《计算机程序设计》实验报告

姓名： 宋亦寒 **学号：** PB25000200 **实验日期：2025 年 11 月 17 日**

实验名称	C 程序设计入门

一、 实验目的要求

1. 熟练运用 C 语言的控制语句与基本语法；
2. 掌握数组的查找、插入和排序；
3. 尝试实现较为复杂的算法流程。

二、 实验内容

1. 实验指导书 3.1.2 程序填空练习 2 移位编码；
2. 实验指导书 3.1.3 自主编程练习 2；
3. 实验指导书 3.1.3 自主编程练习 5；
4. 实验指导书 3.2.3 自主编程练习 2；
5. 实验指导书 3.1.3 自主编程练习 4；
6. 采用主函数随机生成 6 个不同的数字，并编写两个函数：

函数 1：对数组进行排序（升序）

```
void sort(float a[ ], int n);
```

函数 2：计算出数组中最接近且小于等于所有元素平均值的元素值下标

```
int search(float a[ ], int n);
```

如： a[6]={0.2, 0.3, 0.5, 0.7, 0.9,1.2}返回 0.5 所在的下标 2。

要求通过主函数对两个功能函数(函数声明已给出)的调用，输出结果。

选做题 1：非对称加密问题

问题描述：

不对称的计算：RSA（三位提出者姓氏的首字母）公开密钥密码体制设计了一种不对称的加密密钥与解密密钥，从而达到“由已知加密密钥推导出解密密钥在计算上是不可行的”目标。其思路是寻求两个大素数比较简单，而将它们的乘积进行因式分解却极其困难。RSA 算法的设计与应用过程描述如下：

（1）任意选取两个不同的大素数 p 和 q，计算乘积 n=pq， $\phi(n)=(p-1)(q-1)$；

（2）任意选取一个大整数 e（通常是一个大于 p 和 q 的素数），满足 gcd (e, $\phi(n)$)=1，将 e 用做加密密钥；

（3)寻找整数 d,使其满足(de)mod $\phi(n)$=1,即,使其满足(de)mod $\phi(n)$=1,即 de=k $\phi(n)$+1,,k 是≥1 的任意整数。

（4）把（n, e）公开作为公钥，而 d 不公开，（n, d）组成私钥；

（5）将小于 n 的整数 m（称为明文）加密成密文 c，加密算法为 c=E(m)=$m^e \mod n$；

（6）将密文 c 解密为明文 m，解密算法为 m=$D(c)=c^d \mod n$；

在当前的计算能力下，只根据 n 和 e（p 和 q 不公开）要计算出 d 几乎是不可能的。因此，任何人都能对明文进行加密并在网络上公开传输，而只有知道 d 的授权用户才能对密文进行解密。

要求：编程完成上述加密、解密算法，输入整数明文 m，输出密文 c 与解密明文 m。

撰写实验报告，格式为 学号-姓名-实验 6.pdf，在上机平台提交

三、 上机程序（有注释）

1. 实验指导书 3.1.2 程序填空练习 2 移位编码

```
#include<stdio.h>
#include<math.h>
unsigned  long  numEncode(unsigned  num){
        unsigned  long  bitnum=0,newnum=0,i=0;
        while(num!=0){
                bitnum=num%10;//获得当前数的最低位数值
                if(bitnum==num&&bitnum==9)//当前数值为最高位并且数值为 9
                {
                        newnum+=9*pow(10,i);
                        i++;
                }
                bitnum=(bitnum+1)%10;//对获取的数值按规则编码
                newnum+=bitnum*pow(10,i++);
                num/=10;
        }
        return newnum;
}

int  main(void){
        unsigned  origin_num;
        unsigned  long  new_num;
        printf("请输入一个正整数：\n");
        scanf("%u",&origin_num);
        new_num=numEncode(origin_num);
        printf("%u 编码为%lu",origin_num,new_num);
        return  0;
}
```

2. 实验指导书 3.1.3 自主编程练习 2

```
#include<stdio.h>
int search(float a[], int n){
    float sum = 0.0;

    // 计算总和
    for (int i = 0; i < n; i++) {
        sum += a[i];
    }

    float average = sum / n;  // 求平均值
```

```
    // 查找满足条件的位置
    for (int i = 0; i < n; i++) {
        if (a[i] <= average && a[i + 1] > average) {
            return i;
        }
    }
}

void sort(float a[], int n){
    // 选择排序
    for (int i = 0; i < n - 1; i++) {
        int min_index = i;
        for (int j = i + 1; j < n; j++) {
            if (a[j] < a[min_index]) {
                min_index = j;  // 找到更小值及其下标
            }
        }

        // 交换
        float temp = a[min_index];
        a[min_index] = a[i];
        a[i] = temp;
    }
}

int main(){
    float a[6] = {0};
    int n = 6;
    int i;

    // 输入
    for (i = 0; i < 6; i++) {
        scanf("%f", &a[i]);
    }

    sort(a, 6);   // 排序

    // 输出排序后的数组
    for (i = 0; i < 6; i++) {
        printf("%f ", a[i]);
    }

    // 输出 search 结果
```

```
        printf("\n%d", search(a, 6));
}
```

3. 实验指导书 3.1.3 自主编程练习 5

```
#include <stdio.h>
int gcd(int a, int b)
{
    // 确保 a ≥ b，方便后续求余
    if (a<b){
        int temp=a;
        a=b;
        b=temp;
    }

    // 若整除，则 b 为最大公约数
    if (a%b==0) return b;
    // 若 b 已为 1，则最大公约数为 1
    else if (b==1) return 1;
    // 否则递归继续求 gcd
    else return gcd(a%b,b);
}

int main()
{
    int a, b;
    scanf("%d %d", &a, &b);

    if (a<=0 || b<=0)
        printf("分子分母必须为正整数");
    else if (gcd(a,b)==1)
        // 最大公约数为 1：即最简分数
        printf("%d/%d 为最简分数", a, b);
    else{
        // 用最大公约数化简
        int c=a/gcd(a,b);
        int d=b/gcd(a,b);
        printf("%d/%d=%d/%d", a, b, c, d);
    }
}
```

4. 实验指导书 3.2.3 自主编程练习 2

```
#include <stdio.h>
int bitsum(int n)
{
```

```
    int sum=0;
    while (n!=0){
        sum+=n%10;   // 取个位相加
        n/=10;       // 去掉最低位
    }
    return sum;      // 返回各位数字之和
}

int main()
{
    int x;
    scanf("(x=)%d", &x);   // 按格式读取形如 "(x=)12" 的输入

    int i;
    int count=0;           // 统计符合条件的三位数个数
    int a[1000];           // 用来存储符合条件的三位数

    // 枚举 100~999 的所有三位数
    for (i=100; i<1000; i++){
        if (bitsum(i)==x){   // 如果数字之和等于 x
            a[count]=i;      // 保存这个数
            count++;         // 数量加一
        }
    }

    // 输出所有满足条件的三位数，每行最多 5 个，且避免尾部多逗号
    for (int k=0; k<count; k++){
        if (k%5==4){                        // 每满 5 个换行
            printf("%d\n", a[k]);
        } else if (k==count-1){             // 最后一个数字后不加逗号
            printf("%d\n", a[k]);
        } else {
            printf("%d, ", a[k]);          // 其他数字后加逗号和空格
        }
    }

    // 输出总数
    printf("符合各位上数字之和为%d 的整数个数：%d", x, count);
}
```

5. 实验指导书 3.1.3 自主编程练习 4

```
#include <stdio.h>
int isPrime(int n)
{
```

```
        int flag=0;    // flag=0 表示当前认为是质数，flag=1 表示不是质数
        if (n==2 || n==3) flag=0;    // 2 和 3 是质数
        else{
            // 从 2 到 sqrt(n) 试除
            for (int i=2; i*i<=n; i++){
                if (n%i==0){         // 能整除，说明不是质数
                    flag=1;
                    break;
                }
            }
        }
        return flag;   // 返回 0 表示质数，1 表示非质数
    }

    int main()
    {
        unsigned int n;
        scanf("%d", &n);

        // 遍历所有可能的 i，使得 n = i + (n - i)
        for (int i=2; i*2<=n; i++){
            // isPrime(i)==0 表示 i 是质数
            // isPrime(n-i)==0 表示 n-i 也是质数
            if (isPrime(i)==0 && isPrime(n-i)==0){
                printf("%u=%d+%d\n", n, i, n-i);    // 输出 Goldbach 分解
            }
        }
}
```

