KLAUSUR im FACH ***OOSWE*** im Sommersemester 2024

Name, Vorname: …….

Studiengang/Semester: ……

Prüfer: Dipl.-Ing. (BA) Nico Klaas

Bearbeitungshinweise

1. Tragen Sie auf jeder Seite in der Kopfzeile Ihre Matrikelnummer ein.

2. Der Aufgabensatz (inkl. Deckblatt und Anhang), der aus 10 Seiten besteht (Seite 1 bis 10) ist auf Vollständigkeit zu überprüfen.

3. Der Aufgabensatz ist mit den Lösungsblättern abzugeben.

4. Lösungen auf selbst mitgebrachten Lösungsblättern werden nicht ausgewertet. Verwenden Sie die Ihnen ausgeteilten Lösungsblätter und tragen Sie auch dort Ihre Matrikelnummer ein.

5. Bei Rechenaufgaben muss der Lösungsweg ersichtlich und lesbar sein, sonst erfolgt keine Bewertung der Aufgabe oder des Aufgabenteils.

6. Die Bearbeitungszeit beträgt 60 Minuten.

7. Es wird hiermit darauf hingewiesen, dass vom Prüfungsamt nicht vorher geprüft wurde, ob Sie das Recht bzw. die Pflicht zur Teilnahme an dieser Klausur haben. Die Teilnahme erfolgt auf eigene Gefahr, gleichzeitig bekundet die Teilnahme die Zustimmung zu diesem Passus.

8. Hilfsmittel:
   - Coding Styleguide (ist selbst mitzubringen) – nur Markierungen mit einem Marker sind erlaubt (keine eigenen Notizen)
   - Taschenrechner sind **nicht** erlaubt – auch **keine** Smartphones und weitere digitale Geräte

9. **Die Nichteinhaltung des Coding Styleguides führt zu Punktabzug**

10. Bewertung:
    Gesamtpunktzahl: 70 Punkte (17% Überhang)
    Note 1,0: 60 Punkte Note 4,0: 30 Punkte

Aufgabe	**1**	**2**	**3**	**4**	**5**	**SUMME**
Punkte	**20**	**18**	**10**	**10**	**12**	**70**
Erreichte Punkte						

Aufgabe 1: (20 Punkte)

1.1 Das ++ in C++ steht für _____ (1 P)

1.2 Welcher Sprachumfang von Super C (OO Erweiterung von C++) ist deutlich einfacher anzuwenden als der entsprechende Sprachumfang von C. Hier kommt schon Operatorüberladen zur Anwendung

_____. (1 P)

1.3 Der ::-Operator wird auch _____-Operator genannt. (1 P)

1.4 Gegeben sei die Klasse A mit dem Defaultkonstruktor.

Wie sieht hier der Most Vexing Parse aus? _____ (1 P)

Statt einer Instanziierung eines Objektes handelt es sich jetzt um eine

_____ (1 P)

1.5 Bei welchem Datentyp ergibt sich der konkrete Datentyp aus der Initialisierung? Fügen Sie einfach den Datentyp ein.

```
_____ x = u32Var;
```
(1 P)

1.6 Welches OO-Grundprinzip wurde anhand eines Eisberges erklärt: _____ (1 P)

1.7 Legen Sie im folgenden Code eine Klassenvariable für die Klasse A (Coding Styleguide!) an. Es soll sich um eine vorzeichenlose 32-Bit Zahl handeln. Initialwert sei 42; (2 P)

A.cpp

```
#include <stdint>
#include "A.h"
```

A.h

```
class A
{
public:


};
```

1.8 Begründen Sie **kurz**, warum es zu einem Fehler kommt. (1 P)

Main.cpp

```
#include <stdint>
#include "B.h"

//somewhere in main
B* pb = new B{};
```

B.h

```
class B
{
public:
   B(int32_t s32V) : s32V_{s32V} { }
private:
   int32_t s32V_;
};
```

1.10 Gegeben sei:

```
class C
{
};
```

Was ergibt `sizeof (C)` auf einem 32-Bitsystem: _____ (1 P)

Welche Funktionen hat die Klasse C per default: (1 P)

Welche Operatoren hat die Klasse C per default: (1 P)

1.11 Der Vorteil eines static_cast im Gegensatz zu einem C-Cast ist darin zu sehen, dass schon zur

_____ überprüft wird, ob der Cast überhaupt möglich ist. (1 P)

1.12 Mit dem Keyword _____ wird eine Exception geworfen und mit dem Keyword

_____ wird eine Exception gefangen. (2 x 0,5 P)

1.13 Java und C# haben einen Garbage Collector. C++ hat diesen noch nicht. Nennen jeweils einen Vorteil und einen Nachteil eines Garbage Collectors: (2 P)

- Vorteil:
- Nachteil:

1.14 Mit welcher Funktionalität von C++ versucht man den Nachteil eines fehlenden Garbage Collectors zu kompensieren? (2 P)

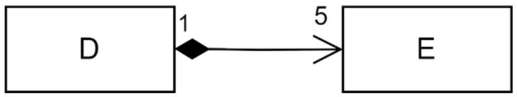
_____

1.15 _____ ist derzeit das führende Testframework im Bereich von C++.

