Version C++ 23

Übungsblatt 4

Aufgabe 1 - Referenzen

Gegeben seien folgende überladene Funktionen in der Programmiersprache C++:

(a)
```
int inkrementiere(int &v)
{
  return ++v;
}
```

(b)
```
int inkrementiere(const int &v)
{
  return v+1;
}
```

1. Welche Implementierung (a), (b) wird durch welchen Funktionsaufruf (i) und (ii) ausgeführt?

   (i)
   ```
   int x = 10;
   int y = inkrementiere(x); -> (a), da x nicht const ist.
   ```

   (ii)
   ```
   int z = inkrementiere(10); -> (b), da 10 ein Literal, und somit const, ist.
   ```

2. Wie lauten die Werte in x, y und z nach der Ausführung der Aufrufe in (i) und (ii) ?
   x = 11, y = 11, z = 11
3. Worin unterscheidet sich die Implementierung in a) von der Implementierungen in b)? Warum ist in b) eine andere Implementierung notwendig?

Aufgabe 2 - Werte, Zeiger und Referenzen

```
double a {2.3};                    R-Wert   | 2,3
double *b {a};                     L-Wert   | 9
double c {10.3-0.3};               R-Wert   | 10
float &d {0};                      R-Wert   | falsch
double &e {a};                     L-Wert   | a=2,3
double *f {&e};                    L-Wert   | 2,3
short *g {b+1};                    R-Wert   | falsch
const double &h {4.0+a};           R-Wert   | 6,3
double *i = &(c-0.5);              R-Wert   | falsch?
const int &j {20};                 R-Wert   | 20
int &&k = j + 10;                  R-Wert   | 30
```

Listing 1: Initialisierung von Variablen in C++.

1. Geben Sie in Listing 1 für **jeden** Ausdruck, der die Variablen a-k initialisiert, an, ob es sich um einen L-Wert oder R-Wert handelt.
2. Markieren Sie richtige und falsche Quelltext-Zeilen (C++ 23) in Listing 1. Geben Sie wenn möglich den Wert an, der in den Variablen a - k gespeichert oder - im Falle von Zeigern und Referenzen - referenziert wird.

Aufgabe 3 - Dynamische Speicherverwaltung

1. Geben Sie mindestens drei Nachteile der dynamische Speicherverwaltung in C++ an.
2. Welche Vorteile haben intelligente Zeiger (Smart Pointer)? Nennen Sie mind. drei.
3. Gegeben sei die Klasse *Quadrat* mit Konstruktor *Quadrat(int groesse)* und einer Member-Funktion *void zeichnen()*. In unten stehendem Quelltext zum Zeichnen eines Quadrates (der Seitenlänge 5) haben sich kleine Fehler eingeschlichen. Korrigieren Sie diese!

```
Quadrat *quad {new Quadrat[5]}; Quadrat *quad(new Quadrat(5));
quad->zeichnen(); quad->zeichnen();
delete Quadrat; delete quad;
```

Listing 2: Dynamische Objekterzeugung

4. Schreiben Sie den Quelltext um und verwenden Sie den Smart Pointer `std:unique_ptr` aus der C++-Standardbibliothek (C++ 23).

3]

1) Man muss sich selbst um Speicherreservierung/-freigabe
- Gefahr von „dangling pointers"
- ist von der Laufzeit langsamer

2) Speicherbedarf ist erst zur Laufzeit bekannt
- Objekte werden nicht direkt am Ende des Wertebereichs gelöscht
- Deklaration und Initialisierung kann an verschiedenen Orten statt finden