面向对象程序设计A  
（08305138，4学分）实验安排

**课程小项目从第三周开始，以小组形式进行。四人一组，自由组合。**

**注意：（1）程序题源代码打包格式为“10008642陈圣波.zip”并提交；**

**编写程序的时候，程序源代码（.h或者.cpp）的文件名（以及项目名称）不要用中文。**

【腾讯文档】OOP-2023实验分组chen01

<https://docs.qq.com/sheet/DWmhSSFNFWmdZaGdk> ，点击并填写。

第2周 实验二（单向链表设计）

2.1 函数设计基本原则（供参考）

C/C++程序是由函数组织的，程序的运行时函数之间相互调用驱动的。编写C/C++程序主要是编写各种各样的函数，包括主函数。

我们认为，学习了C语言程序设计基础后，“微观程序设计”能力应该基本具备。即：能熟练处理顺序结构、分支结构、循环结构语句。从现在起，需要将注意力集中到函数的首部（函数返回类型、函数名、形式参数）设计上来，特别是函数的参数设计。

2.1.1 函数名

在符合标识符命名规则的前提下，函数名要在一定程度上反映函数的功能。同一程序中，函数名命名风格一致，且函数名不宜过长。

2.1.2 函数返回类型

(1) 如果函数确实没有需要返回的量，则返回类型设计为**void**。

(2) 一般地，有些函数原本并不需要返回值或返回变量，在不增加程序设计负担、同时还能增加函数应用的灵活性的情况下，可以设计返回类型，返回一个数值、或返回一个变量（引用返回）。返回类型的设计应尽量保持已有函数、或运算符的相似特性。

(3) 如果可能，返回类型应**尽可能地采用引用返回**，以避免值返回时创建、销毁临时无名变量。

2.1.3 函数形式参数

(1) 形参的名称应尽可能表意，形参个数不宜过多，参数之间应相互独立。精心安排参数的顺序，并考虑是否设置默认参数。

(2) 函数所需的数据最好都从形参传入，尽量减少对全局变量的依赖（仅允许类似cin、cout这样的全局变量）；尽量减少对某个具体题目所设定数值的依赖，使函数具有灵活性、适应性、可移植性（例如：将数组传递给函数时，需要传递数组的“三要素”，而不要默认元素个数为某个具体值）。

(3) 根据参数是单纯地“传入”给函数，还是“传入并传回”的需求，考虑采用值传递或引用传递。运用指针、多级指针亦可以间接地“传回”函数计算的结果；运用变量（包括指针变量）的引用可以直接地“传回”函数的计算结果。

(4) 如果形参是指针，且仅用于将目标数据“传入”给函数，则应尽可能用常量指针；如果形参是引用，且仅用于将所绑定的变量或常量“传入”给函数，则应尽可能采用常量的引用。其好处有二：① 增加了数据的安全性；② 扩大了函数的适用面（能够处理真正的常量）。

(5) 如果形参是以值传递的方式传递对象，可改用传递常引用“const &”方式来传递，这样可以省去值传递对象时的拷贝构造形参对象和析构该形参对象的过程，从而提高时间效率和空间效率。

2.1.4 函数体

(1) 功能单一，不要将众多可以独立的计算、处理过程放在同一个函数中。

(2) 除专门处理I/O的函数外，将计算与I/O相分离。尽量不在执行计算的函数体内执行I/O操作，函数所需的数据应从参数传入给函数；函数的计算结果应从返回类型或有“传回”功能的形参“返回”。因为，不同的应用场景中I/O的方式方法有很大的差别（如：字符模式、图形模式）。

(3) 函数体代码规模不宜过大，要有足够的注解。

2.2 单向链表基本操作

给定构成单向链表结点的结构体Node。编写若干函数对链表进行操作。

|  |
| --- |
| **struct** Node  {  **int** data; // 数据域(虽然这里仅有一个数据，但还是用数据datum的复数形式)  Node \*next; // 指针域  }; |

程序Link\_int编写了主控函数，可以边设计函数边进行测试。建议按表2.1中的顺序逐个设计并逐个测试函数如下6个函数。请将所设计的函数原型写入头文件Link\_int.h中。在Link\_int.cpp文件中编写函数定义。对每一个函数，给出了设计与实现中的要点，特别需要注意的问题。

