

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	
Allgemeine Fragen zu C++		
TOTAL		

1. Zeiger und Referenzen

- a) In folgender Tabelle findest zu jeweils 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

<pre>int english = 2; int& german = english; OK</pre>
<pre>double pi = 3.1415; int* pPi = &pi; FALSCH, int-Zeiger auf double geht nicht</pre>
<pre>double* pNeu = 0; pNeu = 3.66; FALSCH, Zeiger kann nicht den Wert 3.66 haben</pre>
<pre>char buchstabe = 'z'; char& leBuch = buchstabe; OK</pre>
<pre>Test = 55; FALSCH, Datentyp fehlt int* pTest = &Test;</pre>
<pre>int Zahl = 3.55; int* pZahl = &Zahl; OK (implizite Typwandl.)</pre>
<pre>int* pp = 0; char buchstabe = pp; FALSCH, ungleiche Typen</pre>
<pre>int l = 2000; int* pl = &l; *pl = 3000; OK</pre>
<pre>char* pc = 0; pc = 3.55; FALSCH, Zeiger kann nicht den Wert 3.55 haben &pc = 'v';</pre>
<pre>**A = &&55; FALSCH, macht keinen Sinn int t = A;</pre>

- b) Schreibe zwei Funktionen SwapChars, die zwei Variablen vom Datentypen char tauschen. Eine soll mit Referenzen, die andere mit Zeigern als Parameter definiert werden.

<pre>void SwapChars(char& a, char& b) { char z = a; a = b; b = z; }</pre>	<pre>void SwapChars(char* a, char* b) { char z = *a; *a = *b; *b = z; }</pre>
---	---

2. Namensräume

- a) Folgende Funktion ist in einem bestimmten Namensraum definiert. Schreibe darunter zwei kleine main-Funktionen, die diese Funktion auf zwei verschiedene Arten aufruft.

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

namespace pruefung
{

void SagHallo()
{
    cout << "Hallo" << endl;
}

}
```

```
int main()
{
    pruefung::SagHallo();
    return 0;
}
```

```
using namespace pruefung;
int main()
{
    SagHallo();
    return 0;
}
```

3. Klassen

- a) Definiere eine Klasse Buch, die ein Buch für einen Online-Buchhandel abstrahiert. Jedes Buch hat einen Titel, eine Nummer und einen Preis. Definiere also die richtigen Datenelemente. Schreibe einen Konstruktor **mit Parametern**, mit dem die Datenelemente initialisiert werden und eine Methode verkaufen. Es genügt die Klassendefinition, du musst also nur den Code schreiben, den man in einer Header-Datei schreiben würde.
-

```
class Buch
{
    public:
        // Konstruktor
        Buch(const string& titel, long nummer, double Preis);

        void verkaufen();
    private:
        string m_Titel;
        long   m_Nummer;
        double m_Preis;
}; // Semikolon nicht vergessen !
```

- b) Schreibe eine Klasse Kreis. Ein Kreis hat einen Radius. Die Klasse soll eine Methode "BerechneFlaeche" und eine Methode "BerechneUmfang" besitzen. Sie soll einen Konstruktor mit Parametern haben und zusätzlich eine Methode "SetzeRadius". Überlege bei jeder Methode was für einen Rückgabewert sie hat und was für Parameter der Methode übergeben werden müssen. Natürlich musst du auch die richtigen Datenelemente definieren. Versuche auch Methoden const zu definieren, wenn möglich... Übrigens Kreisfläche=Radius*Radius*PI; Umfang = 2*Radius*PI. Schreibe zuerst den Code für die Header-Datei und direkt darunter den Code, den du in eine .cpp Datei schreiben würdest.
-

```
class Kreis
{
    public:
        // Konstruktor
        Kreis(double Radius);

        void SetzeRadius(double Radius);
        double BerechneFlaeche() const;
        double BerechneUmfang() const;

    private:
        double m_Radius;
};

// CODE :
const double PI = 3.1415; // nützlich und gut !
// Konstruktor
Kreis::Kreis(double Radius)
{
    m_Radius = Radius;
}

void Kreis::SetzeRadius(double Radius) // nicht const, denn
{
    m_Radius = Radius;                // das Objekt ändert
                                     // HIER !
}

double Kreis::BerechneFlaeche() const // die Funktion ändert
{
    return m_Radius * m_Radius * PI; // das Objekt nicht, also
}

double Kreis::BerechneUmfang() const // Auch hier const !
{
    return m_Radius * 2 * PI;
}
```

4. Programmverständnis

a) Was gibt folgendes Programm aus ?

```
#include <iostream>

using namespace std;

class Winky
{
    public:
        Winky()
        {
            cout << "Ein Winky kommt" << endl;
        }
        ~Winky()
        {
            cout << "Ein Winky geht" << endl;
        }
        void winke()
        {
            cout << "Ein Winky winkt" << endl;
        }
};

int main()
{
    Winky winkyEins;
    winkyEins.winke();

    bool b = true;
    if(b)
    {
        Winky winkyZwei;
    }
    winkyEins.winke();
    return 0;
}
```

Ein Winky kommt
Ein Winky winkt
Ein Winky kommt
Ein Winky geht
Ein Winky winkt
Ein Winky geht

5. Weitere Fragen

a) Wieviele Objekte existieren in der dieser main-Funktion ?

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

struct Person
{
    Vorname;
    Name;
};

int main()
{
    string name("Moier");
    string vorname("säpp");
    Person einePerson;
    einePerson.Vorname = vorname;
    einePerson.Name = name;
    return 0;
}
```

1. String Objekt; Variablenname=name, Inhalt="Moier"

2. String Objekt; Variablenname=vorname, Inhalt="säpp"

**3. Person Objekt(zusammengesetzt); Variablenname=einePerson,
Inhalt: weiteres string Objekt : Vorname und string Objekt Name**

Total = 5 Objekte !

Das Objekt von der Klasse/Struktur Person ist selber ein Objekt, enthält aber noch zwei weitere Objekte von der Klasse string. Die ersten beiden Objekte sind die string, die zuerst definiert werden.