Klausur im SS2016
Modul Informationstechnik

Institut für Technik der Informationsverarbeitung – ITIV
Prof. Dr.-Ing. E. Sax

Modul Informationstechnik
Datum: 06.09.2016
Name: Max Mustermann
Matr. Nr.: 0000000
ID: 1

Hörsaal: Neue Chemie
Platz: 1

Hinweise zur Klausur

Hilfsmittel:

- Erlaubte Hilfsmittel sind Lineal und eine Notizen-/Formelsammlung, handgeschrieben DIN A4 Blatt (zweiseitig beschrieben).
- Verwenden Sie zum Bearbeiten der Aufgaben nur dokumentenechte Schreibgeräte – keinen Bleistift, keine Rotstifte!
- Alle nicht genannten Hilfsmittel sind untersagt. Dies beinhaltet jegliche Kommunikation mit anderen Personen sowie die Benutzung eines Taschenrechners.

Klausurdauer:
Die Prüfungsdauer für die Klausur beträgt 120 Minuten.

Klausurunterlagen:
Die Klausurunterlagen bestehen aus insgesamt 29 Seiten Aufgabenblättern (inklusive dieses Titelblatt, 8 Aufgabenblöcke und 1 zusätzliches Lösungsblatt).
Bitte prüfen Sie vor der Bearbeitung der Aufgaben auf jeder Seite oben Ihre Matrikelnummer sowie Ihre ID und zusätzlich Ihren Namen auf der ersten Seite.
Falls Sie zusätzliche Blätter zur Lösung der Aufgaben benötigen, fragen Sie nach zusätzlichem Lösungspapier bei der Aufsicht. Vermeiden Sie generell das Beschreiben der Rückseiten. Die Verwendung von eigenen Blättern ist nicht erlaubt. Geben Sie zu jeder Aufgabe ein detailliertes Rechenweg an. Lösungen ohne Rechenweg können trotz richtigem Ergebnis zu Punktabzug führen.

Klausurabgabe:
In den letzten 30 Minuten der Klausur ist eine vorzeitige Abgabe der Klausur nicht möglich. Am Ende der Klausur bleiben Sie bitte sitzen. Alle Aufgaben- und Lösungsblätter sowie dieses Deckblatt sind in den ausgehändigten Umschläge abzugeben. Diese werden von der Aufsicht eingesammelt.

		Seite	≈ Pkt. [%]	Punkte
Aufgabe 1:	Rechnerarchitekturen	2	14	**17**
Aufgabe 2:	Software Engineering	6	10	**12**
Aufgabe 3:	Programmiersprachen	8	13	**15**
Aufgabe 4:	Objektorientierung in C++	11	13	**16**
Aufgabe 5:	Datenstrukturen	16	11	**13**
Aufgabe 6:	Algorithmenanalyse	18	13	**16**
Aufgabe 7:	Hardwarenahe Programmierung	22	9	**11**
Aufgabe 8:	Projektmanagement	25	13	**15**
				∑ **115**

17

Aufgabe 1: Rechnerarchitekturen

Aufgabe 1.1: Verständnisfragen

A) Was versteht man unter den Begriffen Pipelining und Superskalarität im Kontext Performanzsteigerung eines Prozessors? Erklären Sie.

2

Pipelining: Zeitliche Überlappung der Befehlsausführungsphasen resultiert in einer höheren Taktrate und einem höheren Durchsatz.
Superskalar: Wie Pipelining, zusätzlich mehrere Ausführungseinheiten und deren dynamische Befehlszuweisung zur parallelen Ausführung mehrerer Befehle – im Idealfall Vervielfachung der Leistung.

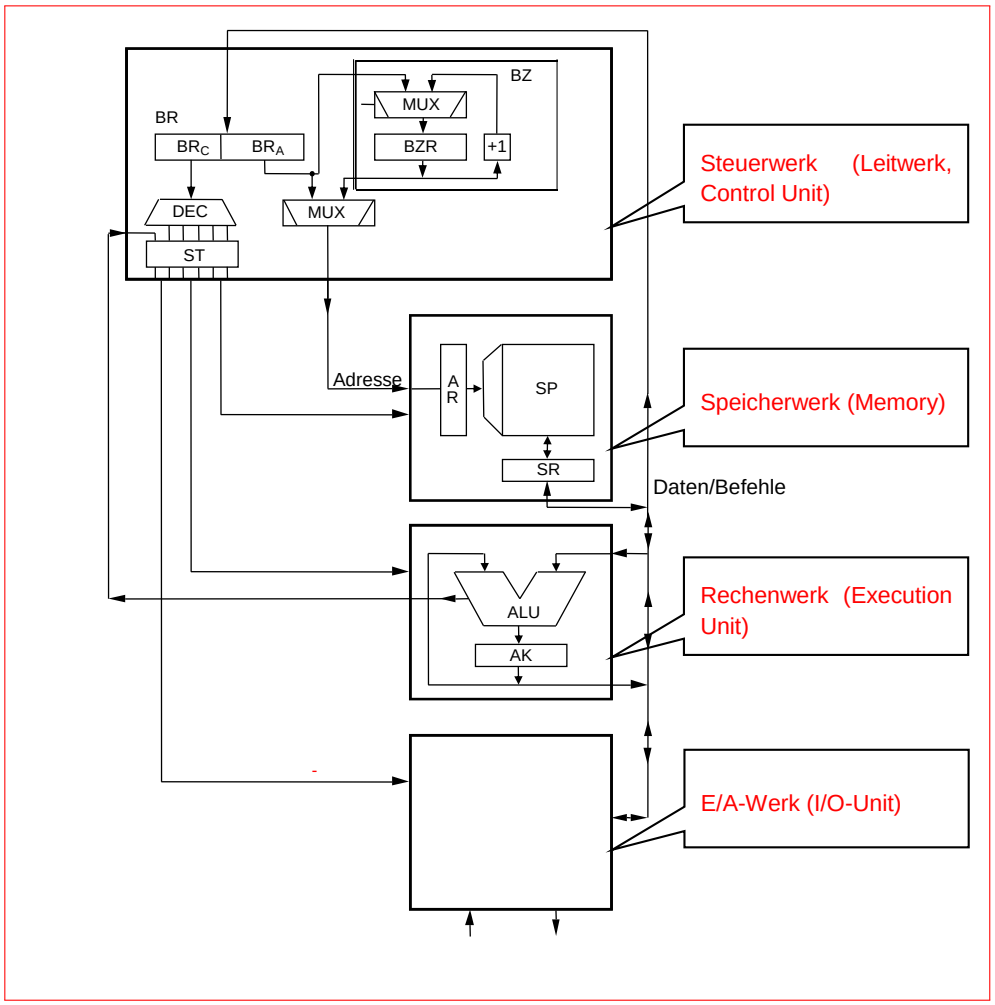
B) Erklären Sie den Unterschied zwischen der *internen* und *externen* Architektur eines Prozessors.

1

Intern: Definiert die interne Hardware-Architektur des Rechners.
Extern: Repräsentiert die Sicht des Programmierers auf den Rechner, den Maschinenbefehlssatz bzw. Instruction Set Architecture (ISA).

C) Gegeben ist das Schema eines einfachen von-Neumann Rechners. Beschriften Sie dabei die einzelnen Komponenten und beschreiben Sie jeweils deren Hauptfunktion.

4



Steuerwerk: Befehlsdekodierung, -adressierung und Ansteuerung der anderen Komponenten
Rechenwerk: Führt arithmetisch-logische Operationen, die in Maschinenbefehlen enthalten sind, aus
Speicherwerk: Arbeitsspeicher speichert Maschinenbefehle und zugehörige Daten
Ein-/Ausgabewerk: Schnittstelle zur Ansteuerung von Peripherie

D) Nennen Sie vier verschiedene Speicherformen eines Rechners und sortieren Sie diese nach ihrer Speicherkapazität (von links groß nach rechts klein). Warum werden Speicher in einem Rechner hierarchisiert? **3**

Platten-Speicher (extern), Hauptspeicher, Cache (LN-L1), Register des Prozessors
Grund Hierarchie:
1) Lokalitätsprinzip - häufig benutzte Daten/Befehle nah am Prozessor halten, um schneller auf diese zugreifen zu können im Vergleich zum großen Massenspeicher
2) Kosten/Chipfläche.

E) Erklären Sie den Unterschied zwischen "Little Endian" und "Big Endian"? **1**

Little Endian: Das niedrigstwertige Byte eines Datenworts/Befehls wird an der kleinsten Adresse gespeichert.
Big Endian: Das niedrigstwertige Byte eines Datenworts/Befehls wird an der höchsten Adresse gespeichert.

Aufgabe 1.2: Caches: Verständnisfragen

A) Nennen Sie die drei Möglichkeiten der Cacheorganisation. **1**

Direct Mapped, N-Wege assoziativ, voll-assoziativ

B) Was versteht man in Bezug auf Caches unter Verdrängungsstrategie? Nennen Sie ein Beispiel für eine Strategie und erklären Sie dessen Funktionsweise. **2**

Eine Verdrängungsstrategie ist eine Methode, die entscheidet, welcher Block im Cache ersetzt wird, falls ein neuer Block für neue Daten gebraucht wird.

Bsp: *Least Recently Used (LRU)*: Der Eintrag, auf den am längsten nicht zugegriffen wurde, wird verdrängt.
Random: Zufällig ausgewählter Block wird verdrängt.
Least frequently Used (LFU): Der am seltensten gelesene Eintrag wird verdrängt.
FIFO: Der jeweils älteste Eintrag wird verdrängt.

Aufgabe 1.3: Cacheorganisation

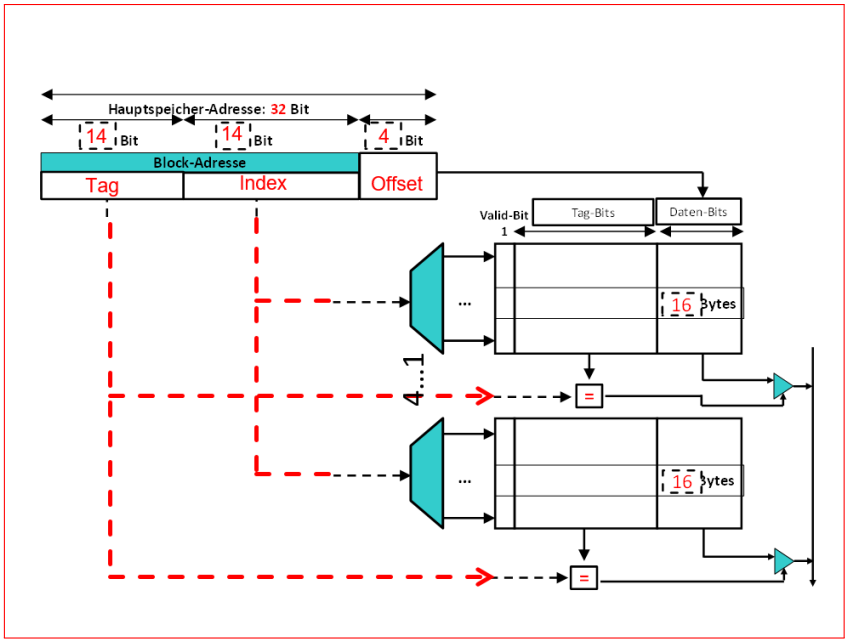
Angenommen, ein Prozessor hat einen Hauptspeicher der Größe 1 GByte, wobei jedem einzelnen Byte im Speicher eine 32 Bit lange Adresse zugeordnet sei. Zusätzlich verfügt der Prozessor über einen Cache Baustein in SRAM Technologie, der 1024 KByte (nur Daten) vom Hauptspeicher aufnehmen kann (unabhängig vom notwendigen Speicher zur Speicherung der weiteren Bits, wie z.B. Tag). Der Cache sei 4-fach assoziativ und habe eine Blockgröße von 128 Bit. Zusätzlich wird auf dem Cache für jeden Block ein Valid-Bit vorgesehen.

Annahmen:

- Setzen Sie das Cache-Modell, wie es in der Vorlesung eingeführt wurde, voraus.

A) Berechnen Sie die Bitbreite von Offset, Index und Tag für den oben beschriebenen Cache. Beschriften Sie zudem die untenstehende Grafik indem Sie alle leeren Felder mit durchgehendem Rahmen mit Text oder Symbolen und Felder mit gestricheltem Rahmen mit Werten ausfüllen. Ergänzen Sie zudem die 4 gestrichelten Pfeile.

3



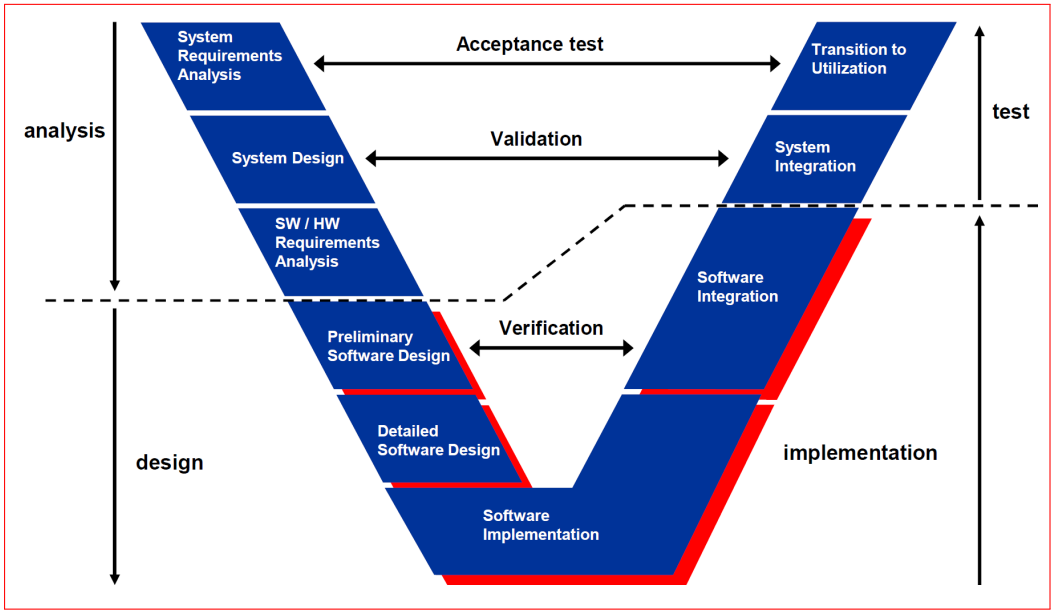
$Offset = ld(\#Bytes\,pro\,Block) = ld(128/8) = ld(16) = 4Bit$
$Index = ld(Datenmenge\,in\,Byte\,/\,Bytes\,per\,Block\,in\,Cache/Sets) = ld((1024 * 1024/16)/4) = ld(16384) = 14Bit$
$Tag = Adressbreite - Offset - Index = 32 - 4 - 14 = 14Bit$

12

Aufgabe 2: Software Engineering

Aufgabe 2.1: Entwurfsmethodiken

Gegeben ist das V-Modell:



A) Welche vier Entwurfsphasen deckt das V-Modell ab? Tragen Sie diese an den entsprechenden Pfeilen im V-Modell oben ein. **2**

B) Was versteht man im Bereich Software Engineering unter Verifikation, was unter Validierung? Tragen Sie sowohl die Verifikation als auch die Validierung im V-Modell oben ein. **2**

Verifikation: Überprüfung, ob das Produkt richtig entwickelt/gebaut wird.
Validierung: Überprüfung, ob das richtige Produkt entwickelt/gebaut wird.

Aufgabe 2.2: Programmstrukturen

A) Nennen Sie einen Vor- und einen Nachteil des Nassi-Shneidermann-Diagramms gegenüber dem Programmablaufplan. **1**

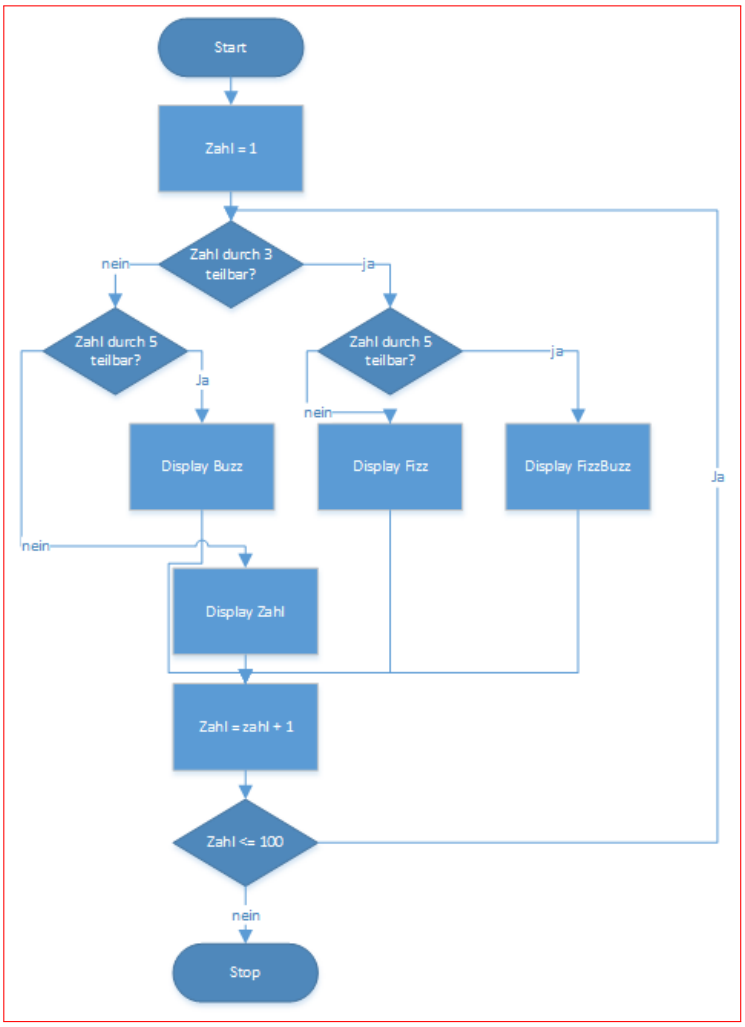
Vorteile: Repräsentiert gut das strukturierte Programmierparadigma, strukturierte Planung verhindert Sprungbefehle, Blöcke fassen Schleifen eindeutig zusammen
Nachteile: Von Hand schwer zu zeichnen, alle konkreten Aufgaben sind nur Prozesse, Blockform führt oft zum Eindruck der Überfülltheit
Hinweis: Weitere Vor- und Nachteile denkbar

B) Zeichnen Sie den Programmablaufplan für ein Programm, dass den Fizz-Buzz-Algorithmus ausführt. Dieser Algorithmus ist wie folgt definiert: **7**

- Durchlaufe alle natürlichen Zahlen von 1 bis 100
- Ist die Zahl durch 3 teilbar, soll "Fizz" ausgegeben werden
- Ist die Zahl durch 5 teilbar, soll "Buzz" ausgegeben werden
- Ist die Zahl durch 15 teilbar, soll "FizzBuzz" ausgegeben werden
- Ist keiner dieser Fälle zutreffend, soll die Zahl selbst ausgegeben werden

Verwenden Sie in ihrem Diagramm die Zählervariable "zahl". Die Ausgabe soll über einen Block "Display" erfolgen, z.B. "Display 'Fizz'" oder "Display zahl". Beispielhaft sei hier die Ausgabe des Programms für die ersten 15 Zahlen gegeben:

1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz



15

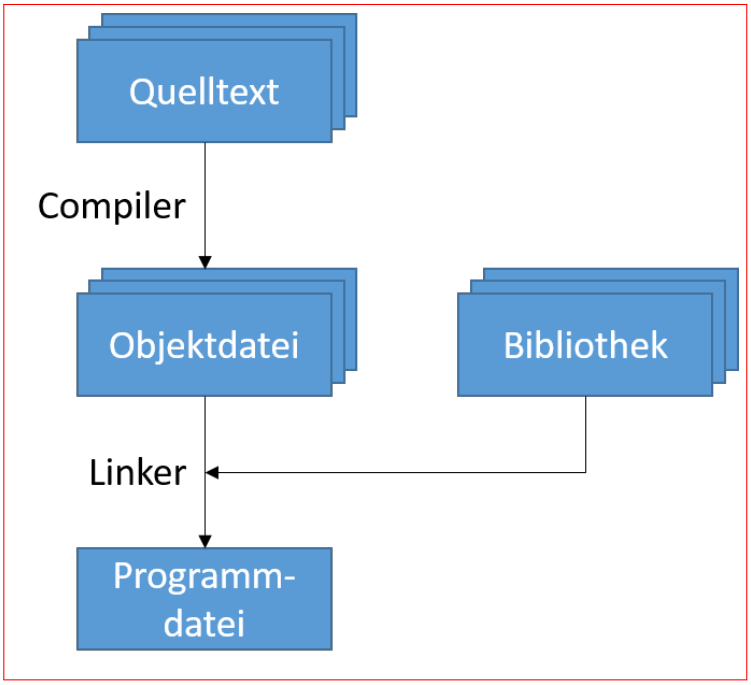
Aufgabe 3: Programmiersprachen

Aufgabe 3.1: Begrifferklärung

5

A) Erklären Sie kurz folgende sechs Begriffe im Kontext von Programmiersprachen und skizzieren Sie in einem Bild die Interaktion zwischen diesen sechs Begriffen:
Bibliothek, Compiler, Linker, Objektdatei, Programmdatei, Quelltext

- **Bibliothek**: Stellt Funktionalität zur Verfügung, die dem Programmierer das Erstellen von Software erleichtert, ohne jedes mal das Rad neu zu erfinden
- **Compiler**: Übersetzt Programmiersprache in Maschinencode oder Assembler. Erzeugt aus einzelnen Programmquellen Objektdateien
- **Linker**: Verknüpft Objektdateien. Bettet aus Bibliotheken verwendete Funktionen ein. Dynamic Binding: Verweis auf ladbare Bibliotheken. Bildet ausführbare Datei
- **Objektdatei**: Ergebnis des Compilers
- **Programmdatei**: Ausführbare Datei. Enthält Befehle, die vom Computer ausgeführt werden können; z.B. exe
- **Quelltext**: Code



B) Erklären Sie *Bitweises UND* und *Logisches UND* und geben sie jeweils die dazugehörigen Operatoren in C++ an.

