

MUSTERLÖSUNG BLATT 5

A5.1:

```
#include<iostream.h>
#include<string.h>

typedef char str5[5];
typedef char str10[10];
typedef char str15[15];
typedef char str20[20];

typedef struct {
    str5   Anrede;
    str15  Name;
    str15  Vorname;
    str20  Strasse;
    int    Hausnr;
    int    PLZ;
    str20  Ort;
} Adresse;

typedef struct ListenElement * Listenpointer;

struct ListenElement {
    Adresse Adr;
    Listenpointer Nachfolger;
};

void ausgabe(Listenpointer Cur){
    // Wertübergabe des Pointers
    // Die Funktion gibt nur Adr.Name aus
    while(Cur!=0){
        cout<<Cur->Adr.Name<<endl;
        Cur=Cur->Nachfolger;
    }
    cout<<endl;
};

void einbinde(Listenpointer& Anfang, Listenpointer NewElement){
    // Funktion zum Einordnen eines neuen Elements (NewElement)
    // in eine sortierte Liste (Anfang)
    if (strcmp(NewElement->Adr.Name,Anfang->Adr.Name)<0){
        //NewElement kommt ganz nach vorne
        NewElement->Nachfolger=Anfang;
        Anfang=NewElement;
    }
    else{
        // Zyklus mit zwei Pointer (Vorgaenger und Cur)
        Listenpointer Vorgaenger=Anfang;
        Listenpointer Cur=Anfang->Nachfolger;
        // Finde die passende Stelle
        while(Cur!=NULL && strcmp(NewElement->Adr.Name,Cur->Adr.Name)>0){
            Vorgaenger=Cur;
            Cur=Cur->Nachfolger;
        }
        //Binde dae neue Element ein
        NewElement->Nachfolger=Cur;
        Vorgaenger->Nachfolger=NewElement;
    }
}

void sort(Listenpointer& Anfang){
```

```

// Basiert auf der Funktion einbinde
if (Anfang != NULL){ //Liste nicht leer
    Listenpointer Cur=Anfang->Nachfolger;
    // Trennung der akten liste
    // Initialisation des sortierten Teiles
    Anfang->Nachfolger=NULL;

    // Einbinden der Elemente der unsortierten Liste
    // in die sortierte
    while(Cur!=NULL){
        Listenpointer Add=Cur;
        Cur=Cur->Nachfolger;
        einbinde(Anfang,Add);
    };
};

};

void main( ) {
    Adresse Element3={"Frau", "Graf", "Steffi",
        "Am Tennisplatz", 1 , 35039, "Marburg"};

    Adresse Element2={"Herr", "Kahn", "Oliver",
        "Am Tor", 1 , 35039, "Marburg"};

    Adresse Element1={"Herr", "Schumacher", "Michael",
        "Am Auto", 1 , 35039, "Marburg"};

    Adresse Element4={"Herr", "Neuer", "Sportler",
        "Am Trainieren", 1 , 35039, "Marburg"};

    // Aufbau der Liste
    Listenpointer Anfang(new ListenElement), Ende=Anfang;

    Ende->Adr=Element1;
    Ende->Nachfolger=(new ListenElement);

    Ende=Ende->Nachfolger;
    Ende->Adr=Element2;
    Ende->Nachfolger=(new ListenElement);

    Ende=Ende->Nachfolger;
    Ende->Adr=Element3;
    Ende->Nachfolger=NULL;

    // Testausgabe der unsortierten Liste
    cout<<"Dies ist die Initialisierungsliste"<<endl;
    ausgabe(Anfang);

    // Eingabe des neuen Elementes
    // in diesem Fall ist das Element4
    Listenpointer NewElement(new ListenElement);
    NewElement->Adr=Element4;
    // Einbinden des neuen Elementes in die Liste
    einbinde(Anfang,NewElement);
    cout<<"Dies ist die unsortierte Liste"<<endl;
    ausgabe(Anfang);

    // Sortieren der Liste + Ausgabe
    sort(Anfang);
    cout<<"Dies ist die sortierte Liste"<<endl;
    ausgabe(Anfang);}

```