

身份证号码查验程序 设计文档

软件程序：身份证号码查验程序

文档类型：设计文档

课 程：程序设计基础课程设计

授课教师：胡建伟

学生姓名：雷沛弘

学生班级：230087 班(23 计算机类 14 班)

学 号：23009200885

目录

一、 题目解析.....	1
1.1 需求.....	1
1.2 功能概述.....	1
1.3 查验原理.....	1
1.3.1 检验长度	1
1.3.2 检验字符	1
1.3.3 检验校验码	2
二、 关键数据结构和功能模块设计.....	2
2.1 关键数据结构.....	2
2.2 内存处理.....	3
2.3 功能模块设计和程序流程.....	4
三、 调试与测试.....	6
3.1 测试目标.....	6
3.2 测试方法.....	6
3.2.1 测试 <i>validateIDCard()</i> 函数.....	6
3.2.2 测试用户输入和内存分配	6
3.2.3 测试身份证号码的有效性判断和输出	7
3.2.4 测试程序的稳定性和鲁棒性	7
3.3 测试记录.....	7
3.4 测试结果.....	9
四、 使用程序的注意事项.....	10
4.1 程序的使用.....	10
4.2 github 仓库	10
4.3 更新时间.....	10
五、 程序源代码（C 语言）	10

身份证号码查验程序设计文档

一、 题目解析

1.1 需求

用户需要通过一个程序查验向该程序输入的身份证号码是否有效。

1.2 功能概述

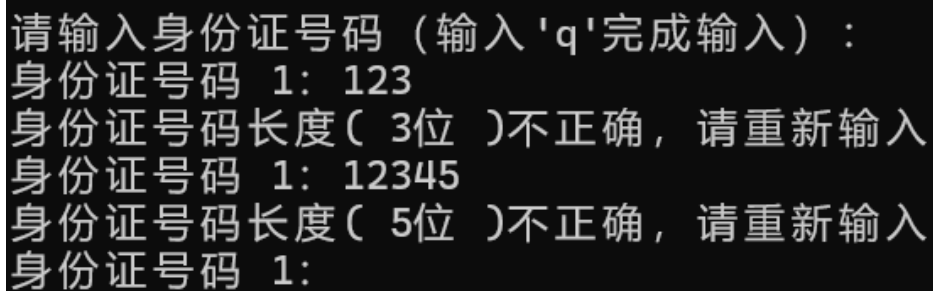
身份证号查验程序将验证输入的所有的身份证号码 (一次最多输入 10000 个, 按 *Enter* 键结束当前身份证号输入, 按按 q 或 Q 结束所有身份证号输入) 是否有效, 并对所有无效号码进行提示与输出。

1.3 查验原理

一个合法的身份证号码由 17 位地区、日期编号和顺序编号加 1 位校验码组成, 共计 18 位。

1.3.1 检验长度

程序会先判断输入的身份证号长度是否为 18 位。如果输入长度不是 18 位 (包括不足或超出 18 位), 程序将提示 “身份证号码长度 (x 位) 不正确, 请重新输入”, 其中 x 是检测的当前输入身份证号码的长度, 如图 (1) 所示。



```
请输入身份证号码 (输入 'q' 完成输入):  
身份证号码 1: 123  
身份证号码长度 ( 3位 ) 不正确, 请重新输入  
身份证号码 1: 12345  
身份证号码长度 ( 5位 ) 不正确, 请重新输入  
身份证号码 1:
```

图 1 检测输入身份证号码的长度

1.3.2 检验字符

在输入的身份证号码长度为 18 位后, 程序将判断号码中是否存在除了 0 9 和 X 以外的任何字符。如果存在不合法字符, 程序将提醒用户重新输入身份证号, 如图 (2) 所示。

```
请输入身份证号码（输入 'q' 完成输入）：
身份证号码 1: 123
身份证号码长度（3位）不正确，请重新输入
身份证号码 1: 12345
身份证号码长度（5位）不正确，请重新输入
身份证号码 1: 12345678912345678A
身份证号码包含不合法字符，请重新输入
```

图 2 检测输入身份证号码是否含有非法字符

1.3.3 检验校验码

在检验完身份证号的长度和字符符合要求后，程序会检验身份证号的校验码是否正确。

校验码的计算规则如下：

- 1. 先对前 17 位数字加权求和，权重分配为：7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2；
- 2. 然后将计算的和对 11 取模得到值 Z；
- 3. 最后按照表 (1) 检查值 Z 有没有对应的校验码 M：Z: 0 1 2 3 4 5 6 7 8 9 10 M: 1 0 X 9 8 7 6 5 4 3 2

表 1 Z 值和校验码 M 值对应关系表

Z	0	1	2	3	4	5	6	7	8	9	10
M	1	0	9	8	7	6	5	4	3	2	1

二、 关键数据结构和功能模块设计

2.1 关键数据结构

- 1. `char ** idCards`: 动态分配的指针数组，用于存储用户输入的身份证号码，初始化为 `NULL`
- 2. `int numIDCards`: 记录已输入的身份证号码数量，初始化为 0
- 3. `int maxIDCards`: 设置的最大输入身份证号码数量为 10000 个
- 4. `int len`: 验证当前输入的身份证号码长度，初始化为 0

2.2 内存处理

起初，程序将为指针数组动态分配内存，如下所示：

```
idCards = (char**)malloc(maxIDCards * sizeof(char*));
if (idCards == NULL) { // 检验内存是否分配成功
    printf("内存分配失败\n");
    return 1;
}
```

随后，程序将要求用户逐个输入身份证号码，并通过动态分配内存来存储当前身份证号码：

```
// 动态分配内存来存储每个身份证号码
idCards[numIDCards] = (char*)malloc(20 * sizeof(char)); //
    身份证号码长度为18位，最后一位为'\0'
if (idCards[numIDCards] == NULL) { // 检验内存是否分配成功
    printf("内存分配失败\n");
    return 1;
}
```

在内存分配成功后，程序将读取用户输入的身份证号码，去除输入的换行符并检验长度与字符：

```
// 使用 fgets 安全地读取用户输入的身份证号码
if (fgets(idCards[numIDCards], 20, stdin) == NULL) {
    printf("读取输入失败\n");
    return 1;
}

// 去除输入中的换行符
idCards[numIDCards][strcspn(idCards[numIDCards], "\n")] = '\0';
len = strlen(idCards[numIDCards]);
if (idCards[numIDCards][0] == 'q' || idCards[numIDCards][0] == 'Q') {
    break; // 用户输入 'q' 或 'Q' 结束输入
}
if (len != 18) {
    printf("身份证号码长度( %d位 )不正确，请重新输入\n", len);
    //free(idCards[numIDCards]);
    continue;
}
for(i=0;i<len;i++){
    if (!(idCards[numIDCards][i] >= '0' && idCards[numIDCards][i] <='9' ||
        idCards[numIDCards][i] == 'X')) {
        printf("身份证号码包含不合法字符，请重新输入\n");
        //free(idCards[numIDCards]);
        break;
    }
}
```

```
}
```

数据处理完毕，程序将检验每个身份证号码的校验码是否正确：

```
// 判断并输出所有无效身份证号码
for (i = 0; i < numIDCards; i++) {
    if (!validateIDCard(idCards[i])) {
        printf("%s\n", idCards[i]);
    }
}
```

最终，程序将释放所有动态分配的内存以避免造成内存溢出和泄露。

```
// 释放动态分配的内存
for (i = 0; i < numIDCards; i++) {
    free(idCards[i]);
}
free(idCards);
```

