0x06 – Übungen

’Reeds Field’

C-String

- Legen Sie einen C-String mit einem beliebigen Text an: `char* str = ”hallo”;`
- Legen Sie einen Zeiger auf char an und laufen Sie mit diesem durch das Feld str (einschließlich des Null-Zeichens), um den jeweiligen Charakter, auf den der Zeiger zeigt, auszugeben. Geben Sie den jeweils aktuellen Charakter einmal als Charakter und einmal als ASCII Wert aus. Tipp: (int)-cast für den ASCII-Wert,

Erweiterung:

- Geben Sie zusätzlich auch den Wert des Zeigers, d.h. die Adresse aus. Tipp: geeigneter cast.

0x06 – Übungen

’South Birds Gale’

Zeigern, Felder

- Definieren Sie vier Worte ’Dies’, ’ist’, ’ein’, ’Satz’ in einem Feld mit vier Zeigern.
- Übergeben Sie dieses Feld von Zeigern einer Funktion, um dort die Worte in umgekehrter Reihenfolge auszugeben.

Erweiterung:

- Drehen Sie die die Reihenfolge der Worte in dem Feld vor der Ausgabe um (nicht die Worte selbst).
- Drehen Sie auch die Worte selbst für die Ausgabe um.

0x06 – Übungen

’Green Mound’

new, Felder, C-Strings

- Programmieren Sie eine Funktion ’copy’, die einen C-String als Parameter erhält, dynamischen Speicher mit ’new’ anfordert, den übergebenen C-String dorthin inkl. des Null-Zeichens kopiert und den Zeiger auf den neuen Speicher zurückgibt.
- Ermitteln Sie die Länge mit Hilfe einer eigenen Funktion. Schreiben Sie Testcode, der Worte kopiert und ausgibt.

Erweiterung:

- Schreiben Sie eine Funktion ’free’ zum Freigeben des zuvor reservierten Speichers und nutzen Sie sie.
- Testen Sie ggf. mit Tools wie valgrind, ob Ihr Code Speicherlecks enthält.

0x06 – Übungen

’Rose Pond’

new, dynamischen Strukturen, smart pointer

- Legen Sie eine Struktur ’address’ an, die einen Namen und eine Tel.Nr., beides vom Typ ’string’, enthält.
- Speichern Sie drei fiktive Datensätze jeweils unter einer ID in einer ’unordered_map<int,address*>’. Die ’address’-Struktur erhalten Sie dynamisch mit ’new’ und speichern nur den Zeiger.
- Geben Sie die komplette Map aus.

Erweiterung:

- Geben Sie die Daten der Map am Ende wieder frei. Testen Sie Ihren Code auf Speicherlecks.
- Verwenden Sie ’unique_ptr’ statt ’address*’.

0x06 – Übungen

’Kenford’

new, dynamischen Strukturen, smart pointer

- Realisieren Sie eine verkettete Liste, in der jeder Knoten einen ’int’ und einen Zeiger (’unique_ptr’) auf den nächsten Knoten enthält. Die Liste enthält einen Zeiger ’root’ auf den ersten Knoten, sowie einen Zähler ’count’.
- Die Liste bietet eine Funktion ’add(int n)’ zum Hinzufügen eines Knotens, eine Funktion ’clear’ zum Löschen bzw. Freigeben aller Daten sowie einen Ausgabeoperator.

Erweiterung:

- Testen Sie Ihren Code auf Speicherlecks.
- Machen sie aus der Liste ein Template für beliebige Datentypen (nicht nur ’int’).

0x06 – Übungen

Hausübung

- Verstehen Sie den Code in ’work_ptrptr’ (Beispiel für Zeiger auf Zeiger).
- Verstehen Sie den Code in ’work_reserve’ (Unterschied Zeiger zu Referenzen).

Selbstkontrolle

- Ich habe alle Codes und Übungsthemen verstanden.
- Ich weiß was C-Strings sind.
- Ich kenne den Unterschied zwischen ’delete’ und ’delete[]’.
- Ich kann Vor- und Nachteile von Zeigern nennen.
- Ich weiß, wie ’call-by-ref.’ mit Zeigern funktioniert.
- Ich kann konstante Zeiger von Zeigern auf Konstanten unterscheiden.
- Ich nutze ’smart pointer’ statt ’raw pointer’.