PR3 - Übungsblatt B.1

Prof. Dr. Holger Peine
Hochschule Hannover
Fakultät IV – Abteilung Informatik
Raum 1H.2.60, Tel. 0511-9296-1830
Holger.Peine@hs-hannover.de

Thema

Einführung in C++, Klassen und Objekte

Termin

Ihre Arbeitsergebnisse zu diesem Übungsblatt führen Sie bitte bis zum 18.12.2025 vor. Stellen Sie aber zumindest die Lösungen für die Aufgaben 1 und 2 schon bis zum 11.12. vor.

1 new, delete, Referenzparameter (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 9.d

Schreiben Sie in C++14 eine Funktion `createArray`, die ein Ganzzahl-Array gewünschter Länge dynamisch allokiert und als Rückgabewert zurück gibt. Die Funktion soll einen Parameter haben, der die gewünschte Anzahl der Array-Elemente angibt.

Schreiben Sie eine zweite Funktion `releaseArray`, die ein Ganzzahl-Array als Referenzparameter entgegen nimmt. Die Funktion gibt das Array wieder frei und setzt die übergebene Zeigervariable auf `nullptr`. Vergleichen Sie die Implementierung mit der Lösung zu Aufgabe 5.4 ("`free0`").

Verwenden Sie nicht `malloc/free`, sondern die neuen Möglichkeiten von C++.

2 Einfache Personen-Klasse (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 10.a

Schreiben Sie in C++14 eine Klasse `Person` mit folgenden Membern:

- Private Attribute für einen Namen (`string`) und ein Geburtsjahr (`int`)
- Öffentlicher Standard-Konstruktor, der einen leeren Namen und das Geburtsjahr 0 setzt
- Öffentlicher Konstruktor, bei dem Name und Geburtsjahr übergeben und gesetzt werden
- Öffentliche Methode `print()`, die die Daten der Person in der Form "*name* ist *geburtsjahr* geboren." auf die Standard-Ausgabe ausgibt (mit den Mitteln von C++, nicht etwa mit `printf`)

Schreiben Sie außerdem eine `main()`-Methode, die einen (C-) Array von zwei Personen jeweils mit Namen und Geburtsjahr anlegt und anschließen in einer Schleife für alle Elemente des Arrays `print` aufruft. Verwenden Sie die neue C++-Schleife ("range based for loop", "foreach-Schleife").
Teilen Sie Ihr Programm in drei Dateien `Person.cpp`, `Person.h` und `main.cpp` auf. Achten Sie darauf, die nötigen `#include` 's und `using namespace` zu benutzen, damit alle benötigten externen Symbole auch bekannt sind.

Für alle nachfolgenden Aufgaben gilt:

L.10	Lesen Sie zunächst bis einschließlich Abschnitt L.10.1.1 im Dokument PR3_10_L.pdf im Vorlesungsordner.

3 Einkaufswagen (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 10.b

Schreiben Sie in C++14 eine Klasse `ShoppingCart`, die eine Menge von einzukaufenden Artikeln enthält, und eine Klasse `CartItem`, die einen einzukaufenden Artikel zusammen mit der gewünschten Anzahl repräsentiert (z. B. „vier Dosen Hundefutter“).

Die Klasse `CartItem` soll intern die Anzahl der gekauften Artikel und den Einzelpreis speichern. Gewünschte Methoden der Klasse `CartItem` sind:

Methode	Beschreibung
Konstruktor	Initialisiert einen einzukaufenden Artikel mit den Daten Name (Typ `string`), Anzahl (ganzzahlig) und Preis-pro-Einheit (Fließkommazahl)
getCost	Liefert den Gesamtpreis des Artikels. D. h. bei z. B. vier Dosen Hundefutter zu je 3,20 liefert diese Methode 12,80.
get- und set-Methoden für Name, Anzahl und Preis-pro-Einheit	Liest und setzt den entsprechenden Wert.