2.3 功能模块设计和程序流程

该程序的功能模块流程图如下图所示：

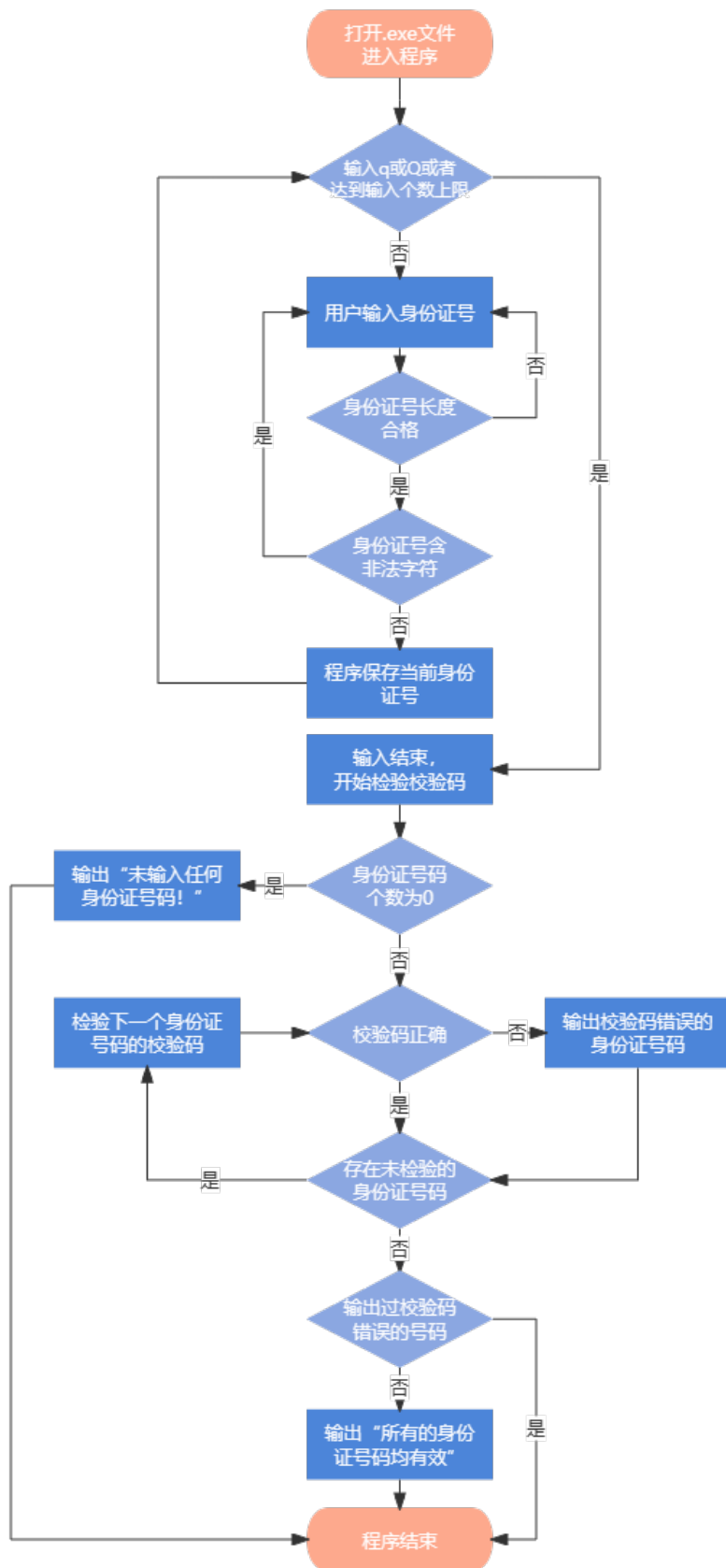


图 3 功能模块设计和程序流程图

三、 调试与测试

3.1 测试目标

- 确保身份证号码查验函数 *validateIDCard()* 能正常运行并返回正确的结果。
- 验证程序可以正确读取用户输入的身份证号码。
- 确保程序能够正确判断身份证号码的有效性并输出无效的身份证号码。
- 验证程序在各种输入情况下的稳定性和鲁棒性。

3.2 测试方法

3.2.1 测试 *validateIDCard()* 函数

创建多个测试用例，包括合法的身份证号码和非法的身份证号码。对每个测试用例，手动计算预期的 Z 值，并将其与实际结果进行比较以验证函数是否返回了预期的结果。确保函数对于各种情况下的边界条件都能正确处理。

3.2.2 测试用户输入和内存分配

执行程序并按照提示逐个输入身份证号码。测试不同情况下的输入，包括合法和非法的身份证号码，长度不正确的输入以及输入 *q* 或 *Q* 终止输入等。确保程序能够正确读取用户输入的身份证号码，并进行相应的处理。通过在程序中设计相应的测试输出以检查内存分配是否成功，以防止内存泄漏或分配失败的情况，比如下面两段代码：

1) 检验是否成功指针数组动态分配内存：

```
idCards = (char**)malloc(maxIDCards * sizeof(char*));  
if (idCards == NULL) { // 检验内存是否分配成功  
    printf("内存分配失败\n");  
    return 1;  
}
```

2) 检验是否成功通过动态分配内存来存储当前身份证号码：

```
// 动态分配内存来存储每个身份证号码  
idCards[numIDCards] = (char*)malloc(20 * sizeof(char)); //  
    身份证号码长度为18位，最后一位为'\0'  
if (idCards[numIDCards] == NULL) { // 检验内存是否分配成功  
    printf("内存分配失败\n");  
    return 1;  
}
```


3.2.3 测试身份证号码的有效性判断和输出

输入多个合法和非法的身份证号码。确保程序能够正确判断身份证号码的有效性。验证程序是否能够正确输出无效的身份证号码。

3.2.4 测试程序的稳定性和鲁棒性

输入大量的身份证号码，包括合法和非法的号码。验证程序在处理大量输入时是否仍能正常运行，且不会出现崩溃或异常行为。尝试各种边界情况和异常情况，如无输入、输入 q 或 Q 终止输入等，确保程序能够正确处理并给出合适的提示。

3.3 测试记录

测试过程记录如下：

先测试能否发现不符合 18 位长度要求的身份证号码：

```
请输入身份证号码（输入 'q' 或 'Q' 完成号码的输入）  
身份证号码 1: 123456  
身份证号码长度（ 6位 ）不正确，请重新输入  
身份证号码 1: 123q  
身份证号码长度（ 4位 ）不正确，请重新输入  
身份证号码 1:  
身份证号码长度（ 0位 ）不正确，请重新输入
```

图 4 检测当前身份证号码长度

在长度符合要求后，测试能否发现含有非法字符的身份证号码：

```
身份证号码 1: 12345678912345678q  
身份证号码包含不合法字符，请重新输入
```

图 5 检测当前身份证号码是否含有非法字符

如果当前输入的身份证号码的长度和字符均符合要求，则保存该号码。随后，检查输入的身份证号码个数是否达到最大值，如果否，让用户继续输入下一个号码。

```
身份证号码 2: 123456789123456789
```

图 6 保存当前号码

检测能否输入 q 或 Q 就能立刻停止输入并开始判断校验码：

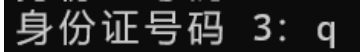


图 7 检测能否终止输入

最后，检测所有输出的错误身份证号码的校验码是否确实有误。以下是测试全程：

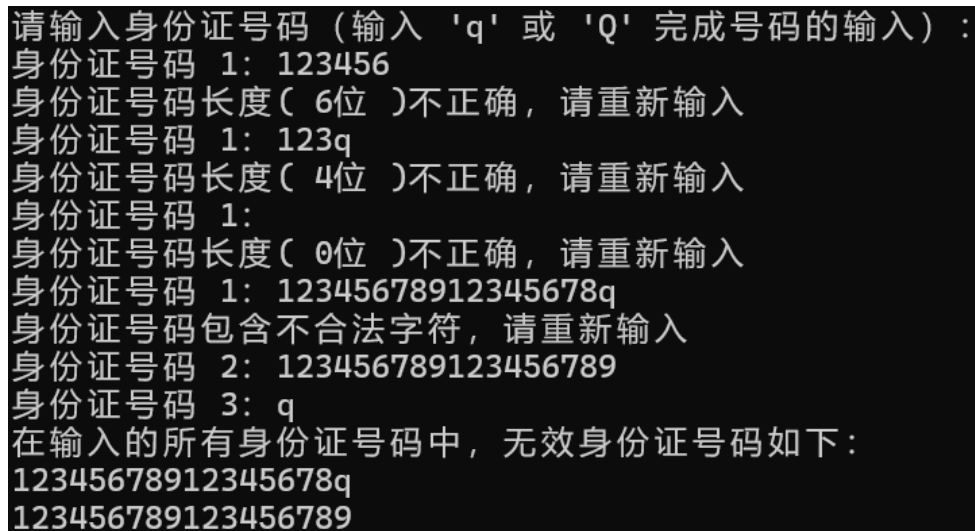


图 8 测试全程

检测输入为空时输出是否为“未输入任何身份证号码！”

