Übung Zeiger/Vektoren
File-Streams

Einführung

Arrays/Vektoren können gleich oder ähnlich wie Zeiger verwendet werden, das heisst es ist möglich mit dem Index-Operator zu arbeiten. Genauso ist es möglich mit Zeigerarithmetik auf Arrays zuzugreifen.

Funktion strlen

Schreibe eine Funktion strlen. Diese Funktion bestimmt die Anzahl Buchstaben in einem C-String (ohne abschliessende 0) und gibt diese Anzahl als Rückgabewert an den Aufrufer zurück.

```
char* text = „Hallo wie geht es dir ?“;
int anzahlBuchstaben = strlen(text);
```

Die Variable anzahlBuchstaben sollte bei diesem Beispiel 23 betragen. Der Funktionsprototyp dieser Funktion sieht also etwa so aus:

```
int strlen(char* einText);
```

Oder was auch möglich wäre:

```
int strlen(char einText[]);
```

Diese Funktion existiert bereits und ist mit #include <cstring> erreichbar. Wir wollen diese aber selber definieren. Da wir einen Funktionsnamen verwenden, der bereits vergeben ist, wollen wir diese Funktion in einem eigenen Namensraum (namespace) definieren.

Funktion strcat

Schreibe eine Funktion strcat (kommt string concatinate), die zwei C-Strings zusammenfügt. Sie nimmt als Parameter zwei C-Strings und fügt den zweiten String an den ersten an. Beachte, dass beide C-Strings mit einer 0 terminiert sind. Um als ein C-String betrachtet zu werden darf der resultierende C-String nur am Ende eine 0 haben.

C-String 1	H	a	l	l	i	0
C-String 2	H	a	l	l	o	0

Ergebnis

H	a	l	l	i	H	a	l	l	o	0

Der Funktionsprototyp sieht so aus :

```
char* strcat(char* text1, char* text2);
```

Schreibe also diese Funktion so, dass der Inhalt des C-Strings text2 hinter text1 kopiert wird. Wie bei strlen ist die abschliessende 0 der „Schlüsselpunkt“. Setze diese Funktion auch in den Namensraum, den du für die Funktion strlen definiert hast, denn auch diese Funktion exisiert bereits. Beim Aufruf dieser Funktion ist es wichtig, dass das char-Array von text1 genügend gross, ist dass beide strings darin Platz haben !

```
int main()
{
    char ergebnis[50] = „Halli“; // Array genügend gross !
    char dazu[] = „Hallo“;

    strcat(ergebnis, dazu);

    return 0;
}
```

Sonst überschreibst Du Speicher, der bestenfalls nicht Dir gehört und das Programm stürzt aber. Wahrscheinlicher ist hier aber, dass Du deinen eigenen Speicher „zerschiesst“.
Wir sind darum froh, dass der C++ Standard die Klasse string definiert, die alles sauber und logisch behandelt:

```
int main()
{
    string ergebnis;
    string h1 = „Halli“;
    string h2 = „Hallo“;
    ergebnis = h1 + h2; // sicher und sauber!

    return 0;
}
```

Erweiterung der Notenliste

Nimm die Notenlistenübung vom letzten mal. Ergänze die Klasse Notenliste, so dass es möglich ist die Notenliste in einer Datei zu speichern (z.B. Methode: speichereInDatei(const string& dateiName)).
Als zweites gilt es die Klasse so zu erweitern, dass es möglich ist eine Notenliste ohne Namen zu erzeugen (definiere den neuen Konstruktor) dafür mit soll es möglich sein mit einer Methode ladeAusDatei(const string& dateiName) die Notenliste zu füllen.