[TOC]

崇实战队26赛季算法组第一次作业

> 相信大家经过一个多月的学习，已经感受到编程与算法的独特魅力，沉浸在这场名为算法的艺术舞台中，为了更好了解大家的学习情况，我们准备了算法组的第一次作业供大家巩固学习。
>
> 第一次作业内容相对简单 希望大家认真对待 给自己现阶段的学习一份满意的答卷

概述

这次作业主要分为两个部分，分为文本部分和编程部分。 主要考察简单的代码编程能力，以及动手操作解决问题的能力，同时为了解这段时间的学习情况，需同学们自学并使用markdown语言将第一阶段学习内容及成果做成报告提交，此外，整个作业需打包提交至自己github的仓库，将自己的仓库地址填写在群（**QQ群聊：776367199**）里的收集表中（收集表预计在作业发布后24h内发出，请及时关注后续群消息）。 作业截止时间为10.19（周日）23：30前。

ps.收集表与作业完成时间截止时间一致，请完成任务的同学一定记得在群里填写自己的仓库地址

文本部分

关于Markdown的简要介绍

> Markdown是一种轻量化的标记语言，并且排版标记简单，是一种易读易写的纯文本格式编写的文档。因简洁、高效、易读、易写等特性，Markdown被大量使用，例如在Github、Wikipedia、简书等。

当然，在初次接触Markdown语言时，不要被“标记”、“语言”等专业术语吓到，在实际使用中，Markdown的语法在1个小时内就能完全掌握并快速上手，在日常写作和记录上更是绰绰有余。

为了让大家更快掌握该语法，我检索了几个实用的教程供大家参考，可根据自己实际情况进行参考，同时鼓励大家独立查询相关资料自主学习。

- Markdown官方文档，界面简洁，便于学习，包含所有语法及使用方式，同时具有速查表，忘记某语法时可快速查找。 Markdown 官方教程
- B站up主剪的一部兼具Markdown和VScode的教程影片，时长为40min,能在短时间内掌握其使用语法，且跟着视频逐帧学习能更好的体验到Markdown的伟大之处，这里给大家一个小tips，可以在学习的过程中尝试用Markdown做笔记，也是一个不错的学习方法。 40分钟学会在VScode上使用Markdown做出好看的笔记

学习进度汇报

显然，这就是需要完成的第一部分作业。经过第一阶段的学习，相信同学们已疯狂汲取了很多新的知识，从刚开始的“Hello,world”再到循环、判断语句，以及后面的数组、指针和结构体，学习中或许也遇到了很多问题，是如何解决的又怎么避免这些问题，这些都需要记录下来，知识是无穷尽的，在漫长的学习过程中，倘若只学不记，很多小瑕疵、基础知识点都会在时间的长河中被遗忘，常常因一个小BUG致程序无法运行，又要耗费大量的时间精力去解决，致使本末倒置。 因此，第一部分的任务是需要大家使用Markdown语言完成第一阶段学

习进程的汇报，并且完成后将文档转为pdf格式，对于转格式方面有不懂的可以在网上查询，这里不再赘述。文件结构包含.md和.pdf两项。 在文件夹中展示2个简单例子，供各位参考，**详情见附件1和附件2。**

编程部分

一、输出斐波那契数列

> 了解算法的同学都知道，递归和迭代都能解决复杂的重复问题，二者有着千丝万缕的联系，同时又存在一些区别，具体体现在在实现方式、效率及可读性等方面。递归和迭代在解决问题的方式上有所不同。递归通过函数调用自身实现重复操作，而迭代则通过循环结构实现。在选择使用哪种方法时，需要考虑具体问题的需求和性能要求。例如，如果问题具有自然的递归结构，或者需要简洁的代码表示，则递归可能是更好的选择；反之，如果对性能要求较高，或者需要避免栈溢出的风险，则迭代可能是更优的选择。
>
> 可见，递归和迭代各有优缺点，适用于不同的场景，所以，理解好二者有助于实际编程中做出明智的选择。

