

Einführung in die Programmierung

Klausur — WiSe 2024/25 — 25. Februar 2025

Haupttermin, Prüfungsnr. 23201019

Gruppe: begin

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	8	10	10	10	10	10	6	<b>84</b>
Punkte (erreicht)										

Note:

## Aufgabe 1: Kurzfragen

(10 Punkte)

Beantworten Sie die Fragen jeweils in 1–2 Sätzen.

### (a) C++-Bestandteile

(2 Punkte)

Was ist die Aufgabe des Linkers in C++?

### (b) Fachbegriffe Programmierung

(2 Punkte)

Was versteht man unter einem „Typ“ in der Programmierung?

### (c) Aggregat-Initialisierung

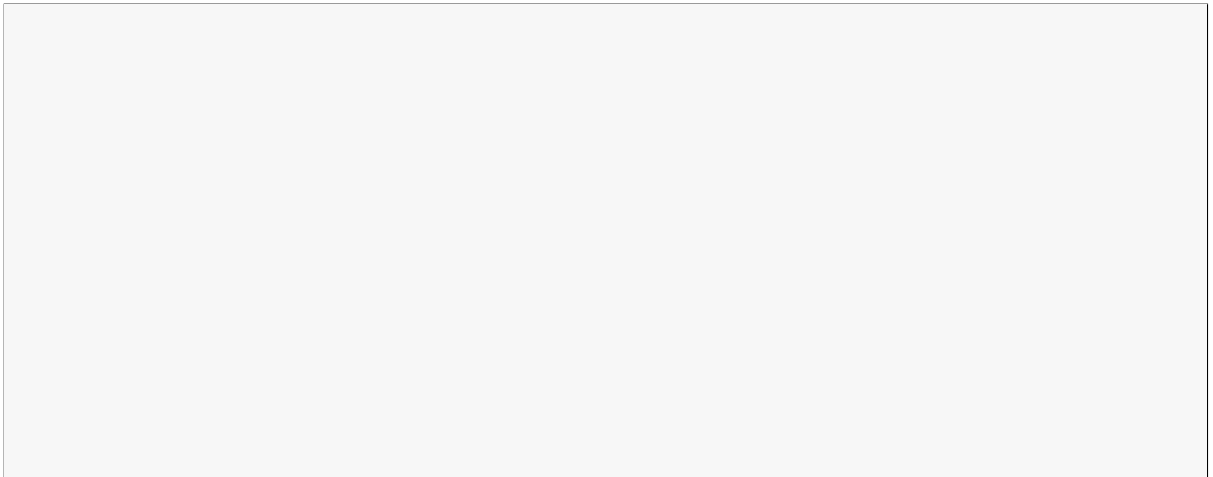
(2 Punkte)

Welche Voraussetzungen müssen erfüllt sein, damit eine Klasse in C++ mittels Aggregat-Initialisierung initialisiert werden kann?

**(d) RAII**

(2 Punkte)

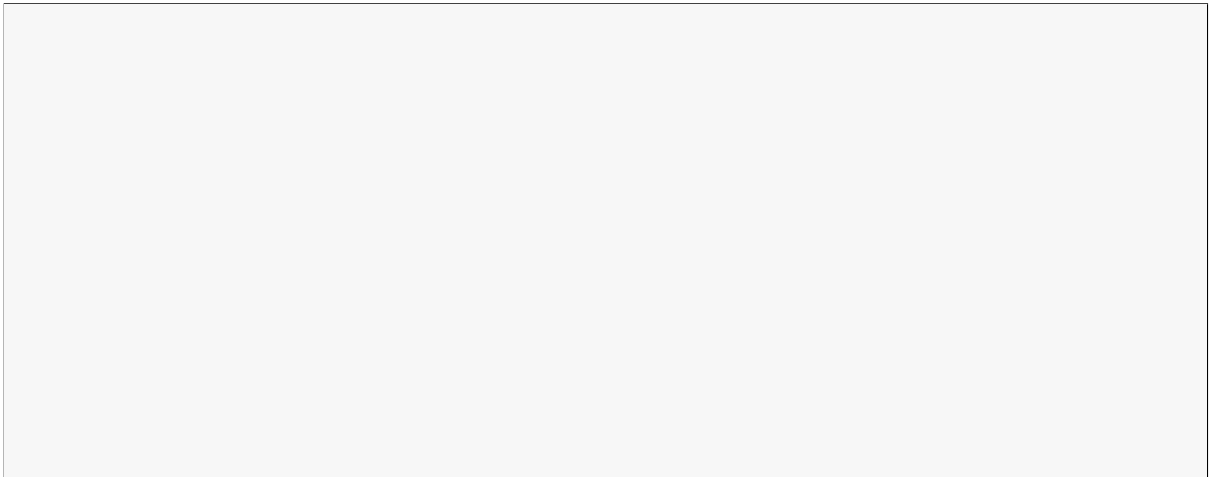
Welches Problem soll RAII (Resource Acquisition is Initialization) verhindern und was ist die Grundidee hinter RAII?

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

**(e) Vererbung**

(2 Punkte)

Was ist eine abstrakte Klasse?

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. Gehen Sie zudem von den typischen Speichergrößen der Typen aus. 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>// Addiert zwei ints und gibt das // Ergebnis zuruck. int add(int a, int b);  int main() {     // Gib die Summe von 3 + 4 aus.     cout &lt;&lt; add(3, 4) &lt;&lt; '\n';     return 0; }  int add(int a, int b) {     return a + b; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>int main() {     int variable = "Hallo, Welt";     // Gib "Hallo, Welt" aus.     cout &lt;&lt; variable &lt;&lt; '\n';     return 0; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>struct MyClass {     int value;     explicit MyClass(const int&amp; number)         : value(number * 1000) {} };  int main() {     int my_number = 1073741824;     MyClass instance(my_number);     // Gib den Wert von instance aus     cout &lt;&lt; instance.value &lt;&lt; "\n";     return 0; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

	kein Fehler	Fehler zur...			Logikfehler
		Kompilierzeit	Linkzeit	Laufzeit	
<pre>// Addiert 5 auf den Wert des Ziels // eines uebergebenen Zeigers und gibt // den Wert zurueck. <b>template</b>&lt;typename T&gt; T add_5(T* ptr) {     <b>return</b> *ptr + 5; }  <b>int</b> main() {     <b>int</b>* int_ptr;     cout &lt;&lt; add_5(int_ptr) &lt;&lt; '\n';     <b>return</b> 0; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>// Gibt das Ergebnis der Gleitkommadivision // numerator / denominator zurueck. <b>double</b> divide(<b>int</b> numerator, <b>int</b> denominator) {     <b>if</b>(denominator == 0) {         <b>return</b> 0;     }     <b>return</b> numerator / denominator; }  <b>int</b> main() {     cout &lt;&lt; divide(10, 3) &lt;&lt; "\n";     <b>return</b> 0; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

### Aufgabe 3: Programmsemantik

(8 Punkte)

Geben Sie exakt die jeweilige Ausgabe der folgenden C++-Programme an. Sie können davon ausgehen, dass keine Compiler-Optimierung stattfindet und jedes mal die Header-Datei „std\_lib\_inc.h“ eingebunden ist.

(a) (2 Punkte)

```
int main() {
    int x = 8;
    int j = 5;
    if(x >= j)
        if(x == j)
            cout << "OP";
    else
        cout << "QB";
    cout << "000";
    return 0;
}
```

Ausgabe:

(b) (2 Punkte)

```
int main() {
    vector<int> werte
        = {1, 2, 3, 4, 5, 6};
    int* a = &werte[0];
    int* b = a + 3;
    int* c = a + b[1] - a[2];
    int* d = &c[*c - 1];

    cout << *a << *b << *c << *d;
    return 0;
}
```

Ausgabe:

(c) (2 Punkte)

```
struct A {
    A() { cout << "X"; }
    ~A() { cout << "Y"; }
    void f() { cout << "W"; }
    void f() const { cout << "V"; }
};

void f(const A& a) {
    a.f();
}

int main() {
    A a;
    a.f();
    f(a);
    return 0;
}
```

Ausgabe:

(d) (2 Punkte)

```
struct O {
    void f() {
        cout << "O::f()";
    }
};

struct Q : public O {
    void f() {
        cout << "Q::f()";
    }
};

int main() {
    O* o = new Q {};
    o->f();
    return 0;
}
```

Ausgabe:

## (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
3 using namespace std;
4
5 class MeineKlasse
6 {
7     int wert_;
8 public:
9     int wert() { return wert_; }
10 };
11
12 int main()
13 {
14     int x = 5;
15     for(int i = 5; i > 0; i--)
16     {
17         cout << x << '\n';
18     }
19 }
```

**(b) Scopes nachvollziehen**

(4 Punkte)

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

```
#include "std_lib_inc.h"

int x = 7;

namespace X {
    int x = -2;
}

int main()
{
    int x = 0;
    int x_ = 2;
    for(int x = x_; x > 0; --x)
    {
        x_ = x + X::x - ::x;
        X::x = ::x - x_;
    }
    cout << x << x_ << X::x << ::x - 2;
}
```

Ausgabe:

### Aufgabe 5: Rekursion

(10 Punkte)

Betrachten Sie die folgende Funktion  $A: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ .

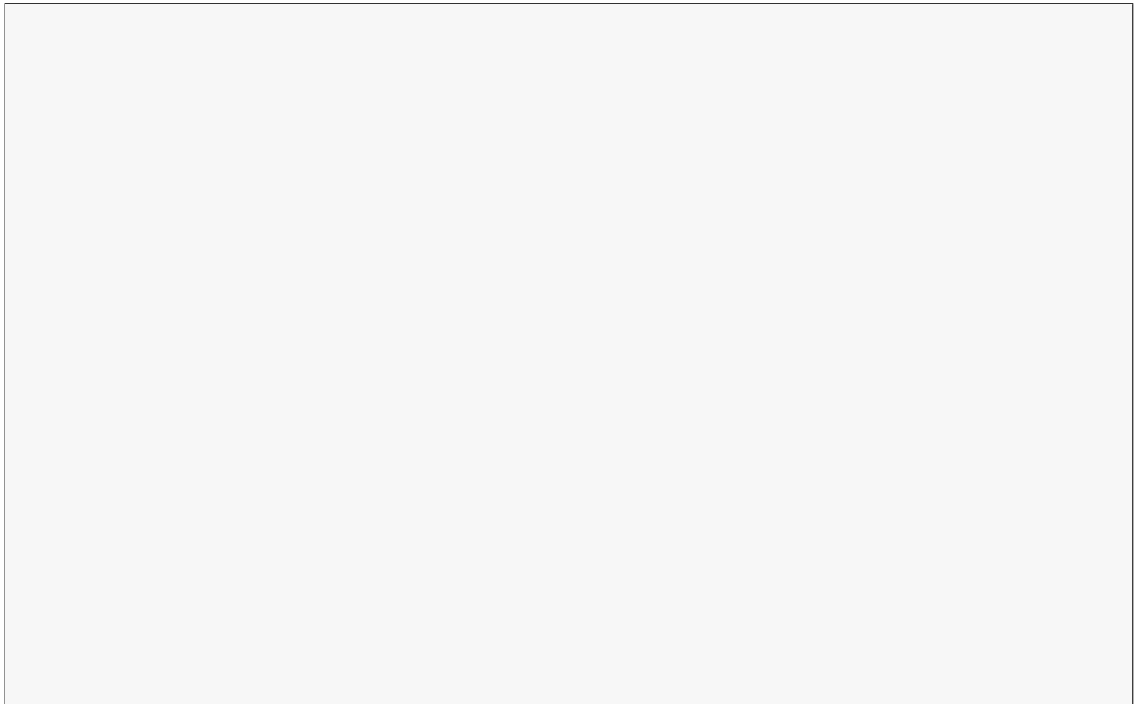
$$A(n, m) = \begin{cases} n + 1 & m = 0 \\ A(1, m - 1) & m > 0 \text{ und } n = 0 \\ A(A(n - 1, m), m - 1) & \text{sonst} \end{cases}$$

#### (a) Rekursive Funktion

(6 Punkte)

Schreiben Sie eine rekursive C++-Funktion, die die obige Funktion berechnet. Sie brauchen sich um Überläufe nicht zu kümmern und können davon ausgehen, dass nur  $n \geq 0$  und  $m \geq 0$  übergeben werden. Sie dürfen hierfür keine Header-Dateien laden.

```
int A(int n, int m) {
```

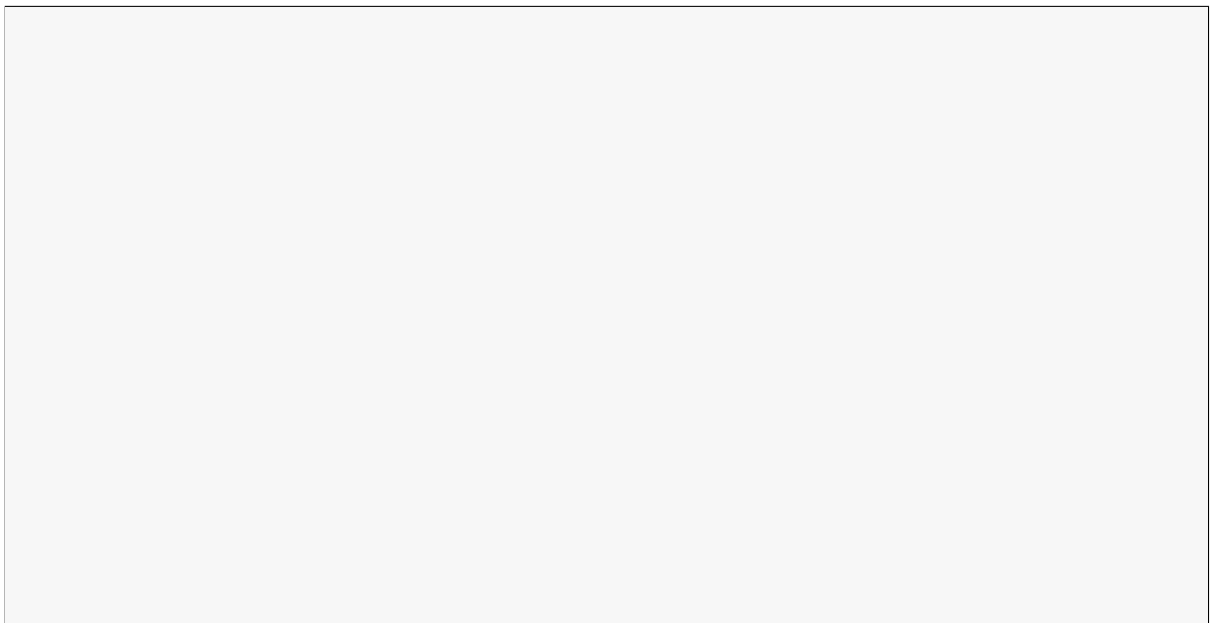


```
}
```

#### (b) Rekursive Berechnung

(4 Punkte)

Was ist  $A(2, 1)$  mit Rechenweg?



## Aufgabe 6: Stream-Modell

(10 Punkte)

Schreiben Sie einen Eingabe-Operator für das folgende Struct Bruchzahl:

```
#include "std_lib_inc.h"

struct Bruchzahl {
    int zaehler;        // Zaehler des Bruchs
    int nenner;        // Nenner des Bruchs
};
```

Es gilt: Der Zähler „zaehler“ ist eine ganze Zahl und der Nenner „nenner“ eine positive ganze Zahl. Das Austauschformat dieser Klasse sieht wie folgt aus: Ist der Bruch negativ, so folgt zuerst ein „-“. Danach kommt der (absolute) Wert des Zählers als ganze Zahl, gefolgt von dem Zeichen „/“ und als letztes der Wert des Nenners als ganze Zahl. Whitespace zwischen den Zahlen und dem „/“ darf ignoriert werden. Alle anderen Eingaben müssen verworfen werden. Berichten Sie Fehler gemäß des Stream-Modells.

**Beispiel**

**Eingabe:** „-7/5“ (ohne Anführungszeichen), gefolgt von Zeilenumbruch.

**Bruchzahl:** zaehler == -7 und nenner == 5.

## Aufgabe 7: Listen

(10 Punkte)

Eine einfach verkettete Liste ist definiert durch das folgende Struct:

```
#include "std_lib_inc.h"

struct Eintrag {
    string wort;
    Eintrag* nachfolger;

    Eintrag(const string& w, Eintrag* n = nullptr)
        : wort(w), nachfolger(n) {}
};

Eintrag* vorne_einfuegen(Eintrag* liste, Eintrag* element);
Eintrag* loesche_alles(Eintrag* liste);
```

Implementieren Sie die Funktionen `vorne_einfuegen` und `loesche_alles`: Die Funktion `vorne_einfuegen` erhält einen Zeiger `liste` auf den ersten Eintrag einer Liste und einen Zeiger `element` auf ein neues Listen-Element. Das neue Element soll vorne in die Liste eingefügt werden. Sind beide übergebenen Zeiger `nullptr`, soll `nullptr` zurückgegeben werden. Ist nur ein übergebener Zeiger `nullptr`, soll der jeweils andere Zeiger zurückgegeben werden. Im Erfolgsfall wird ein Zeiger auf das neue erste Element der Liste zurückgegeben. Die Funktion `loesche_alles` löscht alle Elemente in der Liste und gibt `nullptr` zurück.

## Aufgabe 8: Templates

(10 Punkte)

Erstellen Sie ein Funktionstemplate `count_elements`, das die Anzahl der Elemente zwischen zwei Iteratoren `begin` und `end` in einem C++-Container berechnet und als `int` zurückgibt. Verwenden Sie dabei das Iterator-Entwurfsmuster und nur Operationen aus dem Interface eines Iterators, insbesondere ohne `std::distance` zu nutzen. Schreiben Sie den vollständigen Quelltext des Funktionstemplates auf.

*Hinweis: Sie dürfen davon ausgehen, dass `begin` nicht hinter `end` liegt.*

Beispiel

**Übergeben:** Begin- und End-Iterator eines `std::vector<int>` mit Inhalt (4, 5, 2, 2, 4, 3)  
**Rückgabe:** 6

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

## Aufgabe 9: Vererbung

(6 Punkte)

Betrachten Sie das folgende Teilprogramm:

```
#include "std_lib_inc.h"

struct Kreditkarte {
    void bezahlung_starten(double wert) const;
};

struct Lastschrift {
    void bezahlung_starten(double wert) const;
};

void bezahlen(const BezahlArt& art, double wert) {
    art.bezahlung_starten(wert);
}
```

Erstellen Sie ein Interface `BezahlArt`, damit die Funktion `bezahlen` sowohl `Kreditkarte`- als auch `Lastschrift`-Objekte als ersten Parameter akzeptieren kann. Passen Sie die beiden bestehenden Klassen `Kreditkarte` und `Lastschrift` so an, dass sie dieses Interface implementieren. Die Funktion `bezahlen` darf nicht verändert werden. Es genügt die neuen Definitionen von `BezahlArt`, `Kreditkarte` und `Lastschrift` aufzuschreiben. Sie brauchen die Methoden in `Kreditkarte` und `Lastschrift` nicht zu implementieren, die Deklarationen genügen.

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

*Raum für Notizen:*