3

Bitweise UND: `&`
Zwei Operanden werden bitweise *UND*-verknüpft. Die Verknüpfung darf nur für Integer-Operanden verwendet werden. Das Ergebnis enthält an jeder Stelle eine '1', wenn beide Operanden an dieser Position eine '1' haben, sonst '0'.

Logisches UND: `&&`
Ergebnis Boolscher Ausdruck. Ergebnis des Ausdruckes ist 1/true wenn beide Operanden ungleich 0/false sind, anderenfalls ist das Ergebnis 0/false. (Im Unterschied zu Bitweise UND wird der Ausdruck streng von links nach rechts ausgewertet. Wenn der erste Operand bereits 0 ergibt, wird der zweite Operand nicht mehr ausgewertet, und der Ausdruck liefert in jedem Fall den Wert 0. Nur wenn der erste Operand 1 ergibt, wird der zweite Operand ausgewertet.)

Au

A) lgende Aussage gilt

-
-

7

Schr = 1000 findet. Das sole ausgeben.

```
#in
    int a, b, c;
    for (a = 0; a < 998; a++) {
        for (b = a + 1; b < 999; b++) {
            for (c = b + 1; c < 1000; c++) {
                if ((a*a + b*b == c*c) && (a + b + c == 1000)) {
                    cout << "Found: a=" << a << ", b=" << b << ", c
                        =" << c << endl;
                    return;
                }
            }
        }
    }
}

//Alternative 1:
    int a, b, c;
    for (c = 2; c < 1000; c++) {
        for (b = 1; b < c; b++) {
            for (a = 0; a < b; a++) {
                if ((a*a + b*b == c*c) && (a + b + c == 1000)) {
                    cout << "Found: a=" << a << ", b=" << b << ", c
                        =" << c << endl;
                    return;
                }
            }
        }
    }
}

//Alternative 2:
    int a, b, c;
    for (a = 0; a < 333; a++) {
        for (b = a + 1; b < 500; b++) {
            c = 1000 - a - b;
            if ((a*a + b*b == c*c) && (a + b + c == 1000)) {
                cout << "Found: a=" << a << ", b=" << b << ", c="
                    << c << endl;
                return;
            }
```

```
        }
    }
}

//Weitere Alternativen moeglich!
```

16

Aufgabe 4: Objektorientierung in C++

Aufgabe 4.1: Grundlagen objektorientierter Programmierung

A) Was versteht man unter einem Objekt bei objektorientierter Programmierung?

1

Ein Objekt (auch Instanz genannt) bezeichnet in der objektorientierten Programmierung (OOP) ein Exemplar eines bestimmten Datentyps oder einer bestimmten Klasse (auch „Objekttyp“ genannt). Objekte sind konkrete Ausprägungen („Instanzen“) eines Objekttyps. Klasse ist „Fabrik“ für Objekte.

B) Erklären Sie die Prinzipien Kapselung und Wiederverwendbarkeit in der OOP.

2

Kapselung: Objekt definiert Zugriffsmöglichkeit auf seine Daten und schützt diese so. Bei der Definition einer Klasse wird festgelegt, welche Elemente der Klasse vor einem Zugriff von außen geschützt werden sollen (private) und welche Elemente öffentlich verfügbar sein sollen (public). Ein Objekt kapselt seine Daten und Methoden von der Außenwelt ab und verwaltet sich mit Hilfe seiner Methoden selbst.
Wiederverwendbarkeit: Nur einmal definieren wie ein Objekt funktioniert, und diese Definition (Klasse) immer wieder verwenden. Es können zudem immer neue (verfeinerte) Klassen erstellt werden (Vererbung).

Aufgabe 4.2: Fehlersuche

```
1  #include <string>
2
3  using namespace std;
4
5  class Node {
6  private:
7      string type;
8      int id;
9  protected:
10     void getId() { return id; }
11     void setId(int nodeId) { id = nodeId; }
12 };
13
14 class TempSensor : public Node {
15 private:
16     const string sensorName;
17     int tempValue;
18 public:
19     TempSensor(string name) { sensorName = name; }
20     int getTempValue() { return tempValue; }
21     void setTempValue(int temp) { tempValue = temp; }
22     void setType(string nodeType) { type = nodeType; }
23 };
24
25 int main() {
26     TempSensor sensor1;
27     sensor1.setType("Temp");
28     sensor1.setId(123);
29     return 0;
30 }
```

A) Benennen Sie die fünf Fehler im oben stehenden C++-Code und geben Sie eine mögliche Fehlerbehebung an. **5**

1) Zeile 10: Node::getId() gibt die "id" zurück, hat aber keinen Rückgabewert (void) –> Rückgabwert int.
2) Zeile 19: TempSensor::TempSensor() kann "sensorName" nicht setzen, da dieser const ist –> const entfernen.
3) Zeile 22: TempSensor::setType() kann "type" nicht setzen, da dieser in der Basisklasse private deklariert ist –> TempSensor::setType() als public Methode von Node deklarieren, oder Node::type protected oder public deklarieren.
4) Zeile 26: TempSensor hat keinen Default-Konstruktor –> Sensorname beim Erzeugen des Objekts "sensor1" als Parameter übergeben.
5) Zeile 28: sensor1.setId() nicht möglich, da Node::setId() protected deklariert ist –> Node::setId() public deklarieren.

Aufgabe 4.3: Device-Verwaltung

Gegeben ist folgender C++-Code:

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  class Device {
7  private:
8      string name;
9      int memory; //GB
10 public:
11     void setName(string name) { this->name = name; }
12     string getName(void) { return name; }
13     void setMemory(int memory) { this->memory = memory; }
14     int getMemory(void) { return memory; }
15     virtual string getTyp(void) { return "Device"; }
16     virtual void print(void) {
17         cout << "Name: " << name << endl;
18         cout << "Memory: " << memory << endl;
19     };
20 };
21
22 class Smartwatch : public Device {
23 private:
24     int heartRate; //bpm
25 public:
26     Smartwatch(string name, int memory, int heartRate) {
27         setName(name);
28         setMemory(memory);
29         this->heartRate = heartRate;
30     }
31     virtual string getTyp(void) { return "Smartwatch"; }
32     virtual void print(void);
33 };
34
35 class Smartphone : public Device {
36 private:
37     int memoryExtension; //GB
38 public:
39     Smartphone(string name, int memory, int memoryExtension) {
40         setName(name);
41         setMemory(memory);
42         this->memoryExtension = memoryExtension;
43     }
44     void insertSdCard(int capacity) {
45         memoryExtension = capacity;
46     }
47     virtual string getTyp(void) { return "Smartphone"; }
48     virtual void print(void);
49 };
```

3

A) d Smartphone, die Methodendef- init abe redundant son int() der Ba- sisk

```
void Smartwatch::print(void) {
    cout << "Smartwatch" << endl;
    Device::print();
    cout << "HearRate: " << heartRate << endl;
}

void Smartphone::print(void) {
    cout << "Smartphone" << endl;
    Device::print();
    cout << "Memory Extension: " << memoryExtension << endl;
}
```

In der `main()`-Funktion wird genau ein Array angelegt, um drei Devices zu verwalten. Zu Beginn sollen die folgenden drei Devices der Verwaltung hinzugefügt werden:

- Smartwatch - Name: "Watch1", Speicher: 4GB, HeartRate: 60bpm
- Smartphone - Name: "Phone1", Speicher: 16GB, Speichererweiterung: 0
- Smartphone - Name: "Phone2", Speicher: 32GB, Speichererweiterung: 0

Im nächsten Schritt werden alle verwalteten Devices überprüft. Sollte es sich dabei um ein Sma baut. Dies soll entsp Dan e ausgegeben. Bevo

```
    deviceList[0] = new Smartwatch("Watch1", 4, 60);
    deviceList[1] = new Smartphone("Phone1", 16, 0);
    deviceList[2] = new Smartphone("Phone2", 32, 0);

    for (int i = 0; i < 3; i++) {
        if (deviceList[i]->getTyp() == "Smartphone") {
            ((Smartphone*)(deviceList[i]))->insertSdCard(32);
        }
        deviceList[i]->print();
        delete deviceList[i];
    }
    return 0;
}
```

B) bung.

5

int

Aufgabe 5: Datenstrukturen

13

A) Nennen Sie vier Operationen, die auf Datenstrukturen ausgeführt werden können. 2

Search(S,k), Insert(S,x), Delete(S,x), Minimum(S), Maximum(S), Successor(S,x), Predecessor(S,x), Size(S)
Weitere denkbar.

B) Nennen Sie je eine Möglichkeit um mit STL eine FIFO- (First-In-First-Out) und eine LIFO-Datenstruktur (Last-In-First-Out) zu realisieren. 2

FIFO = queue; LIFO = stack

C) Gegeben seien Vektoren $\vec{a} = (\frac{1}{2}, 4, -\frac{1}{3}, 1)$ und $\vec{b} = (-2, \frac{3}{7}, 1, 9)$. Schreiben Sie in der Programmiersprache C++ einen Algorithmus der das Skalarprodukt $\vec{a} \cdot \vec{b}$ berechnet und auf der Konsole ausgibt. Verwenden Sie für $\vec{a}$ und $\vec{b}$ Arrays als Variablentypen. 3

Hinweis: $\vec{a} \cdot \vec{b} = \sum_{n=0}^{N} a_n \cdot b_n$

```
#in
usi

int
    double a[4] = {(double)1/2, 4, (double)-1/3, 1};
    double b[4] = {-2, (double)3/7, 1, 9};
    int i;
    double result = 0;
    for (i = 0; i < 4; i++) {
        result = a[i] * b[i] + result;
    }
    cout << "Ergebnis = " << result;
    return 0;
}
```

Im Folgenden soll davon ausgegangen werden, dass eine doppelt verkettete Liste mit folgender Element-Struktur vorhanden ist.

```
1  struct node {
2      struct node *left;
3      struct node *right;
4      int data;
5  } Node;
```

D) Schreiben Sie einen Pseudo-Code für die angegebene Funktion um **vor** dem aktuellen Element (`list`) dauerhaft ein neues Element einzufügen. Der Übergabewert `data` soll dabei in der Liste abgespeichert werden. Der Rückgabewert ist das neue Element der Liste. 4

```
struct node * insert_left(struct node *list, int data)
```

(1) Speicherplatz für Struktur node allokieren
(2) Datenwert in data zuweisen
(3) Zeiger right von neuem Element auf list setzen
(4) Zeiger left von neuem Element auf list->left setzen
(5) Zeiger list->left->right auf neues Element setzen
(6) Zeiger list->left auf neues Element setzten
(7) Rückgabe des neuen Elements

Für das Löschen eines Elements ist folgender Code gegeben:

```
1  struct node * delete_node(struct node *list){
2      list->right->left = list->left;
3      list->left->right = list->right;
4      return list->left;
5  }
```

E) Welche zwei Probleme treten beim Ausführen obiger `delete_node` Funktion auf? Wie können die Probleme behoben werden? **2**

Der Speicher des zu löschenden Elements wurde nicht freigegeben und ist jetzt nicht mehr zugreifbar (es gibt keinen Pointer mehr auf den Speicher) –> Memory Leak. Abhilfe schafft die Freigabe des Speichers für das zu löschende Element mit `delete`.
Wenn das Element das erste bzw. letzte in der Liste ist, existiert entweder kein `left` oder kein `right`. Abhilfe schafft eine Abfrage vor dem Setzen der Pointer.

16

Aufgabe 6: Algorithmenanalyse

Aufgabe 6.1: Binäre Suche

2

A) Gegeben sei die folgende, bereits sortierte, Liste.

Führen Sie den Algorithmus der binären Suche nach dem Wert "34" aus und geben Sie alle dafür nötigen Schritte und Vergleiche an. Sollte sich eine nicht natürliche Zahl für den Index des Pivotelements ergeben, so runden Sie auf die nächstgelegene natürliche Zahl ab.

2	3	6	11	18	32	34	45

$34 = 11$? Nein $34 < 11$? Nein
$34 = 32$? Nein $34 < 32$? Nein
$34 = 34$? Ja, Ergebnis erreicht

1

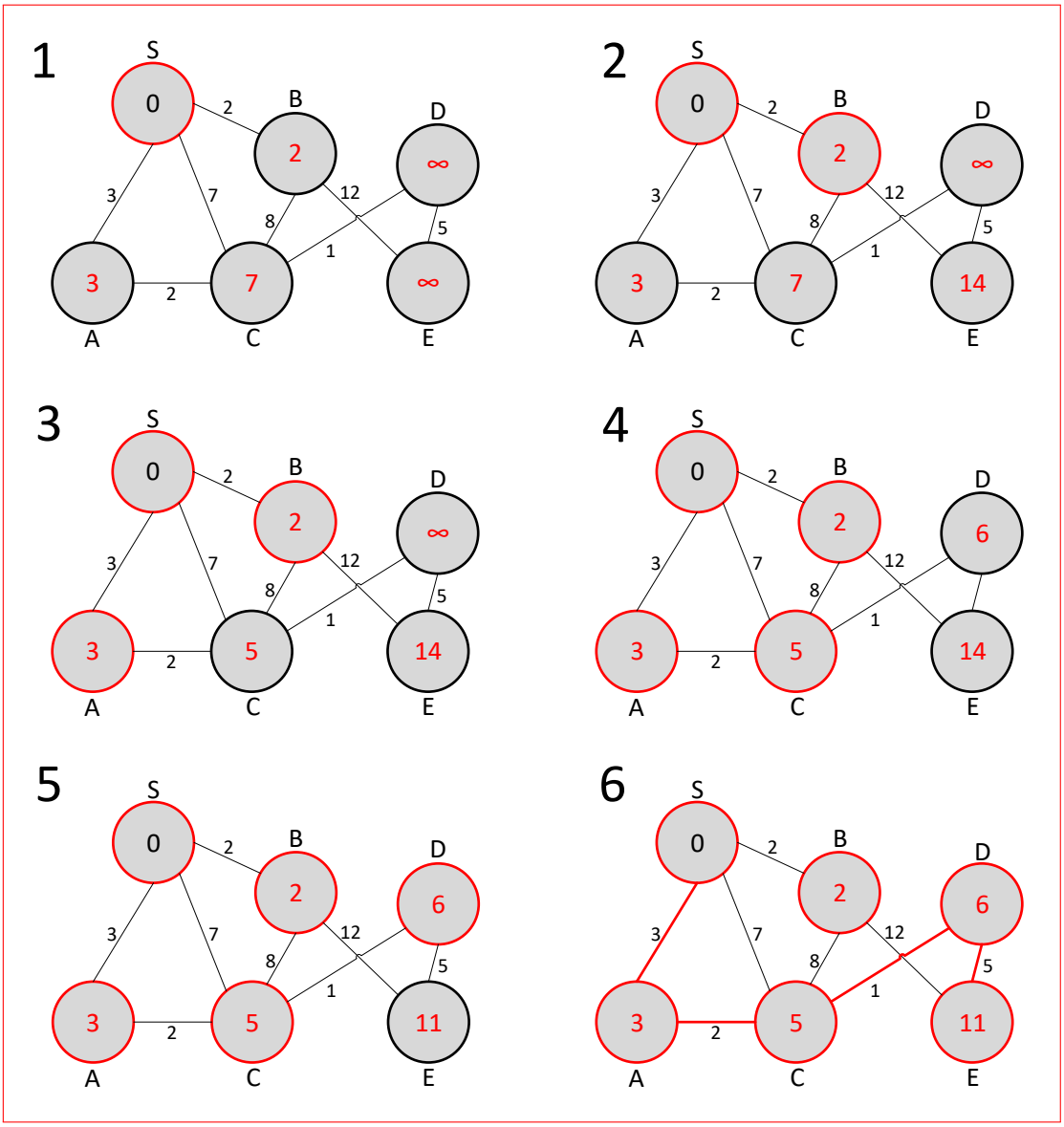
B) Welche worst-case Laufzeit hat die binäre Suche?

$O(ld(n))$

Aufgabe 6.2: Dijkstra Algorithmus

5

A) Führen Sie den Dijkstra-Algorithmus auf dem folgenden Graphen aus. Nutzen Sie dabei die Knoten des vorliegenden Graphen, um die Kosten für deren erreichen einzutragen, wie sie dem Algorithmus entsprechend in den einzelnen Schritten ermittelt werden. Geben Sie zusätzlich die Reihenfolge der permanent markierten Knoten an. Verwenden Sie "S" als Startpunkt und markieren Sie abschließend die kürzeste Route zwischen dem Startpunkt "S" und dem Zielpunkt "E".



Reihenfolge der permanent markierten Knoten:

S - B - A - C - D - E

Kürzester Pfad von S nach E:

S - A - C - D - E

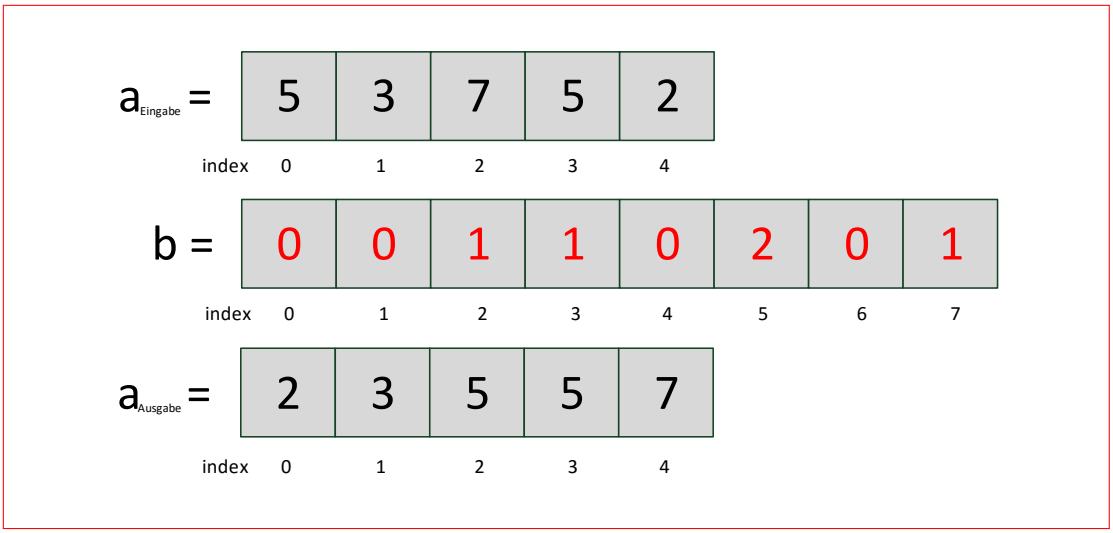
Aufgabe 6.3: Array-index basiertes Sortieren

Gegeben sei der nachfolgende Algorithmus in C++-Code, welcher den Index eines Arrays ausnutzt, um eine Menge von Zahlen zu sortieren. Hierbei wird neben dem zu sortierenden Eingangsarray `a` ein weiteres Hilfsarray `b` verwendet. In einem ersten Schritt werden die zu sortierenden Werte als Index für das Hilfsarray benutzt, bevor in einem zweiten Schritt die sortierte Liste erzeugt wird. Gehen Sie bei dem vorliegenden Code davon aus, dass alle Variablen korrekt initialisiert sind und der Code korrekt ausgeführt wird. Beachten Sie, dass die Eingabemenge auch Duplikate enthalten kann!

```
1  char a[n];
2  char b[m]; //typischer Wert von "m" bei Benutzung des Typ char: 256
3
4  //Schritt 1
5  int i;
6  for(i = 0;i<n;i++) {
7      b[a[i]]++;
8  }
9
10 //Schritt 2
11 i = -1;
12 int j;
13 for(j = 0;j<m;j++) {
14     while(b[j]>0) {
15         b[j]--;
16         a[++i] = j;
17     }
18 }
```

A) Im Folgenden sehen Sie ein Eingabe-Array `a[]`, ein Hilfsarray `b[]` sowie das Ergebnis des Algorithmus, welches als Ausgabe wieder im modifizierten Array `a[]` steht. Führen Sie nun Schritt 1 des Algorithmus auf den im Folgenden angegebenen Arrays aus und tragen Sie hierzu die fehlenden Werte im Array `b[]` ein. Gehen Sie dabei davon aus, dass das Hilfsarray `b[]` zu Beginn auf allen Feldern mit 0 initialisiert ist.

2



B) Wozu wird in dem angegebenen Algorithmus eine `while`-Schleife im zweiten Schritt eingesetzt? Begründen Sie Ihre Antwort indem Sie die Funktion der Schleife kurz erklären. (Hinweis: Was würde passieren wenn man dies in eine if-Abfrage ändert und Zeile 15 streicht?) **2**

Falls es Duplikate unter den Eingangswerten gab müssen diese im zweiten Schritt wiederhergestellt werden. Dies wird durch die while-Schleife realisiert, indem sie die Vorkommen, die im ersten Schritt hochgezählt wurden, hier wieder herunterzählt und in die Ergebnis-Liste einfügt. Mit einer if-Abfrage würden Duplikate "gelöscht", d.h. bei der Wiederherstellung der sortierten Eingangsliste würde jeder Wert nur einmal eingefügt werden.

C) Geben Sie die Komplexität des Algorithmus in Abhängigkeit der beiden Eingangsgrößen `m` und `n` an. **1**
(Gehen Sie in diesem speziellen Fall davon aus, dass Duplikate nicht vorkommen können)

O(m+n)

D) Ist die Laufzeit des Algorithmus abhängig von den Werten in der zu sortierenden Liste? (Hinweis: Macht es also z.B. einen Unterschied, ob die Liste bereits sortiert ist?) Begründen Sie Ihre Antwort. **1**
(Gehen Sie auch in diesem speziellen Fall davon aus, dass Duplikate nicht vorkommen können.)

Nein, die Laufzeit ist immer n+m. Unabhängig vom Zustand der Eingabeliste muss jedes Element einmal untersucht und in das Hilfsarray geschrieben werden. Anschließend muss das Hilfsarray einmal komplett durchlaufen werden um die sortierte Liste zu erstellen.

E) Nennen Sie zwei Nachteile des vorgestellten Algorithmus. **2**

Es muss zusätzlicher Speicherplatz für das Hilfsarray bereitgestellt werden. Die Laufzeit hängt auch von der Größe des Hilfsarrays ab. Die Wertemenge muss im voraus bekannt sein, da sonst das Hilfsarray eventuell zu klein gewählt ist.
Bei vernünftiger und präziser Begründung sind auch andere Lösungen möglich.

Aufgabe 7: Hardwarenahe Programmierung

11

Aufgabe 7.1: Allgemeine Fragen

A) Welche Funktion hat der Zusatz `volatile` bei der Deklaration einer Variablen? 1

Mit dem Typqualifikator `volatile` wird in C und C++ spezifiziert, dass bei jedem Variablenzugriff der Wert direkt aus dem Speicher gelesen bzw. in den Speicher geschrieben wird (eine Zwischenspeicherung in Prozessorregistern etc. wird dadurch unterbunden). Dies ist z.B. bei memory-mapped Peripherieregistern notwendig, wenn die Hardware Registerwerte verändern kann (in Prozessorregistern zwischengespeicherte Werte würden dadurch ggf. nicht mehr den aktuellen Wert repräsentieren).

B) Kann durch den Zusatz `volatile` ein Cache-Miss, d.h. einen fehlerhaften Zugriff im Cache, erzeugt werden? Begründen Sie Ihre Antwort! 1

Mehrere Optionen:
Wenn erzeugt = "erzwungen": Nein, Daten können im Cache liegen.
Wenn erzeugt = "möglich": Ja, volatile zwingt den Prozessor zwar zu einem Re-Read der Variablen, hat aber keinen Einfluss auf den Speicherort. Falls die Variable kürzlich verwendet wurde und im Cache steht, wird ggf. aus dem Cache gelesen (Konsistenz zwischen Cache und Speicher ist immer sicherzustellen - genaue Realisierung ist abhängig von der Hardware).

C) Vervollständigen Sie die folgende Funktion mit der fehlenden Zeile Code. Achten Sie auf entsprechende Casts. 1

```
int add(unsigned int wert) {
    void *void_ptr = (void*) 0x8f000000; //zeigt auf 32 bit Register
    // zum Inhalt von void_ptr wird der Inhalt aus 'wert' addiert




    return 0;
}
```

```
    *((unsigned int*)void_ptr) += wert;
    //oder
    *((unsigned int*)void_ptr) = *((unsigned int*)void_ptr) + wert;
```

Aufgabe 7.2: Register Programmierung

Ein Analog Komparator ist ein Peripheriegerät, dass zwei Analogspannungen vergleicht und das Ergebnis als logisches Vergleichsergebnis zur Verfügung stellt.

Der interne Aufbau des Komparators ist in Abbildung 7.1 dargestellt. Hierbei kann über *Analog Comparator Reference Voltage Control (ACREFCTL)* der negative Eingang des Komparators ausgewählt werden, am positiven Eingang liegt immer die externe Spannung *ve+* an. Durch schreiben einer *"1"* in *Analog Comparator Start (ACSTRT)* wird der Komparator gestartet. Nach einem erfolgten Vergleich wird das *ACSTRT* von der Hardware gelöscht und das Ergebnis ist in *Analog Comparator Output Value Register (ACOVR)* verfügbar. Um das Ergebnis des Komparators zu invertieren kann eine *"1"* in *Analog Comparator Inverter (ACINV)* geschrieben werden.

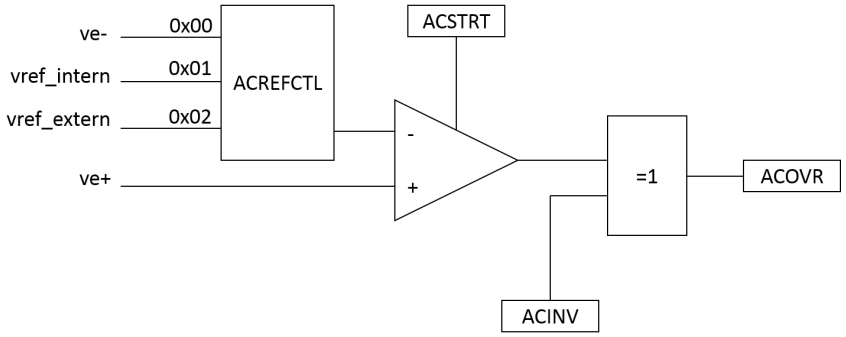


Abbildung 7.1: interner Aufbau des Komparators

Für die funktionale Beschreibung des Komparators gilt:
VIN- <= VIN+, VOUT = 1
VIN- > VIN+, VOUT = 0

Für den Analog Komparator gilt die Basisadresse *0xFFFF8000* und die in Tabelle 7.1 dargestellten Offsets.

Registername	**Offset in Bytes**
Analog Comparator Reference Voltage Control (ACREFCTL)	0x04
Analog Comparator Start (ACSTRT)	0x08
Analog Comparator Inverter (ACINV)	0x10
Analog Comparator Output Value Register (ACOVR)	0x14

Tabelle 7.1: Offsets der Register

Hinweis: Bei allen Registern handelt es sich um 32-bit Register.

A) Initialisieren Sie den Analog Komparator zum Vergleichen einer Eingangsspannung mit der internen Referenzspannung. Der Ausgangswert in *ACOVR* soll *"1"* ergeben, sobald die interne Referenzspannung größer als die Eingangsspannung ist.

4

```
voi
    void *BASISADRESSE;
    unsigned int ACREFCTL, ACINV;
    ACREFCTL = 0x04;
    ACINV = 0x10;
    BASISADRESSE = (void*)0xFFFF8000;
    *((volatile unsigned int*)(BASISADRESSE + ACREFCTL)) = 0x01;
    *((volatile unsigned int*)(BASISADRESSE + ACINV)) = 0x01;

    //Alternativ:
    *(volatile unsigned int*)(0xFFFF8000 + 0x04) = 0x01;
    *(volatile unsigned int*)(0xFFFF8000 + 0x10) = 0x01;
}
```

B) Geben Sie in der nachfolgenden Funktion den Vergleichswert der Eingangsspannungen zurück. Nehmen Sie an, dass `AC_INIT()` bereits ausgeführt wurde. Geben Sie nur den aktuellsten Wert zurück.

4

```
boo
    void *BASISADRESSE;
    unsigned int ACSTRT, ACOVR;
    ACSTRT = 0x08;
    ACOVR = 0x14;
    BASISADRESSE = (void*)0xFFFF8000;
    *((volatile unsigned int*)(BASISADRESSE + ACSTRT)) = 0x01;
    while(*((volatile unsigned int*)(BASISADRESSE + ACSTRT)) ==
        0x01);
    return *((volatile bool*)(BASISADRESSE + ACOVR));

    //Alternativ:
    *(volatile unsigned int*)(0xFFFF8000 + 0x08) = 0x01;
    while(*(volatile unsigned int*)(0xFFFF8000 + 0x08) == 0x01);
    return *((volatile bool*)(0xFFFF8000 + 0x14);
}
```

Aufgabe 8: Projektmanagement

15

Aufgabe 8.1: Verständnisfragen

A) Im Laufe eines Projekts werden Lasten- und Pflichtenheft erstellt. Wer erstellt Lastenheft und wer erstellt Pflichtenheft? Erklären Sie den inhaltlichen Unterschied.

3

Lastenheft: Wird vom Auftraggeber erstellt. Enthält die Anforderungen aus Sicht des Auftraggebers/Kunden.
Pflichtenheft: Wird vom Auftragnehmer auf Basis des Lastenhefts erstellt. Darin enthalten ist eine Detaillierung der Kundenanforderungen aus Sicht des Auftragnehmers.

B) Was ist ein Projektstrukturplan (PSP)?

1

Der Projektstrukturplan gliedert das Projekt in plan- und kontrollierbare Elemente. Im Rahmen dessen, wird das Projekt in Teilaufgaben und Arbeitspakete unterteilt. Er bildet die Grundlage für die Termin-, Ablauf-, Ressourcen- und Kostenplanung.

C) Was ist ein Projektablaufplan (PAP)? In welcher Art von Diagramm wird ein Projektablaufplan typischerweise dargestellt?

2

Der Projektablaufplan definiert hauptsächlich den zeitlichen Ablauf eines Projekts - auf Basis der im PSP definierten Arbeitspakete.
Normalerweise wird der Projektablaufplan in einem Gantt-Chart/Gantt-Diagramm dargestellt.

D) Warum ist es empfehlenswert zu Beginn eines Projekts Programmierrichtlinien festzulegen - speziell wenn mehrere Softwareentwickler involviert sind (min. zwei Punkte)? **1**

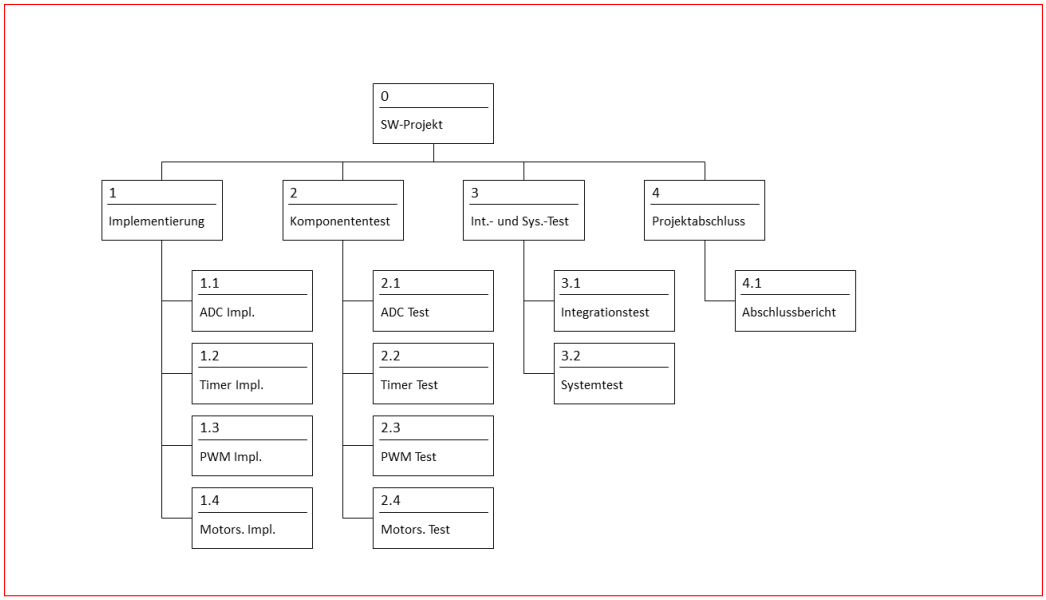
Programmierrichtlinien tragen maßgeblich zur besseren Verständlichkeit und Wartbarkeit von Quellcode bei und können potentielle Fehlerquellen minimieren. Außerdem dienen Sie der Einheitlichkeit des erstellten Quellcodes.

Aufgabe 8.2: Beispielprojekt

In einem spannenden Projekt bearbeitet eine Gruppe von 4 Studenten ein Softwareprojekt. Zu Beginn der Projektplanung wurden folgende Arbeitspakete identifiziert - die Zahlen in den Klammern geben die geschätzte Dauer in Wochen an:

- ADC Implementierung (1)
- Timer Implementierung (1)
- PWM Implementierung (1)
- Motorsteuerung Implementierung (2)
- ADC Test (1)
- Timer Test (1)
- PWM Test (1)
- Motorsteuerung Test (2)
- Integrationstest (2)
- Systemtest (1)
- Abschlussbericht (2)

A) Zeichnen Sie als nächsten Schritt der Projektplanung einen Projektstrukturplan. Verwenden Sie dabei genau eine Teilprojektebene! **3**



3

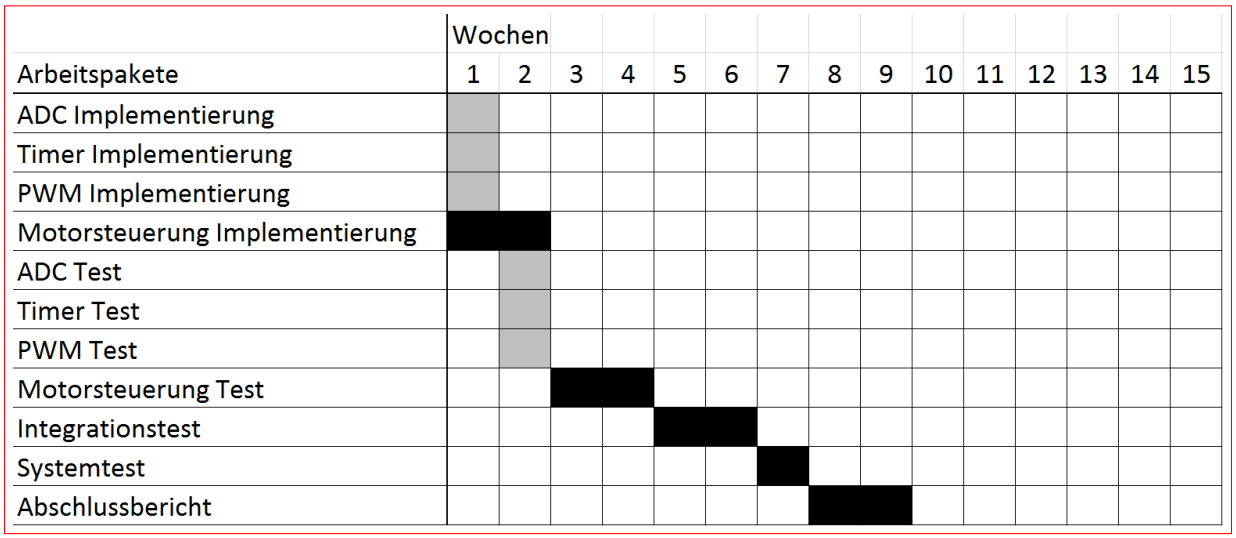
B) Zeichnen Sie nun den Projektablaufplan unter Berücksichtigung der angegebenen Dauer der Arbeitspakete. Beachten Sie dabei die logische Reihenfolge der Arbeitspakete und zusätzlich die folgenden Randbedingungen:

- Ein Student kann zu einem Zeitpunkt immer nur ein Arbeitspaket bearbeiten.
- Die Module ADC, Timer, PWM und Motorsteuerung haben in diesem Projekt keine Abhängigkeiten untereinander.
- Die Gesamtdauer des Projekts soll möglichst kurz gehalten werden.
- Zur Minimierung des Projektrisikos, sollen Arbeitspakete immer zum frühest möglichen Zeitpunkt starten.

	Wochen														
Arbeitspakete	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADC Implementierung															
Timer Implementierung															
PWM Implementierung															
Motorsteuerung Implementierung															
ADC Test															
Timer Test															
PWM Test															
Motorsteuerung Test															
Integrationstest															
Systemtest															
Abschlussbericht															

C) Erklären Sie den Begriff "kritischer Pfad" im Kontext der Projektplanung. Markieren Sie den kritischen Pfad in Ihrem Projektablaufplan.

2



Der kritische Pfad bezeichnet die Folge von Arbeitspaketen, welche die Gesamtdauer des Projekts bestimmt. Die Verzögerung eines Arbeitspakets, das auf dem kritische Pfad liegt, führt sofort zu einer Verzögerung des Gesamtprojekts.

Zusätzliches Lösungsblatt: