《类库与数据结构》

实验指导书

**计算机与信息科学学院**

**20XX年X月**

**目 录**

[实验一 C++语言与面向对象编程概念 3](#_Toc115970540)

[实验二 开发一个可应用于双向链表的Bidirectional Iterator 1](#_Toc115970541)

[实验三 Linked类的进一步扩充和时间复杂度分析 5](#_Toc115970542)

[实验四 递归回溯算法和面向对象通用回溯框架 9](#_Toc115970543)

[实验五 通用顺序容器List类的完整设计和实现 12](#_Toc115970544)

[实验六 扩展洗车仿真 14](#_Toc115970545)

[实验七 Binary Search Tree的平均树高 16](#_Toc115970546)

[实验八 最短路算法及其应用 18](#_Toc115970547)

# 实验一 C++语言与面向对象编程概念

**实验目的**

1. 掌握C++语言中类、对象、封装、继承、多态等面向对象编程的基本概念和特点；
2. 掌握数据抽象、开放和封闭、子类替换等面向对象编程的基本思想和原则；
3. 掌握在Visual Studio中以定义和实现分离的方式建立包含多个类的C++项目的方法.

**预习与实验要求**

1. 实验前应仔细复习上学期课程中C++语言的基本语法，以及类、对象、继承、重载等面向对象的基本概念。
2. 熟悉Visual Studio开发平台，了解新建Windows Console应用程序的方法。
3. 实验前应该先理解实验内容，形成解题的基本思路，实验应独立完成，积极应用所学知识完成程序调试、测试，完成实验后认真填写实验报告，如实反映实验情况。

**实验设备与器材**

* 1. 仪器：PC机
  2. 软件：Windows 2000以上操作系统，Visual Studio 6.0 或以上开发平台

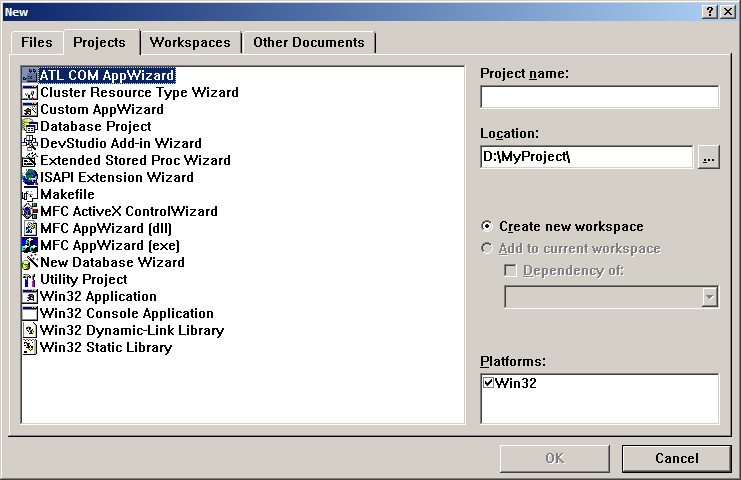
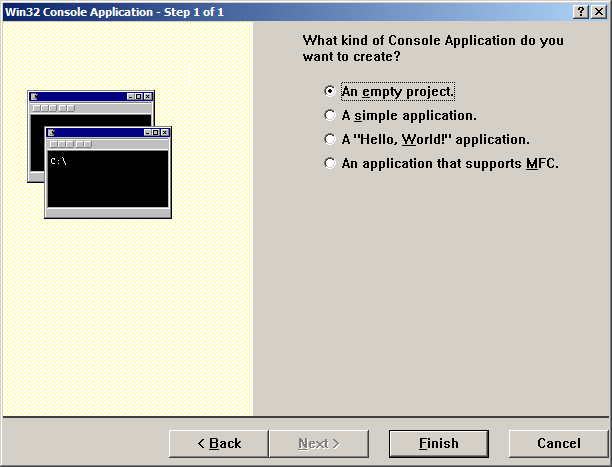
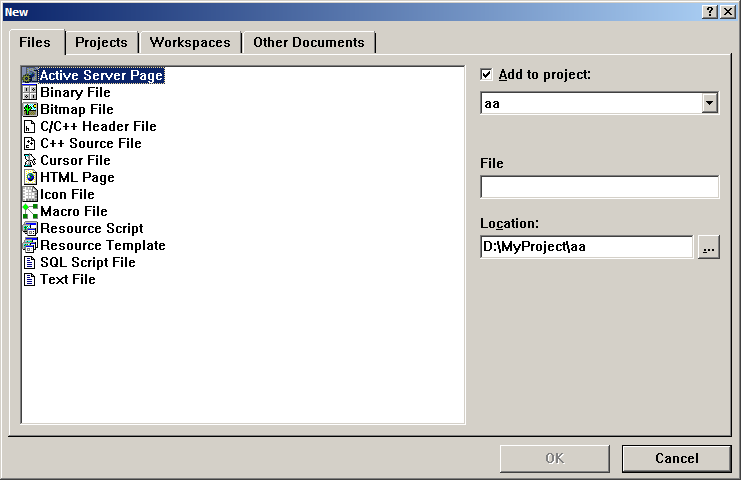
**核心实验原理**

1. 在已经提供的代码中定义了Employee类和Company类，Employee类具有grossPay成员，Company类实现了findBestPaid方法可以边输入Employee的信息边比较查找收入最高的雇员。如果希望将雇员的工资支付方式明确为按月支付，可以通过继承的方式定义新的Employee类和Company类，最大程度地复用已经存在的代码。Employee类的子类必须定义新的成员工作月数和月薪，此外要定义新的信息录入方式，实际上是通过对父类input方覆写（over-ride）实现；Company类的子类必须覆写findBestPaid方法。除此之外所有Employee父类和Company父类的成员和方法均可以继承。
2. 在Visual Studio中创建Console应用程序

（如果你已经很熟悉Visual Studio的使用，可以直接跳过这个部分。）

在本课程实验中开发的程序一般没有必要使用GUI（Graphic User Interface），使用命令窗口或字符界面就能够满足输入和输出的要求，同时把精力集中在数据结构、容器类定义和算法实现上。

下面以在Visual Studio中创建一个新的工程为例简单说明：

1. 启动Visual C++；
2. 选择 file| new…，出现如下对话框：  
   
3. 选择Win32 Console Application，并在右侧填入项目名称，选择项目存储目录，然后确定，随后出现Win32 Console Application向导。  
   
4. 选择“An empty project”，点击Finish, 就得到一个空的项目。
5. 下面点击 file | new 菜单，就会出现如下对话框：  
   
6. 根据需要在选择C/C++ header file (\*.h)或 C++ source file (\*.cpp)，在右侧输入文件名，建立文件，然后输入程序代码。  
   注意：应该在某个\*.cpp文件中包含main()函数，该项目才能正确运行。

**实验内容**

1. 阅读并理解给定的Employee类和Company类的代码，建立工程项目，编译并运行，验证findBestPaid功能。
2. 在给定的Employee类子类和Company类子类的代码框架上进行完善，实现两个子类缺少的方法（覆写父类方法）；在main函数中验证录入并查找按月付酬方式下收入最高雇员的功能。
3. 为Employee类的比较和拷贝功能重载>和=运算符，在此基础上改写Company类的findBestPaid功能，提高代码的可读性。

**实验报告**

1. 在实验设计和实验步骤中明确说明你要增加或改写的变量和语句；
2. 在实验报告中附上所实现方法的**核心**代码（非核心代码不要附）；
3. 在实验结果中展示所实现方法的测试效果；
4. 在实验总结中说明实验中遇到哪些问题，如何解决；实验的收获是什么；重点谈一下对面向对象编程中继承机制的优势和开闭原则的理解和认识。

**思考题**

1. 子类Company2继承了父类Company的Empoyee类型的成员bestPaid. 思考：为什么父类的bestPaid访问控制关键字定义为protected？在你完成本次实验的过程中，查找到的收入最高的雇员（按月付酬，Employee2类型）为什么能够赋值给bestPaid？

# 实验二 开发一个可应用于双向链表的Bidirectional Iterator

**实验目的**

1. 掌握容器类的基本思想。
2. 掌握定义容器类基本框架的方法。
3. 掌握容器中Iterator类的定义方法。

**预习与实验要求**

1. 实验前应仔细复习课程中关于Linked简单容器类的基本框架的内容，熟悉掌握Linked类的程序框架。
2. 复习迭代器（Iterator）类的基本概念和基本思想，掌握Linked类中迭代器（Iterator）类的实现方法。
3. 思考双向链表和单向链表的区别，其Iterator的行为有什么不同？
4. 实验前应该先理解实验内容，形成解题的基本思路，实验应独立完成，积极应用所学知识完成程序调试、测试，完成实验后认真填写实验报告，如实反映实验情况。

**实验设备与器材**

1. 仪器：PC机
2. 软件：Windows 2000以上操作系统，Visual Studio 6.0 或Visual Studio.Net开发平台

**实验原理**

1. 单向链表与迭代器Iterator

链表是指由包含数据和指针的结点集合并通过指针形成链接关系的数据结构，在课程中我们学习了如何以面向对象的方式建立一个链表类Linked，该类可以看成是一个很简单的容器类，其中包含了一个表示链表结点的结构，同时包含相应的头指针Head、表示链表长度的Length等属性，同时实现了一些相应的操作。我们把这个Linked类称为容器，主要原因是该类产生的对象可以存储各种类型的数据形成链表，所以该类定义为一个模板类，链表中的每个结点的数据域的类型为待定模板参数。

为了遵守数据抽象原则，外界需要访问像Linked这样的容器类的内部数据，但如果将数据设置为公有，将违反数据抽象原则。因此需要在容器类定义一个可以提供给外界访问容器内部数据的指针，它而且隐藏容器内部存储特性，为外界提供一个统一的访问方法例如\*（析取运算），就像一个数组指针P++能够指向下一个元素一样简单，从而实现各种容器操作的范型化（通用化）。这个“聪明”的指针还应该提供一些必要的运算如比较大小，判断相等、判断到达容器尾部等。因此这个指针应该实现为一个指针类，嵌套在容器类中表示属于这个容器，这就是所谓的Iterator类。我们可以使用Linked<int>::Iterator itr; 定义一个迭代器对象，然后像指针一样使用。

Linked容器类的总体框架如下，详细代码请参考教材和课程提供的源代码。

#ifndef LINKED

#define LINKED

#include <iostream>

using namespace std;

template<class T>

class Linked {

protected:

struct Node

{

T item;

Node\* next;

}; // struct Node

Node\* head;

long length;

public:

class iterator{

...

};

Linked()；

long size() const；

void push\_front (const T& newItem)；

void pop\_front()；

void operator= (const Linked<T>& otherLinked)；

Iterator begin()；

Iterator end()；

~Linked()；

}; // class Linked

#endif

1. 双向链表和双向迭代器（Bidirectional Iterator）

通过对Linked链表的学习，我们知道链表的每个结点包含一个指向后继结点的指针，我们可以从Head出发，遍历每一个结点。但是不可能从后一结点到达前一结点，因为结点之间是单向链接关系。如果每个结点除包含一个指向后继结点的指针，还包含指向前驱的指针，这样的链表就是双向链表。可以修改Linked类中的结构如下：

struct Node

{

T item;

Node\* next;

Node\* prev;

};

对于单向链表的Iterator，能够定义其++（自增）运算，通过后继指针移动到下一个结点，因为没有前驱指针所以不能完成--（自减）操作。显然双向链表的迭代器既能够完成++操作，也能够完成-- 操作，我们把能够两个方向移动的迭代器就称为双向迭代器（Bidirectional Iterator）。

**实验内容**

1. 参考单向链表Linked类，分析设计双向链表的结构，确定结点的结构和应该包含的属性，确定双向链表应提供的方法，分析双向迭代器应该包含的属性和方法。
2. 根据设计的结果，使用C++语言开发对应的双向链表头文件和源程序文件，通过编译发现并修正错误。
3. 开发一个main()函数，使用已开发的双向链表定义双向链表对象，调用相应的方法生成多个结点，并使用其双向迭代器完成正序输出和逆序输出。作为一个特例，基于实验一的Company和Employee类，实现Employee数据的录入和存储，以及逆向查找收入最高雇员的操作findBestPaidReverse.

**实验报告**

1. 在实验步骤中准确描述所设计的双向链表容器类的框架，说明其包含的结点结构、主要属性成员和主要支持的成员函数（方法）的接口等。
2. 在实验步骤中还应该描述所设计的迭代器类的属性、方法接口和重要方法的代码。
3. 可以在实验报告中附上使用双向链表类的main()函数代码。
4. 可在总结中说明程序调试和运行的结果是否正确，出现了哪些问题，是否解决及如何解决等。
5. 可以对双向链表和单向链表的优缺点进行分析，说明一些操作的差异。

**思考题**

1. 在链式容器中，为什么将容器类定义为其内部迭代器类的友元类？
2. 双向链表是否需要头指针Head，还需要一个尾指针Tail，才能实现双向遍历？如果只有头指针，如何逆序遍历整个链表？

# 实验三 Linked类的进一步扩充和时间复杂度分析

**实验目的**

1. 掌握简单链表容器的实现。
2. 逐渐学习算法时间复杂度分析的方法。
3. 学习掌握Runtime Analysis（运行时分析）的思想和技术。
4. 分析理论时间复杂度和实际运行时间的关系。

**预习与实验要求**

1. 实验前应仔细复习课程中关于Linked简单容器类的基本框架的内容，熟悉掌握Linked类的程序框架。
2. 复习Runtime Analysis的基本方法和程序代码。
3. 思考Linked容器这样的单向链表前端插入和后端插入的操作的区别。
4. 实验前应该先理解实验内容，形成解题的基本思路，实验应独立完成，积极应用所学知识完成程序调试、测试，完成实验后认真填写实验报告，如实反映实验情况。

**实验设备与器材**

1. 仪器：PC机
2. 软件：Windows 2000以上操作系统，Visual Studio 6.0 或Visual Studio.Net开发平台

**实验原理**

1. 时间复杂度

在课程中我们学习了时间复杂度和大O标记法，一个算法的时间复杂度往往和该算法中包含循环的次数有关。例如A操作本身的时间复杂度为常量，完成n次A操作的时间复杂度为常量还是O(n)？如果B操作的时间复杂度为O(n)，完成M次B操作的时间复杂度？

1. 单向链表的插入和删除操作

在Linked这样的单向链表中，每个结点只包含一个指向下一个节点的指针。在Linked类中我们已经实现了push\_front的操作（AddHead），这个方法的时间复杂度为常量。

如果考虑在Linked容器的后端进行插入（push\_back）和删除(pop\_back)操作，就需要首先完成遍历到链表的最后一个节点，然后通过指针调整，加入或删除结点。显然后端的插入和删除因为需要遍历链表而使时间复杂度增加。

1. Runtime Analysis（运行时时间分析）

虽然基于渐进时间复杂度的分析能够帮助我们总体上把握某个数据结构和算法的复杂度，但是在开发实践中，仍然需要针对某个机器配置和硬件条件去分析算法的实际运行时间情况，这就需要使用运行时时间分析。

分析的思想很简单，在耗时操作前计时，完成操作后再计时，两个时刻的差就是这些操作实际运行的时间，例如：

#include <iostream>

#include <time.h>

#include <string>

#include <stdlib.h>

#include "linked.h"

using namespace std;

int main ()

{

const string PROMPT =

"Please enter the number of integers to be push\_fronted: ";

const string TIME\_MESSAGE\_1 = "The elapsed time was ";

const string TIME\_MESSAGE\_2 = " milliseconds.";

const string CLOSE\_WINDOW\_PROMPT =

"Please press the Enter key to close this output window:";

Linked<int> myLinked;

long start\_time,

finish\_time,

elapsed\_time;

int n;

cout << PROMPT;

cin >> n;

**start\_time = clock();**

for (int i = 0; i < n; i++)

myLinked.push\_front (i);

// Calculate how long the task took:

**finish\_time = clock();**

elapsed\_time = finish\_time - start\_time;

cout << TIME\_MESSAGE\_1 << elapsed\_time

<< TIME\_MESSAGE\_2 << endl;

cout << endl << endl << CLOSE\_WINDOW\_PROMPT;

cin.get();

return 0;

} // test\_time

在上面的程序段中，我们使用已完成的Linked类定义了一个myLinked对象，然后前端插入i个结点，程序中带下划线的代码完成了开始计时和结束计时，最后输出插入结点所消耗的时间。

**实验内容**

1. 参考已完成的单向链表Linked类，为该类添加一个push\_back（AddTail）方法。
2. 设计一个main()函数，调用这个Linked类，能够根据用户输入生成指定个数的结点并插入链表。作为一个特例，参考实验一的Employee类和Company类，为Company设计实现两种录入员工数据的方法inputEmployeeHead(int n)和inputEmployeeTail(int n)，要求：
   * 1. n为录入方法参数，表示待录入的员工总数
     2. 根据参数n循环生成新的员工，其姓名格式为“Employee＋员工编号”，如“Employee123”，其总收入为随机生成的正整数；为此你需要为Employee类设置不同于以往的input方法。然后调用AddHead（inputEmployeeHead方法）或AddTail（inputEmployeeTail方法）将新员工插入链表。
3. 使用Runtime Analysis的方法，测试2中实现的inputEmployeeHead和inputEmployeeTail两种录入方法的时间开销。要求录入的员工总数n逐渐增大（例如：1000，10000，100000等），记录每一种总数n下两种方法的运行时时间数据。
4. 分析两组运行时时间数据差异的原因以及这种差异的规律。要求测试多组员工总数n对应的时间数据，同时绘制两种方法各自的耗时曲线图（纵轴为耗时，横轴为n）。对获取的数据进行曲线拟合并对结果给出理论分析。

**实验报告**

1. 在实验步骤中准确描述AddTail方法的实现过程。同时应给出main()运行时时间分析的主要代码。
2. 在测试数据及实验结果中给出所记录的两组插入在不同总数n下的执行时间。
3. 在实验分析中分析两组插入的理论时间复杂度，同时对所记录数据进行曲线拟合等分析，并与运行时时间数据进行对比，说明其代表的含义。
4. 总结本次实验遇到的问题和解决方法，指出仍然存在的疑问或问题。

**思考题**

1. 实验二要求实现一个双向链表和相应的迭代器。本实验不允许使用双向链表，只能使用单向链表，为什么？
2. 在本实验中，员工的总收入是随机生成的，为此需要设置随机数生成器的种子（seed），请思考这个设置放置在代码的什么位置为好？为什么？

# 实验四 递归回溯算法和面向对象通用回溯框架

**实验目的**

1. 掌握递归回溯算法的基本思想。
2. 学习掌握应用面向对象通用回溯程序框架解决实际问题。
3. 提高面向对象编程的技能。

**预习与实验要求**

1. 实验前应仔细复习课程中关于回溯法的基本思想和示例。
2. 复习面向对象的回溯法程序框架，分析递归和回溯的过程。
3. 实验前应该先理解实验内容，形成解题的基本思路，实验应独立完成，积极应用所学知识完成程序调试、测试，完成实验后认真填写实验报告，如实反映实验情况。

**实验设备与器材**

1. 仪器：PC机
2. 软件：Windows 2000以上操作系统，Visual Studio 6.0 或Visual Studio.Net开发平台

**实验原理**

1. 回溯法

在课程中我们已经学习过回溯法，这是一个通过指定规则采用不断尝试并在必要时回退的探索型算法，这个算法适合表示为递归算法，学习回溯法时常常解决迷宫、路径方面的问题。针对不同的问题，我们需要建立不同的问题模型，以便在程序中表达要解决的问题，然后使用回溯法探索解法。

回溯法的递归和回溯的过程，实际上就是一般递归程序的递归和返回的过程。

1. 面向对象回溯法程序框架（Framework）

对于不同的实际问题，回溯法在递归和回溯的代码上没有什么不同，不同的是问题表示、递归和回溯的具体条件。因此我们学习了一个面向对象的回溯法程序框架，该框架包含一个通用的backtrack类(backtrack.h, backtrack.cpp)，实现了通用的递归和回溯的代码，同时该框架还提供了一个针对不同问题的方法接口定义和迭代器接口定义（application.h），针对不同问题，我们需要实现这些方法接口和迭代器接口（application.cpp），以便于使用backtrack类。此外，该框架还要求针对不同问题实现相应的position类（position.h, position.cpp），用于表示具体问题中构成路径的某个位置。

详细的程序框架参考教材P117迷宫应用的例子，源代码教师将在理论课后提供。

1. 回溯问题的表示

在回溯法解决的实际问题中，不同的问题我们需要不同的数据结构来存储、表示。我们学习的迷宫应用例子中，采用了一个二维数组来存储迷宫，数组元素值代表走廊、墙、死路、路径等。在本实验中，我们也只需要使用数组这样的简单数据结构就能够存储、表示相应的问题。

**实验内容**

1. 八皇后问题：在一个8×8的国际象棋棋盘上放置8个皇后，要求每个皇后不能被其它皇后攻击到（在国际象棋中皇后能够攻击8个方向、任意距离）。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  | Q |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

1. 在回溯法程序框架的基础上，开发实现application.h中的方法接口和迭代器接口，源程序文件可以是application.cpp或者自主命名。开发实现Position类的头文件和源程序文件。
2. 开发一个main()函数，使用数组存储棋盘，实例化backtrack、application类，输出棋盘和8个皇后的放置结果。

**实验报告**

1. 在实验步骤中准确描述application.h中valid(), record(), done(), undo()等方法的实现思想，并给出迭代器自增运行的实现代码。同时应给出main()中表示棋盘和皇后的方案。
2. 在测试数据及实验结果中给出第一个皇后的位置后，输出的棋盘和皇后的位置。
3. 在实验分析中说明在八皇后问题中什么时候发生回溯。
4. 总结本次实验遇到的问题和解决方法，指出仍然存在的疑问或问题。

**思考题**

1. 如何修改使你的程序以找出这个棋盘上所有可能的八个皇后放置位置？
2. 在Application的内部类Iterator中封装的无类型指针其作用是什么？

# 实验五 通用顺序容器List类的完整设计和实现

**实验目的**

1. 进一步提高顺序容器的开发应用能力。
2. 掌握链表数据结构的各种操作方法与编程。
3. 学习通用容器的存储设计和操作设计，加深Iterator的应用和开发能力。

**预习与实验要求**

1. 实验前应仔细复习课程中关于List容器的存储结构和操作。
2. 学习教材6.1.6节关于list类的另一种实现思想，了解此实现的特点。
3. 实验前应该先理解实验内容，形成解题的基本思路，实验应独立完成，积极应用所学知识完成程序调试、测试，完成实验后认真填写实验报告，如实反映实验情况。

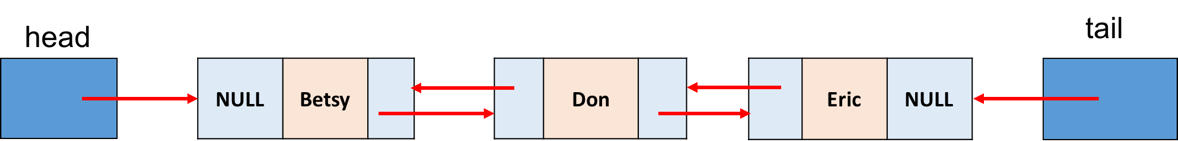
**实验设备与器材**

1. 仪器：PC机
2. 软件：Windows 2000以上操作系统，Visual Studio 6.0 或Visual Studio.Net开发平台

**实验原理**

1. List容器

List也被称为Linked list（链表），它和数组、向量、双端队列等都被称为顺序容器。链表的优势在于不需要大块的连续存储空间，存储灵活，端点处的插入、删除操作时间复杂度为常量；主要缺点是缺乏随机访问能力。为了使链表功能更强大，本实验的list容器被实现为双向环状链表，存储形式如下：



在以上实现中，存在一个头指针和一个尾指针，能够快速到达链表头和链表尾。

1. 在设计和实现中保持容器的通用性

我们已经在课程中学了STL的多个容器，它们都是通用容器。为了保持通用性，一般采用下列设计和实现原则：

* 1. 采用模板类保持存储数据的通用；
  2. 使用Iterator向外提供统一的访问接口；
  3. 类型的范型表示，容器内部数据结构也尽量采用通用形式，如value\_type等；
  4. 提供完善的操作、方法。

**实验内容**

1. 设计实现一个双向链表容器，可采用头-尾指针的结构。
2. 设计该容器较完善的操作和方法，并实现其迭代器类及其方法。
3. 开发一个程序使用完成的双向链表容器，测试其正确性。

**实验报告**

1. 在实验步骤中采用图示说明你设计的容器的存储结构，并叙述所设计的方法和迭代器的操作和功能。
2. 在测试数据及实验结果中给出调用程序的思路，给出程序执行的结果。
3. 在实验分析中分析所实现的容器中pop\_front, pop\_back, erase方法的时间复杂度。
4. 总结本次实验遇到的问题和解决方法，指出仍然存在的疑问或问题。

