

**Lab Report**

**实验报告**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2024-2025** |
| **Major**: | Software Engineering |
| **Class**: | 2023 |
| **Student Name**: | 吴仲雄 |
| **Student ID:** | 222023321062007 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Time Complexity and Runtime Analysis  时间复杂度和C++运行时分析 | | | |
| Date | | Oct，2024 | Type | | ☑Confirmatory （验证确认型）  ☑Design（设计型）  🗆Comprehensive（综合型） |
| 1. **Objective & Requirements（实验目的）**    1. Understand the theoretical time complexity of an algorithm and know how to analyze it   理解算法理论复杂度的概念和表示方法，掌握算法理论复杂度的分析方法   * 1. Grasp the use of random numbers and techniques for measuring execution time in C++   掌握在C++中使用随机数生成器和计时工具进行算法运行时时间复杂度的测量方法   * 1. Grasp runtime analysis of programs to show the effect of theoretical complexity on the time cost of real programs by running programs and measuring the time cost   掌握对大量实验数据进行统计和理论分析的技术，能够从实验上验证算法的理论时间复杂度 | | | | | |
| 1. **Experimental environment (**platform and software**)（实验环境）**   Windows 7 (or higher versions) + Visual Studio 2010 (or higher versions) | | | | | |
| 1. Experimental content and design (Main Content, Procedure, Codes and Results)（此部分应包含每一个实验内容的详细设计，含实验思路、详细实验步骤、核心代码说明等） 2. Task 1 （任务1）   You are provided with a template container based on singly linked list. Please read the source code and implement a new method addTail() that can add a new element at the end of the linked list. Note that your are NOT allowed to modify any existing codes of the lined container, including adding new data or function member.  请阅读并理解所给单向链式模板容器，为其实现一个新的元素添加方法addTail()以实现在链式容器的末尾添加新的元素。注意，在实现addTail()方法时，请不要修改该容器中任何已经存在的代码，包括添加新的数据或方法成员。  实现代码：     1. Task 2 （任务2）   Please rewrite the input() method for the Employee class. The new input() method takes a parameter int id, and set the calling employee with name “employee+id”, e.g. “employee123”, and with a randomly generated integer as the grosspay.  为Employee类改写input方法。新的input方法接收一个整型参数id，将调用者的姓名置为“employee+id”，例如“employee123”，同时将其总收入置为一个随机生成的自然数。  实现随机数应包含新的头文件：    实现代码：     1. Task 3 （任务3）   Please implement two methods for the Company class i.e.   * + - * void inputEmployeeHead(int total\_num);       * void inputEmployeeTail(int total\_num);   The integer total\_num is the method argument that specifies the total number of employee to input. The information of each input employee is set by the method in Task 2. inputEmployeeHead() is based on addHead() and inputEmployeeTail() is based on addTail() that is implemented in Task 1.  为公司类实现inputEmployeeHead()和inputEmployeeTail()方法，这两个方法均接收一个整型数total\_num为参数，表示待录入信息的员工的总数；每一个员工的信息使用任务2的input方法录入；要求inputEmployeeHead()基于容器类的addHead()方法实现，inputEmployeeTail()基于上述任务1（Task 1）实现的容器类的addTail()方法实现。  实现代码：     1. Task 4 （任务4）   Using runtime analysis to measure the time costs of inputEmployeeHead() and inputEmployeeTail() by increasing the total number of employee total\_num, e.g., 1000, 2000, …, 10000, 20000, …, 100000, and so on. Record the time costs of the two methods for each value of total\_num. Plot the data in a figure and try to fit the data using a curve (数据拟合，曲线拟合).  Analyze the theoretical time complexity of inputEmployeeHead and inputEmployeeTail. Compare your theoretical analysis to the experimental data you obtained.  使用运行时分析工具统计inputEmployeeHead() 和 inputEmployeeTail()方法的耗时；设置不同的参数测试并记录两个算法的多组时间数据。使用所得数据绘图并进行数据拟合，比较你获取的实验分析结果和两个算法的理论复杂度分析结果。  main.cpp:  #include "listTemp.h"  #include "company.h"  #include <ctime>  #include <iostream>  using namespace std;  void TimeTest(Company& cmp, int total\_num)  {  //runtime analysis  clock\_t start\_t, end\_t, elapsed\_t;  start\_t = clock();  cmp.inputEmployeeHead(total\_num);  end\_t = clock();  elapsed\_t = end\_t - start\_t;  cout << "Clock ticks per second: " << CLOCKS\_PER\_SEC << endl;  cout << "Time elapsed for inputEmployeeHead: " << elapsed\_t << " (in ticks)" << endl;  start\_t = clock();  cmp.inputEmployeeTail(total\_num);  end\_t = clock();  elapsed\_t = end\_t - start\_t;  cout << "Time elapsed for inputEmployeeTail: " << elapsed\_t << " (in ticks)" << endl;  cout << endl;  }  int main()  {  Company cmp;  //please test your implementation  int total\_nums[] = {1000, 2000, 5000, 10000, 20000, 50000};  for (int i = 0; i < 6; i++)  {  TimeTest(cmp, total\_nums[i]);  }  system("pause");  return 0;  }  测试结果：  在插入1000, 2000, 5000, 10000, 20000, 50000个员工信息时程序的分别耗时：    曲线拟合： | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）此部分应包含实验结果，对实验结果的分析，实验收获的总结，实验中存在问题的讨论等；另外，需要回应一下如下思考题：   1. Lab1中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？  由实验结果和理论分析，inputEmployeeHead()方法的时间复杂度为O(n), inputEmployeeTail()方法的时间复杂度为O(n2)。  思考题：   1. 因为在单向链表中，添加节点到头部的时间复杂度为 O(1)，而添加节点到尾部的时间复杂度为 O(n)。在双向链表中，添加节点到尾部的时间复杂度与添加到头部的时间复杂度一致为O(1)，无法达到比较在头尾插入节点的时间复杂度差异的实验目的。 2. 在本实验中，设置随机数生成器的种子（seed）是为了确保每次运行程序时生成不同的随机数。通常，设置随机数生成器的种子应该放在程序的初始化阶段，这样可以确保在整个程序运行期间，随机数生成器只被初始化一次。 | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |