

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2019-2020** |
| **Major**: | Software Engineering |
| **Class**: | 2018 |
| **Student Name**: | SONG,Xingjian（宋行健） |
| **Student ID:** | 222019321062006 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Iterator and Time Complexity | | | |
| Date | | Nov 21，2019 | Type | | √ Confirmatory  √ Design  □Comprehensive |
| 1. **Objective & Requirements**    1. Know the use of iterator and understand its implementation details    2. Can use iterator to traverse a list in the reverse order to finish a certain task    3. Understand the time complexity and know how to analyze it; show the effect of theoretical complexity on the time cost of real programs by running programs and measuring the time cost | | | | | |
| 1. **Experimental environment (**platform and software**)**   Windows 7 (or higher versions) + Visual Studio 2010 (or higher versions) | | | | | |
| 1. **Experimental content and design** (Main Content, Procedure, Codes and Results)   Task 1   * Using the source code sent to you about the linked list template with iterator, implement three methods   + operator--(int) for the iterator inner class   + Begin() for the container class   + End() for the container class   + Based on the container and iterator class, implement the method findBestPaidReverse() for the Company class that find the best paid employee by traversing the list of employees in the reverse order * Test your implementation in the main() method   Task 2   * Based on the source code sent to you, implement   the following two methods for the Company class   * + findBestPaid()   + findBestPaidReverse()   **!!!**Note that the container class is implemented in Task 1 so is not provided here**!!!**   * Test your implementation in main() and measure the time   costs of both methods by setting the two constants in  company.h   * + NUM\_EMPLOYEE   + MAX\_SALARY   Task1:   * 首先定义Begin和End 函数，Begin的head和curr两个参数均为链表的head位置，End函数的两个参数一个为head，而curr参数为NULL。   template<class T>  typename ListTemp<T>::Iterator ListTemp<T>::Begin() const  {  return Iterator(head, head);  }  template<class T>  typename ListTemp<T>::Iterator ListTemp<T>::End() const  {  return Iterator(head,NULL);  }   * 其次定义operator—函数，先创建一个迭代器类型的位置在链表头的临时变量，然后用一个循环进行查找，当迭代器的下一个节点是当前节点的时候跳出循环，返回当前迭代器，并且this指针指向迭代器所指位置。   template<class T>  typename ListTemp<T>::Iterator ListTemp<T>::Iterator::operator--(int)  {  Iterator temp = Iterator(head, head);  while (!(temp.curr->next == this->curr))  {  temp++;  }  \*this = temp;  return temp;  }   * 逆向查找函数，创建一个在End位置前一位的迭代器，让它依次与最高薪水作比较，当到链表起始位置时跳出循环。   void Company::findBestPaidReverse()  {  bestPaid = Employee();  ListTemp<Employee>::Iterator itr = (container.End()--); //最后一位没有数据，所以这里要进行自减操作，否则第一次比较会报错  while (!(itr == container.Begin()))  {  if (\*itr > bestPaid)  bestPaid = \*itr;  itr--;  }  }  if (\*itr > bestPaid)  bestPaid = \*itr;  Task2:  将第一问的代码粘贴到第二问后，根据我的电脑性能，多次调试后将样本数调到10000，最大工资调到100。 | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   Task1:     * 定义Begin 和End函数的时候，要注意是两个参数，第一个参数均以链表起始节点传入，第二个参数是curr是当前位置指针，Begin将head位置传入，End则将NULL位置传入； * 定义自减操作时要创建一个迭代器类型的临时变量，在判断是否为前一位置时，要用temp中的curr结构体中的next指针与this中的curr作比较。另外要注意的是在返回之前，要将this的位置设置为前一节点的位置，这样才可以进行连续自减； * 在逆向查找最高工资时，注意查找要从最后一位有数据的节点开始，而End返回的是NULL，所以在比较之前要进行一次自减运算。 * 循环在到Begin的位置跳出，所以Begin位置的数据并没有参与比较，所以要在循环外面单独对Begin位置的数据与bestpaid作比较。 * 在自减运算时，通过逐语句调试，发现return temp并没有用，这一点不是很明白原因。   Task2:     * 由结果可知，逆向查找的所消耗的时间远远大于顺序查找，由此可知逆向查找的时间复杂度更大，因为它每往前走一个节点就要进行一次循环。 * 另外，通过这次实验熟悉了程序计时的方法，通过对程序进行计时可以比较他们运行的速度和时间复杂度。 | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |