Dr.Yun的学习报告

自动化2025级易长勋

学号202520613064

文本部分

一、学习内容

1.基本数据类型

- all

2.运算符与表达式

- all

3.基本输入输出

- all

4.选择结构

- all

5.循环结构

- all

6.数组

- all

7.函数

- all

8.指针

- 指针与指针变量的概念
- 指针变量的定义与应用
- 指针和地址运算
- 指针与数组
- 数组的指针和指向数组的指针变量
- 指向多维数组的指针——数组指针
  剩下的没学

9.预处理命令

- all

注：本人学习C语言的参考书是学校发的教材

二、自身感悟

- **实践是检验真理的唯一标准**
- > ***漫漫长夜之后，我们会目睹黎明***

三、后一阶段的学习目标

- 学习C++（《C++ Primer plus》）
- 学习Open CV

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

Dr.Yun的作答

```
#include <stdio.h>
double fibonacci_nth(int n);    //声明自定义函数，用于返回斐波拉契数列第n项的值
void fibonacci_sequence(int n);
double fibonacci_sum(int n);     //声明自定义函数，用于返回斐波拉契数列前n项的和
int main()                                  //主函数
{
    int n;
    double Fibonacci_n, Fibonacci_sum;
    while (1)
    {
        printf("请输入一个不大于90的正整数，以得出斐波那契数列的第n项，前n项和\n");
        scanf_s("%d", &n);
        if (n >= 0 && n <= 90)  break;
        printf("输入错误，请再次输入\n");
    }
    Fibonacci_n = fibonacci_nth(n);
    printf("斐波那契数列的第n项为%.0lf\n", Fibonacci_n);
    fibonacci_sequence(n);
    Fibonacci_sum = fibonacci_sum(n);
    printf("斐波那契数列的前n项和为%.0lf\n", Fibonacci_sum);
    return 0;
}

double fibonacci_nth(int n)          //定义函数斐波拉契数列
{
    int i = 0;
    double fibonacci_n = 0;
    double fibonacci[99];
```

```
    fibonacci[0] = 0;
    fibonacci[1] = 1;
    while (i<=n-2)
    {
        fibonacci[i + 2]=fibonacci[i + 1]+fibonacci[i];
        i++;
    }
    fibonacci_n = fibonacci[n];
    return fibonacci_n;
}
double fibonacci_sum(int n) //定义计算斐波那契数列前n项和的函数
{
    int i = 0;
    double Fibonacci_sumup = 0;
    while (i<=n)
    {
        Fibonacci_sumup += fibonacci_nth(i);
        i++;
    }
    return Fibonacci_sumup;
}

void fibonacci_sequence(int n)      //定义，输出斐波那契数列的前n项
{
    int i = 0;
    while (i <= n)
    {
        printf("F(%d)=%.0lf\n", i, fibonacci_nth(i));
        i++;
    }
    return;
}
```

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。

Dr.Yun的作答

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
#include <ctype.h>

void generateArray(int a[],int n);      // 生成随机数组
void printArray(int a[],int n);         // 打印数组
void bubbleSort(int a[],int n);         // 冒泡排序
```

```
void selectionSort(int a[],int n);      // 选择排序
void insertionSort(int a[],int n);      // 插入排序
void quickSort(int a[],int low,int high);           // 快速排序
void quickSortandPrint(int a[], int low, int high);

int main()
{
    int arr[100];
    int n;
    char ch;
    printf("请你决定生成多少个由100以内的整数组成的随机数列（上限为100）：");
    scanf_s("%d", &n);
    generateArray(arr,n);               // 生成随机数组
    printArray(arr,n);              // 打印数组
    printf("\n请你从以下4种排序方式中选择一种：\nA.冒泡排序\nB.选择排序\nC.插入排序\nD.
快速排序\n\n");
    ch=_getch( );
    ch=toupper(ch);
    if (ch = 'A')
        bubbleSort(arr, n);                     // 冒泡排序
    else if (ch = 'B')
        selectionSort(arr, n);                  // 选择排序
    else if (ch = 'C')
        insertionSort(arr, n);                  // 插入排序
    else if (ch = 'D')
        quickSortandPrint(arr, 0, n - 1);                   // 快速排序
    else
        printf("输入错误");
    return 0;
}
//生成随机数组
void generateArray(int a[],int n)
{
    int i = 0;
    srand((unsigned)time(NULL));            //生成随机种子
    for (i=0; i <n; i++)
    {
        a[i] = rand() % 100;        //生成一个随机数（0到99），并赋值给a[i]
    }
    return;
}
//打印原始数组
void printArray(int a[],int n)
{
    int i = 0;
    printf("\n原始数组：");
    for(i=0;i<n;i++)
    {
        printf("%d ", a[i]);
    }
    printf("\n");
    return;
}
//冒泡排序
```

```
void bubbleSort(int a[],int n)
{
    int i ,j,t;
    for (i = 0; i < n-1; i++)
    {
        for(j=0;j<n-1;j++)
        {
            if (a[j] > a[j+1])
            {
                t = 0;
                t = a[j];
                a[j]=a[j+1];
                a[j+1] = t;
            }
        }
    }
    printf("\n冒泡排序: ");
    for (int d=0;d<n;d++)
        printf("%d ", a[d]);
    printf("\n");
    return ;
}
//选择排序
void selectionSort(int a[],int n)
{
    int i, j, k,t;
    for (i = 0; i < n-1; i++)
    {
        k = i;
        for (j = i + 1; j < n; j++)
            if (a[k] > a[j])
                k = j;
        if (k != i)
        {
            t = 0;
            t = a[i];
            a[i] = a[k];
            a[k] = t;
        }
    }
    printf("\n选择排序: ");
    for (int d = 0; d < n; d++)
        printf("%d ", a[d]);
    printf("\n");
    return ;
}
//插入排序
void insertionSort(int a[],int n)
{
    int i, j, k;
    for (i = 1; i < n; i++)
    {
        k = a[i];
        j = i - 1;
```

```
        while ((j >= 0) && (a[j] > k))
        {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = k;
    }
    printf("\n插入排序: ");
    for (int d = 0; d < n; d++)
        printf("%d ", a[d]);
    printf("\n");
    return ;

}
//快速排序
void quickSort(int a[],int low,int high)
{

    int i = low, j = high;
    int k = a[i];
    {
        while (i < j)
        {
            while (i < j && a[j] >= k)
                j--;
            a[i] = a[j];
            while (i < j && a[i] <= k)
                i++;
            a[j] = a[i];
        }
        a[i] = k;
        if ((i - 1) > low)
            quickSort(a, low, i - 1);
        if ((j + 1) < high)
            quickSort(a, i + 1, high);
    }
    return;
}
void quickSortandPrint(int a[], int low, int high)
{
    int n;
    n = high - low + 1;
    quickSort(a, low, high);
    printf("\n快速排序: ");
    for (int d = 0; d < n; d++)
        printf("%d ", a[d]);
    printf("\n");
    return;
}
```

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
```

Dr.Yun的作答

```c
#include <stdio.h>

/*
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)
 */
```

```c
void countGrades(int* scores, int n, int* counts);



int main()
{
    int numStudents;

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

    int scores[100] = { 0 };
    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 = 0;
    int minScore = 0;

    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;
}

/*
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)
{
   ChongHui: printf("请输入这%d个学生的成绩(0~100): \n", n);
    for (int i = 0; i < n; i++)
        scanf_s("%d", &scores[i]);
```

```
    for (int d = 0; d < n; d++)
        if (scores[d] < 0 || scores[d]>100)
            goto ChongHui;
    return;
}

/*
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)
{
    double sum = 0;
    for (int i = 0; i < n; i++)
        sum += scores[i];
    double ave;
    ave = sum / n;
    return ave;
}

/*
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)
{
    *max = scores[n-1];
    *min = scores[0];

    for (int i = n - 1; i >= 0; i--)
        if (*max < scores[i])
            *max = scores[i];

    for (int j = 0; j<=n-1; j++)
        if (*min > scores[j])
            *min = scores[j];

    return;
}

/*
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)
{
    for(int i=0;i<n-1;i++)
        for(int j=i+1;j<n-1-i;j++)
            if (scores[j] < scores[j - 1])
            {
                int t = scores[j];
                scores[j] = scores[j-1];
                scores[j - 1] = scores[j];
            }
    return;
}

/*
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)
{
    for (int i = 0; i < n; i++)
        if (scores[i] >= 90)
            counts[0]++;
        else if (scores[i] >= 80)
            counts[1]++;
        else if (scores[i] >= 70)
            counts[2]++;
        else if (scores[i] >= 60)
            counts[3]++;
        else
            counts[4]++;

}
```