Die Klasse `ShoppingCart` soll intern eine Ansammlung der `CartItem`s verwenden und folgende Methoden anbieten:

Methode	Beschreibung
Konstruktor	initialisiert einen leeren Einkaufswagen
add	Fügt ein `CartItem`-Objekt hinzu
getTotalCost	Liefert den Gesamtpreis aller Artikel im Einkaufswagen
getNumberOfItems	Liefert die Anzahl der enthaltenen `CartItem`-Objekte
getItem	Liefert den i-ten Item

Generell zu beachten: Objekte als Parameter sollen nach Möglichkeit als Referenzen übergeben werden. Dasselbe gilt für als Rückgabewert gelieferte Objekte.

Für die interne Verwaltung der `CartItem`s bietet sich der Einsatz der Klasse `vector<CartItem>` (oder `vector<CartItem*>`[1] - machen Sie sich den Unterschied klar!) aus der C++-Standardbibliothek an (Namespace `std`, Headerdatei `<vector>`). Diese Klasse ist ein sog. Klassen-Template (entspricht ungefähr der generischen Klasse `ArrayList<E>` in Java). Der Elementtyp wird wie in Java in spitzen Klammern angegeben.
Beispiel:

```
vector<int> zahlen;
zahlen.push_back(6);  // 6 am Ende von zahlen einfügen
zahlen.push_back(4);
zahlen.push_back(7);
int summe;
for (auto zahl: zahlen) {
  summe += zahl;
```

[1] … aber nicht `vector<CartItem&>` : *vector verlangt von seinem Elementtyp, dass er zuweisbar ist, aber Referenzen sind nicht zuweisbar: https://stackoverflow.com/questions/33144419/stl-container-of-references-c11*

```
}
```

Mehr Informationen zur `vector`-Klasse finden Sie z. B. hier: https://cplusplus.com/reference/vector/vector/. Beachten Sie, dass die `push_back`-Methode Kopien der übergebenen Elemente (was auch immer der genaue Elementtyp ist – vielleicht ja nur ein Zeiger oder eine Referenz) im Vektor speichert; überlegen Sie sich, welche Auswirkungen das auf Ihr Programm hat und wie dies vom genauen Elementtyp abhängt, den Sie wählen.

Schreiben Sie schließlich ein Client-Programm, das folgende Artikel einkauft und anschließend den Gesamtpreis anfordert:

- 6 Dosen Hundefutter zu je 3,20
- 4 Packungen Kekse zu je 1,59
- 1 Flasche Milch zu 0,69
- 3 Glas Erdbeermarmelade zu je 2,19

Außerdem soll Ihr Clientprogramm wenigstens eine Zuweisung eines `CartItem` an ein anderes enthalten (das wird im Zusammenhang mit späteren Aufgaben eine Rolle spielen!).

Denken Sie daran, Ihr Programm in mehrere Quellmodule (am besten eines je Klasse und eines für das Hauptprogramm) aufzuteilen. Schreiben Sie in die Header-Dateien nur die Klassendeklaration (einschließlich Methodeklarationen/-schnittstellen) ohne jegliche Methodendefinitionen/-implementierungen.

4 Kassenbon (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 10.b

Schreiben Sie Programmcode zur Ausgabe eines Kassenbons für einen gefüllten Einkaufswagen. Implementieren Sie dazu je eine `toString`-Methode in `CartItem` und in `ShoppingCart`. Letztere soll einen `string` folgenden Inhalts zurückgeben:

```
  6 x Hundefutter                          3,20       19,20
  4 x Kekse                                1,59        6,36
  1 x Milch                                0,69        0,69
  3 x Erdbeermarmelade                     2,19        6,57

Summe:                                                32,82
```

Die Methode `toString` der Klasse `CartItem` liefert hiervon jeweils eine der ersten vier Zeilen als Rückgabewert zurück. Die Methode `toString` der Klasse `ShoppingCart` baut aus diesen Teilstrings den gesamten Kassenbon zusammen. Sie dürfen davon ausgehen, dass ein Artikelname höchstens 30 Zeichen lang ist.

