边界时才相等的话 说明之前无相同的tag*

                        if (k == tag\_size)  tag\_size++;

*//搜索到相同的标签后，遍历下一条做题记录*

                        break;

                    }

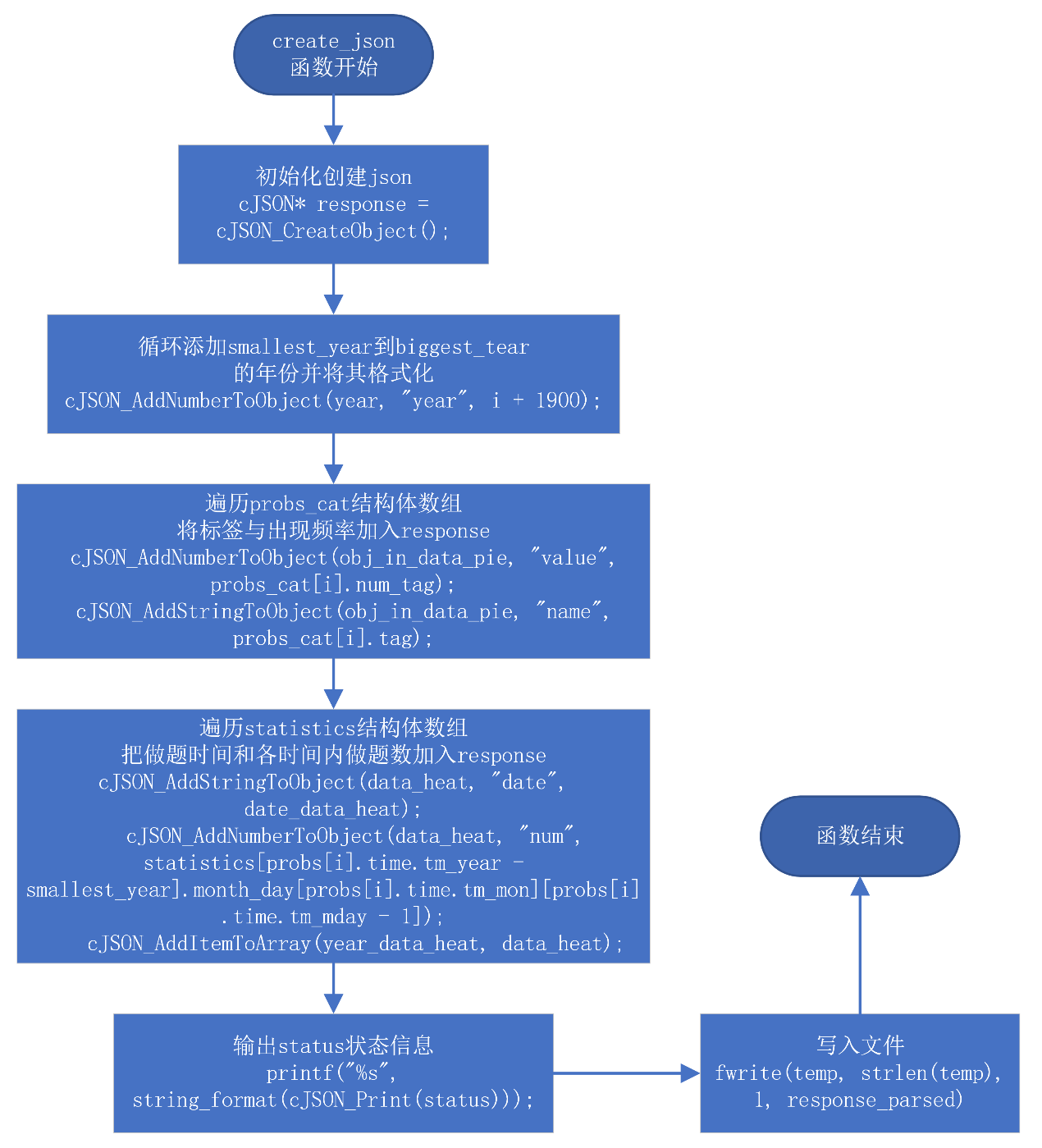
                }

            }

        }

    }

}



void create\_json(char\* *handle*)

{

    cJSON\* response = cJSON\_CreateObject();

    cJSON\* status\_json = cJSON\_AddStringToObject(response, "status", string\_format(cJSON\_Print(status)));

    cJSON\* handle\_json = cJSON\_AddStringToObject(response, "handle", *handle*);

    cJSON\* min\_year = cJSON\_AddNumberToObject(response, "min\_year", smallest\_year);

    cJSON\* max\_year = cJSON\_AddNumberToObject(response, "max\_year", biggest\_year);

    cJSON\* year = cJSON\_AddArrayToObject(response, "year");

    cJSON\* data\_tag = cJSON\_AddArrayToObject(response, "data\_tag"); *//tags*

    cJSON\* data\_all\_submitted = cJSON\_AddNumberToObject(response, "data\_all\_submitted", num\_all\_submit);

    cJSON\* data\_all\_solved = cJSON\_AddNumberToObject(response, "data\_all\_solved", num\_all\_ok);

    cJSON\* group\_month\_data\_per\_year = cJSON\_AddArrayToObject(response, "group\_month\_data\_per\_year");

    cJSON\* group\_year\_data = cJSON\_AddArrayToObject(response, "group\_year\_data");

    for (int i = smallest\_year; i <= biggest\_year; i++)

    {

        cJSON\_AddNumberToObject(year, "year", i + 1900);

    }

    for (int i = 0; i < tag\_size; i++) *//输出tag的json*

    {

        cJSON\* obj\_in\_data\_pie = cJSON\_CreateObject();

        cJSON\_AddNumberToObject(obj\_in\_data\_pie, "value", probs\_cat[i].num\_tag);

        cJSON\_AddStringToObject(obj\_in\_data\_pie, "name", probs\_cat[i].tag);

        cJSON\_AddItemToArray(data\_tag, obj\_in\_data\_pie);

    }

    cJSON\* group\_data\_heat = cJSON\_AddArrayToObject(response, "group\_data\_heat"); *//输出热力图数据的json*

    cJSON\* year\_data\_heat = cJSON\_CreateArray();

    cJSON\* data\_heat = cJSON\_CreateArray();

    for (int i = num\_all\_ok - 1, j = smallest\_year; i >= 0; i--)

    {

        if (probs[i].time.tm\_year != j)

        {

            year\_data\_heat = cJSON\_CreateArray();

            j++;

        }

        data\_heat = cJSON\_CreateArray();

        char date\_data\_heat[11] = { 0 };

        sprintf(date\_data\_heat, "%04d-%02d-%02d", probs[i].time.tm\_year + 1900, probs[i].time.tm\_mon + 1, probs[i].time.tm\_mday);

        cJSON\_AddStringToObject(data\_heat, "date", date\_data\_heat);

        cJSON\_AddNumberToObject(data\_heat, "num", statistics[probs[i].time.tm\_year - smallest\_year].month\_day[probs[i].time.tm\_mon][probs[i].time.tm\_mday - 1]);

        cJSON\_AddItemToArray(year\_data\_heat, data\_heat);

        if (i == 0 || probs[i - 1].time.tm\_year != j)

        {

            cJSON\_AddItemToArray(group\_data\_heat, year\_data\_heat);

        }

    }

    for (int i = 0; i < (biggest\_year - smallest\_year + 1); i++)

    {

        cJSON\* month\_data\_per\_year = cJSON\_CreateArray(); *//输出有做题记录年份的每月做题数*

        for (int j = 0; j < 12; j++)

        {

            cJSON\_AddNumberToObject(month\_data\_per\_year, "month\_data\_per\_year", statistics[i].month[j]);

        }

        cJSON\_AddItemToArray(group\_month\_data\_per\_year, month\_data\_per\_year);

        cJSON\_AddNumberToObject(group\_year\_data, "year\_data", statistics[i].year);

    }

    cJSON\* data\_level = cJSON\_AddArrayToObject(response, "data\_level"); *//输出题目难度分布的json*

    cJSON\* obj\_in\_data\_bar = cJSON\_CreateObject();

    cJSON\_AddNumberToObject(data\_level, "easy", easy);

    cJSON\_AddNumberToObject(data\_level, "intermediate", intermediate);

    cJSON\_AddNumberToObject(data\_level, "moderate", moderate);

    cJSON\_AddNumberToObject(data\_level, "challenging", challenging);

    cJSON\_AddNumberToObject(data\_level, "difficult", difficult);

    cJSON\_AddNumberToObject(data\_level, "expert", expert);

    char\* temp = cJSON\_Print(response);

    printf("%s", string\_format(cJSON\_Print(status)));

    FILE\* response\_parsed = fopen("response\_parsed.json", "w");

    fwrite(temp, strlen(temp), 1, response\_parsed);

}

## 测试分析

1. 测试与调试

以某用户为例：

1. 在Visual Studio Code平台打开以该用户名作为API handle参数调用API返回的json原文件，比较自行统计结果与原文件的差异；
2. 在codeforces平台查找该用户，比较官方平台数据与自行统计结果的差异；
3. 测试结果分析
4. json原文件含有93条记录，做题年份由2011至2017年，dsu标签计数1次，2017.12.22通过两题，challenging难度解题一道，皆与自行统计结果相同

（93条记录）

电脑萤幕的截图

描述已自动生成

图形用户界面, 文本

描述已自动生成

（做题年份由2011至2017年）

图形用户界面, 文本

描述已自动生成 文本

描述已自动生成

图形用户界面, 文本

描述已自动生成

（dsu标签计数1次）

图形用户界面, 应用程序

描述已自动生成

图示

描述已自动生成

（2017.12.22通过两题）

文本

描述已自动生成

文本

描述已自动生成

图形用户界面, 应用程序, 网站

描述已自动生成

（challenging难度解题一道）

文本

描述已自动生成 手机屏幕的截图

描述已自动生成

1. 自行统计过题数略大于官方统计数。自制热力图缺少2014.8.1，增加2014.8.13数据。

图片包含 徽标

描述已自动生成

图片包含 文本

描述已自动生成

应为官方统计数据不会重复统计相同题目的过题数，自制统计数据未作排除重复题目处理

图表, 散点图

描述已自动生成

图片包含 图表

描述已自动生成据查询codeforces-submission 2014.8.1数据



为该记录异常，分析得出，官方热力图统计数据为提交记录，而自制热力图统计数据为过题记录

据查询codeforces-submission 2014.8.13数据

图形用户界面, 应用程序, Teams

描述已自动生成

在submission提交记录中可查询两条2014.8.13数据。具体原因不明，推测官方热力图数据统计规则较特殊，不仅仅参考submission记录。

1. 结论

官方统计数据不会重复统计相同题目的过题数，自制统计数据未作排除重复题目处理

热力图绘制数据为过题数据，须格外注意

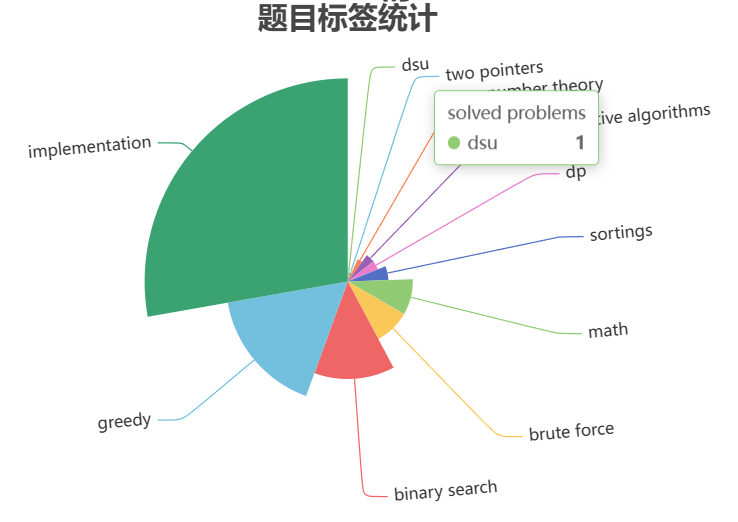
其余功能正常实现

## 系统演示

图形用户界面, 文本, 应用程序

描述已自动生成图形用户界面, 应用程序

中度可信度描述已自动生成



图表, 条形图

描述已自动生成

图表, 条形图

描述已自动生成

表格

中度可信度描述已自动生成

## 设计体会

1. 设计过程中的问题及其解决方案

问题1：c语言程序如何向服务器发送GET请求

解决方案：使用curl库发送GET请求

问题2：如何实现数据可视化

解决方案：利用Echats库，搭配cjson把数据包装成json文件提供给HTML生成图表

1. 设计过程中的错误及其原因分析

错误1：自行统计过题数略大于官方统计数

原因分析：应为官方统计数据不会重复统计相同题目的过题数，自制统计数据未作排除重复题目处理

错误2：自制热力图缺少2014.8.1，增加2014.8.13数据

原因分析：据查询codeforces-submission 2014.8.1数据



为该记录异常，分析得出，官方热力图统计数据为提交记录，而自制热力图统计数据为过题记录

据查询codeforces-submission 2014.8.13数据

图形用户界面, 应用程序, Teams

描述已自动生成

在submission提交记录中可查询两条2014.8.13数据。具体原因不明，推测官方热力图数据统计规则较特殊，不仅仅参考submission记录。

1. 体会和收获
   1. 学习了C语言程序的更多用途。使用C语言发送GET请求。
   2. 强化了设计程序的思维能力。分模块、分功能实现等。
   3. 增强了对json和c语言交互的理解。