《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 10 月 27 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 2.3.3 自主编程练习 12；
2.实验指导书 2.3.3 自主编程练习 14；
3.实验指导书 2.3.3 自主编程练习 17；
4.实验指导书 2.4.3 自主编程练习 4；
5.实验指导书 2.4.3 自主编程练习 7；
6.输出 100-200 之间不能被 7 整除的数，每输出 5 个数换一行，并画出流程图；
7.输出数组（数值和字符）的最大值以及它的下标位置；
8.画出 2.3.3 自主编程练习的 12 题和 14 题的流程图

选做题 1：求解全排列问题

撰写实验报告，格式为 学号-姓名-实验 3.pdf，在上机平台提交

三、 上机程序（有注释）

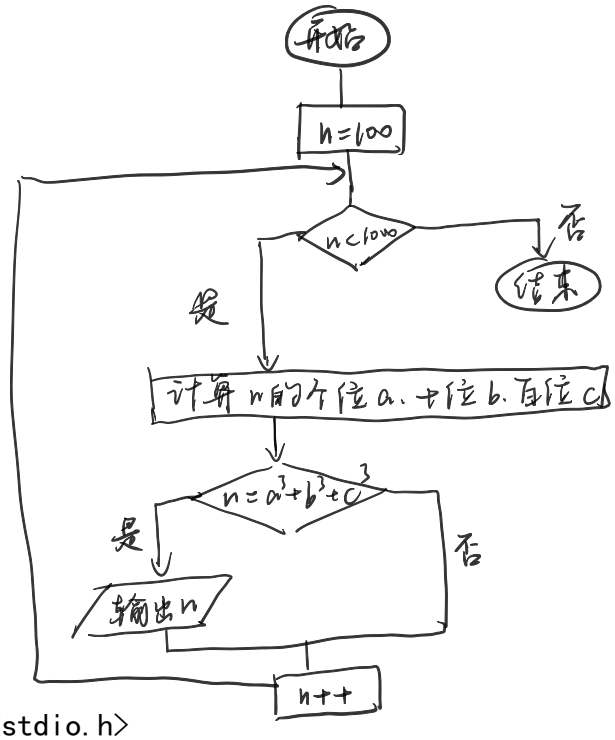
1：

```
    #include <stdio.h>

    int main()
    {
        for (int n = 100; n < 1000; n++) {
            int a = n % 10;                     // 取个位
            int b = n / 10 % 10;                // 取十位
            int c = n / 100;                    // 取百位

            if (n == a*a*a + b*b*b + c*c*c)     // 判断是否为水仙花数
                printf("%d ", n);
        }

        return 0;
}
```

```c
#include <stdio.h>
int main()
{
   for (int n=100; n<1000; n++){
    int a = n%10;
    int b = n/10%10;
    int c = n/100;

    if (n == a*a*a + b*b*b + c*c*c){
        printf("%d ", n);
        }
    }
    return 0;
}
```

```
C:\Windows\System32\cmd.e
153 370 371 407
Press any key to continue . . .
```

流程图：



2:

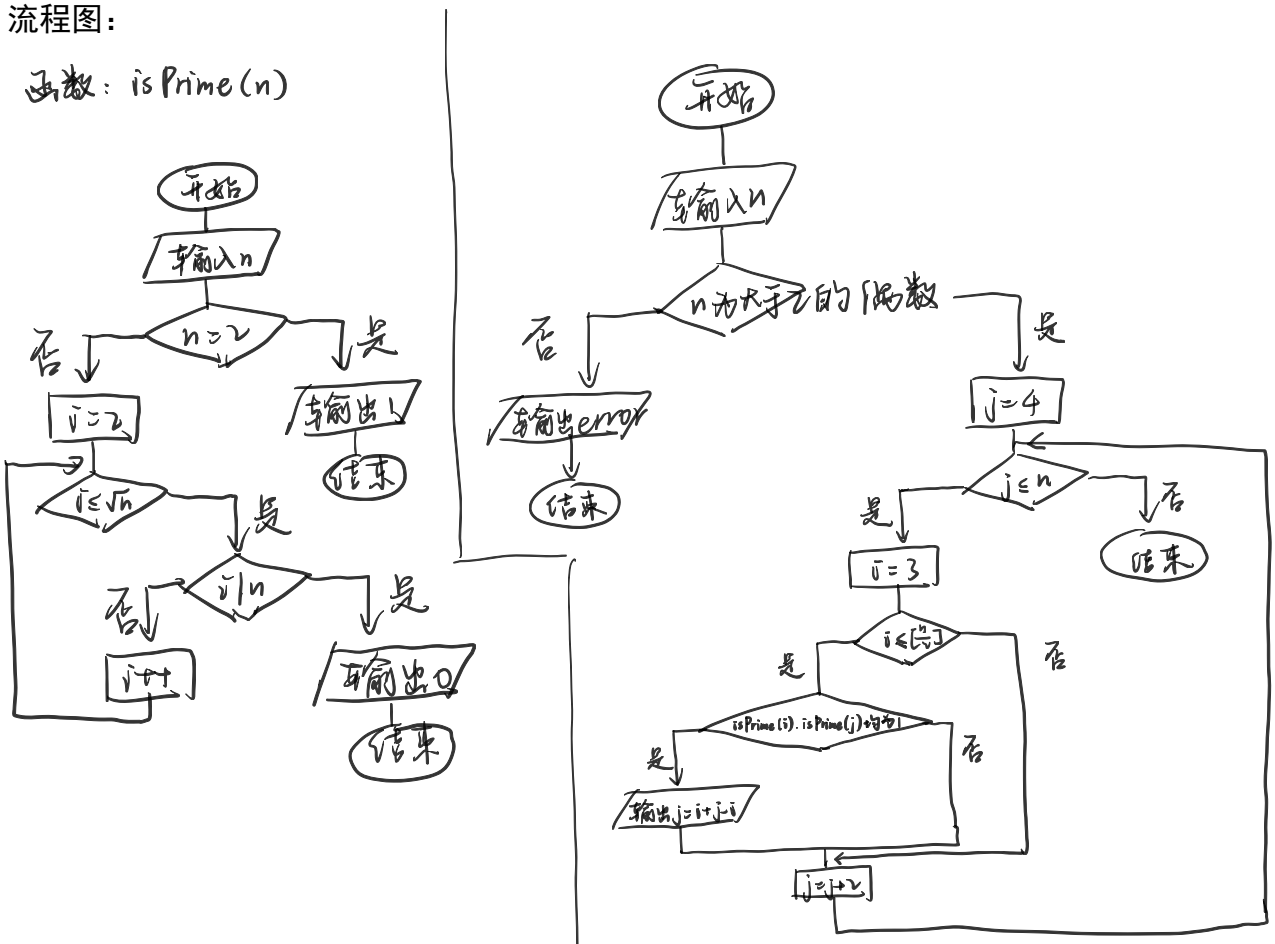
```c
#include <stdio.h>

// 判断一个整数 n 是否为素数（质数）
// 返回 1 表示是素数，返回 0 表示不是素数
int isPrime(int n)
{
    if (n == 2) return 1;
    else {
        for (int i = 2; i*i <= n; i++) {
            if (n % i == 0) return 0;      // 能被整除，则不是素数
        }
        return 1;                          // 否则是素数
    }
}
```

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

        if (n % 2 != 0 || n < 4)
            printf("error");
        else {
            for (int j = 4; j <= n; j = j + 2) {      // 枚举所有小于等于 n 的偶
数
                for (int i = 2; i*2 <= j; i++) {     // 枚举 j 的一半以内的数
                    if (isPrime(i) && isPrime(j-i))  // 判断 i 和 j-i 是否都是素
数
                        printf("%d=%d+%d\n", j, i, j-i); // 输出偶数分解成两个素
数
                }
            }
        }

        return 0;  // 程序结束
}
```

```
#include <stdio.h>
int isPrime(int n)
{
    if (n==2) return 1;
    else{
        for (int i=2; i*i<=n; i++){
            if (n%i==0) return 0;
        }
        return 1;
    }
}

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

    if (n%2!=0 || n<4) printf("error");
    else {
        for (int j=4; j<=n; j=j+2){
            for (int i=2; i*2<=j; i++){
                if (isPrime(i) && isPrime(j-i)){
                    printf("%d=%d+%d\n", j, i, j-i);
                }
            }
        }
    }

    return 0;
}
```

```
C:\Windows\System32\cmd.e
30
4=2+2
6=3+3
8=3+5
10=3+7
10=5+5
12=5+7
14=3+11
14=7+7
16=3+13
16=5+11
18=5+13
18=7+11
20=3+17
20=7+13
22=3+19
22=5+17
22=11+11
24=5+19
24=7+17
24=11+13
26=3+23
26=7+19
26=13+13
28=5+23
28=11+17
30=7+23
30=11+19
30=13+17
```

流程图：



3:

```
    #include <stdio.h>

    int main()
    {
        double n;
        scanf("%lf", &n);

        if (n > 0) {
            int i = 0;
            double m = n;                   // 保存原始输入，用于计算平方
            while (i < 1000) {              // 迭代 1000 次
                n = n - (n*n - m) / n / 2.0;  // 牛顿迭代法求平方根
                i++;
            }
        }
        else return 0;                      // 输入非正数，程序直接结束

        printf("%f", n);                    // 输出平方根近似值
        return 0;
}
```

```c
#include <stdio.h>
int main()
{
    double n;
    scanf("%lf", &n);

    if (n > 0){
        int i = 0;
        double m = n;
        while (i < 1000){
            n = n - (n*n - m)/n/2.0;
            i++;
        }
    }
    else return 0;

    printf("%f", n);
    return 0;
}
```

```
C:\Windows\System32\cmd.e
3
1.732051
Press any key to continue . . .
```

4:

```c
#include <stdio.h>

int main()
{
    int a[11];  // 声明一个长度为 11 的数组，存储 10 个初始整数和插入的新数
    printf("输入 10 个整数:");
    for (int i = 0; i < 10; i++){
        scanf("%d", &a[i]);  // 输入前 10 个整数
    }

    int x, y;
    printf("\n 输入插入到数组中的位置（0~10)和要插入的数据:");
    scanf("%d %d", &x, &y);  // 输入插入位置和要插入的数

    for (int j = 10; j > x; j--){
        a[j] = a[j-1];           // 将插入位置及之后的元素依次向后移动一位
    }
    a[x] = y;                    // 在指定位置插入新元素

    printf("\n");
    for (int k = 0; k < 11; k++){
        printf("%d ", a[k]);     // 输出插入后的数组
    }
    return 0;
```

}

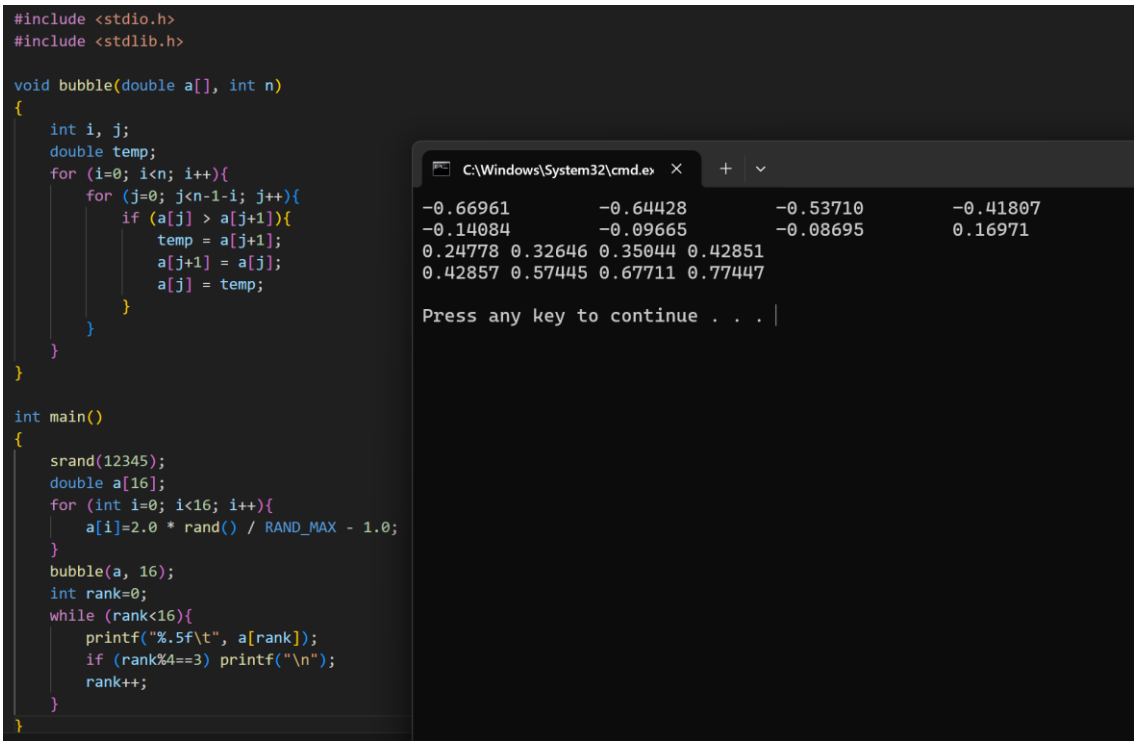
```c
#include <stdio.h>
int main()
{
    int a[11];
    for (int i=0; i<10; i++){
        scanf("%d", &a[i]);
    }

    int x, y;
    scanf("%d %d", &x, &y);

    for (int j=10; j>x; j--){
        a[j]=a[j-1];
    }
    a[x]=y;

    for (int k=0; k<11; k++){
        printf("%d ", a[k]);
    }
    return 0;
}
```

```
C:\Windows\System32\cmd.e
1 3 5 7 9 2 4 6 8 10
0 666
666 1 3 5 7 9 2 4 6 8 10
Press any key to continue . . .
```

5：

```c
#include <stdio.h>
#include <stdlib.h>

// 冒泡排序将数组 a 按升序排序
void bubble(double a[], int n)
{
    int i, j;
    double temp;
    for (i = 0; i < n; i++){
        for (j = 0; j < n-1-i; j++){          // 每趟将最大的元素改到末尾
            if (a[j] > a[j+1]){
                temp = a[j+1];                // 交换相邻两个元素
                a[j+1] = a[j];
                a[j] = temp;
            }
        }
    }
}

int main()
{
    srand(12345);                             // 设置随机数种子 12345
    double a[16];
    for (int i = 0; i < 16; i++){
        a[i] = 2.0 * rand() / RAND_MAX - 1.0;  // 生成 [-1, 1] 区间的随机小
```

```
数
        }

        bubble(a, 16);                          // 对数组 a 进行升序排序

        int rank = 0;
        while (rank < 16){
            printf("%.5f\t", a[rank]);          // 输出排序后的数组，保留 5 位小
数
            if (rank % 4 == 3) printf("\n");    // 每输出 4 个元素换行
            rank++;
        }
}
```



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

void bubble(double a[], int n)
{
    int i, j;
    double temp;
    for (i=0; i<n; i++){
        for (j=0; j<n-1-i; j++){
            if (a[j] > a[j+1]){
                temp = a[j+1];
                a[j+1] = a[j];
                a[j] = temp;
            }
        }
    }
}

int main()
{
    srand(12345);
    double a[16];
    for (int i=0; i<16; i++){
        a[i]=2.0 * rand() / RAND_MAX - 1.0;
    }
    bubble(a, 16);
    int rank=0;
    while (rank<16){
        printf("%.5f\t", a[rank]);
        if (rank%4==3) printf("\n");
        rank++;
    }
}
```

```
C:\Windows\System32\cmd.e×
-0.66961        -0.64428        -0.53710        -0.41807
-0.14084        -0.09665        -0.08695        0.16971
0.24778 0.32646 0.35044 0.42851
0.42857 0.57445 0.67711 0.77447

Press any key to continue . . .
```

6：

```
#include <stdio.h>
#include <stdlib.h>
// 将 value 插入到已排序数组 a 的正确位置，保证数组仍然有序
void insert(int a[], int n, int value)
{
    int i;
    for (i = n-1; i >= 0 && a[i] > value; i--){  // 从后往前寻找插入位置
        a[i+1] = a[i];                           // 元素后移
    }
    a[i+1] = value;                              // 插入 value
}
```

```
    // 二分查找函数，查找 target 在已排序数组 a 中的位置
    // 找到返回下标，找不到返回 -1
    int BinarySearch(int a[], int n, int target)
    {
        int left = 0, right = n-1;
        while (left <= right){
            int mid = (left + right)/2;              // 取中间下标
            if (a[mid] == target) return mid;        // 找到目标
            else if (a[mid] < target) left = mid+1; // 目标在右半部分
            else right = mid-1;                      // 目标在左半部分
        }
        return -1;                                   // 未找到
    }

    int main()
    {
        int seed;
        scanf("%d", &seed);
        srand(seed);                                 // 设置随机数种子

        int a[20] = {0}, n = 0;
        for (int i = 0; i < 20; i++){
            int num = rand() % 101;                  // 生成 0~100 的随机数
            insert(a, n, num);                       // 插入到有序数组中
            n++;
        }

        int i = 0;
        while (i < 20){
            printf("%4.d", a[i]);                    // 输出数组元素，宽度 4
            if (i % 5 == 4) printf("\n");            // 每行输出 5 个元素
            i++;
        }

        for (int j = 0; j < 3; j++){
            int key;
            scanf("%d", &key);                       // 输入要查找的值
            if (BinarySearch(a, n, key) == -1)       // 查找结果
                printf("Not found\n");
            else
                printf("%d\n", BinarySearch(a, n, key)); // 有则输出下标
        }
}
```

```c
void insert(int a[], int n, int value)
{
    int i;
    for (i=n-1; i>=0&&a[i]>value; i--){
        a[i+1]=a[i];
    }
    a[i+1]=value;
}

int BinarySearch(int a[], int n, int target)
{
    int left=0, right=n-1;
    while (left<=right){
        int mid = (left+right)/2;
        if (a[mid]==target) return mid;
        else if (a[mid]<target) left = mid+1;
        else right = mid-1;
    }
    return -1;
}

int main()
{
    int seed;
    scanf("%d", &seed);
    srand(seed);
    int a[20]={0}, n=0;
    for (int i=0; i<20; i++){
        int num = rand() % 101;
        insert(a, n, num);
        n++;
        }
    int i=0;
    while (i<20){
        printf("%4.d", a[i]);
        if (i%5==4) printf("\n");
        i++;
    }
    for (int j=0; j<3; j++){
        int key;
        scanf("%d", &key);
        if (BinarySearch(a, n, key) == -1)
```

```
C:\Windows\System32\cmd.ex
12345
   5   6   9  10  11
  15  17  17  37  40
  40  41  49  54  60
  71  73  74  75  85
5
0
6
1
7
Not found

Press any key to continue . . .
```