**思考题**

1. 请调研C++标准模板库STL中list容器的实现原理，并对比本实验中你的实现，分析在技术上和功能上的异同点。

# 实验六 扩展洗车仿真

**实验目的**

1. 培养针对实际应用问题建模、表示和实现的能力。
2. 训练应用队列容器进行开发的能力。
3. 掌握随机数的原理和应用。

**预习与实验要求**

1. 实验前应仔细复习课程中关于queue容器的存储结构和操作。
2. 复习教材中关于洗车仿真的例子，熟悉问题模型，深入理解关于车辆到达、离开的模拟方法。
3. 实验前应该先理解实验内容，形成解题的基本思路，实验应独立完成，积极应用所学知识完成程序调试、测试，完成实验后认真填写实验报告，如实反映实验情况。

**实验设备与器材**

1. 仪器：PC机
2. 软件：Windows 2000以上操作系统，Visual Studio 6.0 或Visual Studio.Net开发平台

**实验原理**

1. 洗车仿真

洗车这样的排队问题是典型的队列应用，其难点在于如何表示车辆的到达、洗车、离开等一系列事件。通过理论课程的学习，我们已经知道车辆的排队可以通过queue容器很方便的表示，为了表示事件，在队列中排队的车辆实际表示为其到达时间（事件），这种忽视无关特性（车辆的具体信息）的方法在问题建模中常常使用。在理解时，还应注意借助于currentTime, nextArrivalTime, nextDepartureTime几个变量，实现了车辆到达、离开的有序处理，并最终实现了CarWash类。

1. 泊松分布的基本公式

F(x)=exp(-x/meanArrivalTime), 若timeTillNext为下一辆车到达的时间，则

F(timeTillNext)=EXP(-timeTillNext/meanArrivalTime)。

1. 产生0-1之间随机数的方法

randomDouble=rand()/double(rand\_max+1)

可以得出根据平均到达时间meanArrivalTime随机产生到达时间的公式：

timeTillNext=floor(-meanArrivalTime\*log(1-randomDouble)+0.5)

1. 处理由平均服务时间产生随机服务时间

采用上面相同的方法可以处理由平均服务时间产生随机服务时间，在洗车仿真的程序框架改造后使用，能够输入平均到达时间，平均服务时间和最大到达时间，达到最大到达时间程序就结束。

**实验内容**

1. 使Speedo的洗车仿真更接近真实。根据泊松分布，到达时间应当由平均到达时间随机产生。
2. 增加一个新的特点，服务（洗车）时间不必是10分钟，而依赖于顾客的需要，例如只清洗、清洗加打腊或清洗并吸尘。
3. 为实现以上功能改造已有的CarWash类，并开发相应的主程序进行模拟。

**实验报告**

1. 在实验步骤中详细说明你对已有的CarWash类的那些地方进行了改造，可给出部分关键代码。
2. 在测试数据及实验结果中给出相应的模拟输入数据，并给出程序执行的结果。
3. 在实验分析中分析采用符合泊松分布随机到达时间比随手输入的到达时间有什么好处。
4. 总结本次实验遇到的问题和解决方法，指出仍然存在的疑问或问题。

**思考题**

1. 假设你仿真获得了一些洗车服务的指标，如平均等待时长，最大队列长度，等等，发现这些指标不满足洗车公司的设计需求，不能为顾客提供较好的服务。这时，一种改进的策略是增设一个服务台，用两个服务台同时提供汽车清洗服务。那么，怎么修改你的程序能够为这种双服务台系统进行仿真分析？

# 实验七 Binary Search Tree的平均树高

**实验目的**

1. 掌握Binary Search Tree的实现思想和方法。
2. 训练树型结构的开发应用能力。
3. 培养理论联系实际的学习能力，提高对理论的实验验证能力。

**预习与实验要求**

1. 实验前应仔细复习课程中关于Binary Search Tree的基本理论和结论。
2. 复习教材中关于关于Binary Search Tree（BinSearchTree）的存储设计和实现方法。
3. 实验前应该先理解实验内容，形成解题的基本思路，实验应独立完成，积极应用所学知识完成程序调试、测试，完成实验后认真填写实验报告，如实反映实验情况。

**实验设备与器材**

1. 仪器：PC机
2. 软件：Windows 2000以上操作系统，Visual Studio 6.0 或Visual Studio.Net开发平台

**实验原理**

1. 二叉树的树高

通过对二叉树理论的学习，我们知道满树的树高height(t)与结点数成对数关系，在EX. 8.7中我们还证明了完全二叉树的树高也和结点树成对数关系。当一颗二叉树退化为Chain（链）时，其树高显然和结点数成线性关系。由于二叉树的很多算法的复杂度与树高相关，因此树高决定了很多算法的时间复杂度。

Binary Search Tree是非常重要的二叉树，由于其有序的特点，可以广泛应用于很多对查找和搜索效率要求高的应用中。Binary Search Tree的树高有规律吗？显然，它可以是满树（树高与结点数成对数关系），也可以是链（树高与结点数成线性关系），因此我们在第8章中声明Binary Search Tree的平均树高与结点数成对数关系。除了可以推导证明外，我们还可以通过实际的实验结果来支持这一结论。

1. 树高的计算

if t is empty,

height (t) = -1;

otherwise,

height (t) = 1 + max (height (leftTree (t)), height (rightTree (t))

利用已有的BinarySearchTree类，可以把height()作为类中的方法，例如:

binSearchTree myTree;

myTree.height()

1. 产生0-1之间随机数的方法

randomDouble=rand()/double(rand\_max+1)

**实验内容**

1. 利用已有的BinSearchTree类（bst.h）的代码，开发一个完整的Binary Search Tree的容器。
2. 对于任给一结点数n，调用BinSearchTree类产生20颗结点数为n而各结点值为随机实数（0-1之间）Binary Search Tree，利用height()方法计算其树高。
3. 计算20颗数的平均树高，同时计算log2n，对比分析它们的关系。

**实验报告**

1. 在实验步骤中详细说明对树高计算的实现方法，并简单说明主控程序的结构和操作。
2. 在测试数据及实验结果中给出相应的每颗树的树高、平均树高、log2n等数据。
3. 在实验分析中说明为什么各结点值设置为随机数，并分析数据之间的关系，并得出相应结论。
4. 总结本次实验遇到的问题和解决方法，指出仍然存在的疑问或问题。

**思考题**

1. 如何推导证明Binary Search Tree的平均树高与结点数成对数关系？

# 实验八 最短路算法及其应用

**实验目的**

1. 掌握图型数据结构基于邻接表和邻接矩阵的设计思想和实现方法。
2. 掌握最短路算法(Dijkstra算法)的原理和实现方法。
3. 训练图型结构的开发应用能力。

**预习与实验要求**

1. 实验前应仔细复习课程中关于Graph的基本概念、存储设计和实现方法。
2. 实验前应仔细复习前期课程所学C++中map、priority queue、list、pair等数据结构的基本操作。
3. 复习教材中关于关于求最短路问题的Dijkstra算法的基本原理。
4. 实验前应该先理解实验内容，形成解题的基本思路，实验应独立完成，积极应用所学知识完成程序调试、测试，完成实验后认真填写实验报告，如实反映实验情况。

**实验设备与器材**

1. 仪器：PC机
2. 软件：Windows 2000以上操作系统，Visual Studio 6.0 或Visual Studio.Net开发平台

**实验原理**

1. 图的存储方式

见教材14.7.1节Adjacenty List的存储方式和教材14.7.8节Adjacency Matrix的存储方式。

1. 最短路问题的Dijkstra算法

见教材14.5.4节的理论讲解和14.7.6节的代码展示，结合幻灯片中的示例理解。

**实验内容**

1. 利用已给定的Graph类的部分代码，实现求最短路问题的Dijkstra算法。该算法的输入是路径的起点顶点和目的地顶点；如果存在这样一条路径，则返回路径上所有顶点的列表，以及路径上所有边的权重之和；如果不存在，则返回空列表，返回的路径长为 -1.
2. 利用校园电子地图和百度地图，选定校园内经常到访的标志性位置，总数应不少于20个。以所选定标志性位置及其相互间连接构建图的结构并通过计算机存储，位置间的距离可使用百度地图测距功能完成。在此基础上计算校园内任意两点间的最短距离。

**实验报告**

1. 在实验步骤中详细说明最短路算法的设计原理，所采用的数据结构和具体的实现过程。
2. 在实验步骤中详细说明构建校园地图时顶点的选择、边的选择和权重的设定过程。
3. 在测试数据及实验结果中展示最短路算法的测试结果、校园地图的构建结果和两个位置间最短距离的计算结果。
4. 总结本次实验的收获、遇到的问题和解决方法，指出仍然存在的疑问或问题。

**思考题**

1. 理论上，Dijkstra算法不能应用于带有负权边的图，为什么？
2. 本次实验所提供代码实现了一个网络类，即带权的有向图。而校园地图通常是无向图，怎么实现用网络类存储一个无向图？