

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2020-2021** |
| **Major**: | Software Engineering |
| **Class**: | 2019 |
| **Student Name**: | 冯春霖 |
| **Student ID:** | 222019321062074 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Iterator | | | |
| Date | | November 6，2020 | 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 to finish a certain task | | | | | |
| 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 the method in the company class   * + - findBestPaid()   and test your implementation in the main function.  Task 2   * + - 1. Based on the source code of the container template using array storage, implement an iterator inner class for it       2. Based on the implemented iterator, implement the findBestPaid() method for the Company class, just as what you do in Task 1   Task 1：  Implement of findBestPaid() in company.cpp:  void Company::findBestPaid()  {  ListTemp<Employee>::Iterator itr(container.Begin());  bestPaid = \*itr;  while (true)  {  if (\*itr > bestPaid)  bestPaid = \*itr;  itr++;  if (itr == container.End())  break;  }  }  Test in main.cpp:    Task 2:  Implement of contTemp.h:  Implement of the inner class:  template<class T>  ContTemp<T>::Iterator::Iterator()  {  curr\_ptr = NULL;  }  template<class T>  ContTemp<T>::Iterator::Iterator(T\* ptr)  {  curr\_ptr = ptr;  }  template<class T>  typename ContTemp<T>::Iterator ContTemp<T>::Iterator::operator++(int)  {  Iterator temp = \*this;  this->curr\_ptr = curr\_ptr++;  return temp;  }  template<class T>  T& ContTemp<T>::Iterator::operator\*() const  {  return (\*curr\_ptr);  }  template<class T>  bool ContTemp<T>::Iterator::operator==(const Iterator other) const  {  return curr\_ptr == other.curr\_ptr;  }  Implement of Begin() and End():  template<class T>  typename ContTemp<T>::Iterator ContTemp<T>::Begin() const  {  return Iterator(&elemArray[0]);  }  template<class T>  typename ContTemp<T>::Iterator ContTemp<T>::End() const  {  return Iterator(&elemArray[size]);  }  Implement of findBestPaid():  void Company::findBestPaid()  {  ContTemp<Employee>::Iterator itr(container.Begin());  bestPaid = \*itr;  while (true)  {  if (\*itr > bestPaid)  bestPaid = \*itr;  itr++;  if (itr == container.End())  break;  }  }  Test in main.cpp: | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   I was able to complete the experimental requirements and achieve the experimental objectives in this experiment. In this experiment, I have a deeper understanding of iterators, and can implement chain table iterator and array iterator, and distinguish the difference between them, which is very helpful for me to use iterators to complete programs in the future. There are no unsolved problems left from this experiment. | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |