

	Prof. Dr.-Ing. K. D. Müller-Glaser	
<b>Informationstechnik</b>		
SS 2012		
Institut für Technik der Informationsverarbeitung, KIT		

<b>Klausur SS 2012</b>
Fr., 28.09.2012

### Hinweise zur Klausur

#### Hilfsmittel

Zur Prüfung ist nur eine handgeschriebene 2-seitige A4 Notiz-/Formelsammlung zugelassen. Nicht erlaubt sind die Verwendung eines Mobiltelefons und jegliche Kommunikation mit anderen Personen.

#### Prüfungsdauer

Die Prüfungsdauer beträgt 120 Minuten.

#### Prüfungsunterlagen

Die Prüfungsunterlagen bestehen aus insgesamt 27 Seiten Aufgabenblättern (diesem Titelblatt, 8 Aufgabenblöcken)

### Bitte kontrollieren Sie vor der Bearbeitung der Aufgaben auf jeder Seite oben Ihren Namen und Ihre Matrikelnummer!

Falls Sie zusätzliche Blätter zur Lösung der Aufgaben benötigen, fragen Sie nach zusätzlichem Papier. Auf jedes zusätzliche Lösungsblatt ist neben dem Namen auch die Aufgabennummer mit einzutragen. Vermeiden Sie das Beschreiben der Rückseiten. Die Verwendung von eigenen Blättern ist nicht erlaubt. Am Ende der Prüfung sind die 27 Seiten Aufgaben- und Lösungsblätter und alle zusätzlichen Lösungsblätter im ausgehändigten Umschlag abzugeben. Verwenden Sie zum Bearbeiten der Aufgaben nur dokumentenechte Schreibgeräte – keinen Bleistift, keine Rotstifte!

Wie verabredet enthält die Klausur mehr Aufgaben als, bei einer richtigen Lösung, zum Erreichen einer sehr guten Note (1,0) notwendig sind (Nutzen Sie die Auswahlmöglichkeit). Bitte beachten Sie: Die mit \* gekennzeichneten Teilaufgaben können Sie nicht unabhängig von den anderen Teilaufgaben lösen.

Aufgabe	1	2	3	4	5	6	7	8	$\Sigma$
$\approx$ Gewichtung ca. [%]	12	12	15	11	12	13	12	13	100
Ergebnis [P]									



- 
- C) Nennen Sie 3 allgemeine Suchverfahren der Graphentheorie und geben Sie jeweils ihren Laufzeitaufwand im Worst-Case an.
- D) Was ist der Unterschied zwischen der Effektivität und der Effizienz von Algorithmen?
- E) Erläutern Sie die Funktionsweise der Queue (Warteschlange).
- F) Was versteht man unter der unmittelbaren Nachbarschaft von Knoten in einem Graphen?
- G) Nennen Sie 4 Merkmale guter Software.

## Aufgabe 2 C++ Verständnisfragen



- A) Schreiben Sie die Funktion `selbstswap`, die den Inhalt von zwei übergebenen Integer Variablen vertauscht. Gegeben sind der Funktionsaufruf und die Ausgaben.

```
int a = 5, b = 10;
cout << a << " " << b << endl; // Ausgabe: 5 10
selbstswap( &a, &b );
cout << a << " " << b << endl; // Ausgabe: 10 5
```

```
void selbstswap(           ) {
```

```
}
```

- B) Richtig oder falsch?

Bitte beachten Sie, dass falsche Antworten einen Punktabzug bedeuten. Negative Punkte werden allerdings nicht auf andere Teilaufgaben übertragen.

- i. Ein Zeiger ist eine veränderbare Variable, deren Inhalt die Adresse einer anderen Variablen im Speicher enthalten kann.  Falsch  Richtig
- ii. Der Operator `&&` verknüpft zwei Operanden und gibt **true** zurück, wenn beide Operanden den Wert **true** haben, sonst **false**.  Falsch  Richtig
- iii. Der Operator `==` kann dazu verwendet werden, um die Inhalte zweier Variablen zu vergleichen.  Falsch  Richtig
- iv. Die zwei Ausdrücke in der letzten Zeile sind äquivalent in C++:  

```
int var;
// ...
var = var << 2;    var = var * 2;
```

 Falsch  Richtig

- C) Gegeben sind die folgenden Funktionsaufrufe. Schreiben Sie zu den Funktionsaufrufen die passenden C++ Prototypen.

```
float result = min( 7.45, 12.12 );
```

Antwort:

```
char c = get( 'A' );
```

Antwort:

```
bool error = false;
string ergebnis = intToStr( 1234, &error );
```

Antwort:

- D) Ergänzen Sie die folgende Funktion um `if`-Anweisungen in C++. Der Funktion wird eine Arbeitszeit in Minuten als Integer übergeben. Wenn die übergebene Arbeitszeit mehr als 4 Stunden beträgt, sollen 30 Minuten vom übergebenen Wert abgezogen werden, und wenn die übergebene Arbeitszeit mehr als 6 Stunden beträgt, sollen nochmals (zusätzlich) 15 Minuten abgezogen werden. Das Ergebnis soll als **return**-Wert zurückgegeben werden.

```
int arbeitszeitRechnen( int zeit ) {
```

```
}
```

- E) Welche Werte speichern die Variablen `x` und `y`, wenn die folgende Schleife verlassen wird?

```
int x = 0;
```

```
int y = 1;
```

```
do {
```

```
    y *= 10;
```

```
    y = y + x;
```

```
    x = x + 1;
```

```
} while( y <= 1000 );
```

x:

y:

F) Welchen Wert speichert der String u nach Ausführung folgender C++ Anweisungen?

```
string u( "Kaffeesahne" );  
string v( "Tassenkuchen" );  
u.erase( 5, 6 );  
u = u + v;  
u.erase( 5, 6 );  
u = u.substr( 0, u.length() - 1 );
```

G) Was gibt das folgende C++ Programm auf dem Bildschirm aus?

```
#include <iostream>  
using namespace std;  
int main() {  
    long n[3] = {100, 200, 300};  
    long* ref = n;  
    n[1]++;  
    cout << n[1] << " " << *ref << endl;  
    ref++;  
    cout << n[1] << " " << *ref << endl;  
    *ref = *ref + 1;  
    cout << n[1] << " " << *ref << endl;  
    return 0;  
}
```

## Aufgabe 3      Objektorientierung in C++



A) Benennen Sie die vier Fehler im folgenden C++-Code:

```
class Base {
private:
    int value;

protected:
    void set_value( int x ) const {
        value = x;
    }

public:
    int get_value() const {
        return value;
    }
};

class Derived : public Base {
public:
    Derived( int x ) {
        set_value( x * 10 );
    }

    int get_value() const {
        return value * 10;
    }
};

int main() {
    Base b;
    b.set_value( 100 );

    Derived d;
    d.get_value();

    return 0;
}
```



B) Gegeben sei folgender C++-Code:

```
#include <iostream>
using namespace std;

class figure {
protected:
    int start_x;
    int start_y;

public:
    figure() {
        start_x = 0;
        start_y = 0;
    }

    figure( int x, int y ) {
        start_x = x;
        start_y = y;
    }

    void set_start( int x, int y ) {
        start_x = x;
        start_y = y;
    }

    void print_position() const {
        cout << "figure:" << endl;
        cout << "start_x=" << start_x << endl;
        cout << "start_y=" << start_y << endl;
    }

    virtual void print_area() const {
        cout << "figure area=0" << endl;
    }
};

class circle : public figure {
private:
    int radius;

public:
    circle( int x, int y, int r ) {
        set_start( x, y );
        radius = r;
    }

    void print_position() const {
        cout << "circle:" << endl;
        cout << "start_x=" << start_x << endl;
        cout << "start_y=" << start_y << endl;
    }

    virtual void print_area() const {
        cout << "circle area=" << 2 * radius * radius << " x PI"
        << endl;
    }
};
```

```
class quad : public figure {
private:
    int end_x;
    int end_y;

public:
    quad( int s_x, int s_y, int e_x, int e_y ) {
        set_start( s_x, s_y );
        end_x = e_x;
        end_y = e_y;
    }

    void print_position() const {
        cout << "quad:" << endl;
        cout << "start_x=" << start_x << endl;
        cout << "start_y=" << start_y << endl;
    }

    virtual void print_area() const {
        int a = abs( end_x - start_x );
        int b = abs( end_y - start_y );
        cout << "quad area=" << a * b << endl;
    }
};

int main() {
    figure f( 10, 20 );
    circle c( 100, 150, 50 );
    quad q( 200, 200, 300, 400 );

    f.print_position();
    f.print_area();
    c.print_area();
    q.print_area();

    figure& f1 = c;
    f1.print_position();
    f1.print_area();

    figure& f2 = q;
    f2.print_position();
    f2.print_area();

    return 0;
}
```

i. Geben Sie die Ausgabe des Programms an:



- ii. Schreiben Sie eine alternative Main-Funktion, die die figure `f` auf dem Heap anlegt. Anschließend sollen ihre Position sowie ihre Fläche mit den zugehörigen Methoden ausgegeben werden. Sorgen Sie außerdem dafür, dass der verwendete Heap-Speicher vor Beendigung der Main-Funktion wieder freigegeben wird.

```
int main() {
```

```
    return 0;  
}
```

C) Gegeben seien die folgenden C++-Klassen, deren Definition jedoch die Vererbungsbeziehungen fehlen. Tragen Sie die fehlenden Vererbungsbeziehungen in die Kästen ein, falls keine Vererbungsbeziehung notwendig ist, lassen Sie den Kasten leer:

```
class C1  {  
public:  
    C1() {  
        cout << "C1()" << endl;  
    }  
};
```

```
class C2  {  
public:  
    C2() {  
        cout << "C2()" << endl;  
    }  
};
```

```
class C3  {  
public:  
    C3() {  
        cout << "C3()" << endl;  
    }  
};
```

```
class C4  {  
private:  
    C2 c2;  
public:  
    C4() {  
        cout << "C4()" << endl;  
    }  
};
```

Bei der Ausführung folgender Main-Funktion:

```
int main() {  
    C2 c2;  
    C3 c3;  
    C4 c4;  
  
    return 0;  
}
```

erzeugt das Programm diese Ausgabe:

```
C1()  
C2()  
C1()  
C3()  
C1()  
C3()  
C1()  
C2()  
C4()
```



## Aufgabe 4 STL

Es soll eine Studentendatenbank erstellt werden, die zur internen Datenspeicherung mit einem Container der C++ Standard Template Library arbeitet. Der folgende C++ Code ist bereits gegeben:

```
#include <string>
#include <list>
#include <iostream>
using namespace std;

class Student {
private:
    string name;
    string vorname;
    int matrikelnr;

public:
    const string& getName() const { return name; }
    const string& getVorname() const { return vorname; }
    int getMatrikelnr() const { return matrikelnr; }

    void setName( const string& n ) { name = n; }
    void setVorname( const string& n ) { vorname = n; }
    void setMatrikelnr( int nr ) { matrikelnr = nr; }
};

class Studentenverwaltung {
private:
    list<Student> studentenliste;

public:
    void addStudent( const string& vorname, const string& name,
        int matr );
    void deleteStudent( list<Student>::iterator it );
    list<Student>::iterator findStudent( int matrikel );
    void printStudent( list<Student>::iterator it ) const;
};

int main() {
    Studentenverwaltung s;

    s.addStudent( "Peter", "Mueller", 1234567 );

    list<Student>::iterator it = s.findStudent( 1234567 );
    s.printStudent( it );

    s.deleteStudent( it );
    it = s.findStudent( 1234567 );
    s.printStudent( it );

    return 0;
}
```

A) Implementieren Sie die folgenden vier Methoden der Klasse Studentenverwaltung in C++.

- i. `addStudent( const string& vorname, const string& name, int matr )` soll einen Studenten mit den übergebenen Daten zu `studentenliste` hinzufügen. Es muss nicht überprüft werden, ob ein entsprechender Student bereits in der Liste existiert.

```
void Studentenverwaltung::addStudent( const string& vorname,  
    const string& name, int matr ) {
```

```
}
```

- ii. `findStudent( int matrikel )` soll einen beliebigen Studenten mit entsprechender Matrikelnummer zurückgeben. Die Methode soll `studentenliste.end()` zurückgeben, falls der gesuchte Student nicht in der Datenbank existiert.

```
list<Student>::iterator Studentenverwaltung::findStudent(  
    int matrikel ) {
```

```
}
```

- iii. `deleteStudent( list<Student>::iterator it )` soll den angegebenen Studenten aus der Liste löschen.

```
void Studentenverwaltung::deleteStudent(  
    list<Student>::iterator it ) {  
  
  
  
}
```

- iv. `printStudent( list<Student>::iterator it )` soll die Daten eines Studenten aus der Liste ausgeben und für den in `main()` angegebenen Beispielcode die folgende Ausgabe liefern:

Vorname: Peter, Name: Mueller, Matrikel-Nr.: 1234567

Student nicht gefunden

```
void Studentenverwaltung::printStudent( list<Student>::iterator it )  
    const {
```

```
}
```

- B) Geben Sie die Definition des Attributs `studentenliste` mit einem alternativen STL-Container an, der eine effizientere Implementierung der Methode `findStudent( int matrikel )` ermöglicht. Nennen Sie außerdem einen Vorteil und einen Nachteil des verwendeten Containers gegenüber dem `list`-Container.

## C) Verständnisfragen zur STL



- Nennen Sie zwei Sequenzielle Container der STL
  
- Nennen sie zwei Container Adapter der STL
  
- Nennen sie vier Assoziative Container der STL

## Aufgabe 5 Studentendatenbank (Datei- / Stream - / Stringverarbeitung)



Die Verwaltung der Studentendatenbank soll so erweitert werden, dass sie Daten mit einem anderen Programm austauschen kann. Dazu sollen die enthaltenen Daten in eine Textdatei geschrieben werden.

Die Klasse Student hat analog zu Aufgabe 4 folgenden Aufbau, sie wurde nur um das Attribut note und die entsprechenden Zugriffsmethoden erweitert:

```
class Student {
private:
    string name;
    string vorname;
    int matrikelnr;
    double note;

public:
    const string& getName() const { return name; }
    const string& getVorname() const { return vorname; }
    int getMatrikelnr() const { return matrikelnr; }
    double getNote() const { return note; }

    void setName( const string& n ) { name = n; }
    void setVorname( const string& n ) { vorname = n; }
    void setMatrikelnr( int nr ) { matrikelnr = nr; }
    void setNote( double n ) { note = n; }
};

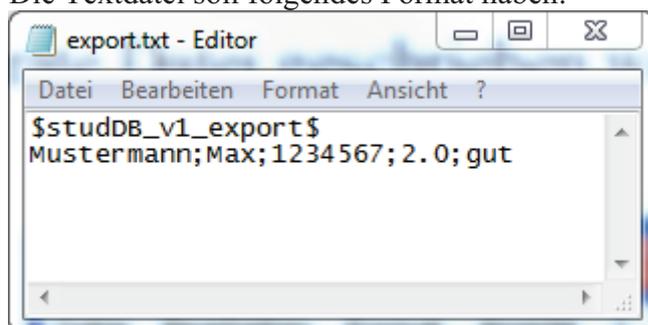
class StudDatabase {
// ...
public:
// ...

    /* Schreibt den Inhalt des aktuellen Datensatzes in die Attribute des übergebenen Objekts*/
    void read( Student* stud );

    /* Schreibt die Werte der Attribute des übergebenen Student-Objektes in die Datenbank */
    void write( Student* stud );

    void first(); // springe zum ersten Datensatz
    void next(); // springe zum nächsten Datensatz
    bool eof(); // gibt true zurück, wenn das Ende erreicht wurde; sonst false
};
```

Die Textdatei soll folgendes Format haben:





B) Schreiben Sie eine Funktion, welche alle Datensätze der Datenbank in eine Textdatei schreibt. Vergessen Sie nicht die Prüfzeile hinzuzufügen. Außerdem soll der Benutzer gefragt werden, unter welchem Dateinamen er die Daten speichern will. Wenn der angegebene Name nicht auf „.txt“ endet, soll das Programm diese Erweiterung an den eingegebenen Dateinamen anhängen. Die Funktion `benotung(double note)` aus der obigen Teilaufgabe soll benutzt werden, um den entsprechenden letzten Teil der Zeile in der Textdatei zu generieren.

```
void daten_in_datei( StudDatabase* database ) {
    Student* stud = new Student;
    const string pz = "$studDB_v1_export$";
    string dateiname;

    cout << "Bitte geben Sie einen Dateinamen ein: ";
    cin >> dateiname;
```

```
        delete stud;
    }
```

- C) Beschreiben Sie eine Methode, indem Sie ein Nassi-Shneiderman-Diagramm mit Pseudocode zeichnen. Die Methode soll eine Datei zeilenweise einlesen und die Werte in die Datenbank schreiben. Die Werte aus der Datei sollen zunächst in die Attribute eines Studenten-Objektes geschrieben werden, und dann mit der entsprechenden Methode der Datenbank hinzugefügt werden. Überprüfen Sie dabei, ob die Prüfzeile korrekt in der Datei vorhanden ist, weitere Fehlerprüfungen sind nicht erforderlich. Wenn die Datei ungültig ist, gibt die Methode nur eine entsprechende Fehlermeldung aus. Eine Instanz der Datenbank ist als Objekt mit dem Name `database` gegeben.



## Aufgabe 6 Graphentheorie



A) Gegeben ist der Graph G durch die folgende Adjazenzmatrix in Tabelle 6-1, beantworten Sie hierzu die unten stehenden Fragen.

Quelle \ Ziel	A	B	C	D	E	F	G
A	0	0	0	0	0	1	0
B	1	0	1	0	0	0	0
C	1	0	0	0	0	0	0
D	0	1	0	0	0	0	0
E	0	1	0	1	1	0	0
F	0	0	0	1	0	0	1
G	0	0	0	1	0	1	0

Tabelle 6-1: Adjazenzmatrix zu Graph G

a) Wieviele Knoten und Kanten hat dieser Graph:

Anzahl Knoten:  Anzahl Kanten:

b) Ist der Graph streng zusammenhängend – begründen Sie Ihre Antwort:

c) Besitzt der Graph eine Schleife – begründen Sie Ihre Antwort:

d) Bestimmen Sie den Grad des Knotens E:

$d(E) =$    $d^+(E) =$

e) Geben Sie eine beliebige einfache Kantenprogression der Länge 5 an:

f) Geben Sie einen beliebigen Zyklus an:

- B) Gegeben ist ein Binärbaum. Ein Beispiel hierzu ist in Abbildung 6-1 dargestellt. Jeder Knoten im Binärbaum hat einen Schlüssel, einen Wert, sowie einen linken Nachfolger und einen rechten Nachfolger. Dabei gilt, dass der Schlüssel des linken Nachfolgers (bzw. aller über den linken Nachfolger erreichbaren Knoten) immer kleiner und der Schlüssel des rechten Nachfolgers (bzw. aller über den rechten Nachfolger erreichbaren Knoten) immer größer als der eigene Schlüssel sind. Weiterhin ist kein Schlüssel mehr als einmal im Baum vorhanden.

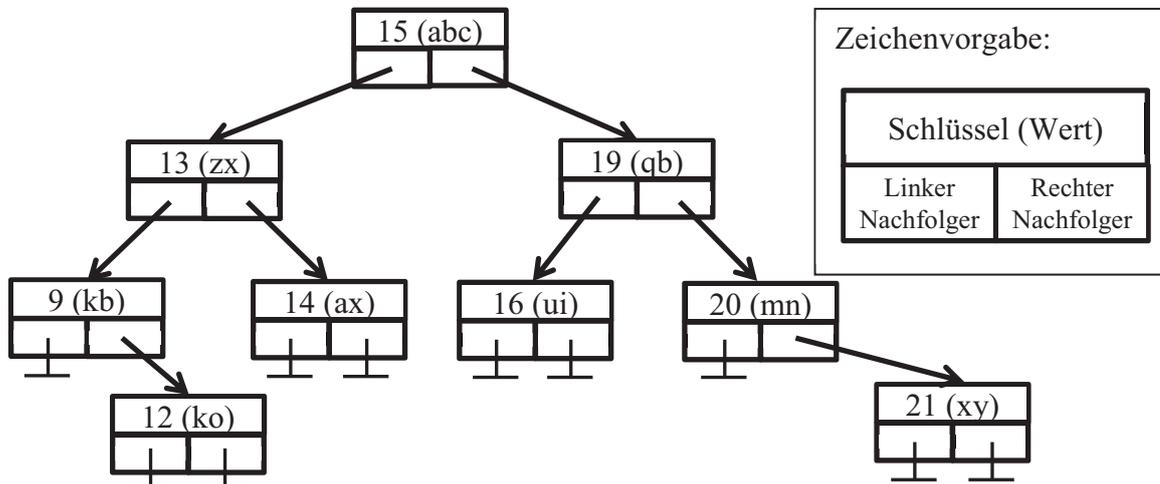


Abbildung 6-1: Beispiel Binärbaum

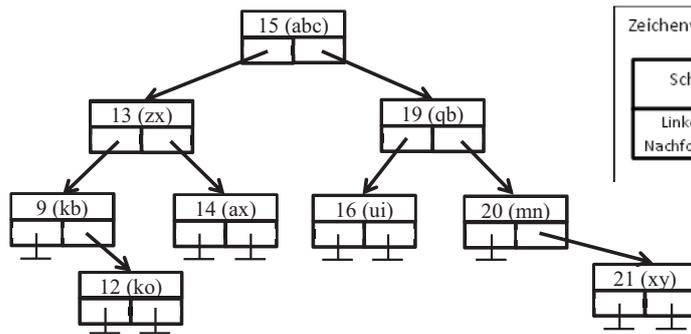
Schreiben Sie in Pseudocode einen Algorithmus, welcher in einem Binärbaum alle Knoten findet, welche einen Schlüssel in einem bestimmten Bereich besitzen. Dieser Bereich wird durch die Variablen *min* und *max* (inkl. *min* und *max*) gekennzeichnet. Es sollen die Werte für alle Knoten im Bereich (Reihenfolge beliebig) ausgegeben werden. Der Aufruf erfolgt mit dem Wurzelknoten (15) - im Beispiel: **suche ( startknoten, 14, 20 )** ; Bitte schreiben Sie einen Algorithmus der für einen allgemeinen Binärbaum gilt, die Abbildung 6-1 dient nur zur Veranschaulichung. Der geschriebene Pseudocode muss ausreichend detailliert sein, um direkt in Programmcode umgesetzt werden zu können.

Die folgende Klasse beschreibt die Attribute eines Knotens:

```

class Knoten {
public:
    int schluessel;
    string wert;
    Knoten* links;
    Knoten* rechts;
};
  
```

Lösen Sie die Aufgabe auf der nächsten Seite.



Zeichenvorgabe:

Schlüssel (Wert)

Linker Nachfolger	Rechter Nachfolger
----------------------	-----------------------

```

class Knoten {
public:
  int schluessel;
  string wert;
  Knoten* links;
  Knoten* rechts;
};
  
```

```

suche( knoten, min, max ) {
  
```

```

}
  
```

C) In Abbildung 6-2 ist ein CPM (Critical Path Method) Graph dargestellt. Hierbei sind Vorgänge als Pfeile mit der Vorgangsdauer  $t$  in Tagen und Ereignisse als Knoten (A-G) dargestellt (siehe Legende). Setzen Sie in die leeren Kästchen jeweils die richtigen Werte für den frühesten Termin, spätesten Termin und den Puffer mit Hilfe der Critical Path Method ein. Anschließend geben Sie den „kritischen Pfad“ unter dem Graph an.

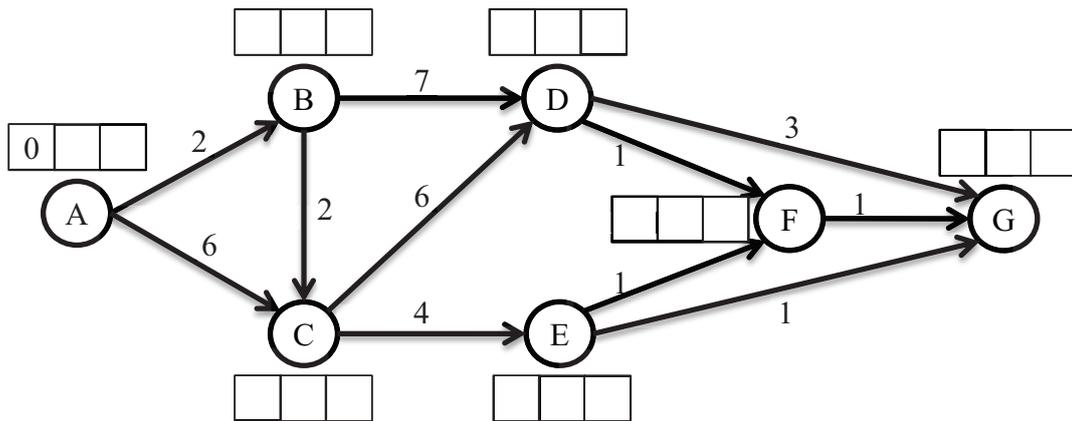
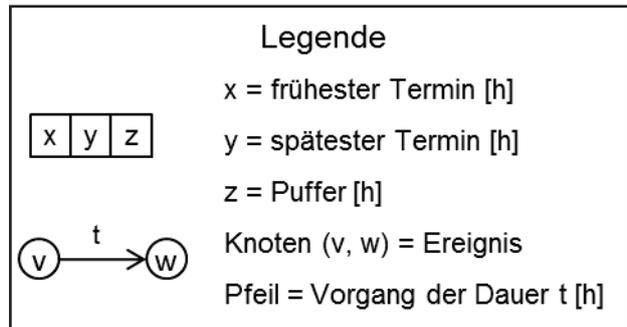


Abbildung 6-2: CPM Graph

**Kritischer Pfad:**

## Aufgabe 7 Selection-Sort Algorithmusanalyse



Gegeben ist der C++ Code des Selection-Sort Algorithmus. Dieser sortiert ein Array in aufsteigender Reihenfolge. Der gezeigten Funktion wird ein Pointer auf ein Array `arr` und die Anzahl der Elemente mit der Übergabevariablen `anzahlElemente` übergeben.

```

01 void mySelectionSort( int* arr, int anzahlElemente ) {
02     int min = 0;
03     int links = 0;
04     while( links < anzahlElemente ) {
05         min = links;
06         for( int i = links + 1; i < anzahlElemente; i++ ) {
07             if( arr[i] < arr[min] ) {
08                 min = i;
09             }
10         }
11         int temp = arr[min];
12         arr[min] = arr[links];
13         arr[links] = temp;
14         links = links + 1;
15     }
16 }

```

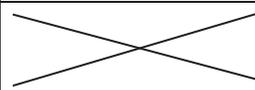
- A) Der Funktion wird für den Worst-Case Fall das folgende Array übergeben. Ermitteln Sie die Ausführung des Algorithmus und füllen Sie jeweils eine Zeile in der folgenden Tabelle aus, wenn der Algorithmus jeweils die Zeile 5 und 9 erreicht hat. Die Werte sollen dabei in die Tabelle NACH Ausführung der entsprechenden Zeile eingetragen werden. Wenn der Wert der Variablen nicht definiert ist tragen Sie ein "-" ein.



				Array arr			
				0	1	2	3
Zeile	min	i		8	6	4	2
5							

- B) Führen Sie zum oben gezeigt Code eine Laufzeitanalyse für den Worst-Case Fall durch. Die Laufzeit von Zeile 8-10, 15-16 müssen Sie nicht in der Tabelle ausfüllen. Geben Sie jeweils die Anzahl der Ausführungen in Abhängigkeit von  $n$  (Anzahl der Elemente im Array) an und unter der Tabelle die Gesamtkomplexität. Die Zwischenrechnungen zu Zeile 6 & 7 können wenn nötig auf einem extra Blatt ausführen.

Es gilt allgemein: 
$$\sum_{j=1}^n (j) = \frac{n \cdot (n+1)}{2}$$

Zeilen- nr		Anzahl der Ausführungen
02	<code>int min = 0;</code>	
03	<code>int links = 0;</code>	
04	<code>while( links &lt; anzahlElemente ) {</code>	
05	<code>min = links;</code>	
06	<code>for( int i = links +1; i &lt; anzahlElemente; i++ ) {</code>	
07	<code>if( arr[i] &lt; arr[min] ) {</code>	
08	<code>min = i;</code>	
09	<code>}</code>	
10	<code>}</code>	
11	<code>int temp = arr[min];</code>	
...	<code>arr[min] = arr[links];</code>	
...	<code>arr[links] = temp;</code>	
...	<code>links = links + 1;</code>	
15	<code>}</code>	
...	<code>}</code>	

Gesamtkomplexität des Algorithmus:

- C) Vergleichen Sie gedanklich den Best-Case mit dem Worst-Case Fall für den Selection-Sort Algorithmus.
- a) Welche Zeile würde eine andere Laufzeit aufweisen im Best-Case Fall im Vergleich zum Worst-Case Fall? Bitte begründen Sie Ihre Antwort.
- b) Wie verändert sich die Gesamtkomplexität des Algorithmus im Best-Case Fall im Vergleich zum Worst-Case Fall? Bitte begründen Sie Ihre Antwort.

**Aufgabe 8 Rechnerarchitektur**

A) Allgemeine Fragen:

- i. Erläutern Sie die beiden Begriffe Polling und Interrupt.  
Polling

Interrupt

- ii. Nennen und erläutern Sie die 4 Phasen der Unterbrechungsverarbeitung.

- iii. Nennen und erläutern Sie 3 Verdrängungsstrategien beim Cache.

## B) Cache

Angenommen, dass ein Prozessor einen Hauptspeicher der Größe 4 GByte hat, wo jedem einzelnen Byte im Speicher eine 32 Bit lange Adresse zugeordnet wird. Zusätzlich verfügt der Prozessor über einen Cache Baustein der Technologie SRAM, der 1024 KBytes (nur Daten) vom Hauptspeicher aufnehmen kann (unabhängig vom notwendigen Speicher zur Speicherung der weiteren Bits, wie z.B. Tag). Der Cache hat eine Blockgröße von 64 Bytes. Zusätzlich wird auf dem Cache für jeden Block ein Valid-Bit vorgesehen.

- i. Berechnen Sie für Direct-Mapped und für 4-Wege-assoziativen Cache die Anzahl der Cache-Blöcke und der Cache-Sets.

Direct-Mapped:

4-Wege:

- ii. \*Berechnen Sie für jeden Typ die in Wirklichkeit notwendige Speicherkapazität für den Cache-Baustein

Direct-Mapped:

4-Wege: