

**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 | | C++ Tempaltes | | | |
| Date | | Oct 29，2020 | Type | | √ Confirmatory  √ Design  □Comprehensive |
| 1. **Objective & Requirements**    1. Understand the concept of containers; can use template to define generic containers    2. Understand the concept of generic algorithms; use generic method to implement container template    3. Understand the difference between contiguous memory allocation and linked memory allocation; use template to implement a container with linked storage | | | | | |
| 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) 2. Task 1    1. In the template container class sent to you, implement a method remove(i) that can remove the item of index **i** in the container (Note that the first item is of index 0)    2. Based on remove(i), implement a method removeEmployee() in the company class that allows the user to input an index i and then remove the employee of index **i.** (Note that the first employee is of index 0)    3. Test your implementation in the main() function 3. Task 2    1. Implement a method for adding a new element at the head of the linked list for the container template with linked storage  * void AddHead()   1. Based on AddHead(), implement the inputEmployee() method for the company class   2. Test your implementation in the main() function, e.g. using printLength().  1. Task 3    1. Implement a method for adding a new element at the tail of the linked list for the container template with linked storage  * void AddTail()   1. Based on AddTail(), rewrite the inputEmployee() method for the company class   2. Test your implementation in the main() function, e.g. using printLength().   **1）task 1**  Code of Remove() in contTemp.h:  template<class T>  void ContTemp<T>::Remove(const int i)  {  for (int j = i; j < size - 1; j++)  {  elemArray[j] = elemArray[j + 1];  }  T temp;  elemArray[size - 1] = temp;  size--;  }  Code of removeEmployee() in company.cpp:  void Company::removeEmployee()  {  int i;  cout << "\nPlease input the index (starting from 0) of the employee to be removed: ";  cin >> i;  container.Remove(i);  }  Test in main.cpp:    **2）task 2**  Code of AddHead() in listTemp.h:  template<class T>  void ListTemp<T>::AddHead(const T& newData)  {  Node\* newNode = new Node;  newNode->data = newData;  newNode->next = head;  head = newNode;  size++;  }  Code of inputEmployee() in company.cpp and test in main.cpp:    **3）task 3**  Code of AddTail() in listTemp.h:  template<class T>  void ListTemp<T>::AddTail(const T& newData)  {  Node\* newNode = new Node;  newNode->data = newData;  newNode->next = NULL;  if (head == NULL)  head = newNode;  else  {  Node\* itr = head;  while ((itr->next) != NULL)  itr = itr->next;  itr->next = newNode;  }  size++;  }  Code of inputEmployee() in company.cpp and test in main.cpp: | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   Through this experiment, I was able to understand the container class and implement a simple template container class. At the same time, I have an understanding of generic algorithms and generic methods. In the process of implementing the container class, I understood the method of creating discontinuous storage in the memory space by LinkedList and how it differs from the continuous storage method like arrays. In the process of implementing the code. There were no leftover unsolved problems 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: | | | | |