6.补充题

```
    #include <stdio.h>
    void matrix(int A[][100], int B[][100], int C[][100], int m, int p, int n) {
        for (int i=0; i<m; i++) {          // 遍历结果矩阵 C 的行
            for (int j=0; j<n; j++) {      // 遍历结果矩阵 C 的列
                C[i][j]=0;                 // 初始化 C[i][j]
                for (int k=0; k<p; k++) {  // 按定义计算矩阵乘法
                    C[i][j]+=A[i][k]*B[k][j];
                }
            }
        }
    }

    int main() {
        int m1, n1, m2, n2;
```

```
        // 输入矩阵 A 的行列数
        scanf("%d %d", &m1, &n1);
        // 输入矩阵 B 的行列数
        scanf("%d %d", &m2, &n2);

        // 判断矩阵是否可以相乘
        if (n1!=m2) return 0;   // 如果不能相乘，直接退出

        int A[100][100];  // 存储矩阵 A
        int B[100][100];  // 存储矩阵 B
        int C[100][100];  // 存储结果矩阵 C

        // 输入矩阵 A 的元素
        for (int i=0; i<m1; i++)
            for (int j=0; j<n1; j++)
                scanf("%d", &A[i][j]);

        // 输入矩阵 B 的元素
        for (int i=0; i<m2; i++)
            for (int j=0; j<n2; j++)
                scanf("%d", &B[i][j]);

        // 调用矩阵乘法函数
        matrix(A, B, C, m1, n1, n2);

        // 输出结果矩阵 C
        for (int i=0; i<m1; i++) {
            for (int j=0; j<n2; j++) {
                printf("%d ", C[i][j]);
            }
            printf("\n");
        }
}
```

7.选做题

```
    #include <stdio.h>
    #include <math.h>

    #define P 13
    #define Q 23 // 定义两个素数 P 和 Q
    #define N 299 // RSA 模数 N = P * Q
    #define E 5 // 公钥指数 E
    #define PHI_N 264 // 欧拉函数 φ(N) = (P-1)*(Q-1)
```

```
    int main()
    {
        // 私钥 d
        // d * E ≡ 1 (mod PHI_N)
        // 对于 E=5, PHI_N=264, 简化为 d = (k*PHI_N + 1)/E =（取 k=1）= 53
        int d = (1 * PHI_N + 1) / 5;

        int m1 = 110;    // 明文 m1

        // 加密操作：c = m^E mod N
        // 使用 pow 函数计算 m^E
        // 注意：pow 返回 double，转换为 long long int
        long long int c1 = ((long long int)pow(m1, E)) % N;

        int c2 = 2;    // 示例密文 c2

        // 解密操作：m = c^d mod N
        long long int m2 = ((long long int)pow(c2, d)) % N;

        printf("%lld\n%d\n%lld", c1, d, m2);
}

//计算 a^b mod n（快速幂算法）
```

- 任意整数 b 可以写成二进制：

$$b = b_k 2^k + b_{k-1} 2^{k-1} + ... + b_0 2^0$$

- 那么：

$$a^b = a^{b_0 2^0} \cdot a^{b_1 2^1} \cdot .... \cdot a^{b_k 2^k}$$

- 如果 $b_i = 0$，对应的幂可以省略；如果 $b_i = 1$，则乘上当前幂。

举例：

$$a^{13} = a^{1101_2} = a^8 \cdot a^4 \cdot a^1$$

```
//原理：
    long long result = 1;          // 最终结果
    long long base = a % n;        // 当前幂，初始化为 a^1 mod n

    while(b > 0) {
        if(b % 2 == 1) {           // 当前二进制位是 1
            result = (result * base) % n;
        }
        base = (base * base) % n; // 当前幂平方，准备下一位
        b /= 2;                    // 处理下一位
    }
return (int)result;
```

四、 调试中的问题及解决方法（字数不限）

1. pow(a,b)表示 a^b, 返回 double 类型，应该使用（int）进行类型转换；当指数较大或者结果超过 int 范围时，使用（long long int）。
2. 在写矩阵乘法函数时应该这样写：void matrix(int A[][100], int B[][100], int C[][100], int m, int p, int n)，如果写 void matrix(int A[][])会出现报错。原因是：访问 a[i][j] 的地址公式为：地址(a[i][j])=起始地址(a)+i×(列数)+j，需要确定列数。
3. 需要注意一些格式控制符使用：
   unsigned int 应该使用%u
   unsigned long int 应该使用%lu
   long long int 应该使用%lld