Laborationsuppgift

Belyser ADT: Lista, Stack samt Kö

Datastrukturer: länkning samt array

Implementationspecifikt: klassmall (template), länkning, inre klass, rent virtuella (pure virtual) funktioner, abstrakt klass

ADT Lista

1. Gör en klassmall `List` för att representera en lista

   Följande funktioner ska vara medlemsfunktioner i listan:

   ```
   void insertAt(const T& data, int pos);
   const T& get(int pos) const;
   T removeAt(int pos);
   void clear();
   int size() const;
   ```

   Som intern datastruktur ska enkelläkning användas utan huvudnod (head node) och svansnod (tail node). En inre klass benämnd `Node` ska definieras för noderna i den länkade strukturen.

   Medlemsvariabler för `List` ska vara en pekare av typen `Node` benämnd `first` samt en heltalsvariabel som håller antalet element i listan. Benämn denna variabel `nrOfElements`.

   Förutom medlemsfunktionerna angivna ovan ska dessutom defaultkonstrukor, kopieringskonstruktor, destruktor samt tilldelningsoperator deklareras och definieras för `List`.

2. Kan klassmallen `List` utökas med en medlemsfunktion som avgör om ett visst element finns i listan eller ej utan att någon annan förändring behövs? Kommer listan då att fungera för alla datatyper?

3. Vilken är kostnaden (”ordo”) för respektive operation i listan om antalet element i listan är n?

ADT Stack

1. Konstruera en klassmall `IStack` med enbart rent virtuella funktioner enligt följande

```
void push(const T &item)
T pop()
const T& peek() const
bool isEmpty() const
```

2. Konstruera en klassmall `StackLinkingNodes` vilken ärver `IStack.` Definierar en inre klass `Node` för en enkellänkad lista (se tidigare klassmall för List). Deklarera lämplig medlemsvariabel och definiera medlemsfunktiorna angivna ovan samt konstruktorer, destruktor och tilldelningsoperator.

   Gör dessutom ett program som testar `StackLinkingNodes`. Följande ska bland annat ingå:

```
StackLinkingNodes<int> stack;
for (int aValue = 10; aValue<=100; aValue+=10)
{
   stack.push(aValue);
}

StackLinkingNodes<int> anotherStack = stack;
StackLinkingNodes<int> thirdStack;
thirdStack =  stack;

cout<<”peek of stack : “<<stack.peek()<<endl;
cout<<”pop() on stack until empty”<<endl;
while (!stack.isEmpty())
{
   cout<<stack.pop()<<endl;
}
cout<<”peek of anotherStack : “<<anotherStack.peek()<<endl;
cout<<”pop() on anotherStack until empty”<<endl;
while (!anotherStack.isEmpty())
{
   cout<<anotherStack.pop()<<endl;
}
cout<<”peek of stack : “<<thirdStack.peek()<<endl;
cout<<”pop() on thirdStack until empty”<<endl;
while (!thirdStack.isEmpty())
{
   cout<<thirdStack.pop()<<endl;
}
```

3. Konstruera en klassmall `StackArray` vilken ärver `IStack`. Deklarera en medlemsvariabel för att dynamiskt allokera en array samt medlemsvariabel för att hålla kapaciteten på arrayen och en medlemsvariabel för att hålla antalet element i stacken. Definiera dessutom alla medlemsfunktionerna från `IStack`. Använd samma testprogram som för `StackLinkedList` men byt till `StackArray.`

4. Gör ytterligare en klassmall `StackUsingList` vilken ärver `IStack`. Denna ska använda ett List-objekt för att hantera de element som finns på stacken. Definiera alla medlemsfunktioner från `IStack`. Använd samma testprogram som för `StackLinkedList` men byt till `StackUsingList.`

5. Vilken är kostnaden (”ordo”) för push(...), pop() och peek() i stacken för de båda datastrukturerna enkellänkad lista respektive array. Förutsätt att stacken innehåller n element.

ADT Kö:

1. Konstruera en klassmall för `IQueue` med enbart rent virtuella funktioner enligt följande

   ```
   void engueue(const T& item)
   T dequeue()
   const T& front() const
   bool isEmpty() const
   ```

2. Konstruera en klassmall `QueueLinkedNodes` vilken ärver `IQueue`. Definierar en inre klass `Node` för en enkellänkad lista.

   Deklarera medlemsvariabler så att det finns en pekare till den nod som är först och en pekare till den nod som är sist. Definiera dessutom alla funktioner från `IQueue.` Testa att anropa funktionerna för ett objekt av typen `QueueLinkedNodes.` Tillse att test görs så att det görs insättning i tom kö, insättning i kö med flera element, borttagning av element när det finns flera element samt när det endast finns ett element.

3. Konstruera en klassmall `QueueCircularArray` vilken ärver `IQueue`. Som datastruktur används en array cirkulärt. Gör motsvarande tester som för `QueueLinkedNodes`.

4. Vilken är kostnaden (”ordo”) för enqueue(...), dequeue() och front() i kön för de båda datastrukturerna enkellänkad lista respektive array. Förutsätt att kön innehåller n element.