表2.1 单向链表函数设计

|  |  |  |
| --- | --- | --- |
| **序号** | **设计说明**（请考虑形参传入in□，传回out□中是否应该☑或🗵） | |
| 1 | 函 数 名 | Create |
| 形式参数 | 1. 链表(in☑ out🗵) 2. 存放数据域具体数据的数组三要素，两个参数:元素个数(in☑ out🗵)，首地址(in☑out🗵，带默认值NULL) |
| 返回类型 | Node \*& （引用返回链表首地址）或 **void** |
| 功 能 | 根据连续存放的数据域数组创建链表 |
| 思考问题 | ① 用什么表示一条链表？即形参的数据类型是什么？  ② 对入口参数传入的实参链表有何要求？入口参数传入的链表必须完整，也可以为空链表（最好为空）。如果非空，则可以清空原来的所有结点，也可以添加到已有结点前面。  ③ 如何简便地完成操作？（提示：创建动态结点，采用头插法。让所有结点皆为动态结点，以利于FreeList操作的统一性）  ④ 第3个参数带默认值有和作用？如何测试？ |
| 函数原型 | void Create(Node \*&head, int n, int \*arr = nullptr); |
| 2 | 函 数 名 | NumNodes |
| 形式参数 | 链表(in☑加以保护？ out🗵) |
| 返回类型 | **int** （返回结点的个数） |
| 功 能 | 计算并返回链表的结点个数 |
| 注意事项 | 遍历链表 |
| 函数原型 | int NumNodes(const Node \*head); |
| 3 | 函 数 名 | ShowList |
| 形式参数 | 链表(in☑，加以保护？ out🗵) |
| 返回类型 | **int**（返回输出的结点个数） 或 **void** |
| 功 能 | 输出链表各结点的数据域数据值，要求用 head -> 数据 -> 数据 -> NULL的形式。 |
| 注意事项 | 遍历链表 |
| 函数原型 | int ShowList(const Node \*head, ostream &out = cout); |
| 4 | 函 数 名 | Insert |
| 形式参数 | 1. 链表(in☑ out☑) 2. 数据域数据(in☑ out🗵) |
| 返回类型 | Node \*（返回新插入结点的地址） |
| 功 能 | 插入一个新结点至链表首结点前，成为新的链首结点 |
| 注意事项 | 创建一个动态结点（堆结点），保证链表中的所有结点均为堆结点，便于FreeList统一操作。考虑原链表为空、非空两种情形的程序代码是否相同。 |
| 函数原型 | Node \*Insert(Node \*&head, int data); |
| 5 | 函 数 名 | Append |
| 形式参数 | 1. 链表(in☑ out☑) 2. 数据域数据(in☑ out🗵) |
| 返回类型 | Node \*（返回新插入结点的地址） |
| 功 能 | 追加一个新结点至链表尾结点后，成为新的尾结点 |
| 注意事项 | 创建一个动态结点（堆结点），保证链表中的所有结点均为堆结点，便于FreeList统一操作。考虑原链表为空、非空两种情形的程序代码是否相同。 |
| 函数原型 | Node \*Append(Node \*&head, int data); |
| 6 | 函 数 名 | FreeList |
| 形式参数 | 链表(in☑ out☑) |
| 返回类型 | **void** |
| 功 能 | 释放链表中的所有结点 |
| 注意事项 | 本函数要求所有结点皆为动态结点（堆结点）。采用引用型参数（Node \*&head）函数返回后，链表为空链表。如果形参仅仅设计成链首结点指针值传递（Node \*head），虽可以成功释放所有结点，但head的指向依然指向原链首结点处。 |
| 函数原型 | void FreeList(Node \*&head); |
| 已经提供了如下函数原型设计、函数定义的源代码（参见程序文件） | | |
| 7 | 函 数 名 | **operator**<< |
| 形式参数 | ① 输出设备(in☑，out☑)  ② 链表（in☑加以保护，out🗵） |
| 返回类型 | ostream & （返回输出设备） |
| 功 能 | 重载运算<<，使之能够进行链表输出（如：cout << head << endl）。 |
| 注意事项 | 调用本函数时，最常用的实参是cout。cout是在名字空间std中定义的ostream类型的全局对象。本函数的形参out多数情况下是引用（绑定）cout对象，即out可以是cout的别名。当然，out还可以绑定文件对象，从而可以直接输出到文件。 |
| 函数原型 | ostream & **operator**<<(ostream &out, **const** Node \*head); |
| 8 | 函 数 名 | Locate |
| 形式参数 | ① 链表（in☑，out🗵）  ② 数据域数值（用作查找的依据）（in☑，out🗵）  ③ 找到的结点序号（整型数据）（in□传入的数据不重要，out☑）  ④ 重新开始的查找/继续查找标志（bool型，待默认值） |
| 返回类型 | Node \* （返回找到的结点的地址值，或NULL） |
| 功 能 | 根据给定的数值，在链表中查找结点数据域成员的数值与给定的值相等的结点。支持继续查找。 |
| 思 考 题 | ① 形式参数newsearch的作用是什么？  ② 形式参数 num 有何作用？  ③ 为什么要将 p，k，data 设计成静态局部变量？  ④ 局部自动指针变量 temp 的作用是什么？  ⑤ 函数的返回值是什么？如何区别是否找到满足条件的结点  ⑥ 若形参newsearch为false，此时形参 x 有无作用？为什么要这样设计？  答：对于一种条件的搜索，不能穿插另外一种条件的继续搜索。因此，此时需要故意废掉x。 |
| 函数原型 | Node \*Locate(Node \*head, **int** x, **int** &num, **bool** newsearch=**false**); |
| 9 | 函 数 名 | Save |
| 形式参数 | ① 文件名（C-字符串）（in☑加以保护，out🗵）  ② 链表（in☑，out🗵） |
| 返回类型 | **int** （返回写入文件的结点数据域数据的个数） |
| 功 能 | 将链表所有结点数据域数据写入指定文件名的文件，每个结点的数据占一行。 |
| 提 示 | ① 用到文件流类及其对象（参见教材第14章第2节 文件I/O流）。  ② 此处打开文件采用默认的文本文件方式。  ③ 对于链表结点数据域数据类型(int)，要求能进行<<和>>操作，这个要求是满足的。  ④ 输出到文件的格式要能够使Load函数正确地读取。 |
| 函数原型 | **int** Save(**const char** \*filename, **const** Node \*head); |
| 10 | 函 数 名 | Load |
| 形式参数 | ① 文件名（C-字符串）（in☑加以保护，out🗵）  ② 链表（in☑，out☑） |
| 返回类型 | **int** （返回从文件中读取到的结点数据域数据的个数） |
| 功 能 | 从指定的文本文件中读取数据，作为各结点数据域数值，创建链表。 |
| 提 示 | 同Save函数 |
| 函数原型 | **int** Load(**const char** \*filename, Node \*&head); |

2.3 设计任务

主控函数以及部分函数已提供了源代码（参见“实验02.zip”），**请完成剩下的6个函数的定义。**