2. Semester, 2. Prüfung, Lösung

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
- PC's sind nicht erlaubt
- Unterlagen und Bücher sind erlaubt
- Achte auf Details wie Punkte, Kommas und Semikolons

Aufgabe	Punkte	
Vektoren und Arrays	8	
Teilobjekte und statische Elemente	11	
Klassen und Strukturen	15	
Programmverständnis	8	
Weitere Fragen	6	
Total	48	

1. Vektoren und Arrays

a) Was hat die Variable *d* im folgenden Beispiel für einen Wert? (1 Punkt)

```
double werte[8] = { 1.2, 3.4, 4.5, 5.6, 7.8, 9.0 };
double d = werte[5];
```

9.0

b) Erzeuge ein Array von 22 string's (std::string) mit dem Namen *Namenarray*. Du brauchst das Array nicht zu initialisieren. Schreibe auch die nötigen *include*–Anweisungen und *namespace*–Bezeichnungen hin. (2 Punkte)

```
#include <string>
using namespace std;
string Namenarray[22];
```

c) Definiere ein Array von 10 double-Werten und schreibe eine Schleife, die alle Werte mit **1.0** initialisiert. (3 Punkte)

```
double werte[10];
for(long i = 0; i < 10; ++i)
{
    werte[i] = 1.0;
}
```

d) Definiere ein Array von 100 unsigned char - Werten und initialisiere es so, dass alle Elemente den Wert 0 haben. Verwende eine möglichst einfache Schreibweise (das heisst eine Zeile)! (2 Punkt)

```
char einArray[100] = { 0 };
```

2. Teilobjekte und statische Elemente

a) Es sind zwei Klassen gegeben.
Eine Klasse *Name*, die aus einen Vornamen und einem Nachnamen besteht. Diese Klasse *Name* hat nur einen Konstruktor mit Parametern.
Die zweite Klasse *Person* enthält ein Datenelement m_Name der Klasse *Name*. Auch hier gibt es nur einen Konstruktor mit Parametern.
Schreibe den Code für die Klasse *Name*. (3 Punkte)
Schreibe den Code für den Konstruktor der Klasse *Person*. (3 Punkte)
(Total 6 Punkte)

```
#include <string>
using namespace std;
class Name
{
    public:
        Name(const string&  Vorname,
             const string& Nachname)
    private:
        const string m_Vorname;
        const string m_Nachname;
};

class Person
{
    public:
        Person(const string& Vorname,
               const string& Nachname);
    private:
        Name m_Name;
};
// Konstruktor für die Klasse Name
Name::Name(const string& Vorname,
           const string& Nachname)
     :m_Vorname(Vorname), m_Nachname(Nachname)
{
}


// Konstruktor für die Klasse Person
Person::Person(const string& Vorname,
               const string& Nachname)
       :m_Name(Vorname, Nachname)
{
}
```

b) Die folgende Klasse *Objekte* enthält eine statische Variable *s_AnzahlObjekte*. Diese Variable muss initialisiert werden und bei jedem Konstruktoraufruf um eins erhöht werden. Die Variable *s_AnzahlObjekte* enthält dadurch die Anzahl erzeugter Objekte.

Schreibe den Code, der die Variable initialisiert und den Code für den Konstruktor. (3 Punkte)
Schreibe auch den Code für die Funktion *holeAnzahlObjekte*.
(2 Punkte) (Total 5 Punkte)

```
class Objekte
{
    public:
        // Konstruktor
        Objekte();

        // statische Funktion zum lesen
        // der Anzahl Objekte
        static int holeAnzahlObjekte();

    private:
        static int s_AnzahlObjekte;
};
```

Lösung:

```
int Objekte::s_AnzahlObjekte = 0;

Objekte::Objekte()
{
    s_AnzahlObjekte++;
}


int Objekte::holeAnzahlObjekte()
{
    return s_AnzahlObjekte;
}
```

3. Klassen und Strukturen

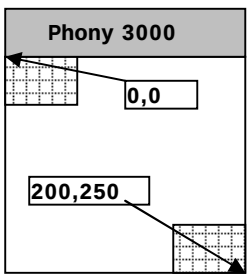
a) Nimm an du arbeitest in einer Firma, die ein Verkehrsleitsystem programmiert. Das Modell für das System besteht aus Ampeln und Kreuzungen.
Definiere eine **Aufzählung** (enum) für die drei möglichen Ampelfarben. (2 Punkte)
Definiere eine Klasse Ampel, die ein Datenelement für die Ampelfarbe (verwende den enum von oben) enthält. Diese Klasse hat einen Konstruktor mit einem Parameter, mit dem die Ampelfarbe initialisiert wird.
Schreiben den Code für diesen Konstruktor direkt unter die Klassendefinition. (5 Punkte)
(Vorsicht, diese Aufgabe hat nichts mit RGB-Werten oder Farbsättigung zu tun.)
(Total 7 Punkte)

```
enum Ampelfarbe
{
    rot,
    gelb,
    gruen
};

class Ampel
{
     public:
          Ampel(Ampelfarbe farbe);
     private:
          Ampelfarbe m_farbe;
};


Ampel::Ampel(Ampelfarbe farbe)
      :m_farbe(farbe)
{
}
```

b) Nimm an du arbeitest in einer Firma, die ein Verkehrsleitsystem programmiert. Das Modell für das System besteht aus Ampeln und Kreuzungen.
Definiere eine Klasse Kreuzung. Diese enthält als Datenelemente vier Ampeln. Die Klasse **Ampel** hat einen Konstruktor mit einem Parameter (siehe auch Aufgabe a) für die Ampelstellung. (3 Punkte)
Schreibe für die Klasse Kreuzung einen Konstruktor (ohne Parameter), der die vier Ampeln so initialisiert, dass sie auf Rot stehen. (2 Punkte)
Beachte, dass man die vier Ampeln nicht in einem Array definieren kann. Du musst sie also einzeln definieren. (Total 5 Punkte)

```
class Kreuzung
{
    public:
        Kreuzung();
    private:
        Ampel m_Sued;
        Ampel m_Nord;
        Ampel m_Ost;
        Ampel m_West;
};

Kreuzung::Kreuzung()
         :m_Sued(rot),
          m_Nord(rot),
          m_Ost(rot),
          m_West(rot)
{
}
```

c) Du arbeitest an der Entwicklung eines mobilen Telefons (Phony 3000). Für die Bildschirm-Funktionen wird jedem Punkt auf dem Bildschirm eine Koordinate zugeordnet.
Als Entwickler wirst Du diese beiden einzelnen Werte (x,y) in einer Struktur oder Klasse zusammenfassen wollen. Die oberste linke Ecke des Displays hat die Koordinate [0,0], die unterste rechte [200, 250]. Definiere diese Struktur oder Klasse. Du kannst sie *Point* nennen. Verwende nicht mehr Speicher als nötig! (3 Punkte)

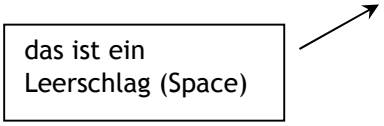



```
struct Point
{
    unsigned char X;
    unsigned char Y;
};
```

4. Programmverständnis

a) Betrachte folgenden Code und finde heraus was er ausgibt. (4 Punkte)

```
#include <iostream>

using namespace std;

char BuchstabenSalat[] =
             {'M','i','u','r','a','B','s','k',' '};
```




```
int main()
{
    char Ausgabe[50] = { 0 };

    Ausgabe[0]  = BuchstabenSalat[0];
    Ausgabe[1]  = BuchstabenSalat[4];
    Ausgabe[2]  = BuchstabenSalat[3];
    Ausgabe[3]  = BuchstabenSalat[7];
    Ausgabe[4]  = BuchstabenSalat[2];
    Ausgabe[5]  = BuchstabenSalat[6];
    Ausgabe[6]  = BuchstabenSalat[8];
    Ausgabe[7]  = BuchstabenSalat[5];
    Ausgabe[8]  = BuchstabenSalat[2];
    Ausgabe[9]  = BuchstabenSalat[3];
    Ausgabe[10] = BuchstabenSalat[3];
    Ausgabe[11] = BuchstabenSalat[1];
    Ausgabe[12] = 0;
    Ausgabe[13] = BuchstabenSalat[0];
    Ausgabe[14] = BuchstabenSalat[5];

    cout << Ausgabe << endl;

    return 0;
}
```

Ausgabe:

Markus Burri

b) Was gibt folgender Code aus? (4 Punkte)

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

//Klassendefinition
class DancyDinky
{
    public:
        DancyDinky(const string& Name);
        void Tanze();
    private:
        string m_Name;
        int m_AnzahlTanz;
        static int s_AnzahlDinkies;
};

// statisch Variable initialisieren
int DancyDinky::s_AnzahlDinkies = 0;

// Konstruktor
DancyDinky::DancyDinky(const string& Name)
          :m_Name(Name)
{
    s_AnzahlDinkies++;
    m_AnzahlTanz = s_AnzahlDinkies;
}

// Methode Tanze
void DancyDinky::Tanze()
{
    for(int i = 0; i < m_AnzahlTanz; ++i)
    {
        cout << m_Name << " tanzt" << endl;
    }
}

// Hier die main-Funktion
int main()
{
    DancyDinky dinkies[3] =
    {
        "Lolo",
        "Floo",
        "Blub"
    };
    for(int i = 0; i < 3; ++i)
    {
        dinkies[i].Tanze();
    }
    return 0;
}
```

```
Lolo tanzt
Floo tanzt
Floo tanzt
Blub tanzt
Blub tanzt
Blub tanzt
```

5. Weitere Fragen

a) Wieviele Bytes an Speicherplatz werden für das Array *Text* gebraucht? (1 Punkte)

```
char Text[] = "Hallo Du";
```

```
9 Bytes
```

b) Beim folgenden Array kann mit dem Index-operator auf einzelne Elemente zugegriffen werden. Schreibe eine kleine Liste mit den gültigen Indices. Oder anders ausgedrückt, mach eine Liste mit den Werten, welche die Variable *Index* haben darf (Vorsicht Falle). (2 Punkte)

```
double dasArray[5] = { 1.2, 2.3, 4.5 };
double Wert = dasArray[Index];
```

```
0, 1, 2, 3, 4
```

c) Was hat die Variable z hier für einen Wert? (2 Punkte)

```
int dasArray[5] = { 0, 1, 2 };
int z = dasArray[3];
```

```
z = 0 !!!
```

d) Willst Du fehlerfreie C++ Programme schreiben? (1 Punkt)

Ja