CSCI 301 Computer Science II Summer 2023

**AAssignment 6 – an ADT Front List**

Due Date: 11:50 pm July19, Wednesday

# Introduction:

This assignment consists of creating three different ATD Front List classes. One private inheritance class via the sortedListAsA, one public inheritance class via the sortedListIsA and finally one containment relationship inheritance class by the method of sortedListHasA. Each of these classes will be implemented and linked to a linkedList class that contains the following member functions: bool insert, bool remove and void replace.Finally, we will be implementing a tester to test these classes.

# Data structure

In this assignment, we have used the following data structures in order to implement this project. String, booleans a linked list as a class, nodes are the most frequent data structures used to successfully design this program.

# Functions

We have design a few functions to successfully implement the program among them we have:

* An insert function to insert data in the list
* A remove function to remove a data in the list
* A retrieve function to retrieve elements from the last

# Structure chart of the main program



# Code list

/\*\*

\* problem description: Design the following classes: FrontListIsA, FrontListHasA, FrontListAsA.

\* Name: Algassimou Diallo

\* startID: el8524jv

\* Instructor: Jie Meichsner

\* Due date: 07/19/2023

\*/

/\*\*

\* Abstract class for the ADT FrontList.

\* Defines the operations that can be performed on the list.

\*/

#ifndef FRONTLIST\_INTERFACE

#define FRONTLIST\_INTERFACE

// Abstract class (interface) for the ADT FrontList

template <typename ItemType>

class FrontListInterface {

public:

/\*\*

\* Inserts a new item at the front of the list.

\* @param newEntry The item to be inserted.

\* @return True if the insertion is successful, false otherwise.

\*/

virtual bool insert(const ItemType& newEntry) = 0;

/\*\*

\* Removes the item at the front of the list.

\* @return True if the removal is successful, false otherwise.

\*/

virtual bool remove() = 0;

/\*\*

\* Retrieves the item at the front of the list.

\* @return The item at the front of the list.

\*/

virtual ItemType retrieve() const = 0;

/\*\*

\* Destructor to ensure proper deallocation of memory.

\*/

virtual ~FrontListInterface() {}

};

#endif

/\*\*

\* problem description: Design the following classes: FrontListIsA, FrontListHasA, FrontListAsA.

\* Name: Algassimou Diallo

\* startID: el8524jv

\* Instructor: Jie Meichsner

\* Due date: 07/19/2023

\*/

#include <iostream>

#include <stdexcept>

#include "LinkedList.h" // Assuming LinkedList class is defined

#include "FrontListInterface.h"

#ifndef FRONTLIST\_AS\_A

#define FRONTLIST\_AS\_A

// Implementation of FrontList using LinkedList (Approach 3: Private Inheritance)

template <typename ItemType>

class FrontListPrivateInheritance : private LinkedList<ItemType>, public FrontListInterface<ItemType> {

public:

using LinkedList<ItemType>::insert;

using LinkedList<ItemType>::remove;

using LinkedList<ItemType>::getEntry;

bool insert(const ItemType& newEntry) override {

// Insert at the front of the list

return LinkedList<ItemType>::insert(1, newEntry);

}

bool remove() override {

// Remove the first item in the list

return LinkedList<ItemType>::remove(1);

}

ItemType retrieve() const override {

// Retrieve the first item in the list

return LinkedList<ItemType>::getEntry(1);

}

};

#endif

/\*\*

\* problem description: Design the following classes: FrontListIsA, FrontListHasA, FrontListAsA.

\* Name: Algassimou Diallo

\* startID: el8524jv

\* Instructor: Jie Meichsner

\* Due date: 07/19/2023

\*/

#include <iostream>

#include <stdexcept>

#include "LinkedList.h" // Assuming LinkedList class is defined

#include "FrontListInterface.h"

#ifndef FRONTLIST\_IS\_A

#define FRONTLIST\_IS\_A

// Implementation of FrontList using LinkedList (Approach 2: Public Inheritance)

template <typename ItemType>

class FrontListPublicInheritance : public LinkedList<ItemType>, public FrontListInterface<ItemType> {

public:

bool insert(const ItemType& newEntry) override {

// Insert at the front of the list

return LinkedList<ItemType>::insert(1, newEntry);

}

bool remove() override {

// Remove the first item in the list

return LinkedList<ItemType>::remove(1);

}

ItemType retrieve() const override {

// Retrieve the first item in the list

return LinkedList<ItemType>::getEntry(1);

}

};

#endif

/\*\*

\* problem description: Design the following classes: FrontListIsA, FrontListHasA, FrontListAsA.

\* Name: Algassimou Diallo

\* startID: el8524jv

\* Instructor: Jie Meichsner

\* Due date: 07/19/2023

\*/

#include <iostream>

#include <stdexcept>

#include "LinkedList.h"// Assuming LinkedList class is defined

#include "FrontListInterface.h"

#ifndef FRONTLIST\_HAS\_A

#define FRONTLIST\_HAS\_A

// Implementation of FrontList using LinkedList (Approach 1: Composition)

template <typename ItemType>

class FrontListComposition : public FrontListInterface<ItemType> {

private:

// Implementation using LinkedList (private member)

LinkedList<ItemType> list;

public:

bool insert(const ItemType& newEntry) override {

// Insert at the front of the list

return list.insert(1, newEntry);

}

bool remove() override {

// Remove the first item in the list

return list.remove(1);

}

ItemType retrieve() const override {

// Retrieve the first item in the list

return list.getEntry(1);

}

};

#endif

/\*\*

\* problem description: Design the following classes: FrontListIsA, FrontListHasA, FrontListAsA.

