《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 12 月 8 日**

实验名称	C 程序设计入门

一、 实验目的要求

1. 理解内存地址的概念，指针和指针变量的概念和使用；
2. 掌握数组的地址空间和元素访问过程，数组和指针的关系，指针的算术运算；
3. 尝试通过指针访问数组和字符串。

二、 实验内容

1.实验指导书 4.1.2 程序填空练习 1
2.实验指导书 4.1.3 自主编程练习 1（用指针移动和指针不动两种方法实现）
3.实验指导书 4.1.3 自主编程练习 4
4.实验指导书 4.1.3 自主编程练习 8
5.实验指导书 4.1.3 自主编程练习 9
6.补充题目：用指向一维数组的指针实现 n 个整数的选择排序（递减），并画出流程图
选做题：
7. 设想一个系列的水槽，每个水槽的宽度相等，水槽的高度由一个长度为 n 的整数数组‘waterTroughHeights‘ 给出。现在要在这些水槽中，选择两个位置安装隔板，使得隔板与水槽底部共同构成的蓄水区域可以积累最多的雨水。计算这个蓄水区域可以积累的最大雨水量。注意：隔板必须垂直放置，且顶部平行于水槽底部。
。
撰写实验报告，格式为 学号-姓名-实验 8.pdf，在上机平台提交

三、 上机程序（有注释）

1.实验指导书 4.1.2 程序填空练习 1

```
    #include <stdio.h>
    int main()
    {
        int i, x[6], sum=0;
        printf("请输入 6 个整数:");
        for (i=0; i<6; ++i){
            scanf("%d", x+i);   // 读取输入到数组
            sum+=*(x+i);        // 累加当前输入的值
        }
        printf("Sum=%d", sum);  // 输出总和
        return 0;
}
```

2.实验指导书 4.1.3 自主编程练习 1（用指针移动和指针不动两种方法实现）

//使用指针完成

```
    #include <stdio.h>

    // 交换两个整型指针所指向的值
    void swap(int *p1, int *p2)
    {
        int temp=*p1;
        *p1=*p2;
        *p2=temp;
    }

    int main()
    {
        int arr[10];
        int *p=arr;
        for (int i=0; i<10; i++){
            scanf("%d", p+i);  // 输入数组元素
        }

        int *min=arr;  // 指向当前最小值
        int *max=arr;  // 指向当前最大值

        for (int i=1; i<10; i++){
            if (*(arr+i)<*min)
                min=arr+i;   // 更新最小值位置
            if (*(arr+i)>*max)
                max=arr+i;   // 更新最大值位置
        }

        if (max==arr)
            max=min;  // 若最大值在第一个位置，避免后续交换冲突

        swap(arr,min);       // 最小值换到开头
        swap(arr+9,max);     // 最大值换到末尾

        for (int i=0; i<10; i++){
            printf("%d ", *(arr+i));  // 输出数组
        }
}

//不使用指针完成
    #include <stdio.h>
    int main()
```

```
    {
        int arr[10];
        int max=0;   // 最大值所在下标
        int min=0;   // 最小值所在下标

        for (int i=0; i<10; i++){
            scanf("%d", &arr[i]);   // 输入数组元素
        }

        for (int i=1; i<10; i++){
            if (arr[i]<arr[min])
                min=i;              // 更新最小值下标
            if (arr[i]>arr[max])
                max=i;              // 更新最大值下标
        }

        int temp1=arr[min];
        arr[min]=arr[0];            // 最小值交换到首位
        arr[0]=temp1;

        if (max==0)
            max=min;                // 若最大值在首位，调整下标避免冲突

        int temp2=arr[max];
        arr[max]=arr[9];            // 最大值交换到末位
        arr[9]=temp2;

        for (int i=0; i<10; i++){
            printf("%d ", arr[i]);  // 输出数组
        }
}
```

3.实验指导书 4.1.3 自主编程练习 4

```
    #include <stdio.h>
    int main()
    {
        char str[1000];
        scanf("%[^\n]", str);        // 读取整行（含空格）
        char *p=str;
        int flag=0;                  //用于控制逗号

        while (*p!='\0'){
            if (*p>='0' && *p<='9'){ // 若当前字符是数字
```

```
            char *start=p;        // 记录数字片段起始位置
            while (*p!='\0' && *p>='0' && *p<='9'){
                p++;              // 移动到数字片段末尾
            }
            if (p-start>=2){      // 数字片段长度≥2 才输出
                if (flag!=0)
                    printf(", "); // 非首段则输出顿号
                for (char *q=start; q<p; q++){
                    putchar(*q); // 输出数字片段
                }
                flag=1;           // 已输出过一段数字
            }
        } else {
            p++;                  // 非数字继续向后扫描
        }
    }
}
```

4.实验指导书 4.1.3 自主编程练习 8

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

int main()
{
    char str[1000];
    scanf("%[^\n]", str);                    // 读取整行字符串（含空格）

    if (strlen(str)>200)
        *(str+200)='\0';                     // 若超过 200 字符则截断

    char *p=str;
    int count=0;                             // 当前最长字母串长度
    char *start_max=p;                       // 指向最长字母串起始位置

    while (*p!='\0'){
        // 判断是否为字母
        if ((*p>='a' && *p<='z') || (*p>='A' && *p<='Z')){
            char *start=p;                   // 当前字母串起始点
            while (*p!='\0' && ((*p>='a' && *p<='z') || (*p>='A' && *p<='Z'))){
                p++;                         // 扫描整个字母串
            }
            if (p-start>count){              // 若当前字母串更长则更新记录
                count=p-start;
```

```
                    start_max=start;
                }
            } else {
                p++;                                // 非字母则继续扫描
            }
        }

        for (char *q=start_max; q<start_max+count; q++){
            putchar(*q);                            // 输出最长字母串
        }
}
```

5.实验指导书 4.1.3 自主编程练习 9

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

    void reverse(char *start, char *end){
        while (start<end){
            char temp=*start;   // 交换首尾字符
            *start=*end;
            *end=temp;
            start++;
            end--;
        }
    }

    void move(char *str, int n) { //通过三次翻转实现
        int len=strlen(str);
        n=n%len;                // 防止移动次数超过长度

        reverse(str, str+len-1);    // 整体翻转
        reverse(str, str+n-1);      // 翻转前 n 个字符
        reverse(str+n, str+len-1);  // 翻转剩余字符
    }

    int main()
    {
        char str[1000];
        scanf("%[^\n]", str);       // 读取字符串（含空格）
        int n;
        scanf("%d", &n);

        if (strlen(str)>50)
```

```
        *(str+50)='\0';          // 超过 50 字符则截断

    move(str,n);                 // 右移 n 位
    printf("%s", str);
}
```

6.补充题目：用指向一维数组的指针实现 n 个整数的选择排序（递减），并画出流程图

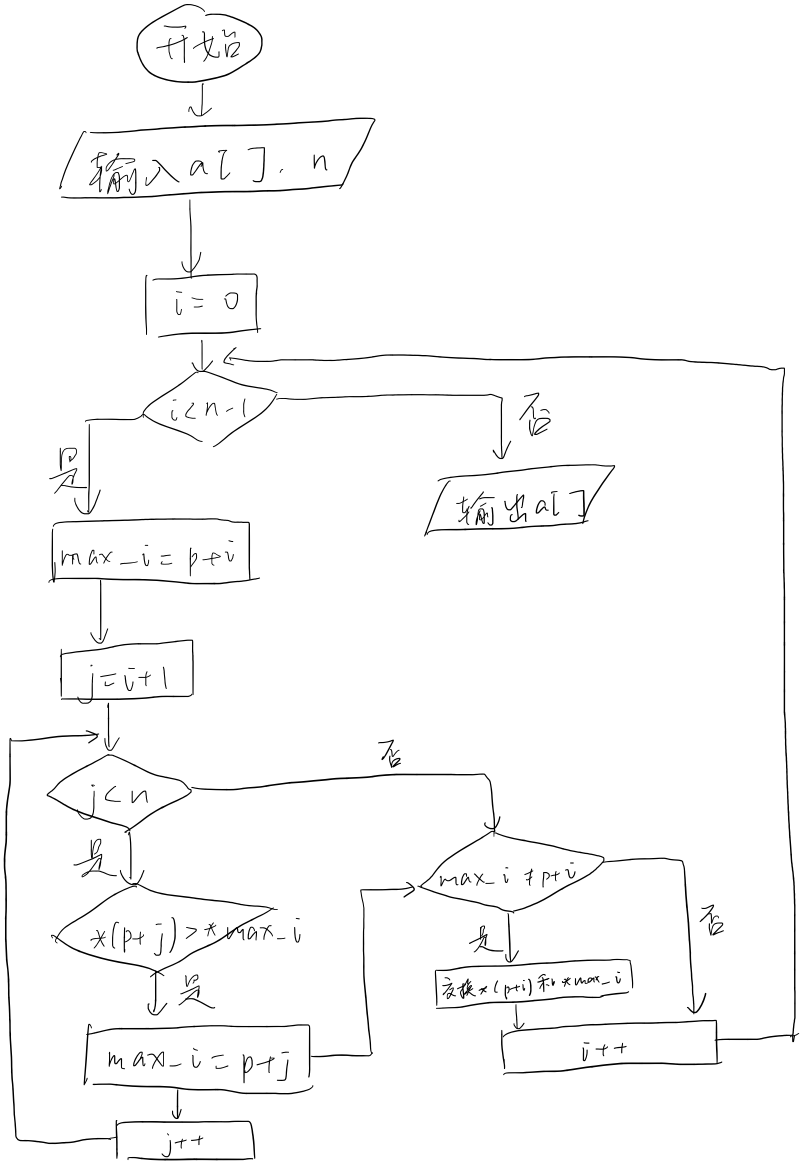
```
#include <stdio.h>

void selection_sort(int *p, int n){
    int *max_i;          // 指向本轮找到的最大值位置
    int temp;
    for (int i=0; i<n-1; i++){
        max_i=p+i;       // 假设当前位置为最大值
        for (int j=i+1; j<n; j++){
            if (*(p+j)>*max_i){
                max_i=p+j;   // 更新最大值位置
            }
        }
        if (max_i != (p+i)){  // 若找到的最大值不是当前位置则交换
            temp=*(p+i);
            *(p+i)=*max_i;
            *max_i=temp;
        }
    }
}

int main()
{
    int n;
    scanf("%d", &n);
    int a[1000];
    int *p=a;
    for (int i=0; i<n; i++)
        scanf("%d", p+i);   // 输入数组

    selection_sort(a, n);   // 选择排序（降序）

    for (int i=0; i<n; i++)
        printf("%d ", *(p+i));  // 输出排序后数组
}
```



7.选做题：

```
#include <stdio.h>
int MaxVolume(int *height, int n){
    int *left=height;           // 左指针
    int *right=height+n-1;      // 右指针
    int max=0;                  // 最大面积记录
    while (left<right){
        int width=right-left;   // 宽度 = 下标差
        int h=(*left<*right)?*left:*right;   // 取较小的高度
        int current=width*h;    // 当前面积
        if (current>max)
            max=current;        // 更新最大面积
        if (*left<*right)
            left++;             // 短板在左 → 左指针右移
        else
```

```
            right--;              // 短板在右 → 右指针左移
        }
        return max;
    }

    int main()
    {
        int arr[100];
        int *p=arr;
        int n=0;
        while (scanf("%d", p+n) == 1){   // 读取一个整数
            n++;
            char ch=getchar();           // 读取分隔符（逗号或换行）
            if (n>=100 || ch=='\n')      // 数组满或行结束 → 停止读取
                break;
        }
        printf("%d", MaxVolume(arr, n));  // 输出计算结果
}
```

四、 调试中的问题及解决方法（字数不限）

1：数组原地操作时的数据丢失

- **问题**：在实现字符串右移或最值交换时，发现直接按照坐标映射修改（如 str[i+n] = str[i]）会导致目标位置的原有数据被覆盖，从而引发连锁错误。
- **解决方案**：
  临时空间法：申请一个 temp 数组作为中转，保存计算结果后再拷回。
  三次翻转法：通过局部逆序从而无需额外空间。

2：指针在交换过程中的“漂移”

- **问题**：在连续执行两次对换（最小换到前，最大换到后）时，如果最大值原本在第一个位置，第一次对换会把最大值移走，导致第二次对换找错目标。
- **解决方案**：在第一次交换后，增加一个条件判断：如果最大值指针指向了首地址，则将其同步更新到交换后的新地址，确保指针始终追踪正确的数据。

3：输入流中的空格处理

- **问题**：使用 scanf("%s") 读取字符串时，遇到空格就停止，无法读取像 USTC room 3c101 这样完整的行。
- **解决方案**：改用 scanf("%[^\n]", str)（正则表达式扫描集）。通过指定“读取直到换行符为止”，成功让指针能够处理包含空格的整行文本。

4：多层循环带来的效率瓶颈

- **问题**：在“盛最多水的容器”和“字符串右移”题目中，使用双重嵌套循环会导致计算量显著增大。
- **解决方案**：采用了“双指针贪心策略”（左右对撞指针）。利用指针相减计算宽度，并只移动高度较矮的一侧。通过一层循环替代了多层循环，大幅提升了程序运行效率。