

Einführung in die Programmierung

Klausur — WiSe 2023/24 — 19. Februar 2024

Haupttermin, Prüfungsnr. 23201019

Gruppe: gcc

Unbedingt ausfüllen

Nachname

Vorname

Matrikelnummer

Identifikator (Beliebiges Wort zur Identifikation im anonymen Notenaushang)

Unterschrift

Hiermit bestätige ich meine Prüfungsfähigkeit.

Grundregeln

- Die Bearbeitungszeit der Klausur beträgt **120 Minuten**.
- Sie schreiben diese Klausur **vorbehaltlich** der Erfüllung der **Zulassungsvoraussetzung**. Das heißt: Wir werden Ihre Zulassung vor Korrektur prüfen; die Tatsache, dass Sie die Klausur mitschreiben, bedeutet keine implizite Zulassung.
- Es sind **keine Unterlagen** und auch **keine anderen Hilfsmittel** erlaubt.
- Benutzen Sie nur dokumentenechten (blauen oder zur Not schwarzen) **Kugelschreiber**. Bleistiftlösungen werden nicht gewertet.
- Es zählt die Antwort, die sich im dafür vorgesehenen Kästchen befindet! Soll eine andere Antwort gewertet werden, so ist diese **eindeutig** zu kennzeichnen und in der vorgesehenen grauen Box darauf hinzuweisen.
- Jegliches Schummeln, und auch der Versuch desselben, führt zum Ausschluss von der Klausur und einer Bewertung mit **5,0**.

Wird vom Korrektor/Prüfer ausgefüllt

Aufgabe	1	2	3	4	5	6	7	8	9	Σ
Punkte (max)	10	10	10	10	10	10	10	10	10	90
Punkte (erreicht)										

Note:

Aufgabe 1: Kurzfragen

(10 Punkte)

(a) Argument vs. Parameter

(2 Punkte)

Was ist der Unterschied zwischen einem (*formalen*) *Parameter* und einem *Argument*?

(b) Exceptions

(2 Punkte)

Geben Sie zwei Gründe an, warum das Werfen von Exceptions gegenüber der Rückgabe von Fehler-Codes bevorzugt werden sollte.

(c) Call-by-value vs. Call-by-reference

(2 Punkte)

Wie unterscheidet sich *call-by-value* von *call-by-reference* und was ist die Konsequenz daraus?

(d) Unsichere Konversion

(2 Punkte)

Wann ist eine Konversion *sicher*?

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question (d).

(e) Kopierkontrolle

(2 Punkte)

Wenn Sie in einer Klasse keinen Kopier-Konstruktor deklarieren, generiert der Compiler einen Kopier-Konstruktor für Sie. Was macht dieser generierte Kopier-Konstruktor und wie heißt das Kopierkonzept, das damit umgesetzt wird?

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question (e).

Aufgabe 2: Fehlertypen

(10 Punkte)

Betrachten Sie die folgenden C++-Programme. Gehen Sie davon aus, dass am Anfang jedes Programms noch die Zeile

```
#include "std_lib_inc.h"
```

steht. Ansonsten sind die Programme vollständig. Jedes Programm verursacht entweder einen oder gar keinen Fehler. Kreuzen Sie entsprechend an.

Hinweis: Das Nicht-ankreuzen einer Teilaufgabe gibt 0 Punkte für diese Teilaufgabe; falsche Kreuze führen zu Punkteabzug. Die Gesamtaufgabe kann jedoch nicht weniger als 0 Punkte bringen.

	kein Fehler	Fehler zur...			Logikfehler
		Kompilierzeit	Linkzeit	Laufzeit	
<pre>int main() { int n = 42; int k = n * 10 return k; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>int main() { int* x = new int; *x = 42; cout << "Adresse: " << x << '\n'; return *x; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>int main() { int a = 0; int b = 7; b = a; // tausche a mit b a = b; cout << a << " " << b << '\n'; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>struct A { virtual void f() = 0; }; struct B : A { virtual void f() {} }; int main() { A a; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>int main() { vector<int> v = {1,2,3,4}; for (int i = 0; i < v.size(); ++i) { cout << v.at(v.size() - i) << '\n'; } // Ausgabe von v rueckwaerts }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>void f(); int main() { f(); }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>void erhoehe(int a) { a += 1; } int main() { int a = 0; erhoehe(a); cout << a << '\n'; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Aufgabe 3: Programmsemantik**(10 Punkte)**

Geben Sie die jeweilige Ausgabe der folgenden C++-Programme an. Sie können davon ausgehen, dass keine Compiler-Optimierung stattfindet.

(a) (3 Punkte)

```
#include "std_lib_inc.h"

int f(int& a, int b) {
    a += 2;
    return b + a;
}

int main()
{
    int x = 1;
    int y = 2;
    y = f(x, 3);
    f(y, 4);
    x = f(x, 5);
    cout << x << ' ' << y;
}
```

Ausgabe:

(b) (2 Punkte)

```
#include "std_lib_inc.h"

struct A {
    void f() const { cout << "R"; }
    void f() { cout << "X"; }
};

int main()
{
    cout << "A";
    A a;
    const A& b = a;
    a.f();
    b.f();
    cout << "B";
}
```

Ausgabe:

(c) (2 Punkte)

```
#include "std_lib_inc.h"

struct A {
    A() { cout << "0"; }
    ~A() { cout << "X"; }
    A(int) { cout << "1"; }
    A(string, int) {
        cout << "2";
    }
};

int main()
{
    A a;
    A b(2);
    A c {};
    A d("X1", 2);
}
```

Ausgabe:

(d) (3 Punkte)

```
#include "std_lib_inc.h"

struct A {
    void f() { cout << "F"; }
    virtual void g() { cout << "G"; }
};

struct B : public A {
    void f() { cout << "f"; }
    virtual void g() { cout << "g"; }
};

int main() {
    A a;
    B b;
    A& c = b;
    a.f(); a.g();
    b.f(); b.g();
    c.f(); c.g();
}
```

Ausgabe:

Aufgabe 4: Scopes

(10 Punkte)

(a) Scopes finden

(6 Punkte)

Markieren Sie alle Scopes im folgenden Programmcode und benennen Sie die Scope-Art. Sie können das entweder durch Umkreisen/Umklammern & Beschriften tun oder als Liste mit Zeilennummern.

```
1  #include "std_lib_inc.h"
2  int x = 0;
3  class A {
4      int x = 2;
5  };
6  using namespace std;
7  int main() {
8      int x = 1;
9      {
10         if(x == 3) ++x;
11         for(int i = 0; i < x; ++i) {
12             cout << "C++ == " << i << " Spass" << '\n';
13         }
14     }
15 }
```

(b) Scopes nachvollziehen

(4 Punkte)

Geben Sie die Ausgabe des folgenden C++-Programms an. Sie können davon ausgehen, dass keine Compiler-Optimierung stattfindet.

```
#include "std_lib_inc.h"

int x = 0;

void f(int& x) {
    x += ::x;
}

int main() {
    ++x;
    cout << x << ' ';
    int x = 2;
    cout << x << ' ';
    int xy = x;
    for(int x = 2*xy; x > 0; --x) {
        x -= 1;
        cout << x << ' ';
    }
    cout << x << ' ';
    f(x);
    cout << x << ' ';
}
```

Ausgabe:

Aufgabe 5: Rekursion

(10 Punkte)

Betrachten Sie die folgende Funktion $f: \mathbb{Z} \rightarrow \mathbb{Z}$.

$$f(x) = \begin{cases} f(\lfloor x/8 \rfloor - 1) + f(\lceil x/6 \rceil - 1) & x \geq 0 \text{ und gerade} \\ 2 \cdot f(x + 3) & x \geq 0 \text{ und ungerade} \\ 5 & x < 0 \end{cases}$$

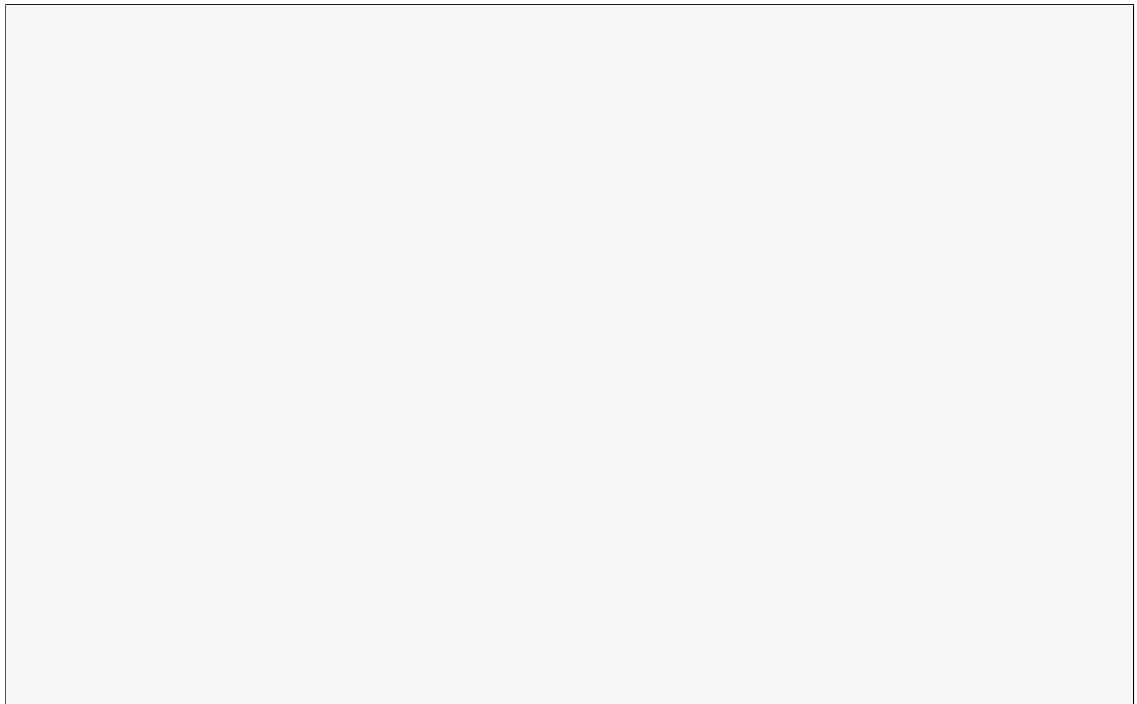
Hinweis: $\lfloor \dots \rfloor$ und $\lceil \dots \rceil$ bezeichnet Ab- bzw. Aufrunden.

(a) Rekursive Funktion

(6 Punkte)

Schreiben Sie eine rekursive C++-Funktion, die die obige Funktion berechnet. Sie dürfen hierfür keine Header-Dateien laden.

```
int f(int x) {
```

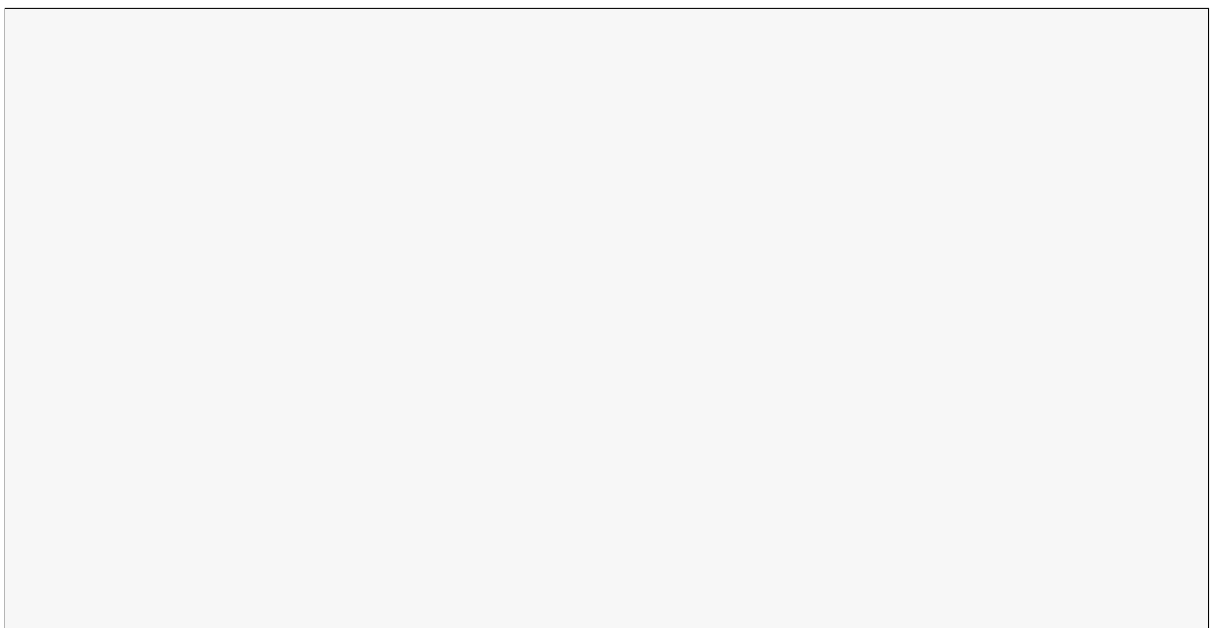


```
}
```

(b) Rekursive Berechnung

(4 Punkte)

Was ist $f(15)$ mit Rechenweg?



Aufgabe 6: Eingabe-Verarbeitung

(10 Punkte)

Schreiben Sie ein C++-Programm, das ganze Zahlen und das Gleichheitszeichen „=“ einlesen kann. Bei einer anderen Eingabe soll das Programm mit einer Fehlermeldung beendet werden. Bei Eingabe eines Gleichheitszeichens wird die Summe der vorher eingegeben Zahlen, gefolgt von einem Zeilenumbruch, ausgegeben und das Programm beendet.

Beispiel	Eingabe:	<input type="text" value="1"/>	<input type="text" value="2"/>	<input type="text" value="Enter"/>	<input type="text" value="3"/>	<input type="text" value="Enter"/>	<input type="text" value="-"/>	<input type="text" value="4"/>	<input type="text" value="Enter"/>	<input "="" type="text" value="="/>	<input type="text" value="Enter"/>	
	Interpretation:	12 + 3 + (-4) =										
	Ausgabe:	11 (gefolgt von einem Zeilenumbruch)										

```
#include "std_lib_inc.h"
```

Aufgabe 7: Stream-Modell

(10 Punkte)

Schreiben Sie einen Ausgabe-Operator für das folgende Struct Schlange:

```
#include "std_lib_inc.h"

enum class Muster {
    Gestreift, // Koerper besteht aus mehreren =
    Kariert,   // Koerper besteht aus mehreren #
    Gewellt   // Koerper besteht aus mehreren v
};

struct Schlange {
    string rufname;
    int laenge;
    Muster muster;
};
```

Die Ausgabe soll aus dem Rufnamen bestehen, gefolgt von einem Ausrufezeichen, einem Leerzeichen und einer grafischen Repräsentation der übergebenen Schlange. Diese grafische Repräsentation enthält `laenge` viele Zeichen, gewählt gemäß des `musters` der Schlange (für `laenge ≤ 0` entfällt die grafische Repräsentation). Die Ausgabe wird mit einem Zeilenumbruch beendet.

Beispiel

Übergeben: Schlange mit `rufname` „Hasso“, `laenge` 5 und `muster` Gestreift.

Ausgabe: „Hasso! =====“ (ohne Anführungszeichen), gefolgt von Zeilenumbruch.

```
#include "std_lib_inc.h"
```

Aufgabe 8: Listen

(10 Punkte)

Sei eine Datei `string_liste.h` gegeben. In dieser Datei wird eine Klasse deklariert, die eine doppelt-zeigerverkettete Liste von Strings repräsentiert:

```
#include "std_lib_inc.h"

struct Element {
    string wort;
    Element* vorgaenger;
    Element* nachfolger;

    Element(const string& w, Element* v, Element* n)
        : wort(w), vorgaenger(v), nachfolger(n) {}
};

class StringListe {
    Element* anfang;
    Element* ende;

public:
    StringListe() : anfang(nullptr), ende(nullptr) {};
    ~StringListe();

    void einfuegenVorne(const string& w);
};
```

Die graue Box unterhalb stellt die zugehörige Datei `string_liste.cpp` dar. Wir gehen davon aus, dass der Destruktor darin schon definiert ist. Implementieren Sie nun die Methode `einfuegenVorne`, die ein neues Listenelement mit `w` als `wort` am Anfang der Liste einfügt.

```
#include "string_liste.h"
```

Aufgabe 9: Templates

(10 Punkte)

Schreiben Sie ein Funktionstemplate namens `nurImErsten` mit einem Template-Parameter `T`. Diesem Funktionstemplate sollen zwei `std::vector<T>` per const-Referenz übergeben werden. Daraufhin soll es dann einen anderen `std::vector<T>` zurückgeben, welcher genau die Elemente des ersten Eingabe-vectors enthält, die nicht im zweiten Eingabe-vector vorkommen.

Beispiel

Übergeben: erster `std::vector<int>` mit Inhalt (1, 2, 3, 4, 3),
zweiter `std::vector<int>` mit Inhalt (2, 4, 5, 4, 2)

Ausgabe: `std::vector<int>` mit Inhalt (1, 3, 3)

```
#include "std_lib_inc.h"
```

Raum für Notizen:

Viel Erfolg!