

## **Labaration 3**

### ***drivers.h***

```
#ifndef _DRVLST_
#define _DRVLST_

#include "List.h"

//Ser till att inmatat värde endast kan anta siffra
//Pre: True
//Post: Returnerat inmatad siffra
int mataIn();

//Lägger till ett element i listan
//Pre: True
//Post: Ett element tillagt i listan
void laggTill(List & listan);

//Tar bort ett element från listan
//Pre: True
//Post: Elementet på angiven position har raderats
void taBort(List & listan);

//Sorterar listan stigande
//Pre: True
//Post: Listan har blivit sorterad i stigande ordning
void sortera(List & listan);

//Hämtar värdet på inmatad position
//Pre: True
//Post: Ett värde har hämtats och skrivits till skärmen
void hamtaVarde(List & listan);

//Skriver ut hela listan
//Pre: True
//Post: Skrivit ut hela listan till skärmen
void visaLista(List & listan);

//Söker efter ett element i listan
//Pre: True
//Post: Skrivit ut elementets position i listan på skärmen
void soek(List & listan);

//Tömmer listan
//Pre: True
//Post: Listan tömd
void toem(List & listan);

//Kontrollerar om listan är tom
//Pre: True
//Post: Skrivit ut listans status
void tom(List & listan);
```

```
//Skriker ut antal positioner i listan
//Pre: True
//Post: Skrivit ut antal positioner i listan.
void antal(List & listan);

#endif
```

### **drivers.cpp**

```
#include <iostream>
#include "drivers.h"
#include "List.h"
#include <iomanip>
using namespace std;

int mataIn()
{
    char buffer[100];
    int varde;
    bool felKoll = false;
    do
    {
        if(felKoll == true)
            cout << "Felaktigt värde! Försök igen: ";
        cin.getline(buffer, 100);
        if(buffer[0]!='0')
            return 0;
        felKoll = true;
    }while(!(varde=atoi(buffer)));
    return varde;
}

void laggTill(List & listan)
{
    int inmat, pos;

    cout << "Lägg till element i listan" << endl
         << "=====" << endl << endl;

    cout << "Ange det HELTAL du vill lägga till i listan: ";

    inmat = mataIn();

    cout << "Ange på vilken position du vill lägga värdet (1 till "
         << listan.size() + 1 << "): ";

    pos = mataIn();

    while((pos > listan.size() + 1 ) || (pos < 1))
    {
        cout << "Ogiltig position, ange ny: ";
        pos = mataIn();
    }
}
```

```
        listan.insert(pos, inmat);

        cout << "Det nya elementet har lagrats!" << endl << endl;
    }

void taBort(List & listan)
{
    int pos;

    cout << "Ta bort ett element från listan" << endl
         << "=====" << endl << endl;

    if(!listan.isEmpty())
    {
        cout << "Ange position (1 till " << listan.size() << "): ";
        pos = mataIn();
        while((pos > listan.size()) || (pos < 1))
        {
            cout << "Ogiltig position, ange ny: ";
            pos = mataIn();
        }

        listan.remove(pos);
        cout << endl << "Elementet på position " << pos << " har raderats" << endl
             << endl;
    }
    else
    {
        cout << "Listan är tom! Går ej radera från tom lista." << endl << endl;
    }
}

void sortera(List & listan)
{
    cout << "Sortera listan" << endl
         << "=====" << endl << endl;

    if(!listan.isEmpty())
    {
        listan.sort();

        cout << "Listan är sorterad i stigande ordning" << endl << endl;
    }
    else
    {
        cout << "Listan är tom! Går ej sortera tom lista." << endl << endl;
    }
}

void hamtaVarde(List & listan)
{
    int kokkos;
    cout << "Hämta element" << endl
         << "=====" << endl << endl;
}
```

```
if(!listan.isEmpty())
{
    cout << "Ange position (1 till " << listan.size() << "): ";
    kokkos = mataIn();
    while((kokkos > listan.size()) || (kokkos < 1))
    {
        cout << "Ogiltig position, ange ny: ";
        kokkos = mataIn();
    }

    cout << endl << "Värdet på position " << kokkos << " är: "
        << listan.getElement(kokkos) << endl << endl;
}
else
{
    cout << "Listan är tom! Går ej hämta värde." << endl << endl;
}
}

void visaLista(List & listan)
{
    cout << "Visa listan" << endl
        << "======" << endl << endl;
    if(!listan.isEmpty())
    {
        cout << "Position\tVärde" << endl << endl;

        for(int i = 1; i <= listan.size(); i++)
            cout << setw(8) << i << "\t" << listan.getElement(i) << endl;

        cout << endl << endl;
    }
    else
    {
        cout << "Listan är tom! Går ej visa lista." << endl << endl;
    }
}

void soek(List & listan)
{
    int sok;

    cout << "Sök ett element" << endl
        << "======" << endl << endl;

    if(!listan.isEmpty())
    {
        cout << "Ange värde att söka efter: ";

        sok = mataIn();

        cout << endl;

        if(listan.isElement(sok))
            cout << sok << " finns på position " << listan.getPos(sok) << endl << endl;
    }
}
```

```
        else
            cout << sok << " finns inte i listan" << endl << endl;
    }
    else
    {
        cout << "Listan är tom! Går ej att söka." << endl << endl;
    }
}

void toem(List & listan)
{
    int stor;

    cout << "Töm listan" << endl
         << "=====" << endl << endl;

    if(!listan.isEmpty())
    {
        stor = listan.size();

        for(int i = 1; i <= stor; i++)
        {
            listan.remove(1);
        }
        cout << "Listan är tömd!" << endl << endl;
    }
    else
    {
        cout << "Listan är redan tom!" << endl << endl;
    }
}

void tom(List & listan)
{
    cout << "Listans status" << endl
         << "=====" << endl << endl;

    if(listan.isEmpty())
        cout << "Listan är tom!" << endl << endl;
    else
        cout << "Listan innehåller element" << endl << endl;
}

void antal(List & listan)
{
    cout << "Antal element i listan" << endl
         << "=====" << endl << endl;

    if(listan.isEmpty())
        cout << "Listan är tom, inga element finns!" << endl << endl;
    else
        cout << "Listan innehåller " << listan.size() << " st element" << endl <<
endl;
```

```
}
```

### **lab3.cpp**

```
#include <iostream>
#include "List.h"
#include "drivers.h"
using namespace std;

//Skriver en meny på skärmen
//Pre: True
//Post: Skrivit en meny på skärmen
void meny()
{
    cout << "  MENY  " << endl
         << "======" << endl
         << "1. Visa listan" << endl
         << "2. Hämta element" << endl
         << "3. Lägg till element" << endl
         << "4. Ta bort element" << endl
         << "5. Sortera listan" << endl
         << "6. Sök efter element" << endl
         << "7. Töm listan" << endl
         << "8. Kontrollera listans status" << endl
         << "9. Antal element i listan" << endl
         << "0. Avsluta" << endl << endl
         << "Ange menyval (0-9): ";
}

int main()
{
    List prgList;

    int menval;

    cout << "Listprogrammet" << endl
         << "======" << endl << endl
         << "Skrivet av: Mathias Andersson" << endl
         << "                Henrik Bäck" << endl << endl;

    while(menval != 0)
    {
        meny();

        menval = mataIn();

        cout << endl;

        switch(menval)
        {
            case 1: visaLista(prgList);break;
            case 2: hamtaVarde(prgList);break;
            case 3: laggTill(prgList); break;
        }
    }
}
```

```
        case 4: taBort(prgList); break;
        case 5: sortera(prgList); break;
        case 6: soek(prgList);break;
        case 7: toem(prgList);break;
        case 8: tom(prgList);break;
        case 9: antal(prgList);break;
        case 0: break;
        default: cout << "Ogiltigt val!" << endl;
    }
}
```

```
}
```

### **List.h**

```
#ifndef _LISTPRG_
#define _LISTPRG_

#include <iostream>
#include <cstring>
#include "Node.h"

class List
{
public:

    //Skapar en ett nytt obejt av klassen List
    //Pre: True
    //Post: Skapat ett nytt obejkt av klassen List
    List();

    //Tar bort dynamiskt allokerat minne
    //Pre:
    //Post: Tagit bort dynamiskt allokerat minne
    ~List();

    //Lägger till ett nytt element
    //Pre: 1 <= position <= size() + 1
    //Post: Elementet har lagts till i listan på position pos
    void insert(int pos, int varde);

    //Tar bort ett element
    //Pre: Elementet finns i listan
    //Post: Elementet har tagits bort
    void remove(int pos);

    //Sortera listan i stigande ordning
    //Pre: True
    //Post: Listan är sorterad i stigande ordning
    void sort();

    //Kontrollerar ifall listan är tom
    //Pre: True
    //Post: Returnerar true om listan är tom, annars false
    bool isEmpty();
};
```

```
//Kontrollerar om ett element finns med i listan
//Pre: True
//Post: Returnerat true om elementet finns i listan annars false
bool isElement(int varde);

//Hämta positionen för ett visst värde
//Pre: True
//Post: Returnerat det första värdet som uppfyller 'varde'
int getPos(int varde);

//Returnerar antal element i listan
//Pre: True
//Post: Returnerat antalet element i listan
int size();

//Returnerar värdet på positionen pos
//Pre: 1 <= pos <= size();
//Post: Returnerat värdet på positionen pos
int getElement(int pos);

private:
Node * head;
int sizen;

};

#endif
```

### **List.cpp**

```
#include "List.h"
#include <iostream>
using namespace std;

List::List()
{
    head = 0;
    sizen = 0;
}

List::~~List()
{
    int stor;
    stor = sizen;
    for(int i = 1; i <= stor; i++)
        remove(1);
}

void List::insert(int pos, int varde)
{
    Node * newNode = new Node();
    newNode -> data = varde;

    if(pos == 1)
```



```
        {
            newNode -> next = head;
            head = newNode;
        }
    else
    {
        Node * temp = head;
        for(int i = 1; i < pos - 1; i++)
        {
            temp = temp -> next;
        }
        newNode -> next = temp -> next;
        temp -> next = newNode;
    }
    sizen ++;
}

void List::remove(int pos)
{
    Node * temp;
    Node * temp2;
    temp = head;

    if(pos == 1)
    {
        temp2 = temp -> next;
        delete temp;
        head = temp2;
    }
    else
    {
        for(int i = 1; i < pos - 1; i++)
            temp = temp -> next;

        temp2 = temp -> next;
        temp2 = temp2 -> next;

        delete temp -> next;

        temp -> next = temp2;
    }
    sizen--;
}

void List::sort()
{
    Node * temp = head;
    Node * temp2;
    Node * temp3;

    for(int i = 1; i < sizen; i++)
        for(int k = 1; k < sizen; k++)
        {

            if(getElement(k) > getElement(k+1))
            {
                temp = head;
```

```
        if(k == 1)
        {
            temp2 = temp -> next;
            temp -> next = temp2 -> next;
            temp2 -> next = temp;
            head = temp2;
        }
        else
        {
            for(int j = 1; j < k-1; j++)
            {
                temp = temp -> next;
            }
            temp2 = temp -> next;
            temp3 = temp2 -> next;
            temp2 -> next = temp3 -> next;
            temp3 -> next = temp2;
            temp -> next = temp3;
        }
    }
}

bool List::isEmpty()
{
    return !sizen;
}

bool List::isElement(int varde)
{
    bool finns = false;
    for(int i = 1; i <= sizen; i++)
    {
        if(getElement(i) == varde)
        {
            finns = true;
            i = sizen + 1;
        }
    }

    return finns;
}

int List::getPos(int varde)
{
    int position = 0;

    for(int i = 1; i <= sizen; i++)
    {
        if(getElement(i) == varde)
        {
            position = i;
            i = sizen + 1;
        }
    }
}
```

```
    return position;
}

int List::size()
{
    return sizen;
}

int List::getElement(int pos)
{
    Node * temp;
    temp = head;

    for(int i = 1; i < pos ; i++)
        temp = temp -> next;

    return temp -> data;
}
```