面向对象程序设计A  
（08305138，4学分）教学安排

|  |  |  |
| --- | --- | --- |
| **周次** | **课堂教学** | **实验教学** |
| 1 | 1. 课程介绍  2. 从C到C++  引用、函数重载、模板、运算符<<,>>,new,delete、默认参数值、内联函数 | 交换C-字符串函数首部设计（参考程序C-String） |
| 2 | C++过程化程序设计  单向链表函数首部设计 | 不带头结点（或带头结点）的单向链表处理函数设计，重点在函数原型设计，特别是**函数的形参、返回值的设计**（参考程序Link\_int） |
| 3 | 封装：**this**指针、常量成员函数、构造函数、析构函数、类模板 | 启动课程项目1：单向链表类模板及其应用（三层结构）。在机房讨论三层结构设计。  阅读分析程序（参考程序LinkList） |
| 4 | 封装：拷贝构造函数、赋值运算符函数重载 | 课程项目1应用程序设计  分组自选具体内容，要求运用链表的链表。 |
| 5 | 封装：静态成员、友元、运算符重载 课程项目1研讨（1课时） | 启动课程项目2：封装C-字符串  启动课程项目3：向量类模板（含异常处理） |
| 6 | 异常处理 | 课程项目2-3功能实现 |
| 7 | 继承 课程项目2-3研讨（2课时） | 启动课程项目4：抽象向量类模板及其派生类 |
| 8 | 多态性 | 多态性体现 |
| 9 | 多态性  课程项目4研讨（2课时） | 自动单向链表（有静态成员，需要考虑深拷贝构造、深赋值运算）、复数类（双目运算符重载，需要考虑两个操作数为同一对象的情形） |
| 10 | 机动（如I/O流） | 机动（如I/O流） |

【注】以小组为单位，分别于第6、8、10周提交课程小项目1、2和3、4的研究报告，提交到“网上教学”平台本课程的PBL（Problem-Based Learning）中。

第1周 实验一（C-字符串处理）

虽然无法“确认眼神”，但从同学们提交的夏季学期实训课程小论文可确认：部分同学仍然没有掌握C-字符处理的方法，因为其提交的程序貌似通过测试，但本质上是错误的。

1.1 需求

交换C-字符串。有如下代码待完成（参见C-String文件夹）。

|  |
| --- |
| **void** test1()  {  **char** array1[10]="Tom", array2[10]="Jerry";  **char** \*p1 = "Snoopy", \*p2 = "Winnie"; // 指针指向字符串常量（常量数组的首地址）  cout << array1 << ", " << array2 << endl;  // (1) 调用SWAP1函数(C/C++的方案相同)  cout << array1 << ", " << array2 << endl;  cout << p1 << ", " << p2 << endl;  // (2) 调用SWAP2\_C函数(C语言的解决方法)  cout << p1 << ", " << p2 << endl;  p1 = "Snoopy"; p2 = "Winnie";  cout << p1 << ", " << p2 << endl;  // (3) 调用SWAP2\_CPP函数(C++的解决方法)  cout << p1 << ", " << p2 << endl;  **char** \*str1, \*str2;  str1 = **new** **char**[4];  str2 = **new** **char**[6];  strcpy(str1, "Tom");  strcpy(str2, "Jerry");  // (4) 调用STRCAT\_C函数  cout << str1 << endl;  // (5) 调用STRCAT\_CPP函数  cout << str1 << endl;  **delete** [] str1;  **delete** [] str2;  } |

1.2 设计任务

首先根据函数的功能要求设计函数原型（写入头文件C-String.h中），然后在C-String.cpp文件中完成函数定义，最后在testA.cpp文件中完成函数调用表达式语句（见注释语句(1)-(5)）。

要求函数具有普适性。例如SWAP1不仅仅只交换容器是元素个数为10的字符型数组。STRCAT函数不仅仅只连接成"Tom & Jerry"。

表1.1 待完成的函数功能描述

|  |  |  |
| --- | --- | --- |
| **序号** | **函数名** | **功能** |
| 1 | SWAP1 | 交换两个以普通数组（数组名为地址常量）为容器的字符串的内容 |
| 2 | SWAP2\_C | 交换两个指针的指向（所指之处可能为字符串常量），纯C语言版 |
| 3 | SWAP2\_CPP | SWAP2\_C的C++版实现 |
| 4 | STRCAT\_C | 用" & "将第二个字符串连接至第一个字符串尾部并“返回”。其中第一个字符串的容器为动态数组。【提示】实现“自动”地按需扩展容器的容量。 |
| 5 | STRCAT\_CPP | STRCAT\_C的C++版实现 |

1.3 测试任务

源程序文件testB.cpp分析了夏季学期程序设计实训作业中的典型错误。请认真阅读，并进行测试。测试函数test2中使用了条件编译预处理技术，根据宏符号MAKE\_DISTANCE是否有定义，实为两个不同的程序。请在定义和不定义该宏（将testB.cpp第50行作为注释语句）的情况下分别重新编译程序，并运行之。通过观察程序运行结果，理解错误原因。

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。

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。

\*\*\*本框里的内容可以暂时忽略\*\*\*

例如：赋值、迭代赋值运算表达式 a = b 和 a += b 的值均为左值a。因此标准函数

**char** \*strcpy(**char** \*dest, **const** **char** \*source);

**char** \*strcat(**char** \*dest, **const** **char** \*source);

分别模仿赋值运算、迭代赋值运算，均**引用返回**第一个参数。是函数调用表达式继续参与其他函数的嵌套（**复合函数**）调用。

cout << strcat(strcat(str1, " & "), str2) << endl;

值得注意的是I/O运算符 >> 和 << 组成的表达式也是**引用返回**第一个参数。即表达式 cout << x 的运算结果仍然是cout；表达式 cin >> x 的结果仍然是 cin。这样就使得连续输出、连续输入成为可能。即I/O运算符函数的一般形式为

istream & **operator**>>(istream &in, 数据类型 &x);

istream & **operator**>>(istream &in, **const** 数据类型 &x);

例如 cout << a << ", " << b << endl;

cin >> x >> y >> z;

实际上分别为 (((cout << a) << ", ") << b) << endl;

即 cout << a; cout << ", "; cout << b; cout << endl;

((cin >> x) >> y) >> z;

即 cin >> x; cin >> y; cin >> z;

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。用心安排参数的顺序，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）；尽量减少具体题目设定的数值的依赖（例如，题目中要求对12个给定的字符串进行排序，设计的排序函数应该是 **void** StringSort(**char** \*str[], **int** n); 这里的形参n不要缺少，保证数组三要素齐全，也使该函数的适应面不仅限于只对12个字符串进行排序）。

(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。其好处有二：一是增加了数据的安全性；二是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，则改用传递常引用“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、处理过程放在一起。

(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 需求

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 |  |
| 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 |  |
| 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 |  |
| 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 |  |
| 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 |  |
| 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 |  |
| 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，请完成剩下的6个函数的定义。

第3周 实验三（阅读程序）

3.1 链表类的层次结构

表3.1 链表类的层次性

|  |  |  |  |
| --- | --- | --- | --- |
| **层次** | **名称** | **特点** | **说明** |
| 底 | 数据域 | 抽象化、统一化 | 暂不设计，用形式数据类型T。编写代码时想象成**int** |
| 中 | 结点类 | 简单化 | **数据成员：** T data;  Node<T> \*next;  **成员函数：** 默认的构造函数、转换构造函数、拷贝构造函数（next指针初始化为NULL）；  赋值运算符函数（next指针）  **其他：**将链表类作为友类 |
| 高 | 链表类 | 泛化  (尽量拥有一定的普适性) | **数据成员：** Node<T> \*head, \*cur\_node;  **int** num;  **成员函数：** 四大函数（构造函数、拷贝构造函数、析构函数、赋值运算符函数）；  标准I/O、文件I/O；  增、删、改、查；  当前结点定位（绝对定位、相对定位、依条件定位）。 |

