**《程序设计基础》实验报告（第 8 次实验）**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学号** | **2021218149** | **姓名** | **贺洋** | **班级** | **计算机21-3班** | **日期** | **1.10** |
| **实验名称：实验8后续综合** | | | | | | | |
| **实验目的：**1. 掌握结构体类型定义；  2．掌握结构体变量、数组、指针的说明；  3．掌握结构体类型量及其成员项的访问方式；  4．掌握动态内存分配方式，理解及熟悉链表的各种操作；  5．掌握位运算的基本操作；  6. 掌握文件的操作过程；  7. 掌握数据存储处理的一般性方法； | | | | | | | |
| **实验内容：**编写程序，实现以下功能，可用多种方法实现。   1. 定义结构类型编程实现如下功能：从键盘输入整数n，代表学生的数量。每个学生信息包括：姓名、学号、以及每一个学生的4门功课的成绩，分别计算每个学生的总分和均分，并将此表按总分的高低排序。要求输入n，动态分配n个连续单元。编程实现上述要求。 2. 输入整数m，n动态分配m×n个浮点型单元，要求采用二维数组的结构化数据表示。通过随机函数（用法自查）产生每个单元的值，计算并输出其中的最小值及其行、列索引位置。 3. 以群文件中的CHAP7EXC.c为蓝本，添加两个函数，第一个函数实现链表的“头插”，第二个函数实现链表的逆序。并将实现的功能添加到主函数的文本菜单中，构成完整的程序。   编写程序，实现以下功能，可用多种方法实现。   1. 要求如下： 2. 通过用户定义类型，将无符号字符型定义成BYTEPORT类型。 3. 通过BYTEPORT类型定义一个变量，其中6bit控制一个十字路口的红绿黄灯，每bit控制不同路口一个颜色的灯，1亮、0灭。 4. 假设路口是东西-南北走向。两个方向各有红、绿、黄三各颜色的交通通行指示，黄灯总是在绿转红之间插入五秒。 5. 要求编程实现一个1s延时函数，输入两个路口的红灯、绿灯亮、灭时间（单位秒，作为参数调用一秒函数实现要求的时间）。   **编程实现交通灯的控制。**   1. 定义一个学生类，包含基本学生的基本信息属性，学号、姓名、年龄性别。构造函数实现初始化。缺省初始化学号：-1、姓名："None"、年龄：0，性别：'M'。成员函数包括设置和限时基本信息。定义继承学生类一个派生类包括属性为：所学专业及四门门课的成绩，并实现构造函数、析构函数及计算总分平均分的成员函数。   要求定义10个人的对象数组，完成输入数据，实现总分和平均分的计算，按以总分降序排序后将学生的属性：学号、姓名、年龄、性别、专业、四门课成绩、总分、平均分存入文本文件（或二进制文件，自选，文件名是学号后四位.dat）。   1. 群文件中的hamlet.txt是莎翁的《哈姆雷特》剧本，编程通过异或运算或循环移位按字符（秘钥长度1Byte）对文本加密（秘钥可输入）。并将加密后的文本存到二进制的文件Cyb.dat中。再编写另一个程序完成相反过程。 2. 将通过随机函数产生86400个双精度数，存入二进制文件s.bin，模拟一张24小时的时间序列表。程序中获取当前时间，取出当前时间（秒）对应的序列表，在屏幕上显示。 | | | | | | | |
| **实验分析：**   * + - 1. 结构体的定义书写以下两种：struct XX{}; 或者是：struct{}变量1，变量2，变量x;后者可以不写出结构题名称。结构体变量的创建，struct XX;       2. 关于结构体变量创建后的赋值行为：方法一：使用构造函数来进行初始化；方法二：在创建结构体变量的时候直接按照数据类型进行初始化，形如：struct stu = {“none”, 18, M};       3. 结构体数组：创建的书写格式：strcut XX[N];这是直接创建结构体固定长度的数组，若是使用动态内存分配去做数组，在c++中应当使用new来开辟内存。因为在c++中new开辟内存的同时会调用构造函数，而c语言中的malloc函数不会调用类内或结构体内的构造函数。       4. 结构题数组的遍历：遍历过程中显示或改变结构体内的变量，通过使用.符号去调用。形如：Arrary[i].xx;并且，结构体内部的指针可以作为连接两位一个变量的方式，从而建立起链表。       5. 结构体指针：在结构体中不能包含自己的结构，但是可以包含自己结构的指针和引用。结构体指针调用成员：形如：p->xx;通过使用->符号来访问结构体内的变量。       6. 结构体的内存：结构体是一个集合体，内部可以包含其他类型的变量，关于结构体所占的字节：是用对齐的方式来计算的，在书写结构体内的变量的时候，也可以适当注意变量的排版。在允许情况下，按照字节的大小顺序规律的排版。       7. 两种动态内存分配的方法：malloc与new：       8. Malloc：使用malloc来分配内存。首先，malloc是在堆中开辟内存，寻找一块空闲的内存，返还首字节的地址。值得注意的是，malooc返回的指针为void类型的指针，开辟的内存的大小也是由程序员自己决定的。因此，使用某种类型的指针来接收malloc返回的地址时，需要进行强制类型转换为我们需要的类型。形如：int \*p = (int \*)malloc(sizeof(int))这样就能在堆中开辟出一块供我们使用的int类型的空间。对于一般的变量，形如float、double、int、long、char等类型，在程序属于的范围内结束时，便会自我消亡，但是malloc开辟的内存不会，这给了程序员更大的操作空间，需要程序员在指定的位置释放这块内存。释放内存的关键字为free，形如free(p);这样我们就释放了之前开辟的那块内存的空间，但是，释放这块空间的意思并不是将这块储存空间清零，而是系统收回了你对这块空间的使用权限。       9. New：new是c++中开辟内存的首选。首先，new的语法和malloc稍有差别。还是以上面的为参考，形如：int \*p = new int;这样便与上面malloc做的事情无异。但是在为结构体、类的变量或数组开辟动态内存时，c++中的new却有着新的功能。C++的new会调用类内或结构体内的构造函数，相比这一点，malloc是无法做到的。对应的new释放内存的关键字与malloc也不一样，在c++中用关键字delete来释放使用new开辟的内存。New开辟的内存也是在堆中，也是需要程序员手动释放。值得一提的是，在释放完new开辟的内存的时候，需要对指针进行滞空处理，防止指针成为野指针。       10. New和malloc开辟数组的空间：关于malloc和int开辟的数组空间，是我们人为的把它看成数组。在实际的内存开辟情况中，我们或得的是一大块连续的空间。在了解这一点的基础上，动态分配二维数组才能够真正的被我们理解。动态分配的二维数组，使用malloc和new来开辟并无多大区别。在理解指针的基础上，将分配的两端内存用二维数组的方式去连接起来。但是，malloc分配出来的二维数组的地址是连续的，但是new开辟出来的二维数组的地址从行和列来看是分开的，行是连续的，但是列是分开的。       11. 链表：创建链表的基础首先要有一个结构体或者是一个类，在其中必须包含指向自己类或结构体的指针。在没有用循环来建立一个链表的时候们可以先单独创建出几个结构体变量出来，用内部的指针来指向下一个变量，如此往下，便能将一堆结构体变量串联起来。但是它们的地址都不是连续的，这当中我们考虑为动态内存分配出的变量。需要就创建，这就是链表的一大优势。与数组对比，链表的优势便是，方便数据常常需要删改，需要添加等等。因为数组中的元素进行删改都要进行全员遍历的行为，在很多时候效率是很低的。而且链表的动态增加，改动也是数组无法做到的。但是，数组创建的快捷性，稳定性也同样是链表所缺失的。在实际问题中，需要我们自己去考虑要使用哪一种。       12. 链表的头结点：链表的头结点要区别于链表的首元节点。链表的头结点是不存放数据的，头结点的类内指针被称为头指针，指向第一个带有存储数据的节点。头结点是可有可无的，但是拥有头结点在对链表进行操作时会带来很多便利，并且也会提高链表的稳定性。       13. 链表的头插及其逆序：一起总结这两点是因为，链表的头插和链表的逆序中使用的头插法很相似。也正是因为拥有了头结点在进行链表逆序的头插发和链表的头插时，显得很轻松。只需要在头结点和首元节点之间插入一个节点便完成了链表的头插。而链表的逆序也同样是这个道理，将头指针往后移动，实现头插法完成链表的逆序。       14. 链表的释放：这一点在之前没有提到，对于链表的释放，可以总结为，开辟了多少此内存空间就释放多少此内存。在最后的最后释放头结点。       15. &：与运算，两个位都为1时，结果才为1。 |：或运算，两个位都为0时，结果才为0。 ^：异或运算，两个位相同位0，相异为1。 ~：取反运算，0变1，1变0。 << ：左移运算，各二进位全部左移若干位，高位丢弃，低位补0。 >> ：右移运算，各二进位全部右移若干位，对无符号数，高位补0，有符号数，各编译器处理方法不一样，有的补符号位（算术右移），有的补0（逻辑右移）       16. 位运算的优势体现在速度上，会比一般的算法更加快，有时候也能防止满栈，就例如快速乘或者是快速幂算法的情况下，一般算法可能会导致程序崩溃，超出参数类型的最大值，而依靠位运算的快速幂和快速乘则不会。       17. 位域和位运算：使用位运算来实现位域节约空间实现目的的方式。冰洁位运算用于加密等也是十分方便书写，在子网掩码等区域应用广泛。       18. 文件打开：打开文件首先要先包含头文件，读写的头文件不同，ifstream与ofstream，但是都可以用fstream来代替。打开一个文件有两种方式，一种是在创建文件变量时直接使用构造函数来初始化打开这个文件，也可以在创建变量之后调用成员函数来打开一个文件。打开文件的的方式分为，读、写、二进制方式等方式。同时也可以用读二进制的方式来打开这个文件。文件的打开可能是不存在这个文件的，在接下去可能就会发生很多无法预料的问题。所以在打开文件的之后要判断文件是否打开成功，若不成功进行处理。       19. 文件的关闭，调用成员函数，形如：fs.close();       20. 文件的读写：文件的写入，二进制与文本文件的写入有区分。二进制写入文件需要调用成员函数的write函数经行操作。而文本文件的操作，只需要类似cout << 这样的输出流方式便能不断的写入文件。文件的读取：二进制文件和文本文件都可以使用fstream的变量的成员函数的get函数来获取文件中的信息，并且是进行逐字符读取，直到文件的末尾，以标志EOF为结尾，意为：end of file。代表的值为整形数据的-1.对于二进制文件的读取也可以使用成员函数read来进行读取。       21. 面向对象的编程：面向对象的三大特征：封装、继承、多态；       22. 封装：尤其是当需要重复使用到一些特征事物的时候，可以将其封装起来，作为一个整体，在后续的使用中带来便利。       23. 继承：继承中还有虚继承，虚继承主要是为了解决菱形问题，防止出现错误。       24. 多态：多态是c++面向对象的重要特征！多态的实现：子类重写父类的虚函数，通过父类的指针去调用子类的对象，实现多接口的实现。虽然这个实现的过程中代码量看起来是变多了，但是有了这个接口，在维护程序和添加改动的过程中更有利于我们。 | | | | | | | |
| **实验结果（结果截屏）** | | | | | | | |
| **第1题结果** | |  | | | | | |
| **第2题结果** | |  | | | | | |
| **第3题结果** | |  | | | | | |
| **第4题结果** | |  | | | | | |
| **第5题结果** | |  | | | | | |
| **第6题结果** | |  | | | | | |
| **第7题结果** | |  | | | | | |