\* Name: Algassimou Diallo

\* startID: el8524jv

\* Instructor: Jie Meichsner

\* Due date: 07/19/2023

\*/

#include <iostream>

#include "LinkedList.h"

#include "FrontListAsA.h"

#include "FrontListHasA.h"

#include "FrontListIsA.h"

int main() {

int choice;

std::cout << "Choose the FrontList implementation:" << std::endl;

std::cout << "1. Composition (Approach 1)" << std::endl;

std::cout << "2. Public Inheritance (Approach 2)" << std::endl;

std::cout << "3. Private Inheritance (Approach 3)" << std::endl;

std::cout << "Enter your choice (1, 2, or 3): ";

std::cin >> choice;

FrontListInterface<int>\* frontList = nullptr;

switch (choice) {

case 1:

frontList = new FrontListComposition<int>();

break;

case 2:

frontList = new FrontListPublicInheritance<int>();

break;

case 3:

frontList = new FrontListPrivateInheritance<int>();

break;

default:

std::cout << "Invalid choice." << std::endl;

return 1;

}

// Test FrontList operations

int item;

char continueChoice;

do {

std::cout << "Enter an item to insert: ";

std::cin >> item;

if (frontList->insert(item)) {

std::cout << "Item inserted successfully." << std::endl;

} else {

std::cout << "Item insertion failed." << std::endl;

}

std::cout << "Front item: " << frontList->retrieve() << std::endl;

if (frontList->remove()) {

std::cout << "Item removed successfully." << std::endl;

} else {

std::cout << "Item removal failed." << std::endl;

}

std::cout << "Do you want to continue? (y/n): ";

std::cin >> continueChoice;

} while (continueChoice == 'y' || continueChoice == 'Y');

delete frontList; // Clean up memory

return 0;

}

#include "LinkedList.h"

#include <stdexcept>

template <typename ItemType>

LinkedList<ItemType>::LinkedList() : head(nullptr), itemCount(0) {}

template <typename ItemType>

LinkedList<ItemType>::LinkedList(const LinkedList<ItemType>& otherList) : head(nullptr), itemCount(0) {

Node\* otherCurrent = otherList.head;

while (otherCurrent != nullptr) {

insert(itemCount + 1, otherCurrent->data);

otherCurrent = otherCurrent->next;

}

}

template <typename ItemType>

LinkedList<ItemType>::~LinkedList() {

clear();

}

/\*\*

\* problem description: Design the following classes: FrontListIsA, FrontListHasA, FrontListAsA.

\* Name: Algassimou Diallo

\* startID: el8524jv

\* Instructor: Jie Meichsner

\* Due date: 07/19/2023

\*/

template <typename ItemType>

bool LinkedList<ItemType>::insert(int position, const ItemType& newItem) {

if (position < 1 || position > itemCount + 1) {

return false;

}

Node\* newNode = new Node(newItem);

if (position == 1) {

newNode->next = head;

head = newNode;

} else {

Node\* current = head;

for (int i = 1; i < position - 1; i++) {

current = current->next;

}

newNode->next = current->next;

current->next = newNode;

}

itemCount++;

return true;

}

template <typename ItemType>

bool LinkedList<ItemType>::remove(int position) {

if (position < 1 || position > itemCount) {

return false;

}

Node\* nodeToDelete;

if (position == 1) {

nodeToDelete = head;

head = head->next;

} else {

Node\* current = head;

for (int i = 1; i < position - 1; i++) {

current = current->next;

}

nodeToDelete = current->next;

current->next = nodeToDelete->next;

}

delete nodeToDelete;

itemCount--;

return true;

}

template <typename ItemType>

ItemType LinkedList<ItemType>::getEntry(int position) const {

if (position < 1 || position > itemCount) {

throw std::out\_of\_range("Invalid position.");

}

Node\* current = head;

for (int i = 1; i < position; i++) {

current = current->next;

}

return current->data;

}

template <typename ItemType>

bool LinkedList<ItemType>::isEmpty() const {

return itemCount == 0;

}

template <typename ItemType>

void LinkedList<ItemType>::clear() {

Node\* current = head;

while (current != nullptr) {

Node\* nextNode = current->next;

delete current;

current = nextNode;

}

head = nullptr;

itemCount = 0;

}

/\*\*

\* problem description: Design the following classes: FrontListIsA, FrontListHasA, FrontListAsA.

\* Name: Algassimou Diallo

\* startID: el8524jv

\* Instructor: Jie Meichsner

\* Due date: 07/19/2023

\*/

#ifndef LINKEDLIST\_H

#define LINKEDLIST\_H

template <typename ItemType>

class LinkedList {

private:

struct Node {

ItemType data;

Node\* next;

Node(const ItemType& item) : data(item), next(nullptr) {}

};

Node\* head; // Pointer to the head of the linked list

int itemCount; // Number of items in the linked list

public:

// Default constructor

LinkedList();

// Copy constructor

LinkedList(const LinkedList<ItemType>& otherList);

// Destructor

~LinkedList();

// Inserts a new node with the given item at the specified position in the linked list

bool insert(int position, const ItemType& newItem);

// Removes the node at the specified position in the linked list

bool remove(int position);

// Retrieves the item at the specified position in the linked list

ItemType getEntry(int position) const;

// Checks if the linked list is empty

bool isEmpty() const;

// Clears the linked list

void clear();

};

#include "LinkedList.cpp" // Include the implementation file

#endif

# User Document

To successfully test these classes, the user must specify the class to be tested. The user will be prompted to enter the number 1, 2 or 3 while 1 corresponds to the composition approach, 2 corresponds to the public inheritance and the number 3 corresponds to the private inheritance. After selecting a specific approach to be tested, the user will be allowed to enter a one digit to insert it into the list and it will be displayed as output and then removed. The program will also allow the user to keep doing this operation until he decides to quit by entering no in the yes/no question.

The files of this project are located in the directory el8524jv/csci301/project6. To successfully compile this program in centOS, the user must run the following script:

g++ -std=c++11 -g main.cpp -o a.out. Then you run ./a.out.

Test cases

| Tests | Input data | Output data |
| --- | --- | --- |
| 1. Composition approach | 3  7  45  30 | 3  7  45  30 |
| 1. Public inheritance | 6  23  76  20 | 6  23  76  20 |
| 1. Private inheritance | 4  56  0  90 | 4  56  0  90 |

Test 1:

Script done on Wed 19 Jul 2023 07:18:52 PM CDT

[el8524jv@csci4 Project6]$ cat test1

Script started on Wed 19 Jul 2023 07:09:57 PM CDT

[el8524jv@csci4 Project6]$ ./a.out

Choose the FrontList implementation:

1. Composition (Approach 1)

2. Public Inheritance (Approach 2)

3. Private Inheritance (Approach 3)

Enter your choice (1, 2, or 3): 1

Enter an item to insert: 3

Item inserted successfully.

Front item: 3

Item removed successfully.

Do you want to continue? (y/n): y

Enter an item to insert: 7

Item inserted successfully.

Front item: 7

Item removed successfully.

Do you want to continue? (y/n): y

Enter an item to insert: 45

Item inserted successfully.

Front item: 45

Item removed successfully.

Do you want to continue? (y/n): y

Enter an item to insert: 30

Item inserted successfully.

Front item: 30

Item removed successfully.

Do you want to continue? (y/n): n

[el8524jv@csci4 Project6]$ exit

Script done on Wed 19 Jul 2023 07:11:15 PM CDT

Test 2

Script started on Wed 19 Jul 2023 07:15:14 PM CDT

[el8524jv@csci4 Project6]$ ./a.out

Choose the FrontList implementation:

1. Composition (Approach 1)

2. Public Inheritance (Approach 2)

3. Private Inheritance (Approach 3)

Enter your choice (1, 2, or 3): 2

Enter an item to insert: 6

Item inserted successfully.

Front item: 6

Item removed successfully.

Do you want to continue? (y/n): y

Enter an item to insert: 23

Item inserted successfully.

Front item: 23

Item removed successfully.

Do you want to continue? (y/n): y

Enter an item to insert: 76

Item inserted successfully.

Front item: 76

Item removed successfully.

Do you want to continue? (y/n): y

Enter an item to insert: 20

Item inserted successfully.

Front item: 20

Item removed successfully.

Do you want to continue? (y/n): n

[el8524jv@csci4 Project6]$ exit

Script done on Wed 19 Jul 2023 07:18:52 PM CDT

test 3:

Choose the FrontList implementation:

1. Composition (Approach 1)

2. Public Inheritance (Approach 2)

3. Private Inheritance (Approach 3)

Enter your choice (1, 2, or 3): 3

Enter an item to insert: 4

Item inserted successfully.

Front item: 4

Item removed successfully.

Do you want to continue? (y/n): y

Enter an item to insert: 56

Item inserted successfully.

Front item: 56

Item removed successfully.

Do you want to continue? (y/n): y

Enter an item to insert: 0

Item inserted successfully.

Front item: 0

Item removed successfully.

Do you want to continue? (y/n): y

Enter an item to insert: 90

Item inserted successfully.

Front item: 90

Item removed successfully.

Do you want to continue? (y/n): y

Enter an item to insert: 1

Item inserted successfully.

Front item: 1

Item removed successfully.

Do you want to continue? (y/n): n

[el8524jv@csci4 Project6]$ exit

Script done on Wed 19 Jul 2023 07:21:29 PM CDT

# Summary

This project was successfully implemented in our local computer using VSCode and then transferred in the centOS. We have registered no failures while compiling this program. We were able to test each class separately by selecting the specific class that we want to test.