阅读LinkList程序。理解单向链表类模板的总体设计。对于每个成员函数、友元函数，重点在其首部设计，理解各个参数的作用。

3.2 LinkList程序中的文件

表3.2 文件说明

|  |  |  |
| --- | --- | --- |
| 序号 | 文件名 | 说 明 |
| **1** | **LinkList.h** | **单向链表类模板全部代码。该文件可移植到其他程序使用。**  **设计了结点类、链表类，数据域类由应用程序设计。数据域类的设计需要满足一些要求：**  **① 四大函数；② 重载I/O运算符函数；③ 能够实现关系运算（因为查找、排序等的需要）。参见下面的通信录程序、银行账户流水记录程序。** |
| **以下都是测试单向链表类模板的代码文件（每个测试函数都是）** | | |
| 2 | Main.cpp | 主函数（主控函数，调用其他测试函数） |
| 3 | Test.cpp | 基本数据类型测试、及链表的链表测试 |
| 4 | AddressBook.cpp | 用户自定义的数据类型测试（通信录） |
| 5 | Banking.h  Banking.cpp  BankingTest.cpp | 用户自定义的数据类型，测试链表的链表。有账户链表（每个结点为一个账户的信息），而每个结点中又包含一个链表（记录账户的流水单）。 |
| **文本文件** | | |
| 6 | AddressBook.txt | 通信录数据文件。可被程序读取，也可将链表各结点数据存盘。 |
| 7 | dLink.txt | 测试函数test1产生的数据文件（double型数据） |
| 8 | LLink.txt | 整型数据链表的链表所产生的数据文件 |
| **工程文件** | | |
| 9 | LinkList.mdsp  LinkList.msp | MinGW Developer Studio IDE的工程文件。旧版（\*.msp）适用于学校计算中心机房。 |
| 10 | LinkList.dev  Makefile.win | Dev-C++ IDE的工程文件。 Makefile.win是编译时产生的文件。 |
| 11 | LinkList.cbp  LinkList.depend  LinkList.layout | Code::Blocks IDE的工程文件。 LinkList.depend和LinkList.Layout为工程文件辅助文件。 |

第4~5周 实验四~五（课程小项目之一：单向链表类模板的应用）

各小组自行选题，利用LinkList类模板设计一个链表综合应用程序。要求使用**链表的链表**。参阅程序Banking。

第6~7周 实验六~七（课程小项目之二、三同步进行）

小组成员角色轮换。要求不同的小项目有不同的组长。

1. 课程小项目之二：字符串类。具体要求如下。

(1) 可以封装C-字符串（即含串结束标志字符'\0'），也可以不含串结束标志；

(2) 对象构造、析构、赋值相关的四大函数；

(3) 其他成员函数（如：length、c\_str等）；

(4) 重载适当的运算符（如：<<、>>、+、+=、<、<=、>、>=、==、!=、[ ]等）；

(5) 有异常处理功能（如：方括号运算时下标越界等）；

(6) 测试用例设计（含异常处理）。

2. 课程小项目之三：向量类模板。具体要求如下。

(1) 向量的维数可缩放；

(2) 对象构造、析构、赋值相关的四大函数；

(3) 其他成员函数（如：返回维数等）；

(4) 重载适当的运算符（如：<<、>>、+、+=、==、!=、[ ]等）；

(5) 有异常处理功能（如：维数不同时无法进行两个向量的和；方括号运算时下标越界）；

(6) 测试用例设计（含异常处理）。

第8~9周 实验八~九（课程小项目之四：抽象向量类模板及其派生类）

1. 抽象向量类模板

(1) 数据成员设计；

(2) 成员函数（包括虚函数、纯虚函数）设计等。

2. 派生向量类模板

(1) 定义纯虚函数；

(2) 设计测试用例单独测试派生类（含异常处理）。

3. 派生字符串类

(1) 定义纯虚函数；

(2) 设计测试用例单独测试派生类（含异常处理）。

4. 联合测试

设计测试用例，对两个派生类进行联合测试，以展示动态多态性。

第10周 实验十（机动，或I/O、异常处理）

略。

【附】2020-2021秋季学期校历

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **周次** |  | **一** | **二** | **三** | **四** | **五** | **六** | **日** |
| 1 | 九月 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 2 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 3 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 4 | 28 | 28 | 30 | **国庆节** | 2 | 3 | 4 |
| 5 | 十月 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 6 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 7 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 8 | 26 | 27 | 28 | 29 | 30 | 31 | 1 |
| 9 | 十一月 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 10 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 考试周 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |