6. Aufgabenblatt

Programmieren in C++ (Tutorium)

Pointer

1. Legen Sie je eine Variable vom Typ `int`, `char`, `float`, `string` an und weisen Sie diesen jeweils einen beliebigen Wert aus dem jeweils zulässigen Wertebereich zu. Lassen Sie nun jeweils einen Pointer auf die eben deklarierten Variablen zeigen. Welche Ausgabe erhalten Sie, wenn Sie die Pointer-Variable mit bzw. ohne den Stern (Dereferenzierungsoperator) ausgeben lassen?

2. Betrachten Sie folgende Situation, in der insgesamt drei Pointer deklariert wurden.

```
char c = 'A';

const char * c_p1 = &c;
char * const c_p2 = &c;
const char * const c_p3 = &c;
```

   Versuchen Sie nun für jeden der Pointer `c_p1`, `c_p2`, `c_p3` sowohl die Speicheradresse des Pointers, als auch den Wert, auf den der jeweilige Pointer zeigt, zu verändern. Was schließen Sie daraus hinsichtlich der Bedeutung des Schlüsselwortes `const` in jedem der drei Fälle?

3. Pointer können nicht nur mit dem `*`-Operator dereferenziert werden, sondern es sind auch weitere Operationen zulässig. Deklarieren Sie je einen Pointer auf eine `bool`- und eine `short`-Variable und führen Sie nacheinander jeweils folgende Operationen aus:

   - Inkrementieren des Pointers,
   - Addition von 5 auf den Pointer,
   - Dekrementieren des Pointers,
   - Substraktion von 4 von dem Pointer.

   Lassen Sie ihr Programm den anfänglichen Wert und den Wert des Pointers nach jeder dieser Operationen ausgeben. Welchen Unteschied beobachten Sie hinsichtlich der verschiedenen Datentypen? Versuchen Sie vor einer Operation den Namen der resultierenden Speicheradresse vorherzusagen.

Arrays

4. Legen Sie ein Array `int data[5];` an und lassen Sie sich den Inhalt der Elemente (ohne das Array zuvor zu befüllen) ausgeben. Testen und vergleichen Sie

die Ausgabe im Debug- und Release-Modus. Vergleichen Sie Ihre Ergebnisse mit denen Ihrer Kommilitonen. Was stellen Sie fest?

5. Zur Erfassung von 1000 Messwerten (Fließkommazahlen) wird ein Datenspeicher benötigt. Legen Sie ein geeignetes Array an und stellen Sie sicher, dass es jedes Element den Wert 0 enthält. Wie können Sie dies am einfachsten erreichen?

6. Legen Sie ein `int[]`-Feld mit einigen Werten an. Geben Sie den Inhalt des Feldes zwei Mal aus, wobei Sie auf verschiedene Arten auf die Elemente zugreifen: Als Array mit dem Indizierungs-Operator `[]` und als Pointer mit dem Dereferenzierungsoperator `*`

7. Legen Sie ein `int[]`-Feld mit einigen Werten an. Geben Sie den Inhalt und die Größe des Feldes [1] aus. Anschließend übergeben Sie das Array an eine Funktion `checkArray(int[] data)` und lassen diese das gleiche machen. Welches Problem stellen Sie fest? Wie müssen die die Funktionsdeklaration erweitern um dieses Problem zu lösen?

C-Strings

8. Auf Übungsblatt 3 (Aufgabe 5) wurde geprüft, ob ein Text ein Palindrom ist (also von vorne und hinten gelesen identisch ist).
   Schreiben Sie erneut ein Programm, das für ein eingegebenes Wort überprüft, ob es sich um ein Palindrom handelt. Verwenden Sie diesmal statt strings aber `char[]` zum Speichern der Zeichenkette. Um den Testablauf zu beschleunigen, kann ein geeigneter Text (z.B. „Lagerregal“) im Programmcode abgelegt werden, anstatt ihn jedesmal von der Tastatus einzulesen.

9. Um Texte zu vergleichen, wird in der Programmierung üblicherweise ein lexikalischer Vergleich angewendet. Implementieren Sie eine Funktion
   `int compare(const char* a, const char* b`, die einen derartigen Vergleich durchführt. Das heißt, sie soll folgende Werte zurückgeben:

   0 Beide Parameter sind exakt gleich (z.B. `compare("abc", "abc")`)

   -1 Der erste ungleiche Buchstabe in `a` hat einen niedrigeren Wert als in `b` (z.B. `compare("aad", "abc")`)

   1 Der erste ungleiche Buchstabe in `a` hat einen höheren Wert als in `b` (z.B. `compare("acb", "abc")`)

10. Um Texte case-insensitive (also ohne Berücksichtigung der Groß-/Kleinschreibung) zu vergleichen, ist es zweckmäßig alle Texte intern in Kleinschreibung umzuwandeln. Erstellen Sie eine Funktion `void strToLower(const char* src, char* dest)`, die einen Text `src` in Kleinschreibung[2] nach `dest` kopiert.
    Zusatzfrage: warum ist es nicht so einfach möglich, die Kopie in der Funktion zu erstellen und per return zurückzugeben?

[1] hierbei hilft der `sizeof`-Operator

[2] Zur Umwandlung einzelner Zeichen können Sie die Ascii-Werte verwenden, oder z.B. die Funktion `char tolower (char c)` aus der locale-Bibliothek nutzen