7：

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int seed;
    unsigned int a[20];
    scanf("%d", &seed);
    srand(seed);                        // 设置随机数种子

    for (int i = 0; i < 20; i++){
        a[i] = rand();                  // 生成随机无符号整数
```

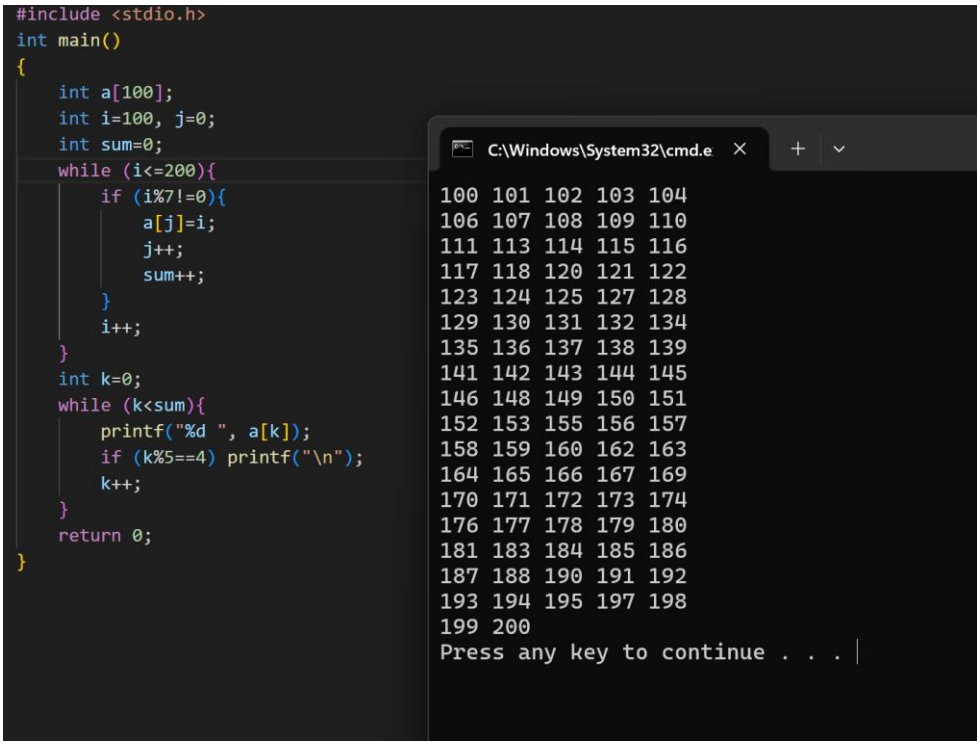
```
    }

    unsigned int max = 0;
    int max_i = 0;
    for (int i = 0; i < 20; i++){
        if (a[i] > max){            // 找到数组中的最大值
            max = a[i];
            max_i = i;              // 保存最大值的下标
        }
    }

    int i = 0;
    while (i < 20){
        printf("%u\t", a[i]);       // 输出数组元素
        if (i % 5 == 4) printf("\n"); // 每行输出 5 个元素
        i++;
    }

    printf("%u\t%d", max, max_i);   // 输出最大值及其下标
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int seed;
    unsigned int a[20];
    scanf("%d", &seed);
    srand(seed);
    for (int i=0; i<20; i++){
        a[i] = rand();
    }
    unsigned int max=0;
    int max_i=0;
    for (int i=0; i<20; i++){
        if (a[i]>max){
            max=a[i];
            max_i=i;
        }
    }
    int i=0;
    while (i<20){
        printf("%u\t", a[i]);
        if (i%5==4) printf("\n");
        i++;
    }
    printf("%u\t%d", max, max_i);
}
```

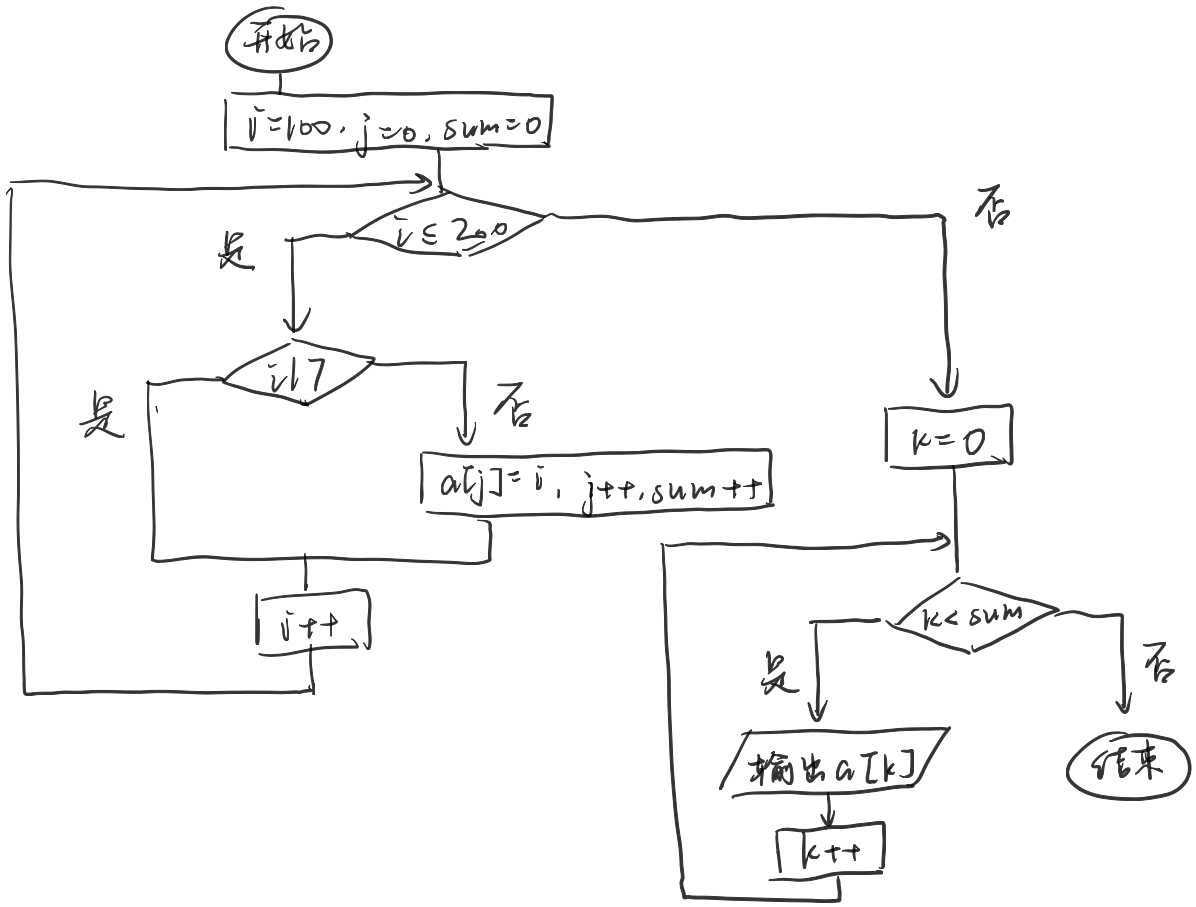
```
C:\Windows\System32\cmd.e×
0
38      7719    21238   2437    8855
11797   8365    32285   10450   30612
5853    28100   1142    281     20537
15921   8945    26285   2997    14680
32285   7
Press any key to continue . . .
```

8：

```
#include <stdio.h>

int main()
{
    int a[100];           // 定义数组
    int i = 100, j = 0;
    int sum = 0;          // 统计符合条件的整数个数

    while (i <= 200){
        if (i % 7 != 0){   // 排除能被 7 整除的数
            a[j] = i;      // 将符合条件的数存入数组
            j++;
            sum++;          // 计数加 1
        }
        i++;
    }

    int k = 0;
    while (k < sum){
        printf("%d\t", a[k]);        // 输出数组中的数
        if (k % 5 == 4) printf("\n"); // 每行输出 5 个数换行
        k++;
    }
    return 0;
}
```



流程图：



选做题 1：

```
    #include <stdio.h>

    int n;              // 全局变量：要排列的数字上界 N（1..N）
    int a[15];          // 存放当前正在构造的排列，使用 1..n 的位置，预留 15 足够 N<=10
    int used[15];       // 标记数字 i 是否已被使用（used[i]==1 表示数字 i 已放在排列中了）

    void dfs(int step) { // 递归函数：负责填第 step 个位置（step 从 1 开始）
       if (step > n) {  // 递归基：当 step 超过 n，说明 1..n 都放完了，形成一个完整排列
          for (int i = 1; i <= n; i++) {
             printf("%d", a[i]);       // 输出当前排列的第 i 个数
             if (i != n) printf(" ");  // 除了最后一个数字外，每两个数字之间输出一个空格
          }
          printf("\n");              // 每个排列占一行，行末换行
          return;                   // 返回到上层递归（回溯）
       }
```

```
    for (int i = 1; i <= n; i++) {    // 依次尝试把数字 i（1..n）放到第 step 位
        if (!used[i]) {              // 如果数字 i 尚未在当前排列中使用
            used[i] = 1;             // 标记 i 已被使用
            a[step] = i;             // 把 i 放到当前位置
            dfs(step + 1);           // 递归去填下一个位置（step+1）
            used[i] = 0;             // 回溯：撤销对 i 的使用，供后续尝试其他排列
        }
    }
}

int main() {
    scanf("%d", &n);   // 从标准输入读入整数 N（题目保证 1<=N<=10）
    dfs(1);            // 从第 1 个位置开始深搜（生成所有排列）
    return 0;          // 程序正常结束
}
//参考 AI 完成
```

```
#include <stdio.h>

int n;                // 全局变量：要排列的数字上界 N（1..N）
int a[15];            // 存放当前正在构造的排列，使用 1..n 的位置，预留 15 足够 N<=10
int used[15];         // 标记数字 i 是否已被使用（used[i]==1 表示数字 i 已放在排列中了）

void dfs(int step) {  // 递归函数：负责填第 step 个位置（step 从 1 开始）
    if (step > n) {   // 递归基：当 step 超过 n，说明 1..n 都放完了，形成一个完整排列
        for (int i = 1; i <= n; i++) {
            printf("%d", a[i]);          // 输出当前排列的第 i 个数
            if (i != n) printf(" ");     // 除了最后一个数字外，每两个数字之间输出一个空格
        }
        printf("\n");                    // 每个排列占一行，行末换行
        return;                          // 返回到上层递归（回溯）
    }

    for (int i = 1; i <= n; i++) {       // 依次尝试把数字 i（1..n）放到第 step 位
        if (!used[i]) {                  // 如果数字 i 尚未在当前排列中使用
            used[i] = 1;                 // 标记 i 已被使用
            a[step] = i;                 // 把 i 放到当前位置
            dfs(step + 1);               // 递归去填下一个位置（step+1）
            used[i] = 0;                 // 回溯：撤销对 i 的使用，供后续尝试其他排列
        }
    }
}

int main() {
    scanf("%d", &n);    // 从标准输入读入整数 N（题目保证 1<=N<=10）
    dfs(1);             // 从第 1 个位置开始深搜（生成所有排列）
    return 0;           // 程序正常结束
}
//参考AI完成
```

```
C:\Windows\System32\cmd.e×
4
1 2 3 4
1 2 4 3
1 3 2 4
1 3 4 2
1 4 2 3
1 4 3 2
2 1 3 4
2 1 4 3
2 3 1 4
2 3 4 1
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 2 1 4
3 2 4 1
3 4 1 2
3 4 2 1
4 1 2 3
4 1 3 2
4 2 1 3
4 2 3 1
4 3 1 2
4 3 2 1

Press any key to continue . . .
```

四、 调试中的问题及解决方法（字数不限）

在调试这些题目时，我遇到了一下问题：数组下标错误，导致输出结果时与期望输出差1；逻辑判断不完善，素数判断时返回结果位置写错导致程序变为只要有不能整除的数就认为是素数。解决方法：在关键步骤使用 printf 打印变量检查值，拆分复杂逻辑逐行检查，借助 AI 工具检查。