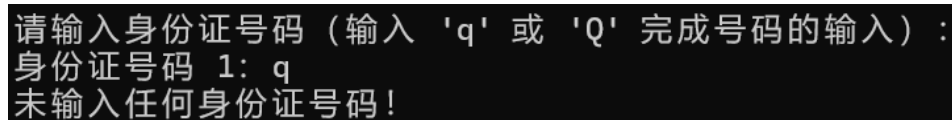


图 9 在未输入任何号码时的测试输出

内存的分配情况通过断点进行测试：

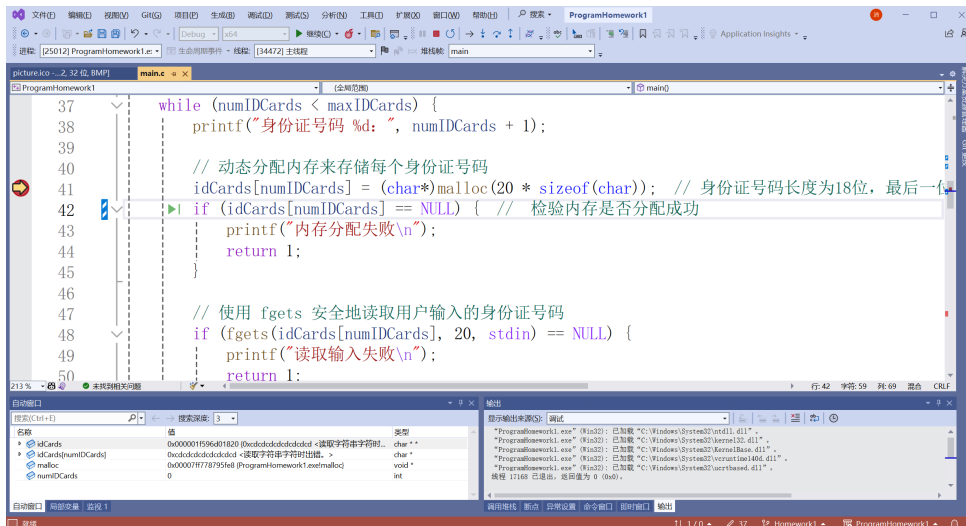


图 10 断点测试 1

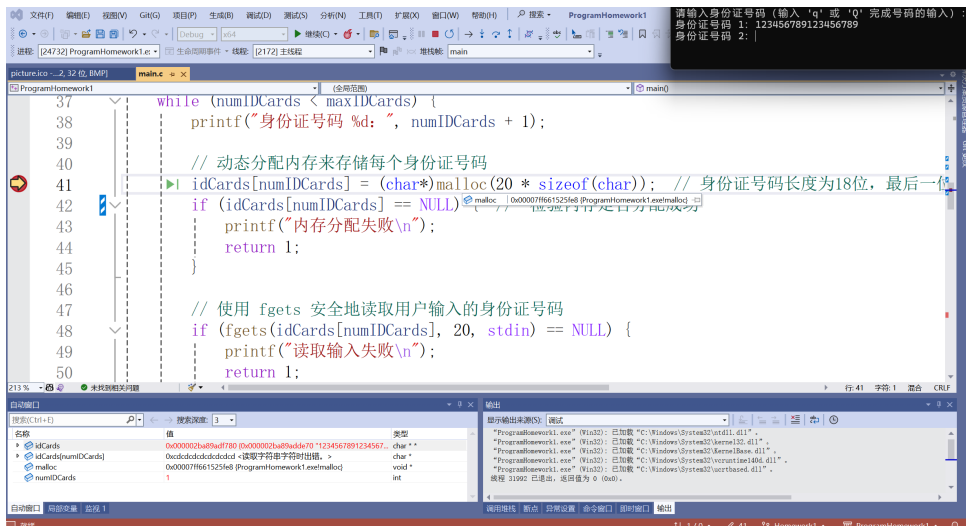


图 11 断点测试 2

3.4 测试结果

在最终版程序的测试中，所有项目测试结果均符合预期，程序测试通过。

四、使用程序的注意事项

4.1 程序的使用

为了启动该程序，用户需要打开“身份证号查验程序.exe”（如图（12）所示），之后按照提示操作即可查验用户输入的身份证号是否正确。

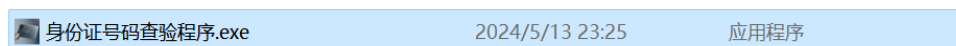


图 12 打开程序

程序使用完毕后，用户按下回车键即可退出该程序。

4.2 github 仓库

该程序的一切源代码（包括本设计文档在内）均可以在 <https://github.com/DSCJ1149270594/ProgramHomework> 的 *Homework1* 分支内下载查看。

4.3 更新时间

该设计文档和源程序最后更新时间为 2024 年 5 月 14 日 22:00

五、程序源代码（C 语言）

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

// 查验身份证号码
int validateIDCard(const char* idCard) {
    int weights[17] = { 7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2 }; //
        对身份证号码前17位数字求加权求和的权重
    char checkCodes[11] = { '1', '0', 'X', '9', '8', '7', '6', '5', '4', '3', '2' }; //
        校验码M的值
    int sum = 0;
    int i = 0;
    for (i = 0; i < 17; i++) {
        sum += (idCard[i] - '0') * weights[i]; // 计算加权和
    }

    int index = sum % 11; // 取模得到值Z
    char checkCode = checkCodes[index]; // 获取Z对应的校验码M
```

```

    return (checkCode == idCard[17]); // 返回校验结果
}

int main() {
    char** idCards = NULL; // 用于存储身份证号码的指针数组
    int numIDCards = 0;
    int maxIDCards = 10000; // 最大输入身份证号码的数量
    int len = 0; // 验证身份证号码长度
    int i = 0;
    // 动态分配内存
    idCards = (char**)malloc(maxIDCards * sizeof(char*));
    if (idCards == NULL) { // 检验内存是否分配成功
        printf("内存分配失败\n");
        return 1;
    }

    // 用户逐个输入身份证号码
    printf("请输入身份证号码 (输入 'q' 或 'Q' 完成号码的输入) : \n");
    while (numIDCards < maxIDCards) {
        printf("身份证号码 %d: ", numIDCards + 1);

        // 动态分配内存来存储每个身份证号码
        idCards[numIDCards] = (char*)malloc(20 * sizeof(char)); //
            // 身份证号码长度为18位, 最后一位为'\0'
        if (idCards[numIDCards] == NULL) { // 检验内存是否分配成功
            printf("内存分配失败\n");
            return 1;
        }

        // 使用 fgets 安全地读取用户输入的身份证号码
        if (fgets(idCards[numIDCards], 20, stdin) == NULL) {
            printf("读取输入失败\n");
            return 1;
        }

        // 去除输入中的换行符
        idCards[numIDCards][strcspn(idCards[numIDCards], "\n")] = '\0';
        len = strlen(idCards[numIDCards]);
        if (idCards[numIDCards][0] == 'q' || idCards[numIDCards][0] == 'Q') {
            break; // 用户输入 'q' 结束输入
        }
        if (len != 18) {
            printf("身份证号码长度( %d位 )不正确, 请重新输入\n", len);
            //free(idCards[numIDCards]);
            continue;
        }
    }
}

```

```

    for(i=0;i<len;i++){
        if (!(idCards[numIDCards][i] >= '0' && idCards[numIDCards][i] <='9' ||
            idCards[numIDCards][i] == 'X')) {
            printf("身份证号码包含不合法字符, 请重新输入\n");
            //free(idCards[numIDCards]);
            break;
        }
    }

    numIDCards++;
}
if(numIDCards == 0){
    printf("未输入任何身份证号码! \n");
    // 释放动态分配的内存
    for (i = 0; i < numIDCards; i++) {
        free(idCards[i]);
    }
    free(idCards);
    printf("\n请按回车键退出该程序。 \n");
    getchar(); // 等待用户按下回车键退出程序
    return 1;
}

printf("在输入的所有身份证号码中, ");
_Bool flag = true;
for (i = 0; i < numIDCards; i++) {
    if (!validateIDCard(idCards[i])) {
        flag = false;
        break;
    }
}

if (flag) {
    printf("所有身份证号码均有效。");
}else {
    printf("无效身份证号码如下: \n");
}

// 判断并输出所有无效身份证号码
for (i = 0; i < numIDCards; i++) {
    if (!validateIDCard(idCards[i])) {
        printf("%s\n", idCards[i]);
    }
}

// 释放动态分配的内存
for (i = 0; i < numIDCards; i++) {

```

```
        free(idCards[i]);
    }
    free(idCards);

    printf("\n请按回车键退出该程序。\\n");
    getchar(); // 等待用户按下回车键退出程序

    return 0;
}
```