3. Semester, 2. Prüfung

Name	

- Prüfungsdauer: 90 Minuten
- Bitte deutlich schreiben
- Lösungen können direkt auf die Aufgabenblätter oder auf die Rückseite der Aufgabenblätter geschrieben werden
- PC's sind nicht erlaubt
- Beliebige Unterlagen und Bücher sind erlaubt
- Achte auf Details wie Punkte, Kommas und **Semikolons**

Aufgabe	Punkte	
Variablen, Datentypen und Kontrollstrukturen	10	
Zeiger, Arrays und dynamischer Speicher	10	
Klassen	10	
Klassen und Vererbung	10	
Verschiedene Fragen	11	
Total	51	

1. Variablen, Datentypen und Kontrollstrukturen

a) Definiere eine Variable vom Datentypen float mit dem Wert 1.0. (1 Punkt)

float f = 1.0;

b) Definiere eine **Konstante** mit der Anzahl Wochentage (7). Wähle den richtigen Datentypen. (1 Punkt)

const int anzahlTage = 7;

c) Schreibe in folgende main-Funktion eine for-Schleife, die 10 mal das Wort **„Hallo“ untereinander** auf der Konsole ausgibt. (3 Punkte)

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

int main()
{
    // Hier Schleife einfügen
    for(int i = 0; i < 10; ++i)
    {
        cout << „Hallo“ << endl;
    }

    return 0;
}
```

d) Bei folgendem Code fehlt auf der ersten Zeile (gestrichelte Linie) der Datentyp für die Variable *pn*. Ergänze diesen Datentypen. (1 Punkt)

```
double*           pn = new double;

*pn = 0.0;
delete pn;
```

e) Wandle folgende if-Abfrage in eine switch-case Anweisung um. (3 Punkte)

```
int a = 0;
a = holeDaten(); // diese Funktion soll irgendeine Zahl
                 // zurückgeben

// diesen Teil ab hier in eine switch-case
// Anweisung umwandeln
if(0 == a)
{
    function1(); // function1 tut irgendwas sinvolles
}
else if(1 == a)
{
    function2();
}
else
{
    function3();
}


switch(a)
{
    case 0:
        function1();
    break;
    case 1:
        function2();
    break;
    default:
        function3();
    break;
}
```

f) Was hat die Variable z in diesem Beispiel für einen Wert. (1 Punkt)

```
unsigned int i = 0;
int z = sizeof(i); // sizeof berechnet den
                   // Speicherplatzverbrauch
Bei der originalen Prüfung war der Datentyp char >> 1
Bei der Prüfung vom Samstag ist das Ergebnis >> 4
```

2. Zeiger, Arrays und dynamischer Speicher

a) Wieviele Elemente hat das Array *array*? (1 Punkt)

```
int array[20];
```

20

b) Wie heisst im folgenden Beispiel der grösste mögliche Index um das letzte Element im Array anzusprechen? (1 Punkt)

```
double daten[19];
```

18

c) Erzeuge ein Array von 10 **unsigned int** Werten mit **new**. (1 Punkt)
Setze jedes Element im Array mit einer Schleife auf *5*. (2 Punkte)
Lösche den dynamisch allozierten Speicher wieder korrekt. (1 Punkt)
(Total 4 Punkte)

```
unsigned int* array = new unsigned int[10];
for(int i = 0; i < 10; ++i)
{
    array[i] = 5;
}

delete [] array;
```

d) Definiere ein Array von 20 char-Werten und initialisiere alle Elemente mit 0 **auf einer Zeile**. (1 Punkt)

```
char array[20] = { 0 };
```

e) Wie heisst in diesem Beispiel der grösste mögliche Index um das letzte Element im Array anzusprechen? (1 Punkt)

```
char text[] = „Hallo Welt!“;

Die abschliessende Null gehört auch dazu!
Bei der Prüfung Version 1 ist die Lösung: 10
Bei dieser Version (Samstagversion) ist die Lösung 11
```

f) Erzeuge einen double dynamisch (also mit new) und initialisiere den Wert dieses double mit 10.0 (eine Zeile)
Gib den Speicher wieder frei. (2 Punkte)

```
double* wert = new double(10.0);
delete wert;
```

3. Klassen

a) Definiere eine Klasse *Vogel*. (1 Punkt)
Definiere in dieser Klasse ein Datenelement für die Vogelart und ein zweites für das Gewicht des Vogels in Gramm (wähle die richtigen Datentypen). (2 Punkte)
Definiere einen Konstruktor mit Parametern. Als Parameter muss man dem Konstruktor die Vogelart und das Gewicht übergeben. Du kannst den Konstruktor inline definieren oder unter die Klasse schreiben. (2 Punkte).
Es gibt keine weiteren Methoden und Datenelemente zu definieren. Die Lösung braucht etwa 10-12 Zeilen. (Total 5 Punkte)

```
#include <string>

using std::string;



class Vogel
{
public:
    Vogel(const string& art, unsigned long gewicht);
private:
    string m_art;
    unsigned long m_gewicht;
};

Vogel::Vogel(const string& art, unsigned long gewicht)
      :m_art(art), m_gewicht(gewicht)
{
}
```

b) Definiere eine Struktur (struct) *Punkt3d* mit **drei** Datenelementen für die Koordinaten im Raum. Es soll nur positive, ganzzahlige Koordinatenwerte geben! (3 Punkte)

```
struct Punkt3d
{
    unsigned int x;
    unsigned int y;
    unsigned int z;
};
```

c) Was gibt folgendes Programm aus? (2 Punkte)

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

class Dinky
{
public:
    Dinky()
    {
        cout << "Ein Dinky wird erzeugt" << endl;
    }
    void sagHallo()
    {
        cout << "Hallo" << endl;
    }
};

const unsigned int Anzahl = 3;

int main()
{
    Dinky dinkies[Anzahl];
    for(int i = 0; i < Anzahl; ++i)
    {
        dinkies[i].sagHallo();
    }
    return 0;
}

Ein Dinky wird erzeugt
Ein Dinky wird erzeugt
Ein Dinky wird erzeugt
Hallo
Hallo
Hallo
```

4. Klassen und Vererbung

a) Welche Klasse ist im folgenden Beispiel die Basisklasse? (1 Punkt)

```
class Vogel
{
};

class Falke : public Vogel
{
};
```

Vogel

b) Leite von folgender Klasse Auto eine Klasse Kombi ab.
Diese Klasse hat zusätzlich ein Datenelement für den Stauraum in Liter (Ganzzahl ist ok).
Der Konstruktor der Klasse Kombi ist fast gleich wie der Konstruktor der Klasse Auto mit einem zusätzlichen Parameter für den Staurraum.
Definiere diesen Konstruktor inline oder unter der Klasse. (4 Punkte)

```
#include <string>

using std::string;

enum Farbe
{
    weiss,
    schwarz,
    gruen,
    blau,
    rot
};

class Auto
{
public:
    Auto(const string& marke,
         Farbe farbe)
    {
        m_marke = marke;
        m_farbe = farbe;
    }
private:
    string m_marke;
    Farbe  m_farbe;
};
```

Lösung auf nächster Seite

```
class Kombi : public Auto
{
public:
    Kombi(const string& marke,
          Farbe farbe,
          unsigned int stauraum);
private:
    unsigned int m_stauraum;
};

Kombi::Kombi(const string& marke,
             Farbe farbe,
             unsigned int stauraum)
      :Auto(marke, farbe)
{
}
```

c) Welcher Konstruktor wird zuerst aufgerufen? Derjenige der Basisklasse oder der Konstruktor der abgeleiteten Klasse? (1 Punkt)

Basisklassenkonstruktor

d) Was gibt folgendes Programm aus? (4 Punkte)

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

class Vogel
{
public:
    Vogel()
    {
        cout << "Ein Vogel kommt" << endl;
    }
    virtual ~Vogel()
    {
        cout << "Ein Vogel geht" << endl;
    }
    virtual void sing()
    {
        cout << "Ein Vogel singt" << endl;
    }
    void bruete()
    {
        cout << "Ein Vogel bruetet" << endl;
    }
};

class Sperling : public Vogel
{
public:
    Sperling()
    {
        cout << "Ein Sperling kommt" << endl;
    }
    ~Sperling()
    {
        cout << "Ein Sperling geht" << endl;
    }
    void sing()
    {
        cout << "Ein Sperling singt" << endl;
    }
    void bruete()
    {
        cout << "Ein Sperling bruetet" << endl;
    }
};

int main()
{
    Vogel* einVogel = new Sperling;
    einVogel->sing();
    einVogel->bruete();
    delete einVogel;
    return 0;
}
```

```
Ein Vogel kommt
Ein Sperling kommt
Ein Sperling singt
Ein Vogel bruetet
Ein Sperling geht
Ein Vogel geht
```

5. Verschiedene Fragen

a) Schreibe eine einfache Funktion *SagHallo*, die „Hallo“ auf der Console ausgibt. Ergänze auch die nötigen #includes und namespace-Anweisungen. (2 Punkte)

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

void SagHallo()
{
    cout << „Hallo“ << endl;
}
```

b) Schreibe eine Funktion *Swap*, die zwei Argumente vom Datentyp *char* vertauscht. (2 Punkte)

```
void Swap(char& a, char& b)
{
    char temp = a;
    a = b;
    b = temp;
}
```

c) Definiere eine Aufzählung (*enum*) für die drei Ampelfarben. (2 Punkte)

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

d) Schreibe eine Funktion *ReverseOut*, die einen C-String rückwärts auf der Konsole ausgibt. Sie nimmt als Argument einen C-String (const char*). In der Funktion wirst du sehr wahrscheinlich eine Schleife brauchen. Um die Länge des Strings zu berechnen kannst du die Funktion *strlen* verwenden. Die #includes kannst du weglassen. (4 Punkte)

```
// Beispiel für strlen
const char* text = „Hallo“;
int test = strlen(text); // test hat hier den Wert 5


// Hier Funktion ReverseOut definieren
void ReverseOut(const char* text)
{
    int len = strlen(text);
    for(int i = len; i >= 0; --i)
    {
        cout << text[i];
    }
}
```

e) Willst du fehlerfreie Programme schreiben? (1 Punkt)

```
Ja
```