

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2020-2021** |
| **Major**: | Software Engineering |
| **Class**: | 2019 |
| **Student Name**: | Fu Ruoxuan |
| **Student ID:** | 222019321062060 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Iterator and Time Complexity | | | |
| Date | | Nov 27，2020 | 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   1. Using the source code sent to you about the linked list template with iterator, implement three methods    1. operator--(int) for the iterator inner class    2. Begin() for the container class    3. End() for the container class    4. 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 2. Test your implementation in the main() method   Task 2   1. Based on the source code sent to you, implement   the following two methods for the Company class   * 1. findBestPaid()   2. findBestPaidReverse()   **!!!**Note that the container class is implemented in Task 1 so is not provided here**!!!**   1. Test your implementation in main() and measure the time   costs of both methods by setting the two constants in  company.h   * 1. NUM\_EMPLOYEE   2. MAX\_SALARY   Task1 code   1. **template**<**class** T> 2. **typename** ListTemp<T>::Iterator ListTemp<T>::Iterator::operator--(**int**) 3. { 4. Iterator temp = \***this**; 5. **for** (Iterator itr(head,head); !(itr == \***this**); itr++) 6. { 7. **if** ((itr.curr)->next == **this**->curr) 8. { 9. **this**->curr = itr.curr; 10. **return** temp; 11. } 12. } 13. } 14. **template**<**class** T> 15. **typename** ListTemp<T>::Iterator ListTemp<T>::Begin() **const** 16. { 17. **return** Iterator(**this**->head,**this**->head); 18. } 20. **template**<**class** T> 21. **typename** ListTemp<T>::Iterator ListTemp<T>::End() **const** 22. { 23. **return** Iterator(**this**->head,nullptr); 24. } 25. **void** Company::findBestPaidReverse() 26. { 27. bestPaid = Employee(); 28. ListTemp<Employee>::Iterator itr = container.End(); 29. **while** (!(itr==container.Begin())) 30. { 31. itr--; 32. **if** (\*itr > bestPaid) 33. bestPaid = \*itr; 34. } 35. }   Task1 result    Task2 code   1. **void** Company::findBestPaid() 2. { 3. bestPaid = 0; 4. ListTemp<**int**>::Iterator itr = container.Begin(); 5. **while** (!(itr == container.End())) 6. { 7. **if** (\*itr > bestPaid) 8. bestPaid = \*itr; 9. itr++; 10. } 11. } 13. **void** Company::findBestPaidReverse() 14. { 15. bestPaid = 0; 16. ListTemp<**int**>::Iterator itr = container.End(); 17. **while** (!(itr == container.Begin())) 18. { 19. itr--; 20. **if** (\*itr > bestPaid) 21. bestPaid = \*itr; 22. } 23. } 24. **template**<**class** T> 25. **typename** ListTemp<T>::Iterator ListTemp<T>::Iterator::operator--(**int**) 26. { 27. Iterator temp = \***this**; 28. **for** (Iterator itr(head,head); !(itr == \***this**); itr++) 29. { 30. **if** ((itr.curr)->next == **this**->curr) 31. { 32. **this**->curr = itr.curr; 33. **return** temp; 34. } 35. } 36. } 37. **template**<**class** T> 38. **typename** ListTemp<T>::Iterator ListTemp<T>::Begin() **const** 39. { 40. **return** Iterator(**this**->head,**this**->head); 41. } 43. **template**<**class** T> 44. **typename** ListTemp<T>::Iterator ListTemp<T>::End() **const** 45. { 46. **return** Iterator(**this**->head,nullptr); 47. }   Task2 result  NUM\_EMPLOYEE 50000  NUM\_EMPLOYEE 10000 | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   Analysis of experimental results   * In task 1, without using a bidirectional list, the method of accessing the previous node of the first node is traversed from the beginning, and if the pointer field to the node is pointed to the current node, the pointer to the current node is assigned to the traversal pointer, thus overloading the operation of the self-detract operator. * Using the method of task 1, we can only change the type of the template class to int.   Harvest  Time Complexity  In computer science, the time complexity of an algorithm is a function that qualitatively describes the running time of the algorithm. This is a function that represents the length of the string in which the algorithm enters values. Time complexity is often expressed as a large O symbol, excluding the lower order and first coefficient of this function. In this way, the time complexity can be described as approaching, i.e. when the input value size is approaching infinity.  Iterator  In task 1, I made a mistake that I try to put subtraction after assignment. This operation may cause that the loop cannot meet the end.    Problems  Insufficient proficiency in the use of dereferencing and class member access operators. | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |