1. Prüfung zu VO Programmiersprache 1 17.03.2022 WS 2021/22

Angabe Nr. 1

Es sind keine Unterlagen erlaubt! Alle Multiple-Choice Aufgaben haben 4 Unterpunkte a) bis d), die entweder wahr oder falsch sind. Sie haken das Antwortkästchen deutlich an, wenn die Aussage wahr ist bzw. lassen das Kästchen frei, wenn die Aussage falsch ist.

1) Folgende Streameingabe für den Objekttyp `Complex` sei definiert:

```
class Complex {
    double re, im;
};

std::istream& operator>>(std::istream& is, Complex& c)
{   is >> c.re >> c.im;
    return is;
}
```

a) Die Streameingabe erzeugt einen Compilerfehler, weil `c.re` privat ist.

b) Die Streameingabe erzeugt einen Compilerfehler, weil `c.re` konstant ist.

c) Damit eine Streameingabe korrekt funktioniert, muss man das einzulesende Objekt als konstante Referenz übergeben werden.

d) Eine Streameingabe, bei der das einzulesende Objekt per Wert übergeben wird, kann nicht korrekt funktionieren.

2) Folgende Streameingabe für den Objekttyp `Complex` sei definiert:

```
struct Complex {
    double re, im;
};

std::istream& operator>>(std::istream& is, const Complex& c)
{   is >> c.re >> c.im;
    return is;
}
```

a) Die Streameingabe erzeugt einen Compilerfehler, weil `c.re` privat ist.

b) Die Streameingabe erzeugt einen Compilerfehler, weil `c.re` konstant ist.

c) Damit eine Streameingabe korrekt funktioniert, muss man das einzulesende Objekt als Referenz übergeben.

d) Eine Streameingabe, bei der das einzulesende Objekt per Wert übergeben wird, kann nicht korrekt funktionieren.

3) a) Der Datentyp `signed char` kann immer die Zahlen 1 und -1 darstellen.
   b) Falls der Datentyp `int` 4 Bytes besitzt, lässt sich 1 Million nicht mehr als `int` darstellen.
   c) C++ verwendet C-kompatibel den Datentyp `int` auch für logische Werte. Jede Zahl ungleich Null gilt als falsch (false).
   d) Der Zahlenumfang der Ganzzahltypen `char`, `short`, `int` und `long` ist in C++ nicht genormt.

4) Dürfen in C++ 2 Funktionen den gleichen Funktionsnamen haben?
   a) Gleiche Funktionsnamen sind in C++ für verschiedene Funktionen nie erlaubt.
   b) Ja, wenn sich die Funktionen im Rückgabewert unterscheiden.
   c) Ja, wenn sich die Funktionen in der Zahl der Parameter unterscheiden.
   d) Ja, wenn sich die Funktionen im Typ von mindestens einem Argument unterscheiden.

5) Welchen Wert haben die Variablen i, j, k nach folgendem Code-Fragment:

```
int i = 5, j = 4, k = 3;
if (i == j++)
        k = ++i;
```

   a) Es gilt $i = 6$
   b) Es gilt $i = 5$
   c) Es gilt $k = 3$
   d) Es gilt $k = 6$

6) Betrachten Sie folgenden C++-Code, bei dem ich zur besseren Orientierung Zeilennummern eingefügt habe (im Quellcode stehen natürlich keine Zeilennummern):

```
1:      struct X {
2:          int value1, value2;
3:          X(int a, int b = 1) : value1{a}, value2{b} {}
4:      };
5:
6:      int main()
7:      {   X x1;
8:          X x2{-1};
9:          x2.value2 = 5;
```

a) Nach Zeile 8 besitzt `x2.value1` den Wert -1, `x2.value2` besitzt einen zufälligen Wert.

b) Der Unterschied zwischen einem `struct` und einer `class` ist, dass eine `struct` keinen Konstruktor und keinen Destruktor haben kann.

c) Alle Teile einer `struct` sind automatisch `public`, wenn man nicht explizit den Zugriff einschränkt.

d) Die Zeile 8 ist korrekt, da der Konstruktor von `X` für `b` einen Defaultwert besitzt.

7) a) Python besitzt den Datentyp `int` für ganze Zahlen. Dieser Typ kann nie überlaufen, da er beliebig viele Stellen speichern kann.

b) C++ besitzt mehrere vordefinierte Datentypen für ganze Zahlen, die alle überlaufen können.

c) Python besitzt den Datentyp `float` für Gleitkommazahlen. Dieser Typ vermeidet Rundungsfehler, da er beliebig viele Stellen speichern kann.

d) C++ besitzt mehrere vordefinierte Datentypen für Gleitkommazahlen, die alle nur eine beschränkte Anzahl von signifikanten Stellen speichern können.

8) In einem C++-Programm sei ein globaler Vector `vector<int> x(100);` mit genau 100 Werten bereits definiert und mit Werten befüllt worden. Alle Array-Elemente, die im Intervall [1,10] liegen und ungerade sind, sollen quadriert werden:

```
for (int i = 0; i++ < 100; )
        if ((1 <= x[i] <= 10) & (x[i] % 2 == 1))
                x[i] = x[i]^2;
```

a) Um alle Elemente eines Containers zu durchlaufen, gibt es ab C++11 den range-based `for` Loop.

b) Eine Zählschleife wie hier ist immer schneller als ein range-based `for` Loop.

c) Die Bedingung `(1 <= x[i] <= 10)` ist immer falsch.

d) Die Elemente des Containers werden durch diesen Code nicht quadriert.

9) Betrachten Sie folgende Schleife:

```
double x, h;
for (x = 0, h = 0.6; x = 6; x += h)
        cout << x << endl;
```

a) Es handelt sich um eine Endlosschleife.
b) `endl` liegt im Namespace `std`.
c) Wenn man 10 Mal 0.6 zu 0 addiert, erhält man sicher exakt 6.
d) Die Summe zweier positiver Gleitkommazahlen kann negativ werden.

10) Kreuzen Sie alle korrekten Codefragmente an, die eine Python-`list x` mit den Zahlen 1, 2, 3, 4, 5 erzeugen:

a) `x = [ 1, 2, 3, 4, 5]`
b) `x = [ range(5) ]`
c) `x = [ range(1, 6)]`
d) `x = list(range(1,6))`

11) Betrachten Sie folgenden C++-Code, bei dem ich zur besseren Orientierung Zeilennummern eingefügt habe (im Quellcode stehen natürlich keine Zeilennummern):

```
1:      class X {
2:          int value1, value2;
3:          X(int a = 0, int b = 1) : value1{a}, value2{b} {}
4:      };
5:
6:      int main()
7:      {   X x1;
8:          X x2{-1};
9:          x2.value2 = 5;
```

a) Die Klasse `X` besitzt keinen Default-Konstruktor.
b) Alle 3 Zeilen von `main()` sind fehlerhaft, da auf private Teile von `X` zugegriffen wird.
c) Private Attribute einer Klasse darf man überall lesen, aber nicht schreiben.
d) C++ fügt zu `X` einen Destruktor hinzu, der `public` ist.

12) Betrachten Sie das folgende Python-Codefragment:

```
x = ( 1, 2, -3., True, "Ein Text")
```

a) `x[:-1]` ist das Tupel `x` ohne das letzte Element.
b) `x[0:]` ist das Tupel `x` ohne ihr Anfangselement.
c) `x[-1::-1]` ist das Tupel `x` in umgekehrter Reihenfolge
d) Eine Schleife der Form: `for i in x:` ist ein Fehler, weil man mit so einer Schleife nur über eine Liste iterieren kann und nicht über ein Tupel.