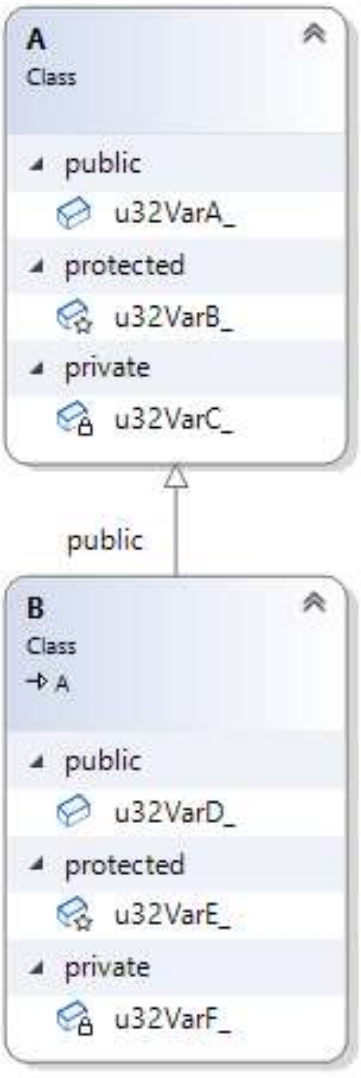
Eine Vergleichsfunktion/-makro lautet z.B. _____ (2 x 0,5 P)

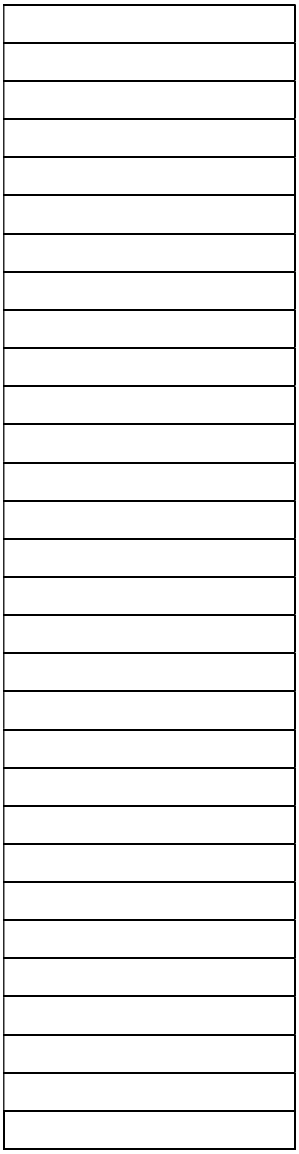
Aufgabe 2: (18 Punkte)

2.1 Implementieren Sie die Klasse D (Komposition). Notwendiger Konstruktor sei vorhanden. E hat Defaultkonstruktor. (2 P)



2.2 Gegeben sei das folgende Klassendiagramm. Mit `B*pb = new B();` wird ein neues Objekt instanziiert. Zeichnen Sie in den rechten Speicherbereich (eine Zelle ist ein Byte) ein, wo welche Attribute eines B-Objektes liegen. Ebenso ist einzuzeichnen, wohin pb zeigt. Der Speicher ist wie folgt organisiert: Oben stehen große Adressen. In u32VarB_ soll 0xBADDCAFE stehen. Es handelt sich um eine LittleEndian-Plattform. Die Reihenfolge der Attribute im Code sind wie im Klassendiagramm aufgezeigt. (6 P)





2.3 Im Programmcode gibt es einen Zeiger auf ein Objekt E mit dem Name pE.
Jetzt soll eine Referenz auf dieses Objekt definiert werden. Wie sieht der exakte Code aus? (2 P)

2.4.1 Erstellen Sie ein Klassentemplate mit dem Namen Box. Die Templateklasse Box hat ein Attribut des generischen Typs. Es werden die Funktionen setValue und getValue zur Verfügung gestellt, welche dieses Attribut lesen und beschreiben. Ebenso ist ein Konstruktor enthalten, der das Attribut initialisiert. Das Klassentemplate soll das Keyword const einmal enthalten. (5 P)

2.4.2 Zeigen Sie kurz anhand von C++ Code auf, wie Sie dieses Klassentemplate benutzten:

- Definition einer Klasse für den Datentyp uint32_t, Initwert sein 79U. (1 P)

- Schreiben Sie den Wert 42U in das Objekt mit setValue und überprüfen Sie den Wert mittels getValue und dem assert-Makro. (2 P)

Aufgabe 3: (10 Punkte)

Gegeben sei die Klasse Vec2D. Fügen Sie den Code der drei Operatoren ein.

```
class Vec2D
{
public:
   Vec2D(f32_t f32X, f32_t f32Y) : f32X_{ f32X }, f32Y_{ f32X } { }
   Vec2D() : f32X_{ 0.f }, f32Y_{ 0.f} { }

   // Überladen Sie hier den * Operator, damit Sie einen Vector mit einem Skalar
   // multiplizieren können: f32V * myVec2D.
   // Das Original ist dabei nicht zu ändern                                (4 P)









   // Überladen Sie hier den ++ Operator Präinkrement, der jeweils auf f23X_
   // und f32Y_ den Wert 1 addiert.
   // Das Original ist zu ändern, der Operator soll verkettbar sein.        (3 P)









   // Überladen Sie den ~ Operator, der den Betrag des Vektors auf 1 normieren soll
   // Das Original ist zu ändern, eine Verkettung ist nicht gewünscht.      (3 P)









friend std::ostream& operator<<(std::ostream& ostr, const Vec2D& v2);

private:
   f32_t f32X_;
   f32_t f32Y_;
};

std::ostream& operator<<(std::ostream& ostr, const Vec2D& v2)
{
  ostr << v2.f32X_ << " - " << v2.f32Y_ << std::endl;
  return ostr;
}
```

Aufgabe 4: (10 Punkte)

Gegeben sei das Decorator-Pattern (Anhang) aus dem Labor. In einem Sales-Objekt werden in einem Vektor die Zeiger auf die erzeugten Objekte gespeichert.

4.1 Zeichnen Sie ein **Objekt-Diagramm**, nachdem ein Espresso sowie ein Houseblend mit Sugar und Soy darin abgespeichert wurde. Vector muss nicht im Diagramm dargestellt werden. (4 P)

4.2 Beim Löschen des Sales-Objektes mussten alle Objekte gelöscht werden, auf die direkt oder indirekt referenziert wird, Es werden keine Smart-Pointers eingesetzt.

```
Sales::~Sales()
{
   for (auto it : VectorBeverages_)
   {
      vDeleteBeverageRecursive(it);
   }
}
```

Implementieren Sie die Funktion vDeleteBeverageRecursive(). Tipp: Sie müssen dabei bestimmen, ob es sich bei pb um ein Getränk (Espresso, Houseblend, Decaf, DarkRoast) oder um eine Zutat (Soy, Milk, Sugar, …) handelt. (6 P)

```
void Sales::vDeleteBeverageRecursive(MyDesignPattern::Beverage* pb)
{









}
```

Aufgabe 5: (12 Punkte)

5.1 Definieren Sie einen Vektor für int32_t und weisen Sie sechs Elemente bei der Definition zu. (2 P)

5.2 Definieren Sie eine Funktion vOutput, die den Vektor aus 5.1 auf der Console ausgibt. Es darf dabei kein neuer Vektor anlegt werden. vOutput darf die Elemente des Vektors nicht ändern. (3 P)

```
void vOutput(                                                    )
{




}
```

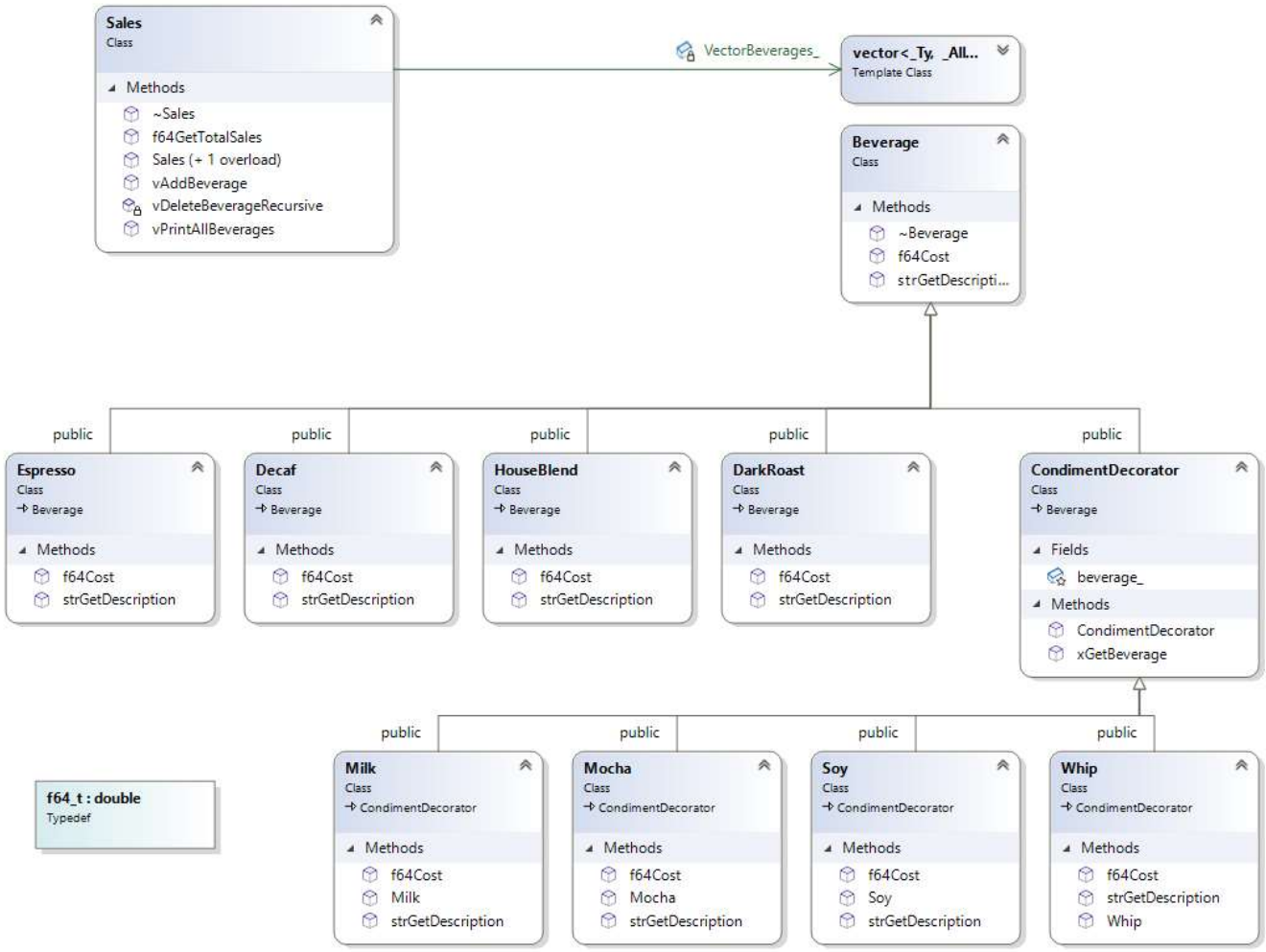
5.3 Definieren Sie eine Funktion vRemove, die aus einem Vektor aus 5.1 die geraden Zahlen entfernt. Es darf dabei kein neuer Vektor angelegt werden. Verwenden Sie die Funktion erase (Anhang, Wichtig: Rückgabewert) und sowie eine while-Schleife. (6 P)

```
void vRemove(                                                    )
{




}
```

5.4 Rufen Sie die Funktion aus 5.3 mit dem Vektor aus 5.1 auf. (1 P)

ANHANG:

Decorator Pattern



API list - Vector

std::vector::erase

<vector>

C++98 C++11

```
iterator erase (const_iterator position);iterator erase (const_iterator first, const_iterator last);
```

Erase elements

Removes from the vector either a single element (***position***) or a range of elements (`[first,last)`).

This effectively reduces the container size by the number of elements removed, which are destroyed.

Because vectors use an array as their underlying storage, erasing elements in positions other than the vector end causes the container to relocate all the elements after the segment erased to their new positions. This is generally an inefficient operation compared to the one performed for the same operation by other kinds of sequence containers (such as list or forward_list).

Parameters

`position`

Iterator pointing to a single element to be removed from the vector.
Member types `iterator` and `const_iterator` are random access iterator types that point to elements.

`first, last`

Iterators specifying a range within the vector] to be removed: `[first,last)`. i.e., the range includes all the elements between ***first*** and ***last***, including the element pointed by ***first*** but not the one pointed by ***last***.
Member types `iterator` and `const_iterator` are random access iterator types that point to elements.

Return value

An iterator pointing to the new location of the element that followed the last element erased by the function call. This is the container end if the operation erased the last element in the sequence.

Member type `iterator` is a random access iterator type that points to elements.

Example

```
// erasing from vector
#include <iostream>
#include <vector>

int main ()
{
  std::vector<int> myvector;

  // set some values (from 1 to 10)
  for (int i=1; i<=10; i++) myvector.push_back(i);

  // erase the 6th element
  myvector.erase (myvector.begin()+5);

  // erase the first 3 elements:
  myvector.erase (myvector.begin(),myvector.begin()+3);

  std::cout << "myvector contains:";
  for (unsigned i=0; i<myvector.size(); ++i)
    std::cout << ' ' << myvector[i];
  std::cout << '\n';

  return 0;
}
```