Institut für Datentechnik
Fachgebiet Echtzeitsysteme
Prof. Dr. rer. nat. Andy Schürr

TECHNISCHE UNIVERSITÄT DARMSTADT

Klausur zum Programmierpraktikum C und C++

14.10.2014

Name: _____

Vorname: _____

Matrikelnummer: _____

Beachten Sie bitte Folgendes:

- **Die Klausur besteht aus 10 Blättern (inkl. Deckblatt) mit 4 Aufgaben.**
  Prüfen Sie die Klausur auf Vollständigkeit.
- **Die Bearbeitungszeit beträgt 90 Minuten. Es sind alle Aufgaben zu bearbeiten.**
- **Hilfsmittel sind nicht erlaubt.**
- **Nur unterschriebene und mit dokumentenechtem Stift (kein Bleistift, kein Rotstift) bearbeitete Klausuren können benotet werden.**
- **Das Code-Handout muss zusammen mit der Klausur abgegeben werden, wird allerding nicht bewertet.**
- **Für eine korrekte Lösung müssen nicht immer alle Lücken ausgefüllt werden.**

Unterschrift _____

Diesen Teil nicht ausfüllen!

Aufgabe	1	2	3	4	∑
maximal	12	29	23	26	90
Punkte					
Zeichen					

Datum _____ **Unterschrift** _____

Note

Datum _____ **Unterschrift** _____

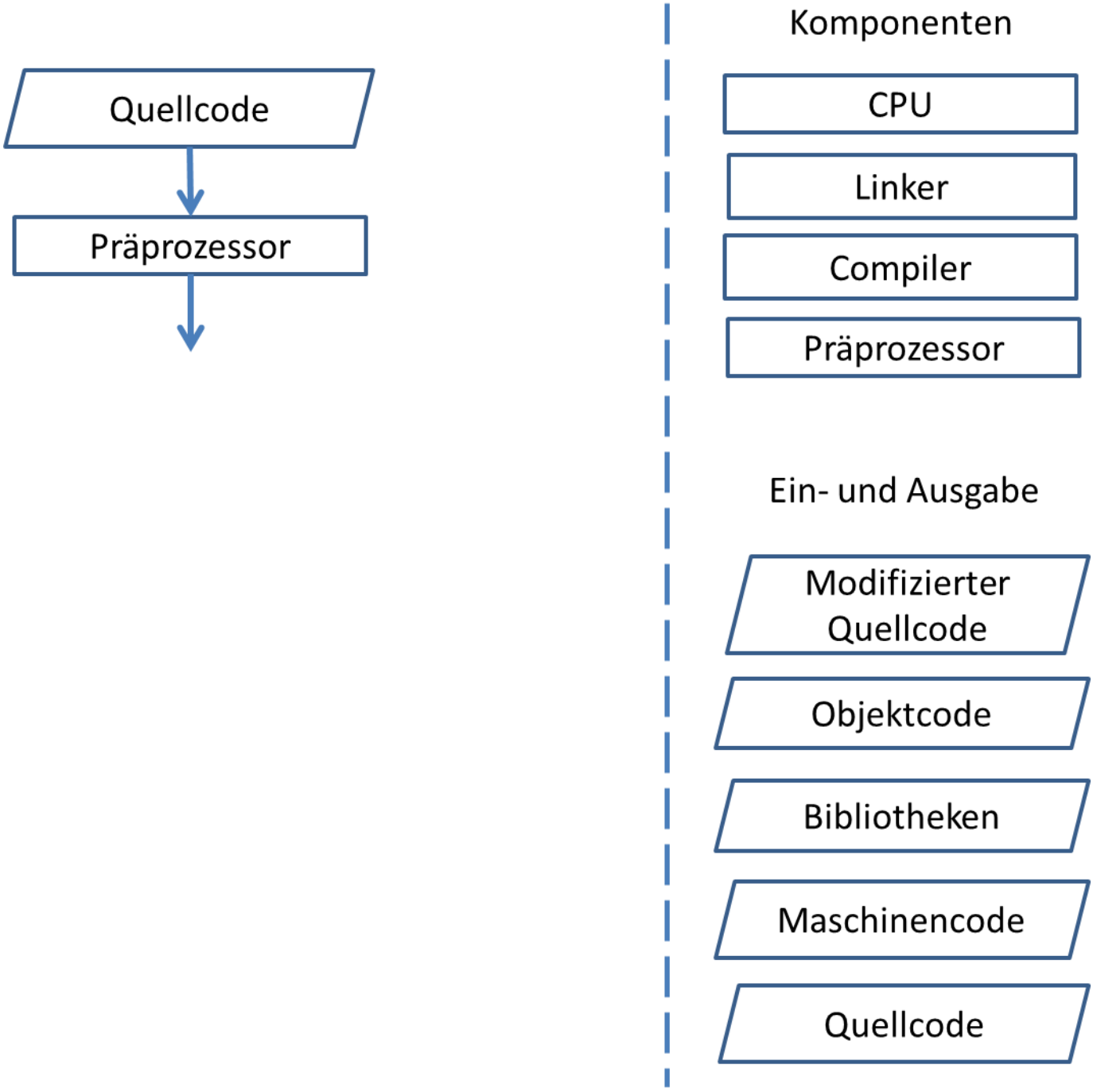
Klausur eingesehen am

Datum _____ **Unterschrift** _____

Diese Seite wurde absichtlich frei gelassen.

Aufgabe 1: Kompilierungsvorgang bei C/C++ ____ / 12

(a.) Vervollständigen Sie das folgende Diagramm, um den Kompilierungsvorgang eines C/C++-Programms darzustellen. Verwenden Sie nur die vorgegebenen Elemente.



(b.) Beschreiben Sie *eine* Aufgabe des Präprozessors

(c.) Erläutern Sie, was dazu führen kann, dass ein C++-Projekt zwar problemlos kompiliert (Compiler), aber nicht gelinkt (Linker) werden kann

(d.) Was ist der Unterschied zwischen einer *Deklaration* und einer *Definition* in C++?

Geben Sie für jeden der Begriffe ein Beispiel beim Anlegen einer Klasse.

Aufgabe 2: Allgemeines Verständnis ____ / 29

Lesen Sie das Code-Handout genau durch.

(a.) Der Quelltext von `main_exercise2.cpp` (im Handout) produziert eine fehlerhafte Ausgabe. Die gewünschte Ausgabe sollte folgende sein:

```
Ship SHIP shown as ?.
Ship CARGO shown as C.
Ship FERRY shown as F.
```

Welche Ausgabe wird stattdessen erzeugt?
Hinweis: Polymorphie.

(b.) Beschreiben Sie knapp, worin das Problem liegt (Prosa) und wie man den Bug behebt (Code).

_____

_____

_____

_____

_____

_____

_____

_____

_____

(c.) Aus Designgründen wollen wir verbieten, `Ship`-Instanzen zu kopieren.

Nehmen Sie eine kleine Änderung in *`Ship.h`* vor, um dies statisch vom Compiler verbieten zu lassen!
Hinweis: Sie müssen nur neuen Code hinzufügen und keinen vorhandenen Code ändern!

```
#ifndef SHIP_H_
#define SHIP_H_
#include <string>

class Ship {
public:
    Ship(const std::string &callSign, double latitude, double longitude);
    virtual ~Ship();

    const std::string getCallSign() const;
    const std::string getMarker() const;
    double getLatitude() const;
    double getLongitude() const;

    _____

    _____

private:
    std::string callSign;
    double latitude;
    double longitude;

    _____

    _____

};

#endif /* SHIP_H_ */
```

(d.) Betrachten Sie den folgenden Prototyp:

```
void describe(const Ship &ship);
```

Schreiben Sie den Ausdruck `const Ship &ship` in einen äquivalenten Ausdruck um, der *nur* Zeiger verwendet:

Wie sieht nun der Aufruf der Methode `Ship::getMarker()` innerhalb von `describe` aus?

Nennen Sie *zwei* Vorteile der Verwendung von Referenzen (allgemein oder an dieser Stelle).

Referenzen haben viele Vorteile, doch wann ist es nötig, Pointer zu verwenden?

Beschreiben Sie *eine* Situation.

_____

_____

_____

_____

_____

_____

_____

_____

Was bewirkt `const` im Ausdruck `const Ship &ship` (Argument der Funktion `describe`)?

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

(e.) Betrachten Sie den folgenden Methoden-Prototyp in der Klasse `Ship`:

```
const std::string getCallSign() const;
```

Welche Bedeutung hat das erste `const`?

Welche Bedeutung hat das zweite `const`?

(f.) Aus Designgründen haben wir uns entschieden, dass wir unbekannte Schiffe (aktuell `Ship`) in Zukunft als separate Klasse (z.B. `UnknownShip`) modellieren wollen. Als Vorbereitung werden Sie `getMarker` *abstrakt* machen.

Wie sieht der Prototyp von `Ship::getMarker()` als *abstrakte* Methode aus? Welche Konsequenzen hat das für die Verwendung der Klasse `Ship`?

Aufgabe 3: Speicherverwaltung ____ / 23

Lesen Sie das Code-Handout *genau* durch(, wenn nicht schon geschehen).

In dieser Aufgabe wollen wir einige Bestandteile von `CoastControlSystem` implementieren und verbessern.

Zur Vereinfachung der Erzeugung von Schiffen wurde eine `ShipFactory` implementiert, die für jeden Schiffstyp eine entspreche Erzeugungsmethode enthält.

Sehen Sie sich folgenden Code näher an:

```
CargoShip* ShipFactory::createCargoShip(
        const std::string &callSign,
        double latitude, double longitude) {
    CargoShip cargoShip(callSign, latitude, longitude);
    return &cargoShip;
}
```

(a.) Wie nennt man diese Art von Fehler? Beschreiben Sie, wie sich der Fehler im weiteren Programmfluss womöglich auswirken wird.

_____

_____

_____

_____

_____

_____

_____

_____

_____

(b.) Implementieren Sie die Methode `createCargoShip` korrekt.

```
CargoShip *ShipFactory::createCargoShip(
        const std::string &callSign,
        double latitude, double longitude) {

        _____________________________________________________________

        _____________________________________________________________

        _____________________________________________________________

        _____________________________________________________________

        _____________________________________________________________
}
```

(c.) Nennen Sie *zwei* Unterschiede zwischen Stack und Heap.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Wir gehen jetzt davon aus, dass alle derartigen Fehler in der `ShipFactory` behoben wurden.

Der folgende Code zeigt, wie die Hauptroutine von `CoastControlSystem` implementiert werden könnte.
`Radar` nutzt dabei `ShipFactory` zum Erzeugen neuer Objekte.

```
// Contains all ships that are currently known to the CCS
std::list<Ship *> shipsInRange;
// ...
while (isRunning()) {
    // Fetch newly entered ships and add them to the list
    std::vector<Ship*> newShips =
        radar.getNewlyEnteredShipsSinceLastPoll();

    ____________________________________________________________

    ____________________________________________________________

    std::copy(newShips.begin(), newShips.end(),
            std::back_inserter(shipsInRange));

    ____________________________________________________________


    // Remove all ships that are no longer in range
    for(std::list<Ship*>::iterator iter = shipsInRange.begin();
            iter != shipsInRange.end(); ++iter) {

        ____________________________________________________________
        if (!isShipInRange(**iter)) {

            ____________________________________________________

            ____________________________________________________
            iter = shipsInRange.erase(iter);

            ____________________________________________________

            ____________________________________________________
        }

        ____________________________________________________
    }

    ____________________________________________________________

    ____________________________________________________________

    cout << "Ships in range: " << shipsInRange.size() << endl;
    // Draw ships on screen and other stuff...
}
```

(d.) Bei einem Systemtest stürzt das System immer wieder nach einiger Zeit ab. Obwohl zwar reger Schiffsverkehr herrscht, pendelt sich die Anzahl der momentan überwachten Schiffe aber stets bei ca. 1000 ein.

Wie nennt man die Ursache des Absturzes?

_____

_____

_____

_____

(e.) Verändern Sie den Code, sodass der Fehler behoben wird.
Hinweis: Es sind nur Einfügungen nötig.

(f.) Auf welche (aus der Vorlesung bekannte) Art könnte man das Problem außerdem elegant lösen?

_____

_____

_____

_____

_____

_____

_____

_____

Aufgabe 4: Fortgeschrittene Themen ____ / 26

Um eine Übersicht bestimmter Bereiche des Seegebiets zu erstellen, soll ein Teil aller sichtbaren Schiffe in einer Liste zusammengefasst werden. Dazu schreiben Sie in dieser Aufgabe eine Filterfunktion und zwei Filter-Prädikate.

Der folgende Code testet den zu implementierenden Filter:

```
int main () {
    Ship ship("SHIP", 53.667, 5.325);
    Ship cargo("CARGO", 54.667, 5.825);
    Ship ferry("FERRY", 53.977, 6.825);

    Ship *shipsArray[3] = { &ship, &cargo, &ferry };
    vector<Ship*> ships(shipsArray, shipsArray + 3);

    vector<Ship*> selectedShips = filter(
         ships.begin(), ships.end(), isFarNorth);

    // output selected ships...
    return 0;
}
```

(a.) Implementieren Sie die Funktion `filter`. Beachten sie auch den Kommentar!

```
/**
 * Returns a vector of all ships in the range of
 * begin (inclusive) to end (exclusive) for which
 * the given predicate evaluates to true.
 */
std::vector<Ship*> filter(
        std::vector<Ship*>::iterator begin,
        std::vector<Ship*>::iterator end,
        bool (*predicate)(const Ship &ship)) {

    std::vector<Ship*> selectedShips;

    _____________________________________________________________
   // Iterate over ships (last ship is the one before 'end').

    _____________________________________________________________

    _____________________________________________________________

    _____________________________________________________________

    // For each ship, check whether it fulfills the predicate
    // and, if yes, add it to 'selectedShips'.

    _____________________________________________________________

    _____________________________________________________________

    _____________________________________________________________

    _____________________________________________________________

    _____________________________________________________________

    _____________________________________________________________

    return selectedShips;
}
```

(b.) Implementieren Sie jetzt die Funktion `isFarNorth`, die zurückgibt, ob ein Schiff nördlicher als 54° Breite (*latitude*) ist.

```
_______ isFarNorth (_____________________________________) {

    return ____________________________________________________;
}
```

In den folgenden Teilaufgaben bauen Sie den Code so um, dass man sowohl Funktionszeiger als auch Funktoren an `filter` übergeben kann.

(c.) Wie sieht der Prototyp von `filter` aus, wenn wir statt des Funktionszeigers einen Templateparameter verwenden?

_____

_____

```
std::vector<Ship*> filter(
        std::vector<Ship*>::iterator begin,
        std::vector<Ship*>::iterator end,

         _____

         _____
       );
```

(d.) Implementieren Sie den Funktor `IsNorthernOfPredicate`, der in seinem Konstruktor einen Breiten-Wert übergeben bekommt und diesen in seinem `operator()` für den Vergleich nutzt.

Ergänzen Sie folgendes Gerüst der Klasse.
Hinweis: Achten Sie auf die korrekte Methodensignatur!

```
class IsNorthernOfPredicate {
public:
    IsNorthernOfPredicate(double latitudeLimit) :
        _____{}

    _____

    _____

    _____

    _____

    _____

private:
    double latitudeLimit;
};
```

(e.) Verwenden Sie `IsNorthernOfPredicate` in folgendem Code, um das Verhalten der Funktion `isFarNorth` nachzuahmen.

```
int main () {
    Ship ship("SHIP", 53.667, 5.325);
    Ship cargo("CARGO", 54.667, 5.825);
    Ship ferry("FERRY", 53.977, 6.825);

    Ship *shipsArray[3] = { &ship, &cargo, &ferry };
    vector<Ship*> ships(shipsArray, shipsArray + 3);

    ____________________________________________________________

    ____________________________________________________________

    vector<Ship*> selectedShips = filter(

    ____________________________________________________________

    ____________________________________________________________

    ____________________________________________________________

    ____________________________________________________________

    );

    // output selected ships...
    return 0;
}
```