Object-Oriented Programming with C++

ENSIA 2024-2025

Tutorial 11 (Templates)

Exercise 1 (Generic Linked List)

The following C++ code defines a class that implements a linked list of integers and a small main program that uses it.

```
// LinkedList.h
#include <ostream>
using namespace std;

class LinkedList {
public:
    LinkedList();      // default constructor
    LinkedList(const LinkedList& lst); // copy constructor
    ~LinkedList();     // destructor

    // A new node containing the given data is inserted at the front
    void add(int data);

    // A new node containing the given data is inserted at the given
   position in the list
    void insertAt(int pos, int data);

    // The first incidence of the given data is removed from the list.
    // Returns true if data is found (and removed), false otherwise
    bool remove(int data);

    // Empties the list, freeing up dynamically allocated memory
    void removeAll();

    // Prints the contents of the list to the standard output stream
    void printList();

private:
    // List node class
    class Node {
    public:
        int data;   // list data
        Node* next; // pointer to next item in the list

        Node(int d) { data = d; }
        Node(const Node* node) { data = node->data; }
    };

    Node* head; // Pointer to the first node in the list
    int size;   // Records the number of nodes in the list
};
```

```cpp
// test.cpp
#include <iostream>
using namespace std;

int main() {
    LinkedList list;
    list.add(14);
    list.add(20);
    list.add(26);
    list.printList();

    LinkedList l2(list);
    l2.printList();

    return 0;
}
```

Tasks

1. Convert the class `LinkedList` to define a template class.
2. Implement the template linked list.
3. Test your linked list template by modifying `test.cpp` so that it creates an `int` linked list and a `string` linked list, and uses each of the `LinkedList` methods at least once for each type of list.

Exercise 2 (Heterogeneous Pairs)

Part 1

We aim to define a template class called `HeterogeneousPair`. An object of this class contains a pair of values; the first and second positions may store values of different types. Use two type parameters `T1` and `T2`; all items in the first position will be of type `T1`, and all items in the second position will be of type `T2`. The default type should be set to `int`.

The template class `HeterogeneousPair` should define:

- constructors: parameterized constructor, default constructor, and copy constructor.
- two mutator functions called `set_first` and `set_second`.
- two accessor functions called `get_first` and `get_second`.
- an overload of the insertion operator to print the two elements of the heterogeneous pair.

Tasks

1. Implement the template class `HeterogeneousPair`.
2. Write a main function that creates pairs with heterogeneous values and tests the various defined functions.

Part 2

A `Point` is represented by a pair of two `double` values. A point is defined as an object of a class `Hpoint` derived from the `HeterogeneousPair` class.

Tasks

1. Implement the derived class `Hpoint` and define within this class a function that calculates the distance between two points.
2. Test the class `Hpoint` in the main function.