

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2020-2021** |
| **Major**: | Software Engineering |
| **Class**: | 2019 |
| **Student Name**: | Ruoxuan Fu |
| **Student ID:** | 222019321062060 |
| **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. Procedure 1 2. **template**<**class** T> 3. **void**  ContTemp<T>::Remove(**int** index) 4. { 5. T\* temp = **new** T[capacity]; 6. **for** (**int** i = 0; i < index; i++) 7. { 8. \*(temp + i) = **this**->elemArray[i]; 9. } 10. **for** (**int** i = index; i < **this**->capacity; i++) 11. { 12. temp[i] = **this**->elemArray[i + 1]; 13. } 14. **this**->capacity--; 15. **delete**[] elemArray; 16. elemArray = temp; 17. } 18. Procedure 2 19. **template**<**class** T> 20. **inline** **void** ListTemp<T>::AddHead(**const** T& newData) 21. { 22. Node\* newHead = **new** Node; 23. newHead->data = newData; 24. newHead->next = head; 25. head = newHead; 26. size ++ ; 27. } 28. Procedure 3 29. **template**<**class** T> 30. **inline** **void** ListTemp<T>::AddTail(**const** T& newData) 31. { 32. Node\* head = **new** Node; 33. Node\* newNode = **new** Node; 34. newNode->data = newData; 35. newNode->next = nullptr; 36. head->next = newNode; 37. head = newNode; 38. size++; 39. } 40. Result 1 41. Result 2 42. Result 3 | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   1) Analysis  a. Task 1  With the template class, we can pass in any type of object to manipulate it, all we need to change is to change Employee to the template class T. Use the pointer to T to create a new array of T.  b. Task 2  First, you need to create a new node with a pointer, store the incoming data into the data field of this new node, and assign the pointer field of this new node as the header pointer. Then assign the header pointer to the address of this new node. Finally, add one to the length of this link list.  c. Task 3  First allocate a memory space for the header pointer. Then create a new node using a pointer to a node whose data domain is assigned to the incoming data and whose pointer domain is assigned to a null pointer. Besides, the pointer field of the element head points to is assigned to the address of the new node, and the head pointer is redirected to this new node. Finally the length of the link list is added by one  2) Harvest  a. Template Class  People need to write multiple functions of similar form and function, so there are function templates to reduce duplication of labor; people also need to write multiple classes of similar form and function, so C++ introduced the concept of class templates, the compiler can automatically generate multiple classes from class templates, to avoid the programmer's duplication of labor. The definitions of function templates and class template member functions are usually placed in a header file, unless separation is achieved using the export keyword.  b. Link list  A link list is a non-continuous, non-sequential storage structure on a physical storage unit, where the logical order of data elements is achieved through the linking order of pointers in the chain table. A link list consists of a series of nodes (each element in the chain table is called a node), which can be dynamically generated at runtime. Each node consists of two parts: a data field that stores the data element and a pointer field that stores the address of the next node.  c. Sequential container & associative container  A sequential container is a linear table with sequential relationships between the elements and is a linearly structured orderable cluster. Each element in a sequential container has a fixed position, unless this position is changed by a deletion or insertion operation. This position is not related to the element itself, but to the time and place of the operation, and the sequential container does not order the elements according to their characteristics but directly preserves the logical order of the elements when they are operated on. For example, if we append three elements to a sequential container at once, the relative positions of these three elements in the container are the same as the logical order of the appending.  Unlike sequential containers, associative containers are non-linear trees, or more precisely, binary trees. There is no strict physical order between the elements, meaning that the elements are not preserved in the logical order in which they were placed in the container. However, associative containers provide an alternative feature of ordering the elements according to their characteristics, so that the iterator can "order" the elements according to their characteristics.  Another distinctive feature of the associative container is that it saves data as keys, i.e. it saves keywords and values in association, whereas the sequential container saves only one (it can be thought of as saving only the keywords, or only the values).  3) Existing Problems  a. In the addTail() method, the head node will allocate a memory space every time the method is called. | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |