

## 2. Semester : 1. Prüfung

Name :	
--------	--

- Die gesamte Prüfung bezieht sich auf die Programmierung in C++ !!
- Prüfungsdauer: 90 Minuten
- mit Kugelschreiber oder Tinte schreiben
- Lösungen können direkt auf die Aufgabenblätter geschrieben werden
- PCs sind nicht erlaubt
- Unterlagen und Bücher sind erlaubt
- Die Aufgaben sind in Fragen unterteilt, die mit Kleinbuchstaben gekennzeichnet sind
- Achte auch auf Details wie Punkte oder Kommas und Semikolons. !!!!!

Aufgabe	Punkte	
Zeiger und Referenzen	10	
Namensräume	4	
Klassen	16	
Programmverständnis	6	
Weitere Fragen	8	
TOTAL		

## 1. Zeiger und Referenzen

- a) In folgender Tabelle findest zu jeweils ein, zwei oder drei Zeilen, die zusammen gehören. Einige davon sind falsch und können nicht kompiliert werden. **Streiche die fehlerhaften Zeilen durch.** Achte dabei vor allem auf die Datentypen, Zeiger oder Referenzen, die zugewiesen werden, aber auch auf fehlende Semikolons oder unvollständige Variablendefinitionen. (5 Punkte)

<code>char Buchstabe = '2'; char&amp; z = Buchstabe; OK</code>
<code>char z = 'B'; double* pz = &amp;z; Falsch, Datentypen</code>
<code>kk = 45 Falsch, Datentyp fehlt int&amp; bb = kk;</code>
<code>double d = 3.66; Falsch, Pointer kann nicht double* pd = &amp;d; 3.66 als Wert haben</code>
<code>int v1 = 55; int v2 = v1; OK</code>
<code>int* pt = 0; int t = 50; pt = &amp;t; OK</code>
<code>int m = 10; int* pm = m; Falsch, müsste Adresse von m</code>
<code>double* pd = 3.55 Falsch, Wert und Semikolon</code>
<code>string name = "Markus"; string&amp; Bez = name; OK</code>
<code>pz = &amp;z; Falsch, macht keinen Sinn *pz = 0;</code>

- b) Schreibe eine Funktionen SwapStrings, die zwei Variablen vom Datentyp string tauscht. Die Funktion soll mit **Referenzen auf strings** arbeiten ! Ergänze danach die main-Funktion darunter mit dem Aufruf der Funktion. (5 Punkte)
- 

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

// hier Funktion SwapStrings hinschreiben :

void SwapStrings(string& x, string& y)
{
    string temp = x;
    x = y;
    y = temp;
}

int main()
{
    string a = "Zig";
    string b = "Zag";

    // Hier Funktion SwapStrings aufrufen

    SwapStrings(a, b);

    return 0;
}
```

---

## 2. Namensräume

- a) Die Klasse `string` ist wie wir wissen im Namensraum **std** definiert. Schreibe zwei kleine aber vollständige `main`-Funktionen, in der du je eine `string`-Variable `Test` definierst. Verwende jeweils unterschiedliche Arten um in den Namensraum **std** zu gelangen. (4 Punkte)
- 

```
#include <string>

using namespace std;

int main()
{
    // Hier Variable Test vom Datentyp string definieren

    string Test;

    return 0;
}
```

---

```
#include <string>

int main()
{
    // Hier Variable Test vom Datentyp string definieren

    std::string Test;

    return 0;
}
```

---

### 3. Klassen

- a) Für ein Adressverwaltungsprogramm brauchen wir eine Klasse Adresse. Zu einer Adresse sollen folgende Angaben gehören: Der **Vorname**, der **Nachname**, die **Strasse**, die **Postleitzahl** und der **Ort**. Definiere also die richtigen Datenelemente um diese Angaben zu speichern und wähle vernünftige Datentypen dafür. Die Klasse soll einen **Konstruktor mit Parametern** haben, mit dem man den Vornamen und den Nachnamen setzen kann. Zusätzlich soll es eine Methode **„SetzeAdresse“** geben, mit der man die Strasse, die Postleitzahl und den Ort setzen kann. Es genügt die Definition der Klasse, also das was man üblicherweise in eine Headerdatei schreiben würde. (10 Punkte)
- 

```
#ifndef ADRESSE_H
#define ADRESSE_H
#include <string>
using std::string;

class Adresse
{
    public:
        Adresse(const string& vorname, const string& nachname);

        void SetzeAdresse(const string& strasse,
                           const string& ort,
                           long plz);

    private:
        string m_Vorname;
        string m_Nachname;
        string m_Strasse;
        string m_Ort;
        long   m_Plz;
};

#endif
```

---

- b) Definiere eine Klasse **Zeitdauer**, die sich eine Zeitdauer in ganzen Sekunden merkt. Definiere einen Konstruktor mit Parametern, mit dem diese Zeitdauer gesetzt wird. Zusätzlich soll es folgende Methoden geben : "HoleGanzeMinuten". Diese Methode soll einfach die Anzahl Minuten (ohne Bruchteile) berechnen und zurückgeben (die Formel lautet hierfür  $\text{AnzahlSekunden} / 60$ ). Eine zweite Methode "HoleMinuten" soll die Anzahl Minuten mit Bruchteilen zurückgeben. Die Formel ist fast gleich wie bei der Methode "HoleGanzeMinuten" ( $\text{AnzahlSekunden} / 60.0$ ), der Rückgabetyt ist aber anders ! Schreibe die Methoden direkt unter die Klassendefinition. (6 Punkte). **Die Klasse hat ein Datenelement, ein Konstruktor mit Parametern und zwei Methoden.**
- 

```
class Zeitdauer
{
    public:
        Zeitdauer(long sekunden);

        long HoleGanzeMinuten() const;
        double HoleMinuten() const;

    private:
        long m_Sekunden;
};

Zeitdauer::Zeitdauer(long sekunden)
{
    m_Sekunden = sekunden;
}

long Zeitdauer::HoleGanzeMinuten() const
{
    return m_Sekunden / 60;
}

double Zeitdauer::HoleMinuten() const
{
    return (double)m_Sekunden / 60.0;
}
```

#### 4. Programmverständnis

a) Was gibt folgender Code auf dem Bildschirm aus (6 Zeilen) ? (6 Punkte)

---

```
#include <string>
#include <iostream>
using namespace std;
class SingyDinky
{
public:
    SingyDinky(string Name) // Konstruktor mit Parameter
    {
        m_Name = Name;
        cout << m_Name << " kommt" << endl;
    }
    ~SingyDinky() // Destruktor
    {
        cout << m_Name << " geht" << endl;
    }
    void singe()
    {
        cout << m_Name << " singt" << endl;
    }
private:
    string m_Name;
};
int main()
{
    SingyDinky einDinky("Fred");
    einDinky.singe();

    bool b = true;
    if(b)
    {
        SingyDinky nochEinDinky("Sepp");
        nochEinDinky.singe();
    }
    return 0;
}
```

---

**Fred kommt**  
**Fred singt**  
**Sepp kommt**  
**Sepp singt**  
**Sepp geht**  
**Fred geht**

## 5. Weitere Fragen

a) Weiviele Objekte werden im folgenden Beispiel erzeugt ? (4 Punkte)

---

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

class Dinky
{
public:
    Dinky(string Name)
    {
        m_Name = Name;
    }

private:
    string m_Name;
};

int main()
{
    Dinky einDinky("Fred");

    Dinky nochEinDinky("Sepp");

    return 0;
}
```

---

**Ein Dinky Objekt besteht aus einem weiteren string Objekt, das macht also zwei Objekte pro Dinky Objekt = 4.**

**Zusätzlich wird beim Konstruktor mit Parametern ein string Objekt erzeugt, ein sogenanntes**

**temporäres Objekt. Ein wenig schwierig...**

**Total 6 Objekte !**

- b) Schreibe eine Funktion "BerechneKreisFlaeche". Diese Funktion hat als Parameter den Radius als Gleitkommazahl und gibt das Ergebnis ebenfalls als Gleitkommazahl zurück. Die Funktion wird so aufgerufen : (Schreibe die Funktion darunter) (3 Punkte)
- 

```
double Radius = 5.5;

double Flaeche = BerechneKreisFlaeche(Radius);

double BerechneKreisFlaeche(double r)
{
    double f = r * r * 3.1415;
    return f;
}
```

---

- c) In einem Wort ! Willst du fehlerfreie C++ Programme schreiben ? (1 Punkt)

**Ja**