```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
//运算符
void Demo11();
void Demo12();

//判断
void Demo21();
void Demo22();
void Demo23();

//简单循环
void Demo31();
void Demo32();
void Demo33();
void Demo34();

//嵌套循环
void Demo41();
void Demo42();
void Demo43();
void Demo44();

//一维数组
void Demo51();
void Demo52();
void Demo53();
void Demo54();

//多维数组
void Demo61();
void Demo62();
void Demo63();
void Demo64();

//指针
void Demo71();
void Demo72();
void Demo73();

//字符/字符串
void Demo81();
void Demo82();
void Demo83();
void Demo84();

//函数
void Demo91();

//结构体
void Demo101();
```

```c
//其他
void extraDemo1();
void extraDemo2();


// 求个十百位
void Demo11() {
    //--变量声明--
    // 输入值
    int a;
    // 个十百位
    int r1, r2, r3;


    //--接收输入--
    scanf_s("%d", &a);

    //--数据处理--
    /*
        a%10表示取个位数
        a/10将数字缩小10倍    也就是十位数变成个位数
    */
    r1 = a % 10;
    r2 = a / 10 % 10;
    r3 = a / 100;
    //--输出--
    printf("%d\n%d\n%d", r3, r2, r1);
}

// 求正方形和圆形的面积差
void Demo12() {

    //--变量声明--
    // PI
    const double PI = 3.14;
    // 输入值
    int l;
    // 正方形面积, 圆形面积.
    double squ_s, cir_s;
    // 结果
    double result;

    //--接收输入--
    scanf_s("%d", &l);

    //--数据处理--
    /*
        正方形 = l^2
        圆形 = PI*r^2
    */
    squ_s = l * l;
    cir_s = l * l * PI / 4;
    result = squ_s - cir_s;
```

```
    //--输出--
    printf("%.2lf", result);
}

//最大数
void Demo21() {
    //--变量声明--
    //输入值
    int a, b, c;
    //最大值
    int max;

    //--接收输入--
    scanf_s("%d %d %d", &a, &b, &c);

    //--数据处理--
    max = a;
    if (b > max) {
        max = b;
    }
    if (c > max) {
        max = c;
    }

    //--输出--
    printf("%d", max);
}


// 打分系统
void Demo22() {
    //--变量声明--
    // 输入
    int n;
    // 输出
    int result;

    //--接收输入--
    scanf_s("%d", &n);

    //--数据处理--
    if (n <= 10) result = n * 6;
    else if (n <= 20) result = 10 * 6 + (n - 10) * 2;
    else result = 10 * 6 + 10 * 2 + (n - 20) * 1;

    //--输出--
    printf("%d", result);
}

// 上班
void Demo23() {
    //--变量声明--
    // 输入
    int n;
    // 负数表示骑车快，正数表示走路快，0表示一样快
```

```
    double result;

    //--接收输入--
    scanf_s("%d", &n);

    //--数据处理--
    //用骑车速度减去走路速度  如果是正数表示骑车耗时多
    result = (27 + 23 + n / 3.0) - (n / 1.2);

    //--输出--
    // 注意double判断会有精度问题  不能直接判断==0
    if (result < 0.0001) printf("骑车");
    else if (result > 0.0001) printf("走路");
    else printf("一样快");
}

// 求平均年龄
void Demo31() {
    //--变量声明--
    // 人数, 临时记录变量, 学长的年龄总和
    int n, temp = 0;
    double age = 0;

    //--接收输入--
    // 注意这里age+=temp   也就是存储了所有学长年龄的和
    scanf_s("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf_s("%d", &temp);
        age += temp;
    }

    //--数据处理--
    age /= n;

    //--输出--
    printf("%.2lf", age);
}

// 求2222222
void Demo32() {
    //--变量声明--
    // 输入n, 输入a, 累加数存储变量, 答案存储变量
    int n, a;
    int num, result = 0;

    //--接收输入--
    scanf_s("%d %d", &n, &a);
    num = a;

    //--数据处理--
    /*
        num用于存储a,aa,aaa这样的数字

        num = aaa
        num*10 = aaa0
```

```
        num*10+a = aaaa
    */
    for (int i = 0; i < n; i++) {
        result += num;
        num = num * 10 + a;
    }


    //--输出--
    printf("%d", result);

}

// 数兔子
void Demo33() {
    //--变量声明--
    // 用户输入值  月数
    int n;
    // 用于存储当月 以及前1,2个月的兔子数
    int m1 = 1, m2 = 1, m3 = 0;;

    //--接收输入--
    scanf_s("%d", &n);

    //--数据处理--
    //前两个月不生兔子
    // 每个月的兔子数量   =  上一个月兔子数 + 上上一个月兔子数
    n -= 2;
    while (n--) {
        m3 = m1 + m2;
        m1 = m2;
        m2 = m3;
    }

    //--输出--
    printf("%d", m3);
}

// 弹球
void Demo34() {
    //--变量声明--
    // 初始条件
    int N, M;
    // 高度和距离
    double h, l = 0;

    //--接收输入--
    scanf_s("%d %d", &M, &N);
    h = M;
    //--数据处理--
    while(N--){
        h /= 2;
        l += h * 3;
    }
```

```
    //--输出--
    printf("%.2lf, %.2lf", h, l);

}


// 乘法表
void Demo41() {
    //--变量声明--

    //--接收输入--

    //--数据处理--

    //--输出--
    for (int i = 1; i <= 9; i++) {
        for (int j = 1; j <= i; j++) {
            printf("%d * %d = %d\t", j, i, i * j);
        }
        printf("\n");
    }

}


// 求素数
void Demo42() {
    //--变量声明--
    // 输入值
    int N;
    // 标识符  值为1表示素数  值为0表示非素数
    int flag;
    //--接收输入--
    scanf_s("%d", &N);
    //--数据处理--
    //--输出--
    for (int i = 2; i <= N; i++) {
        flag = 0;
        for (int j = 2; j < i/*sqrt(i)*/; j++) {
            // 如果i%j为0  则该数字是非素数   0表示假  !取反为真
            if (!(i % j)){
                flag = 1;
                break;
            }
        }
        //如果flag是0  表示是素数 打印
        if (!flag) printf("%d\n", i);
    }


}


// 猜数字
```

```c
void Demo43() {
    //--变量声明--
    // 用户输入 让程序猜的数字
    int n;
    // 程序猜测的次数, 二分猜测法的上下限
    int count = 0, max = 100, min = 0;
    //猜测值
    int mid;

    //--接收输入--
    scanf_s("%d", &n);
    do{
        //--数据处理--
        /*
            猜测值 = (上限 + 下限) /2
            如上限100  下限0   猜测值=50
            上限100  下限50  猜测值=75

            如果猜测值比n小,  则下限 = 猜测值+1
            如果猜测值比n大,  则上限 = 猜测值-1
        */
        count++;
        mid = (max + min) / 2;
        if (mid > n) {
            max = mid - 1;
        }
        else if (mid < n) {
            min = mid + 1;
        }

        //--输出--
        printf("%d\n", mid);

    } while (mid != n);
    printf("最终猜测了%d次", count);

}

// 打印菱形
void Demo44() {
    //--变量声明--
    //菱形边长
    int n;

    //--接收输入--
    scanf_s("%d", &n);

    //--数据处理--
    /*
        用坐标系的思路
        设坐标点(x,y)
        当x的绝对值 + y的绝对值  小于n时,  这个坐标点在菱形内
        当x的绝对值 + y的绝对值  大于等于n时, 这个坐标点在菱形外

        如: (2,2)在菱形内, (3,2)在菱形外, (-4,0)在菱形内, (4,1)在菱形外
```

```
    */
    //--输出--
    // 当n为5时，坐标系的上下限分别是-4~4  最大高/宽度为9
    for (int i = -n+1; i < n; i++) {
        for (int j = -n+1; j < n; j++) {

            // i的绝对值+j的绝对值 小于n  则在菱形范围内
            if (abs(i) + abs(j) < n) {
                printf("*");
            }
            else {
                printf(" ");
            }
        }

        printf("\n");
    }

}

// 去重
void Demo51() {
    //--变量声明--
    //数据总数，去重后的数量
    int N, len;
    //待去重的数据
    int arr[100];

    //--接收输入--
    scanf_s("%d", &N);
    len = N;
    for (int i = 0; i < N; i++) {
        scanf_s("%d", &arr[i]);
    }
    //--数据处理--
    /*
        将每个元素都和后面的元素进行判断
            如果[i]和[j]值重复，将[j]的值设为-1.  并且len--
        {1,3,2,6,-1,-1,4,8,-1,-1}

        都判断一遍后，将值为-1的元素  用后一个元素值进行覆盖
        {1,3,2,6,4,8}
    */
    for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
            if (arr[i] == -1) break;

            if (arr[i] == arr[j]) {
                arr[j] = -1;
                len--;
            }
        }
    }

    /*
```

```
        快慢指针思路

        i表示慢下标，j表示快下标
        快下标：每次循环往后移动一位
        慢下标：条件符合时往后移动一位

        每次循环时，arr[i] = arr[j]
        慢下标条件：覆盖后，如果当前[i]的值不为-1  则i++(往后移动一位)

        下面样例的截取点是在arr[i] = arr[j]后  i++前

        ij
        1        2        -1        3        -1        4        -1        5
    6

                ij
        1        2        -1        3        -1        4        -1        5
    6

                        ij
        1        2        -1        3        -1        4        -1        5
    6

                        i        j
        1        2        3        3        -1        4        -1        5
6

                                i        j
        1        2        3        -1        -1        4        -1        5
    6

                                i                j
        1        2        3        4        -1        4        -1        5
6

                                        i                j
        1        2        3        4        -1        4        -1        5
6

                                        i                        j
        1        2        3        4        5        4        -1        5
6

                                                i                        j
        1        2        3        4        5        6        -1        5
6
    */
    for (int i = 0, j = 0;
        j < N;
        j++)
    {

        arr[i] = arr[j];
        if (arr[i] != -1) {
            i++;
```

```
        }

    }


    //--输出--
    for (int i = 0; i < len; i++) {
        printf("%d ", arr[i]);
    }

}


// 数字环
void Demo52() {
    //--变量声明--
    // 数字环存储数组，长度，移动位数
    int arr[100], len, m;
    // 转换后数组
    int result[100];


    //--接收输入--
    scanf_s("%d", &len);
    for (int i = 0; i < len;i++) {
        scanf_s("%d", &arr[i]);
    }
    scanf_s("%d", &m);


    //--数据处理--
    for (int i = 0; i < len; i++) {
        result[(i + m) % len] = arr[i];
    }


    //--输出--
    for (int i = 0; i < len; i++) {
        printf("%d ", result[i]);
    }

}

//3. 合并有序数组
void Demo53() {
    //--变量声明--
    // 存放有序的数组，存放合并后的数组
    int nums1[100], nums2[100], sorted[200];
    // 数组的下标，用于合并数组
    int p1 = 0, p2 = 0;
    // 当前用于加入新数组的元素
    int cur;
    // 数字的数量
    int n, m;
    //--接收输入--
```

```
    scanf_s("%d %d", &n, &m);
    for (int i = 0; i < n; i++) {
        scanf_s("%d", &nums1[i]);
    }
    for (int i = 0; i < m; i++) {
        scanf_s("%d", &nums2[i]);
    }

    //--合并处理--
    // 遍历数组，使用cur保存要并入新数组的值
    while (p1 < n || p2 < m) {
        if (p1 == n) {
            cur = nums2[p2++];
        }
        else if (p2 == m) {
            cur = nums1[p1++];
        }
        else if (nums1[p1] < nums2[p2]) {
            cur = nums1[p1++];
        }
        else {
            cur = nums2[p2++];
        }
        sorted[p1 + p2 - 1] = cur;
    }

    //--输出--
    for (int i = 0; i < m+n; i++) {
        printf("%d ", sorted[i]);
    }
}

// 约瑟夫环
void Demo54() {
    //--变量声明--
    // 玩家数量，被枪毙数，玩家当前状况
    // 值为1表示存活  值为0表示被枪毙了
    int n, m, arr[100];
    // 目前活着的人数，flag用于标记当前数到几
    int num, flag = 0;

    //--接收输入--
    scanf_s("%d %d", &n, &m);
    num = n;
    for (int i = 0; i < n; i++) {
        arr[i] = 1;
    }

    //--数据处理--
    // 当num值为1时，游戏结束
    for (int i = 0; num - 1; i++) {
        if (i == n) i = 0;

        if (arr[i]) {
            flag++;
```

```
            if (flag == m) {
                arr[i] = 0;
                flag = 0;
                num --;
            }
        }
    }


    //--输出--
    for (int i = 0; i < n; i++) {
        if (arr[i]) printf("%d", i + 1);
    }

}


// 矩阵转置
void Demo61() {
    //--变量声明--
    // 矩阵边长，矩阵本身，转置后数组
    int size, arr[10][10], result[10][10];
    //--接收输入--
    scanf_s("%d", &size);
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            scanf_s("%d", &arr[i][j]);
        }
    }
    //--数据处理--
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            result[j][i] = arr[i][j];
        }
    }


    //--输出--
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }

}


// 颈椎病
void Demo62() {
    //--变量声明--
    // 矩阵边长，矩阵本身，转置后数组
    int size, arr[10][10], result[10][10];
    //--接收输入--
    scanf_s("%d", &size);
```

```
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            scanf_s("%d", &arr[i][j]);
        }
    }

    //--数据处理--
    /*
        假设size为3
        [0][0]->[0][2]
        [0][1]->[1][2]
        [1][2]->[2][1]
        [2][1]->[1][0]
        规律 x转y    y转x
        x转y时 size-x=y
        y转x时 y=x
    */
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            result[j][size - i - 1] = arr[i][j];
        }
    }


    //--输出--
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
}

// 杨辉三角
void Demo63() {
    //--变量声明--
    // 杨辉三角的高度
    int n;
    // 存储用数组，  注意杨辉三角只会占用一半的空间，所以这里可以用malloc动态分配来避免空间浪
费
    int arr[10][10];

    //--接收输入--
    scanf_s("%d", &n);
    arr[0][0] = 1;

    //--数据处理--
    // 左右两侧数据直接填充即可
    for (int i = 1; i < n; i++) {
        arr[i][0] = arr[i][i] = 1;
        for (int j = 1; j < i; j++) {
            arr[i][j] = arr[i - 1][j - 1] + arr[i - 1][j];
        }
    }
    //--输出--
```

```
    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= i; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
}



// 包围圈
void Demo64() {
    //--变量声明--
    int size, tree[10][10];
    // 当前坐标xy, 移动方向toward 0左 1下 2右 3上, 填充数
    // 注意 由于第一次探查前会先移动 x++  所以x需要修正为-1
    int x = -1, y = 0, toward = 0, num = 0;
    // 一个方向的移动步数
    int s;

    //--接收输入--
    scanf_s("%d", &size);
    s = size;

    //--数据处理--
    // 移动规律  如果是4  则是4+3+3+2+2+1+1 = 16  同理 如果是5  5+4+4+3+3+2+2+1+1
    // 外部循环: 每奇数次循环,  最大步数-1.  一个方向移动完后 对方向进行修改
    // 内部循环: 循环(步数)次, 每次移动根据方向对xy进行修改

    for (int i = 0; num<size*size; i++) {
        // 奇数次循环 最大步数-1
        if (i % 2 == 1) s--;
        for (int j = 0; j < s; j++) {
            // 获取方向并移动
            switch (toward) {
            case 0:
                x++;
                break;
            case 1:
                y++;
                break;
            case 2:
                x--;
                break;
            case 3:
                y--;
                break;
            }

            tree[y][x] = ++num;
        }

        //  改变方向  注意 当toward为3  也就是向上时, 需要取模操作 也就是变为0
        // (1+1) %4 = 2   (3+1) %4=0
        toward = (toward + 1) % 4;
```

```
    }


    //--输出--
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            printf("%d ", tree[i][j]);
        }
        printf("\n");
    }
}

void Demo71() {

    //--变量声明--
    //接受输入字符
    char inch;
    //先开辟单个空间
    char str[100] = {0};
    //定义指向字符串的指针
    char* ptr = str;

    //最大单词长度
    int maxLen = 0;
    //最大单词起始位置
    int tag = 0;


    //--接收输入--
    //循环读取输入，直到输入结束符或达到最大长度
    int i;
    for (i = 0; (inch = getchar()) != '.' && i <100 ; i++)
    {
        str[i] = inch;
    }
    len = i;

    //--数据处理--
    //如果没到末尾 则继续循环
    for (; *ptr != '\0'; ptr++) {
        //用于存储当前单词长度
        int tempCount = 0;
        //如果没遇到空格 则长度+1
        for (; *ptr != ' ' && *ptr != '\0';ptr++) tempCount++;

        //遇到空格后  判断当前长度和最大长度谁大  如果当前长度大 重新赋值最大长度
        //并记录当前位置
        if (maxLen < tempCount) {
            maxLen = tempCount;
            tag = ptr - str - tempCount;
        }
    }

    //--输出--
    printf("%d %d", tag, maxLen);
```

```
}

//2. 李四翻笔记时打翻了墨水，但他忘记了这段之前写的什么内容了. 小伙伴们快帮帮他吧.
//用正确的代码替换[墨水]
void Demo72(){}
void swap(int* a, int* b) {
    //--数据处理--
    //--定义一个临时变量，然后交换两个指针指向的整数值--
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
int main() {
    //--变量声明--
    int a = 3, b = 4;
    //--数据处理--
    swap(&a, &b);
    //--输出--
    printf("%d,%d\n", a, b); //此处应打印4,3
}

//3. 王五在写代码时遇到了问题，但他不知道是怎么回事，大家快帮帮他吧.
void Demo73(){}
int* fun() {
    //--变量声明--
    int* a = (int*)malloc(sizeof(int) * 10);
    //--变量赋值--
    for (int i = 0; i < 10; i++)
    {
        a[i] = i;
    }
    return a;//返回动态申请的空间是可以的.
}

int main()
{
    //--变量声明--
    int* b = fun();
    int* c = (int*)malloc(16); // include <stdlib.h>
    //--输出--
    printf("%d\n", b[0]);
}

// 大小写转换
void Demo81() {
    //--变量声明--
    // 存放字符串的数组
    char str[100];

    //--接收输入--
    gets_s(str, 100);

    //--数据处理--
    // 判断ascii码是否在大小写范围内  如果在  则手动转换
    for (int i = 0; str[i] != '\0'; i++) {
```

```
        if (str[i] >= 65 && str[i] <= 90) {
            str[i] += 32;
        }
        else if (str[i] >= 97 && str[i] <= 122) {
            str[i]-= 32;
        }
    }

    //--输出--
    printf("%s", str);
}

// 字符串反转
void strReplace(char* cp, int n, char* str) {
    int lenth = 0;
    int i = 0;
    char* tmp;
    //获取替换子串长度
    lenth = strlen(str);

    //如果需要查找的子串长度大于替换的子串 需要往前移动
    if (lenth < n) {
        tmp = cp + n;
        while (*tmp)
        {
            //n - lenth 是移动的距离
            *(tmp - (n - lenth)) = *tmp;
            tmp++;
        }
        //移动'\0'
        *(tmp - (n - lenth)) = *tmp;
    }
    //否则替换子串大于查找的子串长度 需要往后移动
    else
    {
        if (lenth > n) {
            tmp = cp;
            while (*tmp)
            {
                tmp++;
            }
            while (tmp >= cp + n)
            {
                *(tmp + (lenth - n)) = *tmp;
                tmp--;
            }
        }
    }
    //进行替换
    strncpy(cp, str, lenth);
}

void Demo82() {
    // 用来接受主串
    char str1[1024];
```

```
    //接受查找子串，和替换内容
    char str2[100], str3[100];
    //保存字符长度，当前输入下标以及替换次数
    int i, len, count = 0;
    //用来接收主串输入
    char c;
    //用来接收strstr()结果
    char* p;
    printf("请依次输入查找串和替换串，空格或者回车隔开：\n");
    scanf("%s", str2);
    scanf("%s", str3);
    i = 0;

    //获取主串
    c = getchar();
    while ((c = getchar()) != '\n'&& c != EOF)
    {
        str1[i] = c;
        i++;
    }
    //结尾加上结束符
    str1[i] = '\0';

    //用strstr()来判断主串中是否有子串，有的话才会执行替换操作
    p = strstr(str1, str2);
    while (p)
    {
        count++;
        //执行替换操作
        strReplace(p, strlen(str2), str3);

        //替换后的主串地址
        p = p + strlen(str3);
        //再用新地址判断是否主串中还存在子串
        p = strstr(p, str2);
    }
    printf("替换完成后的字符串为：%s", str1);
}


// 安全密码
void Demo83() {
    //--变量声明--
    // 用户输入
    char password[100];
    // 用于验证是否合法，0:长度，1:开头大写，2:包含小写，3:包含数字，4:包含特殊符号
    int flag[5] = { 0 };

    //--接收输入--
    gets_s(password,100);

    //--数据处理--
    //开头大写判断
    if (password[0] >= 65 && password[0] <= 90) flag[0] = 1;
```

```
    for (int i = 1; password[i] != '\0';i++) {
        //小写字母
        if (password[i] >= 97 && password[i] <= 122) flag[2] = 1;
        //数字
        else if (password[i] >= 48 && password[i] <= 57) flag[3] = 1;
        //特殊符号
        else if (password[i] == '~' ||
            password[i] == '!' ||
            password[i] == '@' ||
            password[i] == '#' ||
            password[i] == '$' ||
            password[i] == '%' ||
            password[i] == '*'
            ) flag[4] = 1;

        // 长度
        flag[1] = (i >= 8 && i <= 16);
    }

    //--输出--
    if (flag[0] && flag[1] && flag[2] && flag[3] && flag[4]) {
        printf("true");
    }
    else {
        printf("false");
    }
}



// 加密
void Demo84() {
    //--变量声明--
    char password[17];

    //--接收输入--
    gets_s(password, 16);

    //--数据处理--
    /*
        思路: 先减到0~25  然后+5取模
        如果超过25了(字母xyz), 则会因为取模成bcd

        比如z
        90 - 65=25
        25 + 5 = 30  %26 = 4
        4+65 = 69  = e;

        数字同理
    */
    for (int i = 0; password[i] != '\0'; i++) {
        //大写字母
        if (password[i] >= 65 && password[i] <= 90) {
            password[i] = (password[i] - 65 + 5) % 26 + 65;
        }
```

```
        //小写字母
        else if (password[i] >= 97 && password[i] <= 122) {
            password[i] = (password[i] - 97 + 5) % 26 + 97;
        }
        //数字
        else if (password[i] >= 48 && password[i] <= 57) {
            password[i] = (password[i] - 48 + 5) % 10 + 48;

        }
    }

    //--输出--
    printf("%s", password);
}

//1. 根据以下要求, 实现一套登录功能
void Demo91() {}
int ids[5] = { 10001,10002,10003,10004 };
char names[5][10] = {
    "张三", "李四", "王五", "赵六"
};
char passwords[5][16] = { "aaaaa","bbbbb","ccccc","ddddd" };
int uNum = 4;

char* selectUserById(int uid);
char* selectPassById(int uid);
int login(int uid, char* password);
void showLoginPage();

int main() {
    showLoginPage();
    return 0;
}

/*
        功能: 根据id 查询用户是否存在, 如果存在返回用户名,  如果不存在返回空
        参数:
            uid: 用户id
        返回值:
            如果用户存在, 返回用户名.
            如果用户不存在, 返回NULL
    */
char* selectUserById(int uid) {
    for (int i = 0; i < uNum; i++) {
        if (ids[i] == uid) {
            return names[i];
        }
    }
    return NULL;
}

/*
        功能: 根据id 查询用户密码, 如果存在返回用户密码,  如果不存在返回空
        参数:
            uid: 用户id
```

```
        返回值:
            如果密码存在, 返回密码.
            如果密码不存在, 返回NULL
    */
char* selectPassById(int uid) {
    for (int i = 0; i < uNum; i++) {
        if (ids[i] == uid) {
            return passwords[i];
        }
    }
    return NULL;
}


/*
        功能: 传入用户id和密码, 根据上面两个函数(selectUserById, selectPassById)来获取相应用户数据, 并判断是否登录成功
            传入用户id 查询用户名是否存在, 并获取用户密码
            如果用户存在  则判断密码是否正确
        参数:
            uid: 用户账户
            password: 用户密码
        返回值:
            如果账号不存在, 返回1
            如果密码错误, 返回2
            如果登录成功, 返回0
    */
int login(int uid, char* password) {
    if (selectUserById(uid) == NULL) {
        return 1;
    }
    char* pass = selectPassById(uid);
    if (strcmp(password, pass) != 0) {
        return 2;
    }
    return 0;

}

/*
        功能: 提示用户输入账号密码, 根据login函数判断是否登录成功,
        如果登录成功提示正在进入加载界面
        如果登录失败
            密码错误: 提示密码错误, 并让用户重新登录
            账号不存在: 提示账号不存在, 并提示正在进入注册界面
        参数: 无
        返回值: 无
    */
void showLoginPage() {
    //--变量声明--
    //用户id
    int uid;
    //用户密码
    char password[17];
    //结果选项
```

```
    /*    如果账号不存在，返回1
            如果密码错误，返回2
            如果登录成功，返回0
            */
    int result;

    //--接收输入--
    scanf_s("%d", &uid);
    //处理掉输入的回车符号
    getchar();
    gets_s(password, 16);

    //--数据处理--
    result = login(uid, password);

    //--输出--
    switch (result) {
        case 1:
            printf("账号不存在，正在进入注册界面");
            break;
        case 2:
            printf("密码错误，请重新登录");
            break;
        case 0:
            printf("登录成功，正在进入首页");
            break;
    }

}


//实现链表结构
void Demo101() {}
//--结构体定义与变量声明--

struct Node {
    int val; //当前节点存储的值
    Node* next; //下一个节点
};
typedef Node* List; // List表示链表本身，List的next是头节点 也就是第一个节点


/*
功能：创建一个链表
参数：void
返回值：返回一个新的空链表
*/
List createList() {
    //创建一个头结点，表示链表的起始结点
    List head = (List)malloc(sizeof(Node));
    head->next = NULL;
    head->next = 0;
    return head;
}
/*
```

```
功能: 新建一个节点，放到链表的头节点 (原本的头节点成为第二个节点)
参数:
List: 链表
val: 新节点的值
返回值: void
*/
void addAtHead(List list, int val) {
    Node* toAdd = (Node*)malloc(sizeof(Node));
    toAdd->next = list->next;
    toAdd->val = val;
    list->next = toAdd;
}
/*
功能: 新建一个节点，放到链表的尾节点 (原本的尾节点成为倒数第二个节点)
参数:
List: 链表
val: 新节点的值
返回值: void
*/
void addAtTail(List list, int val) {
    Node* pred = list;
    while (pred->next != NULL) {
        pred = pred->next;
    }
    Node* toAdd = (Node*)malloc(sizeof(Node));
    toAdd->next = NULL;
    toAdd->val = val;
    pred->next = toAdd;

}
/*
功能: 新建一个节点，放到链表的第index个节点处 (原本的第index个节点成为第index+1个节点)
参数:
List: 链表
index: 新节点的位置
val: 新节点的值
返回值: void
*/
void addAtIndex(List list, int index, int val) {
    //检查插入位置是否合理
    if (index < 0) {
        return;
    }
    //插入新的结点

    Node* pred = list;
    for(int i = 0; i < index-1; i++) {
        pred = pred->next;
        //检查插入位置是否合理
        if (pred == NULL) return;
    }
    Node* toAdd = (Node*)malloc(sizeof(Node));
    toAdd->next = pred->next;
    toAdd->val = val;
    pred->next = toAdd;
```

```
}
/*
功能: 删除第index个节点(原本的第index+1个节点成为第index个节点)
参数:
List: 链表
int: 被删除的节点
返回值: void
*/
void deleteAtIndex(List list, int index) {
    if (index < 0) {
        return;
    }
    //先找到要删除结点的上一个结点

    Node* pred = list;
    for (int i = 0; i < index-1; i++) {
        pred = pred->next;
    }
    //删除当前结点
    Node* p = pred->next;
    pred->next = pred->next->next;
    free(p);
}
/*
功能: 打印这条链表
参数: void
返回值: void
*/
void SListPrint(List list) {
    //略过头结点
    Node* p = list->next;
    //打印结点元素值
    while (p != NULL)
    {
        printf("%d ", p->val);
        p = p->next;
    }
    printf("\n");
}

// 设计一个绘图程序.
void extraDemo1() {}
//#include<graphics.h>
//#include<stdio.h>

//将四个颜色 红，粉，黑，白存入数组
int colors[] = { RED,LIGHTRED,BLACK,WHITE };
//定义colorID作为colors的下表，修改colorID即可更改选择的颜色
int colorID = 0;
//定义brushZise表示笔刷的大小
int brushSize = 10;
//四个颜色选择框的左边位置和间隔
int left = 450, spacing = 50;

//初始化画板界面函数
```

```
void initSketchpad();
//刷新颜色选择框函数
void refreshBrush();
//刷新笔刷大小和颜色函数
void refreshColor();

int main() {
    //先进入画板初始化函数
    initSketchpad();

    //创建消息结构体
    ExMessage m;
    //创建坐标(x,y)
    short x = 0, y = 0;
    while (1) {
        //m接收消息，包括鼠标和键盘消息
        m = getmessage(EX_MOUSE | EX_KEY);
        //根据消息类型进入switch选择
        switch (m.message) {

            //如果是鼠标左键按下
            case WM_LBUTTONDOWN:
                //如果当前鼠标坐标y>60，即不在笔刷、颜色状态栏内
                if (m.y > 60) {
                    //设置线：实线、线宽brushSize
                    setlinestyle(PS_SOLID, brushSize);
                    //根据当前鼠标坐标和选择的颜色绘制点
                    putpixel(m.x, m.y, colors[colorID]);
                }
                //更新点坐标(x,y)
                x = m.x;
                y = m.y;
                break;
                //如果是鼠标移动
            case WM_MOUSEMOVE:
                //如果鼠标左键点击（结合case来看，效果就是点击左键并拖动）
                if (m.lbutton) {
                    //如果当前坐标y>60并且保存的上一个点y>60（即两个点都不在笔刷、颜色状态
栏内）
                    if (m.y > 60 && y > 60) {
                        //设置线：当前选择的颜色colorID，实线，线宽brushSize
                        setlinecolor(colors[colorID]);
                        setlinestyle(PS_SOLID, brushSize);
                        //将上一个点和当前点用直线连接
                        line(x, y, m.x, m.y);
                    }
                    //更新点坐标(x,y)
                    x = m.x;
                    y = m.y;
                }
                break;
                //如果是键盘输入
            case WM_KEYDOWN:
                //如果是A a
                if (m.vkcode == 'A' || m.vkcode == 'a') {
```

```
                    //如果colorID此时为0，则colorID减一后为-1，所以设置colorID为3
                    if (colorID-- == 0)
                        colorID = 3;
                    //刷新颜色选择框
                    refreshColor();
                }
                //如果是D d
                if (m.vkcode == 'D' || m.vkcode == 'd') {
                    //如果colorID此时为3，则colorID加一后为4，所以设置colorID为0
                    if (colorID++ == 3)
                        colorID = 0;
                    //刷新颜色选择框
                    refreshColor();
                }
                //如果是Q q
                if (m.vkcode == 'Q' || m.vkcode == 'q') {
                    //如果brushSize此时为1，则brushSize减一后为0，所以设置brushSize仍为
1
                    if (brushSize-- == 1)
                        brushSize = 1;
                    //刷新笔刷大小和颜色
                    refreshBrush();
                }
                //如果是E e
                if (m.vkcode == 'E' || m.vkcode == 'e') {
                    //如果brushSize此时为25，则brushSize加一后为26，所以设置brushSize
仍为25
                    if (brushSize++ == 25)
                        brushSize = 25;
                    //刷新笔刷大小和颜色
                    refreshBrush();
                }
                break;
            }
        }
    }

//初始化画板界面函数
void initSketchpad() {
    //创建窗口800*600
    initgraph(800, 600);
    //设置背景色为白色
    setbkcolor(WHITE);
    //用背景色清空屏幕
    cleardevice();

    //设置线：黑、实线
    setlinecolor(BLACK);
    setlinestyle(PS_SOLID);
    //画线
    line(0, 60, 800, 60);

    //进入刷新颜色选择框函数
    refreshColor();
```

```
    //设置文字：黑、长50，宽自适应，楷体
    settextcolor(BLACK);
    settextstyle(50, 0, "楷体");
    //绘制QEAD四个字母
    outtextxy(100, 5, 'Q');
    outtextxy(235, 5, 'E');
    outtextxy(left - 65, 5, 'A');
    outtextxy(left + spacing * 3 + 70, 5, 'D');

    //设置填充：黑
    setfillcolor(BLACK);
    //绘制文字旁边的四个三角形
    int pts1[] = { 130,30,150,40,150,20 };
    solidpolygon((POINT*)pts1, 3);
    int pts2[] = { 230,30,210,40,210,20 };
    solidpolygon((POINT*)pts2, 3);
    int pts3[] = { left - 30,30,left - 10,40,left - 10,20 };
    solidpolygon((POINT*)pts3, 3);
    int pts4[] = { left + spacing * 3 + 60 ,30,left + spacing * 3 + 40,40,left +
spacing * 3 + 40,20 };
    solidpolygon((POINT*)pts4, 3);
}

//刷新笔刷大小和颜色函数
void refreshBrush() {
    //用背景色清空圆形笔刷区域
    clearcircle(180, 30, 28);
    //设置线：青色、实线、线宽3
    setlinecolor(CYAN);
    setlinestyle(PS_SOLID, 3);
    //设置填充：根据colorID选择填充的颜色
    setfillcolor(colors[colorID]);
    //绘制有边框的圆形，圆形半径根据brushSize确定
    fillcircle(180, 30, brushSize);
}

//刷新颜色选择框函数
void refreshColor() {
    //用背景色清空颜色选择框区域
    clearrectangle(left - 3, 12, left + spacing * 3 + 33, 48);

    //绘制四个不同的颜色，正方形30*30
    setfillcolor(colors[0]);
    solidrectangle(left + spacing * 0, 15, left + spacing * 0 + 30, 45);
    setfillcolor(colors[1]);
    solidrectangle(left + spacing * 1, 15, left + spacing * 1 + 30, 45);
    setfillcolor(colors[2]);
    solidrectangle(left + spacing * 2, 15, left + spacing * 2 + 30, 45);
    setfillcolor(colors[3]);
    solidrectangle(left + spacing * 3, 15, left + spacing * 3 + 30, 45);

    //设置线：青色、实线、线宽3
    setlinecolor(CYAN);
    setlinestyle(PS_SOLID, 3);
    //设置填充：根据colorID选择填充的颜色
```

```
    setfillcolor(colors[colorID]);
    //绘制有边框的矩形覆盖原来的无边框矩形
    fillrectangle(left + spacing * colorID, 15, left + spacing * colorID + 30,
45);
    //进入刷新笔刷大小和颜色函数
    refreshBrush();
}


//文件操作
void extraDemo1() {}
#include<stdio.h>
#include<windows.h>
#define MAXLEN 10
#define SUC 1
#define DEF -1

/*
    * 链表的实现
    */
typedef struct userPoint
{
    int ranking;
    char username[20];
    int point;
    struct userPoint* pnext;

}USER_T;

/*
    * 打开文件函数。如果文件存在则直接打开，如果不存在则创建
    */
FILE* f_open(const char* name) {
    FILE* fp = NULL;
    fp = fopen(name, "r+");
    if (fp == NULL) {
        fp = fopen(name, "w+");
    }
    return fp;
}

/*
    * 链表的长度
    */
int listCount(USER_T* head) {
    int count = 0;
    USER_T* temp = head;
    while (temp->pnext != NULL)
    {
        count++;
        temp = temp->pnext;
    }
    return count;
}
```

```
/*
    * 删除链表队尾节点
    */

int deleteBackNode(USER_T* head) {
    USER_T* temp = head->pnext;
    USER_T* pre = head;

    while (temp != NULL)
    {
        if (temp->pnext == NULL) {
            pre->pnext = temp->pnext;
            free(temp);
            return SUC;
        }
        pre = pre->pnext;
        temp = temp->pnext;
    }
    return DEF;
}

/*
    * 链表的初始化
    */
USER_T* listInit() {
    USER_T* head = NULL;
    head = (USER_T*)malloc(sizeof(USER_T));
    if (head == NULL) {
        printf("初始化链表失败……");
        exit(0);
    }
    memset(head, '\0', sizeof(USER_T));
    head->pnext = NULL;
    return head;
}

/*
    * 增加链表中的节点，如果超出10个节点则删除最后一个
    */
void listAdd(USER_T* head, USER_T node) {
    if (listCount(head) == MAXLEN) {
        if (deleteBackNode(head) != SUC) {
            printf("删除节点失败……");
            exit(0);
        }
    }
    USER_T* temp = head;
    USER_T* pNewNode = NULL;
    int tag = 1;
    while (temp->pnext != NULL)
    {
        if (temp->pnext->point < node.point) {
            break;
        }
        tag++;
```

```
        temp = temp->pnext;
    }
    pNewNode = (USER_T*)malloc(sizeof(USER_T));
    if (pNewNode == NULL) {
        printf("创建节点失败……");
        exit(0);
    }
    memset(pNewNode, '\0', sizeof(USER_T));
    memcpy_s(pNewNode, sizeof(USER_T), &node, sizeof(USER_T));
    pNewNode->pnext = temp->pnext;
    temp->pnext = pNewNode;
    pNewNode->ranking = tag;
    temp = pNewNode->pnext;
    while (temp != NULL)
    {
        temp->ranking++;
        temp = temp->pnext;
    }

}

/*
    * 链表的释放
    */
void freeList(USER_T* head) {
    USER_T* temp = head;
    while (temp->pnext != NULL)
    {
        head = head->pnext;
        free(temp);
        temp = head;
    }
    free(temp);
}

/*
    * 文件写入
    */

void File_write(FILE* pf, USER_T* head, int nodeSize) {
    USER_T* temp = head;
    int count = listCount(temp);
    pf = f_open("Rank.txt");
    if (pf == NULL) {
        printf("创建或打开文件失败……");
        exit(0);
    }
    while (temp->pnext != NULL)
    {
        fwrite(temp->pnext, nodeSize, 1, pf);
        fflush(pf);
        temp = temp->pnext;
    }
}
```

```
/*
    * 通过读取到的文件数据来创建链表
    */

USER_T* creatListByFlie(FILE* pf) {
    USER_T* head = NULL;
    USER_T* pRead = NULL;
    pRead = (USER_T*)malloc(sizeof(USER_T));
    memset(pRead, '\0', sizeof(USER_T));
    head = listInit();
    pf = f_open("Rank.txt");
    rewind(pf);
    fread(pRead, sizeof(USER_T), 1, pf);
    while (feof(pf) == 0) {
        listAdd(head, *pRead);
        fread(pRead, sizeof(USER_T), 1, pf);
    }
    return head;
}

void main() {
    int cho = 0;
    FILE* fp = NULL;
    USER_T* head = NULL;
    USER_T* pRead = NULL;
    pRead = (USER_T*)malloc(sizeof(USER_T));
    memset(pRead, '\0', sizeof(USER_T));
    head = listInit();

    fp = f_open("Rank.txt");

    if (fp == NULL) {
        printf("创建或打开文件失败……");
        exit(0);
    }
    USER_T* pNode = NULL;
    while (true)
    {
        printf("1.录入\n");
        printf("2.保存&显示\n");
        printf("3.读取\n");
        printf("4.退出\n");
        scanf_s("%d", &cho);
        switch (cho)
        {
            case 1:
                pNode = NULL;
                pNode = (USER_T*)malloc(sizeof(USER_T));
                if (pNode == NULL) {
                    printf("节点生成失败……");
                    exit(0);
                }
                memset(pNode, '\0', sizeof(USER_T));
                printf("请输入姓名：");
                scanf("%s", pNode->username);
```

```
                printf("请输入分数：");
                scanf("%d", &pNode->point);
                listAdd(head, *pNode);
                system("cls");
                break;
            case 2:
                File_write(fp, head, sizeof(USER_T));
                rewind(fp);
                fread(pRead, sizeof(USER_T), 1, fp);
                printf("名词\t姓名\t积分\n");
                while (feof(fp) == 0) {
                    printf("%d\t%s\t%d\n", pRead->ranking, pRead->username,
pRead->point);
                    fread(pRead, sizeof(USER_T), 1, fp);
                }
                break;
            case 3:
                head = creatListByFlie(fp);
                break;
            case 4:
                freeList(head);
                exit(0);
                break;
            default:
                printf("输入有误，请重新输入……");
                break;
        }
        fclose(fp);
    }
}
```