Ch1 作业

班级	R0112030.09	学号	2022040906023	姓名	梁书恺	成绩	

1. 简述下列概念：数据、数据元素、数据项、数据对象、数据结构、逻辑结构、存储结构、抽象数据类型。8 分

数据：对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称。

数据元素：数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。

数据项：数据元素的组成单位，是数据不可分割的最小单位。

数据对象：性质相同的数据元素的集合，是数据的一个子集。

数据结构：相互之间存在一种或多种特定关系的数据元素的集合。

逻辑结构：数据元素之间的抽象逻辑关系。

存储结构：数据结构在计算机中的表示，包括数据元素的表示和关系的表示。

抽象数据类型：指一种数学模型以及定义在该模型上的一组操作。

2. 在数据结构中，从逻辑上可以把数据结构分成（ C ）。5 分

A．动态结构和静态结构　　B．紧凑结构和非紧凑结构

C．线性结构和非线性结构　　D．内部结构和外部结构

3. 以下说法正确的是（ D ）。5 分

A．数据元素是数据的最小单位

B．数据项是数据的基本单位

C．数据结构是带有结构的各数据项的集合

D．一些表面上很不相同的数据可以有相同的逻辑结构

4. 在存储数据时不仅要存储各数据元素的值，还要存储（ C ）。5 分

A 数据的处理方法　　B 数据元素的类型

C 数据元素之间的关系　　D 数据的存储方法

5. 数据的存储结构是指（ B ）　5 分

A. 数据所占的存储空间量

B. 数据的逻辑结构在计算机中的表示

C. 数据在计算机中的顺序存储方式

D. 存储在外存中的数据

6. 算法的时间复杂度取决于（ D ）。5 分

A．问题的规模　　B．待处理数据的初态

C．计算机的配置　　D．A 和 B

7. 算法的时间复杂度是指（ B ）。　5 分

A. 执行算法程序所需要的时间
B. 算法执行过程中所需要的基本运算次数
C. 算法程序的长度
D. 算法程序中的指令条数

8. 算法的空间复杂度是指（ D ） 5 分
A、执行算法程序所占的存储空间
B、算法程序中的指令条数
C、算法程序的长度
D、算法执行过程中所需要的存储空间

9. 分析时间复杂度。8 分

```
1.   i=1;
     while(i<=n)
         i=i*3;
```

答：$O(n)$

```
2.  for (i=n-1; i>=1; i--)
        for (j=1; j<=i; j++)
            if (A[j]>A[j+1]) 交换 A[j]与 A[j+1];
```

答：$O(n^2)$

```
3.  for (i=0; i<n; i++)
        for (j=0; j<m; j++)
            a[i][j]=0;
```

答：$O(n*m)$

```
4.  for (i=1; i<=n; i++)
        if (2*i<=n)
            for (j=2*i; j<=n; j++)
                y=y+i*j;
```

答：$O(n^2)$

10. 设计算法，用伪代码描述。在数组 $r[n]$中删除所有值为 x 的元素，要求时间复杂度为 $O(n)$，空间复杂度为 $O(1)$。9 分

```
输入：数组 r[n]，数组长度 n，目标值 x
输出：新的数组长度 newLen

1. 初始化 newLen = 0;
2. 通过变量 i 遍历数组
    a) 如果 r[i] != x;
      i.    r[newLen] = r[i];
     ii.    newLen++;
3. 输出 newLen;
```

11. 用 C 编程实现：在一维数组中查找最大值，要求用递归实现。10 分

```
int findMax(int arr[], int size)
{
    if (size == 1)
    {
        return arr[0];
    }
    int temp = findMax(arr, size - 1);
    return (temp > arr[size - 1]) ? temp : arr[size - 1];
}
```

12. 用 C 编程实现例 1 的三种解法：将一个具有 n 个元素的数组向左循环移动 i 个位置。30 分

方法一：将前 i 个元素放入临时数组，将其余的元素左移 i 个位置，再将临时数组中的元素放在数组最后。

```
void rotateLeft(int arr[], int n, int i)
{
    int temp[i];

    for(int j = 0; j < i; j++)
    {
        temp[j] = arr[j];
    }

    for(int j = 0; j < n - i; j++)
    {
        arr[j] = arr[j + i];
    }

    for(int j = 0; j < i; j++)
    {
        arr[n - i + j] = temp[j];
    }
}
```

方法二：将第一个元素放入临时变量，将其余的元素左移一个位置，再将临时变量放在数组末尾，循环执行 i 次。

```
void rotateLeft(int arr[], int n, int i)
{
    for(int j = 0; j < i; j++)
    {
        int temp = arr[0];

        for(int k = 0; k < n - 1; k++)
        {
            arr[k] = arr[k + 1];
        }

        arr[n - 1] = temp;
    }
}
```

方法三：首先反转前 i 个元素，然后反转剩余的元素，最后反转整个数组。

```
void reverse(int arr[], int start, int end)
{
    while(start < end)
    {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}

void rotateLeft(int arr[], int n, int i)
{
    reverse(arr, 0, i - 1);
    reverse(arr, i, n - 1);
    reverse(arr, 0, n - 1);
}
```