```c
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <malloc.h>

#include <Eigene Pakete/ausgabe.h>
#include <Eigene Pakete/eingabe.h>
```

```c
#include "includetest4.h"

#define nil NULL

int Anzahl_der_Daten = 0;
char Text[128];

const char *Anzahl_der_Daten_Textdatei = "Aufgabe 2 Anzahl der Daten4.txt",*Ausgabe_Textdatei = "Ausgabe Aufgabe 2 test4.txt";

struct friend
{
  char  *first_name;
  char  *last_name;

  struct friend *next_friend;
};


/*****************************************************************************/
/*  create a new friend; return a pointer to the actual list of all friends */
/*****************************************************************************/

struct friend *create_new_friend(const char *fn, const char *ln)
{
  static struct friend *first_friend = nil;
  struct friend *nf = malloc(sizeof(struct friend));

  nf->first_name = strdup(fn);
  nf->last_name = strdup(ln);

  nf->next_friend = first_friend;
  first_friend = nf;

  return(first_friend);
}

void print_all_friends(struct friend *first)
{
  for (; first; first = first->next_friend)
  {
    printf("%s %s\n", first->first_name, first->last_name);
    sprintf(Text,"%s %s\r\n", first->first_name, first->last_name);
    Anzahl_der_Daten = speichern_in_Datei(Anzahl_der_Daten,Ausgabe_Textdatei,Text);
  }
}

void ausgabe(const char *Ausgabetext,struct friend *Ausgabe)
{
    printf("%s\n",Ausgabetext);
    sprintf(Text,"%s\r\n",Ausgabetext);
    Anzahl_der_Daten = speichern_in_Datei(Anzahl_der_Daten,Ausgabe_Textdatei,Text);
    print_all_friends(Ausgabe);
    printf("\n");
    sprintf(Text,"\r\n");
    Anzahl_der_Daten = speichern_in_Datei(Anzahl_der_Daten,Ausgabe_Textdatei,Text);
}

static void swap(const char **array,int a,int b)
{
    const char *holder;

    holder = array[a];
    array[a] = array[b];
    array[b] = holder;
}
```

```
 66
 67  static void quick_sort(const char **array,const char **array1,int left,int right)
 68  {
 69      int pivot,i,j;
 70      const char *key,*key1;
 71
 72      if(right - left == 1)
 73      {
 74          if(strcmp(array[left],array[right]) > 0 || (strcmp(array[left],array[right]) == 0 && strcmp(array1[
left],array1[right]) > 0))
 75          {
 76              swap(array,left,right);
 77              swap(array1,left,right);
 78          }
 79
 80          return;
 81      }
 82
 83      pivot = (left + right) / 2;
 84      key = array[pivot];
 85      key1 = array1[pivot];
 86      swap(array,left,pivot);
 87      swap(array1,left,pivot);
 88      i = left + 1;
 89      j = right;
 90      while(i < j)
 91      {
 92          while((i <= right && strcmp(array[i],key) < 0) || (i <= right && strcmp(array[i],key) == 0 &&
strcmp(array1[i],key1) < 0))
 93              i++;
 94          while((j >= left && strcmp(array[j],key) > 0) || (j >= left && strcmp(array[j],key) == 0 && strcmp(
array1[j],key1) > 0))
 95              j--;
 96          if(i < j)
 97          {
 98              swap(array,i,j);
 99              swap(array1,i,j);
100          }
101      }
102      swap(array,left,j);
103      swap(array1,left,j);
104      if(left < j - 1)
105          quick_sort(array,array1,left,j - 1);
106      if(j + 1 < right)
107          quick_sort(array,array1,j + 1,right);
108  }
109
110  struct friend *einhaengen(struct friend *original,const char *plus1,const char *plus2)
111  {
112      struct friend *y = original,*neu = nil;
113
114      neu = create_new_friend(plus1,plus2);
115      neu->next_friend = y;
116      y = neu;
117
118      return(neu);
119  }
120
121  struct friend *sortieren_alphabetisch(struct friend *original)
122  {
123      struct friend *y = original,*sortieren_alphabetisch = nil;
124      int Anzahl = 0;
125
126      while(y != nil)
127      {
128          Anzahl++;
```

```
129         y = y->next_friend;
130     }
131
132     const char *Vorname[Anzahl],*Nachname[Anzahl];
133     y = original;
134
135     for(int i = 0;i < Anzahl;i++)
136     {
137         Vorname[i] = y->first_name;
138         Nachname[i] = y->last_name;
139         y = y->next_friend;
140     }
141     quick_sort(Nachname,Vorname,0,Anzahl - 1);
142
143     for(int i = Anzahl - 1;i >= 0;i--)
144         sortieren_alphabetisch = einhaengen(sortieren_alphabetisch,Vorname[i],Nachname[i]);
145
146     return(sortieren_alphabetisch);
147 }
148
149 struct friend *reverse(struct friend *original)
150 {
151     struct friend *t,*y = original,*reverse = nil;
152
153     while(y != nil)
154     {
155         t = y->next_friend;
156         y->next_friend = reverse;
157         reverse = y;
158         y = t;
159     }
160
161     return(reverse);
162 }
163
164 int main()
165 {
166   Anzahl_der_Daten = eingabe_int_aus_Textdatei(Anzahl_der_Daten_Textdatei);
167   struct friend  *anchor, *anchor1, *anchor2;
168
169   anchor = create_new_friend("Ilse", "Ludwig");
170   anchor = create_new_friend("Erich", "Ludwig");
171   anchor = create_new_friend("Paul", "Miller");
172   anchor = create_new_friend("Mike", "Gordan");
173   anchor = create_new_friend("Martha", "Ludwig");
174
175   ausgabe("Originalliste:",anchor);
176
177   anchor1 = sortieren_alphabetisch(anchor);
178
179   ausgabe("Sortierte Liste:",anchor1);
180
181   anchor2 = reverse(anchor);
182
183   ausgabe("Reverse Liste:",anchor2);
184
185   sprintf(Text,"%d\r\n",Anzahl_der_Daten);
186   ausgabe_in_Textdatei(Anzahl_der_Daten_Textdatei,"w",Text);
187
188   return(0);
189 }
```

```c
#include "includetest4.h"

#define nil NULL

int Anzahl_der_Daten = 0;
char Text[128];

const char *Anzahl_der_Daten_Textdatei = "Aufgabe 2 Anzahl der Daten4.txt",*Ausgabe_Textdatei = "Ausgabe Aufgabe 2 test4.txt";

struct friend
{
  char  *first_name;
  char  *last_name;

  struct friend *next_friend;
};


/*****************************************************************************/
/*  create a new friend; return a pointer to the actual list of all friends */
/*****************************************************************************/

struct friend *create_new_friend(const char *fn, const char *ln)
{
  static struct friend *first_friend = nil;
  struct friend *nf = malloc(sizeof(struct friend));

  nf->first_name = strdup(fn);
  nf->last_name = strdup(ln);

  nf->next_friend = first_friend;
  first_friend = nf;

  return(first_friend);
}

void print_all_friends(struct friend *first)
{
  for (; first; first = first->next_friend)
  {
    printf("%s %s\n", first->first_name, first->last_name);
    sprintf(Text,"%s %s\r\n", first->first_name, first->last_name);
    Anzahl_der_Daten = speichern_in_Datei(Anzahl_der_Daten,Ausgabe_Textdatei,Text);
  }
}

void ausgabe(const char *Ausgabetext,struct friend *Ausgabe)
{
    printf("%s\n",Ausgabetext);
    sprintf(Text,"%s\r\n",Ausgabetext);
    Anzahl_der_Daten = speichern_in_Datei(Anzahl_der_Daten,Ausgabe_Textdatei,Text);
    print_all_friends(Ausgabe);
    printf("\n");
    sprintf(Text,"\r\n");
    Anzahl_der_Daten = speichern_in_Datei(Anzahl_der_Daten,Ausgabe_Textdatei,Text);
}

static void swap(const char **array,int a,int b)
{
    const char *holder;

    holder = array[a];
    array[a] = array[b];
    array[b] = holder;
}
```

```
 66
 67  static void quick_sort(const char **array,const char **array1,int left,int right)
 68  {
 69      int pivot,i,j;
 70      const char *key,*key1;
 71
 72      if(right - left == 1)
 73      {
 74          if(strcmp(array[left],array[right]) > 0 || (strcmp(array[left],array[right]) == 0 && strcmp(array1[
left],array1[right]) > 0))
 75          {
 76              swap(array,left,right);
 77              swap(array1,left,right);
 78          }
 79
 80          return;
 81      }
 82
 83      pivot = (left + right) / 2;
 84      key = array[pivot];
 85      key1 = array1[pivot];
 86      swap(array,left,pivot);
 87      swap(array1,left,pivot);
 88      i = left + 1;
 89      j = right;
 90      while(i < j)
 91      {
 92          while((i <= right && strcmp(array[i],key) < 0) || (i <= right && strcmp(array[i],key) == 0 &&
strcmp(array1[i],key1) < 0))
 93              i++;
 94          while((j >= left && strcmp(array[j],key) > 0) || (j >= left && strcmp(array[j],key) == 0 && strcmp(
array1[j],key1) > 0))
 95              j--;
 96          if(i < j)
 97          {
 98              swap(array,i,j);
 99              swap(array1,i,j);
100          }
101      }
102      swap(array,left,j);
103      swap(array1,left,j);
104      if(left < j - 1)
105          quick_sort(array,array1,left,j - 1);
106      if(j + 1 < right)
107          quick_sort(array,array1,j + 1,right);
108  }
109
110  struct friend *einhaengen(struct friend *original,const char *plus1,const char *plus2)
111  {
112      struct friend *y = original,*neu = nil;
113
114      neu = create_new_friend(plus1,plus2);
115      neu->next_friend = y;
116      y = neu;
117
118      return(neu);
119  }
120
121  struct friend *sortieren_alphabetisch(struct friend *original)
122  {
123      struct friend *y = original,*sortieren_alphabetisch = nil;
124      int Anzahl = 0;
125
126      while(y != nil)
127      {
128          Anzahl++;
```

```
129         y = y->next_friend;
130     }
131
132     const char *Vorname[Anzahl],*Nachname[Anzahl];
133     y = original;
134
135     for(int i = 0;i < Anzahl;i++)
136     {
137         Vorname[i] = y->first_name;
138         Nachname[i] = y->last_name;
139         y = y->next_friend;
140     }
141     quick_sort(Nachname,Vorname,0,Anzahl - 1);
142
143     for(int i = Anzahl - 1;i >= 0;i--)
144         sortieren_alphabetisch = einhaengen(sortieren_alphabetisch,Vorname[i],Nachname[i]);
145
146     return(sortieren_alphabetisch);
147 }
148
149 struct friend *reverse(struct friend *original)
150 {
151     struct friend *t,*y = original,*reverse = nil;
152
153     while(y != nil)
154     {
155         t = y->next_friend;
156         y->next_friend = reverse;
157         reverse = y;
158         y = t;
159     }
160
161     return(reverse);
162 }
163
164 int main()
165 {
166   Anzahl_der_Daten = eingabe_int_aus_Textdatei(Anzahl_der_Daten_Textdatei);
167   struct friend  *anchor, *anchor1, *anchor2;
168
169   //anchor = create_new_friend("Ilse", "Ludwig");
170   anchor = create_new_friend("Erich", "Ludwig");
171   anchor = create_new_friend("Paul", "Miller");
172   anchor = create_new_friend("Mike", "Gordan");
173   anchor = create_new_friend("Martha", "Ludwig");
174
175   ausgabe("Originalliste:",anchor);
176
177   anchor1 = sortieren_alphabetisch(anchor);
178
179   ausgabe("Sortierte Liste:",anchor1);
180
181   anchor2 = reverse(anchor);
182
183   ausgabe("Reverse Liste:",anchor2);
184
185   sprintf(Text,"%d\r\n",Anzahl_der_Daten);
186   ausgabe_in_Textdatei(Anzahl_der_Daten_Textdatei,"w",Text);
187
188   return(0);
189 }
```

```c
#include "includetest4.h"

#define nil NULL

int Anzahl_der_Daten = 0;
char Text[128];

const char *Anzahl_der_Daten_Textdatei = "Aufgabe 2 Anzahl der Daten4.txt",*Ausgabe_Textdatei = "Ausgabe Aufgabe 2 test4.txt";

struct friend
{
  char  *first_name;
  char  *last_name;

  struct friend *next_friend;
};


/*****************************************************************************/
/*  create a new friend; return a pointer to the actual list of all friends */
/*****************************************************************************/

struct friend *create_new_friend(const char *fn, const char *ln)
{
  static struct friend *first_friend = nil;
  struct friend *nf = malloc(sizeof(struct friend));

  nf->first_name = strdup(fn);
  nf->last_name = strdup(ln);

  nf->next_friend = first_friend;
  first_friend = nf;

  return(first_friend);
}

void print_all_friends(struct friend *first)
{
  for (; first; first = first->next_friend)
  {
    printf("%s %s\n", first->first_name, first->last_name);
    sprintf(Text,"%s %s\r\n", first->first_name, first->last_name);
    Anzahl_der_Daten = speichern_in_Datei(Anzahl_der_Daten,Ausgabe_Textdatei,Text);
  }
}

void ausgabe(const char *Ausgabetext,struct friend *Ausgabe)
{
    printf("%s\n",Ausgabetext);
    sprintf(Text,"%s\r\n",Ausgabetext);
    Anzahl_der_Daten = speichern_in_Datei(Anzahl_der_Daten,Ausgabe_Textdatei,Text);
    print_all_friends(Ausgabe);
    printf("\n");
    sprintf(Text,"\r\n");
    Anzahl_der_Daten = speichern_in_Datei(Anzahl_der_Daten,Ausgabe_Textdatei,Text);
}

static void swap(const char **array,int a,int b)
{
    const char *holder;

    holder = array[a];
    array[a] = array[b];
    array[b] = holder;
}
```

```
 66
 67  static void quick_sort(const char **array,const char **array1,int left,int right)
 68  {
 69      int pivot,i,j;
 70      const char *key,*key1;
 71
 72      if(right - left == 1)
 73      {
 74          if(strcmp(array[left],array[right]) > 0 || (strcmp(array[left],array[right]) == 0 && strcmp(array1[
left],array1[right]) > 0))
 75          {
 76              swap(array,left,right);
 77              swap(array1,left,right);
 78          }
 79
 80          return;
 81      }
 82
 83      pivot = (left + right) / 2;
 84      key = array[pivot];
 85      key1 = array1[pivot];
 86      swap(array,left,pivot);
 87      swap(array1,left,pivot);
 88      i = left + 1;
 89      j = right;
 90      while(i < j)
 91      {
 92          while((i <= right && strcmp(array[i],key) < 0) || (i <= right && strcmp(array[i],key) == 0 &&
strcmp(array1[i],key1) < 0))
 93              i++;
 94          while((j >= left && strcmp(array[j],key) > 0) || (j >= left && strcmp(array[j],key) == 0 && strcmp(
array1[j],key1) > 0))
 95              j--;
 96          if(i < j)
 97          {
 98              swap(array,i,j);
 99              swap(array1,i,j);
100          }
101      }
102      swap(array,left,j);
103      swap(array1,left,j);
104      if(left < j - 1)
105          quick_sort(array,array1,left,j - 1);
106      if(j + 1 < right)
107          quick_sort(array,array1,j + 1,right);
108  }
109
110  struct friend *einhaengen(struct friend *original,const char *plus1,const char *plus2)
111  {
112      struct friend *y = original,*neu = nil;
113
114      neu = create_new_friend(plus1,plus2);
115      neu->next_friend = y;
116      y = neu;
117
118      return(neu);
119  }
120
121  struct friend *sortieren_alphabetisch(struct friend *original)
122  {
123      struct friend *y = original,*sortieren_alphabetisch = nil;
124      int Anzahl = 0;
125
126      while(y != nil)
127      {
128          Anzahl++;
```

```
129         y = y->next_friend;
130     }
131
132     const char *Vorname[Anzahl],*Nachname[Anzahl];
133     y = original;
134
135     for(int i = 0;i < Anzahl;i++)
136     {
137         Vorname[i] = y->first_name;
138         Nachname[i] = y->last_name;
139         y = y->next_friend;
140     }
141     quick_sort(Nachname,Vorname,0,Anzahl - 1);
142
143     for(int i = Anzahl - 1;i >= 0;i--)
144         sortieren_alphabetisch = einhaengen(sortieren_alphabetisch,Vorname[i],Nachname[i]);
145
146     return(sortieren_alphabetisch);
147 }
148
149 struct friend *reverse(struct friend *original)
150 {
151     struct friend *t,*y = original,*reverse = nil;
152
153     while(y != nil)
154     {
155         t = y->next_friend;
156         y->next_friend = reverse;
157         reverse = y;
158         y = t;
159     }
160
161     return(reverse);
162 }
163
164 int main()
165 {
166   Anzahl_der_Daten = eingabe_int_aus_Textdatei(Anzahl_der_Daten_Textdatei);
167   struct friend  *anchor, *anchor1, *anchor2;
168
169   //anchor = create_new_friend("Ilse", "Ludwig");
170   //anchor = create_new_friend("Erich", "Ludwig");
171   anchor = create_new_friend("Paul", "Miller");
172   anchor = create_new_friend("Mike", "Gordan");
173   //anchor = create_new_friend("Martha", "Ludwig");
174
175   ausgabe("Originalliste:",anchor);
176
177   anchor1 = sortieren_alphabetisch(anchor);
178
179   ausgabe("Sortierte Liste:",anchor1);
180
181   anchor2 = reverse(anchor);
182
183   ausgabe("Reverse Liste:",anchor2);
184
185   sprintf(Text,"%d\r\n",Anzahl_der_Daten);
186   ausgabe_in_Textdatei(Anzahl_der_Daten_Textdatei,"w",Text);
187
188   return(0);
189 }
```

```
Originalliste:
Martha Ludwig
Mike Gordan
Paul Miller
Erich Ludwig
Ilse Ludwig

Sortierte Liste:
Mike Gordan
Erich Ludwig
Ilse Ludwig
Martha Ludwig
Paul Miller

Reverse Liste:
Ilse Ludwig
Erich Ludwig
Paul Miller
Mike Gordan
Martha Ludwig

Originalliste:
Martha Ludwig
Mike Gordan
Paul Miller
Erich Ludwig

Sortierte Liste:
Mike Gordan
Erich Ludwig
Martha Ludwig
Paul Miller

Reverse Liste:
Erich Ludwig
Paul Miller
Mike Gordan
Martha Ludwig

Originalliste:
Mike Gordan
Paul Miller

Sortierte Liste:
Mike Gordan
Paul Miller

Reverse Liste:
Paul Miller
Mike Gordan
```