Lösung Klausur 3b, infT

Fragen zur Sprache C++
Korrekte Antworten sind fett markiert. Vormalige Lücken sind mit fettem Text ergänzt.

1. In C++ ist eine Standard-Header-Datei
   a) eine Objektdatei.
   b) eine Textdatei.
   c) eine ausführbare Datei.

2. Ein Funktionsaufruf ist ein Ausdruck, dessen Typ bestimmt ist durch
   a) den Return-Wert der Funktion.
   b) die an die Funktion übergebenen Argumente.
   c) die im Funktionskopf deklarierten Parameter.

3. Zur Ausführung der Anweisung `cout << "Hello world!";` (in `main` formuliert) genügt es, die Header-Datei `iostream` ins Programm zu inkludieren.
   a) richtig
   b) falsch

4. Beim Erzeugen eines Objekts gilt grundsätzlich, dass
   a) jedes Element mit einem passenden Wert initialisiert wird.
   b) Speicher für die Datenelemente reserviert wird.
   c) Speicher auf dem Heap alloziert wird.

5. Nach den Definitionen `bool flag;  int x = 3, y = 2;` gibt die Anweisung `cout << (flag = x == y);` Folgendes auf dem Bildschirm aus:
   a) `2`
   b) `0` **(oder** `false`**)**
   c) `1` (oder `true`)

6. Im Anschluss an die Definition `int i = 2, j;` bewirkt die Anweisung `i *= j = 4;`
   a) die Zuweisung von `4` für die Variable `i` und `j`.
   b) die Zuweisung von `8` **für** `i` **und** `4` **für** `j`**.**
   c) eine Fehlermeldung des Kompilers.

7. In C++ ist jeder Vergleich ein Ausdruck vom Typ `int` mit dem Wert `0` ("false") oder `1` ("true").
   a) richtig
   b) falsch

8. Wenn eine Source-Datei geändert wird, müssen alle zu einem C++-Programm gehörenden Source-Dateien neu kompiliert werden, damit eine ausführbare Datei erzeugt werden kann.
   a) richtig
   b) falsch

9. Gegeben seien die folgenden Deklarationen: `void func(double&);  double x = 9.7;` . Nach Rückkehr der Programmausführung aus dem Funktionsaufruf wurde der Wert der Variablen möglicherweise geändert.
   a) richtig
   b) falsch

10. Wenn beim Aufruf einer Funktion weniger Argumente übergeben werden als Parameter im Funktionskopf deklariert sind, werden übrige Parameter mit `0` als Default-Wert initialisiert.
a) richtig
b) falsch

11. Angenommen, eine Funktion ist mit Default-Argumenten definiert und bei Aufruf wird ein Argument weggelassen. In diesem Fall müssen
a) auch nachfolgende Argumente weggelassen werden.
b) auch die Argumente davor weggelassen werden.
c) nie weitere Argumente weggelassen werden.

12. Der von lokalen, nicht als `static` deklarierten Objekten belegte Speicher wird freigegeben, sobald der entsprechende **Block** verlassen wird.

13. Ein statisches Objekt, das ausserhalb jeder Funktion definiert ist, kann
a) überall in demselben Source-File
b) überall im Programm
c) nach seiner Definition in derselben Quelldatei
verwendet werden.

14. Die Signatur einer Funktion besteht aus
a) dem ganzen Funktionskopf.
b) dem Namen der Funktion und dem Return-Typ der Funktion.
c) dem Namen der Funktion, der Anzahl und dem Typ der Parameter.

15. Welche der folgenden Anweisungen deklarieren eine überladene Funktion?
a) `double func2(double);  int func2(double);`
b) `double func3(double);  int func3(double, double = 0);`
c) `int func1(double);  int func1(double, double);`

16. Angenommen, die Funktion `myFunc()` ist im Namensraum `mySpace` definiert und wird ausserhalb des Namensraums aufgerufen. Ohne Verwendung einer `using`-Deklaration oder `using`-Direktive wird beim Aufruf dem Namen der Funktion `mySpace::` vorangestellt.

17. Sind im aktuellen Namensraum und in einem mit der `using`-Direktive importierten Namensraum gleiche Bezeichner vorhanden, dann
a) gibt der Kompiler eine Fehlermeldung aus.
b) führt dies nicht automatisch zu einem Namenskonflikt.
c) kann es zu Mehrdeutigkeiten kommen, wenn einer dieser Bezeichner angesprochen wird.

18. Gegeben seien folgende Definitionen: `long z;  long& rz = z;`.
Dann wird im Ausdruck `++rz;` der Wert von `z` nicht inkrementiert.
a) richtig
b) falsch

19. Gegeben sei der folgende Prototyp `int& func(void);`.
Dann wird bei der Anweisung `func() = 7;`
a) der Funktion ein neuer Return-Wert zugewiesen.
b) dem durch den Funktionsaufruf referenzierten Objekt ein neuer Wert zugewiesen.
c) der Kompiler eine Fehlermeldung ausgeben.

20. Gegeben sei die folgende Funktion: `void triple(int& a) { a *= 3; }`.
Dann wird beim Aufruf `triple(2);`
a) der Kompiler eine Fehlermeldung ausgeben.
b) die Zahl `3` verdoppelt.
c) die Zahl `2` verdreifacht.

21. Gegeben sei die folgende Definition: `char ch = 'A';` .
Dann hat der Ausdruck `&ch` den folgenden Typ:
a) `char *`
b) `char`
c) `char&`

22. Speicher für die Member-Daten eines Objekts der Klasse `Y` (mit Konstruktor `Y()`) wird angelegt, wenn
a) der Aufruf `Y *myY = new Y;` zur Ausführung kommt.
b) die Zeile `Y yourY;` ausgeführt wird.
c) die Klasse definiert wird.

23. Nach den Deklarationen `void func(double *p1, double *p2);  double x = 1.1, y = 2.2;` können Sie die Funktion `func()` wie folgt aufrufen:
a) `func(*x, *y);`
b) `func(&x, &y);`
c) `func(x, y);`

24. Wenn zwei verschiedene Objekte derselben Klasse angelegt werden, wird der Maschinencode für jede Methode auch zweimal erzeugt.
a) richtig
b) falsch

25. Wenn eine Funktion eine Referenz oder einen Zeiger auf ein Objekt, das innerhalb der Methode definiert wurde, zurückgibt, muss das Objekt als **`static`** deklariert werden.

26. Wenn bei der Definition eines Objekts der Kompiler den Konstruktor mit der passenden Signatur nicht finden kann, wird
a) kein Objekt angelegt und eine Fehlermeldung ausgegeben.
b) ein Objekt angelegt und mit Default-Werten initialisiert.
c) ein Objekt angelegt und nicht initialisiert.

27. Beim Aufruf einer Methode wird ein "verstecktes" Argument mit der Adresse des aktuellen Objekts übergeben. Dabei handelt es sich um den **`this`-Zeiger**.

28. Gegeben sei eine Matrix `net` mit Elementen vom Typ `double`, die zwei Zeilen und drei Spalten besitzt. Geben Sie die Anweisung an, um dem letzten Element in der letzten Zeile den Wert `9.1` zuzuweisen: **`net[1][2] = 9.1;`**

29. Wenn einer Funktion ein Vektor als Argument übergeben wird, erhält die Funktion
a) nur das erste Vektorelement.
b) eine Kopie des Vektors.
c) die Adresse des ersten Vektorelements.

30. Es sei `X` eine Klasse. Um einen Vektor von Zeigern auf Objekte vom Typ `X` definieren zu können, muss die Klasse einen Default-Konstruktor besitzen.
a) richtig
b) falsch

31. Mit der Definition `long *ptr = new int;`
a) wird ein Objekt vom Typ `int` angelegt und implizit in `long` konvertiert.
b) wird ein Objekt vom Typ `long` erzeugt.
c) wird vom Kompiler eine Fehlermeldung ausgegeben.

32. Die Funktion **`func()`**, die ein Array mit **`Employee`**-Objekten als Argument erwartet und keinen Return-Wert liefert, weist folgenden Prototyp auf

a) **`void func(Employee e[]);`**

b) **`void func(Employee *e[]);`**

c) **`void func(Employee e);`**

33. Angenommen, es soll eine Funktion **`strChr()`** definiert werden, die das erste Auftreten eines Zeichens in einem C-String untersucht und einen Zeiger auf das gefundene Zeichen zurückgibt. Dann kann die Funktion folgenden Prototyp aufweisen:

a) **`char strChar (char s[], int c);`**

b) **`char *strChr(const char *s, int c);`**

c) **`char strChar (const char *s, int c);`**

34. Nach dem Aufruf des **`delete`**-Operators kann nicht überprüft werden, ob der dynamisch reservierte Speicher freigegeben wurde.

a) richtig

b) falsch

35. Angenommen, der Operator **`new`** reserviert dynamisch den Speicher für ein Objekt einer Klasse. Dann sorgt er auch dafür, dass

a) ein passender Konstruktor für das Objekt aufgerufen wird.

b) alle von ihm belegten Bytes mit `0` initialisiert werden.

c) der New-Handler aufgerufen wird, wenn kein Speicherplatz verfügbar ist.

36. Im Anschluss an die Definitionen **`long double z = 1.1, *ptr = &z;`** kann der von **`z`** belegte Speicher wie folgt freigegeben werden: **`delete ptr;`**

a) richtig

b) falsch

37. Angenommen, es soll eine Klasse **`MeasureArr`** entwickelt werden, die einen Vektor variabler Länge mit Elementen einer vorgegebenen Klasse **`Measure`** darstellt. Dann ist bei der Definition der Klasse

```
class MeasureArr {
private:
  int n;      // Vektorlaenge
  // . . .
};
```

ein dynamisches Element **`msPtr`** für den Vektor wie folgt zu deklarieren:

a) **`MeasureArr *msPtr;`**

b) **`Measure *msPtr;`**

c) **`Measure msPtr[n];`**

38. Angenommen, zwei verschiedene Zeiger **`ptr1`** und **`ptr2`** zeigen auf denselben dynamisch reservierten Speicher. Dann wird mit der Anweisung **`delete[] ptr1;`** der dynamische Speicher nicht freigegeben, da der Zeiger **`ptr2`** immer noch dorthin zeigt.

a) richtig

b) falsch

39. Gegeben sei folgender Ausschnitt aus einer Klassendefinition:

```
class LongArr {
private:
  long *ptr;
  // . . .
public:
  LongArr(int n);   // Erzeuge einen Vektor der Laenge n
  // . . .
};
```

In der Definition des Konstruktors wird dann mit folgender Anweisung der erforderliche Speicher reserviert: **`ptr = new long[n];`**