题目背景

斐波那契数列又称黄金分割数列，是一个经典的数学序列，在自然界和计算机科学中都有广泛应用。该数列由意大利数学家莱昂纳多·斐波那契提出，用于描述兔子繁殖的数学规律。 数列定义：

- F(0) = 0
- F(1) = 1
- F(n) = F(n-1) + F(n-2) (n ≥ 2) 数列示例：0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

题目要求

编写一个完整的程序，实现以下功能：

1. 计算斐波那契数列的第n项
2. 输出斐波那契数列的前n项
3. 计算斐波那契数列的前n项和

程序接口如下：

```
// 计算第n个斐波那契数
long long fibonacci_nth(int n);

// 输出前n项斐波那契数列
void fibonacci_sequence(int n);

// 计算前n项斐波那契数列的和
long long fibonacci_sum(int n);
```

输入格式：

一个整数n（0 ≤ n ≤ 90，避免整数溢出）

输出格式：

- 第n项斐波那契数
- 前n项数列（每行输出8个数字）
- 前n项和

二、简单的排序问题

题目要求

请你编写一个C/C++程序，能够实现以下功能：

1. 生成一个包含10个随机整数的数组（随机数范围：0~99）
2. 输出排序前的原始数组
3. 使用至少两种不同的排序算法（）对数组进行升序排序
4. 分别输出每种排序算法的结果

程序接口如下：

```
// 生成随机数组
void generateArray(int arr[], int n);

// 打印数组
void printArray(int arr[], int n);

// 排序算法函数（至少实现两个，不局限于所给四种排序方式）
void bubbleSort(int arr[], int n);          // 冒泡排序
void selectionSort(int arr[], int n);      // 选择排序
void insertionSort(int arr[], int n);      // 插入排序
void quickSort(int arr[], int low, int high); // 快速排序
```

输出格式

```
原始数组: 12 45 3 78 23 56 89 43 67 1
冒泡排序: 1 3 12 23 43 45 56 67 78 89
选择排序: 1 3 12 23 43 45 56 67 78 89
```

一些小提示tips

随机数生成

1. 使用 rand() 函数生成随机数；
2. 通过 srand(time(0)) 设置随机种子；
3. 需包含头文件：<stdlib.h> 和 <time.h>。

排序算法

- 冒泡排序： 相邻元素比较交换
- 选择排序： 每次选择最小元素放到前面
- 插入排序 将元素插入到已排序序列的正确位置
- 快速排序 分治思想，选择基准元素

代码规范要求

1. 将每个排序算法封装为函数；
2. 主函数清晰调用各个算法；
3. 添加必要的注释说明。

三、学生成绩管理系统（拔高）

题目描述

请你完成一个学生成绩管理系统的程序，要求实现以下功能：

1. 输入多个学生的成绩
2. 计算所有学生的平均分
3. 查找最高分和最低分
4. 对成绩进行降序排序
5. 统计各分数段人数

功能要求

输入学生成绩

- 从键盘输入学生人数
- 依次输入每个学生的成绩（0-100分）
- 使用指针遍历数组进行输入

计算平均分

- 编写函数计算所有成绩的平均值
- 返回数据类型为double的平均分

查找最高分和最低分

- 使用指针在数组中查找最大值和最小值
- 通过指针参数返回结果

成绩排序

- 对成绩数组进行降序排序
- 使用指针操作数组元素

统计等级

- 按以下等级统计人数： 优秀：90-100分 良好：80-89分 中等：70-79分 及格：60-69分 不及格：0-59分

函数接口

```
/*
  输入学生成绩
  scores 指向成绩数组的指针
  n 学生人数
 */
void inputScores(int *scores, int n);

/*
 计算平均分
 scores 指向成绩数组的指针
 n 学生人数
 return 平均分
 */
double calculateAverage(int *scores, int n);

/*
 查找最高分和最低分
 scores 指向成绩数组的指针
 n 学生人数
 max 指向存储最高分的变量的指针
 min 指向存储最低分的变量的指针
 */
void findMinMax(int *scores, int n, int *max, int *min);

/*
 对成绩进行降序排序
 scores 指向成绩数组的指针
 n 学生人数
 */
void sortScores(int *scores, int n);

/*
 统计各等级人数
 scores 指向成绩数组的指针
 n 学生人数
 counts 指向等级统计数组的指针
 counts[0]: 优秀人数(90-100)
 counts[1]: 良好人数(80-89)
 counts[2]: 中等人数(70-79)
 counts[3]: 及格人数(60-69)
 counts[4]: 不及格人数(0-59)
 */
void countGrades(int *scores, int n, int *counts);
```

主函数框架

```
int main() {
    int numStudents;

    printf("请输入学生人数: ");
    scanf("%d", &numStudents);

    int scores[numStudents];
    int gradeCounts[5] = {0};

    // 调用各功能函数
    inputScores(scores, numStudents);

    double avg = calculateAverage(scores, numStudents);
    printf("平均分: %.2f\n", avg);

    int maxScore, minScore;
    findMinMax(scores, numStudents, &maxScore, &minScore);
    printf("最高分: %d, 最低分: %d\n", maxScore, minScore);

    sortScores(scores, numStudents);
    printf("成绩降序排列: ");
    for(int i = 0; i < numStudents; i++) {
        printf("%d ", scores[i]);
    }
    printf("\n");

    countGrades(scores, numStudents, gradeCounts);
    printf("等级统计:\n");
    printf("优秀(90-100): %d人\n", gradeCounts[0]);
    printf("良好(80-89): %d人\n", gradeCounts[1]);
    printf("中等(70-79): %d人\n", gradeCounts[2]);
    printf("及格(60-69): %d人\n", gradeCounts[3]);
    printf("不及格(0-59): %d人\n", gradeCounts[4]);

    return 0;
}
```

运行Test（可测试多组数据）

输入示例:

```
请输入学生人数: 5
请输入5个学生的成绩:
85
92
78
```

```
65
88
```

输出样例：

```
平均分：81.60
最高分：92，最低分：65
成绩降序排列：92 88 85 78 65
等级统计：
优秀(90-100)：1人
良好(80-89)：2人
中等(70-79)：1人
及格(60-69)：1人
不及格(0-59)：0人
```

关于作业提交方面

将两部分作业完成后打包在一起，并提交至github（若因其他问题导致无法进入，可提交至gitee），随后将自己所在作业的仓库地址填写至群里收集表中，收集表截止时间与作业时间一致，逾期不候。 文件结构包含：

1. 作业原题文件
2. 编程部分文件
3. 文本部分文件 包含.md和.pdf两种格式
4. 其他

为了让初学者更快了解git和github，在此依旧提供一些简单教程供大家参考，鼓励大家自己搜寻更好的资源和学习教程，可分享至群中共同进步。

- 来自广州航海学院ICEBREAKER破冰船机器人实验室的培训材料，花费1.5小时就能掌握VScode+git进行代码管理，内容简明概要，非常适合初学者学习。 vscode+git本地代码版本管理及远程仓库保存
- 通过短短20min讲明git的工作流及基本原理，并且具有常用的github基本操作，通过形象的比喻讲清git的工作流程，易于理解，讲解的也很清晰。 git基本工作流+github基本操作

结语

第一次作业相对来说比较基础，希望大家能投入一定的时间和精力，认真对待，在干中学，相信一次作业的收获会远超你前期的学习，不以完成作业为目的，量力而行，同时做到精益求精，将自己前期积累的知识转化为解决实际问题的能力，最后，愿诸位度过愉快的一周。

参与编者：刘璧洁　武晓健
通讯信息：QQ 3524679561
1836871898