Für die formatierte Ausgabe in einen `ostringstream` können Sie folgende Möglichkeiten nutzen (dazu müssen Sie `iomanip` und `sstream` inkludieren; weitere Informationen bei Bedarf in der Leseaufgabe L10.4.2):

- *`os << fixed`: stellt die Ausgabe von Gleitkommazahlen auf feste Anzahl von Nachkommastellen ein.*
- *`os << setprecision(n)`: stellt die Anzahl der Nachkommastellen für alle nachfolgenden Ausgaben auf n.*
- *`os << setw(n)`: stellt für die unmittelbar folgende Ausgabe die minimale Ausgabebreite auf n Zeichen ein.*
- *`os << left`: Stellt linksbündige Ausgabe innerhalb der eingestellten Ausgabebreite ein.*
- *`os << right`: Stellt rechtsbündige Ausgabe innerhalb der eingestellten Ausgabebreite ein.*

5 Konstruktion und Übergabe von Objekten (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 10.b

Was gibt das folgende Programm aus und warum? Füllen Sie dazu alle leeren Zellen in folgender Tabelle aus. Falls Sie Probleme mit dieser Aufgabe haben, hilft es vielleicht, sich die Quizaufgaben aus 10b und ihre Lösungen nochmals anzusehen.

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

class A {
public:
  int data;
  int id;
  static int counter;

  A() {
    data = 0;
    id = counter++;
  }

  A(const A& other) {
    data = other.data;
    id = counter++;
  }
};

int A::counter {0};

void foo(A a1, A& a2) {
    ++a2.data;
    a1 = a2;
    cout << a1.data << a1.id << a2.data << a2.id << endl;
}

void bar(A a1, A& a2) {
    ++a1.data;
    a2 = a1;
    cout << a1.data << a1.id << a2.data << a2.id << endl;
}

int main(void) {
  A x, y;
  foo(x, y);
  bar(x, y);
  cout << x.data << x.id << y.data << y.id << endl;
  return 0;
}
```

Zeitpunkt	Wert von `x` bzw. `a1`		Wert von `y` bzw. `a2`	
	`data`	`id`	`data`	`id`
Nach Konstruktion von x und y	0	0	0	1
Nach Eintritt in `foo` (direkt hinter `{`)				
Vor Rückkehr aus `foo` (direkt vor `}`) (entspricht 1. Ausgabe)				
Nach Eintritt in `bar`				

Vor Rückkehr aus `bar` (entspricht 2. Ausgabe)				
Nach Rückkehr aus `bar` (entspricht 3. Ausgabe)				

6 Unerwartete Objekte (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 10.b

L.10

Lesen Sie zunächst mindestens den Abschnitt L.10.1.5 im Dokument PR3_10_L.pdf im Vorlesungsordner.

Die Ausgabe des folgenden Programms (als Quelldatei auf dem Server in den Anlagen zu diesem Übungsblatt) entspricht nicht den Erwartungen:

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

class Kreditgeber {
public:
  Kreditgeber(const string& pName)
    : name {pName}, vergebenesKreditVolumen {0} { }
  void auszahlen(const int betrag) {
    cout << name << " zahlt Kredit über " << betrag <<
      " EUR aus." << endl;
    vergebenesKreditVolumen += betrag;
  }
  void zurueckzahlen(int betrag) {
    cout << name << " bekommt Kredit über " << betrag <<
      " EUR zurückgezahlt." << endl;
    vergebenesKreditVolumen -= betrag;
  }
private:
  const string name;
  int vergebenesKreditVolumen;
};


class Kredit {
public:
  Kredit(Kreditgeber& kreditgeber, const int betrag);
  ~Kredit();
private:
  Kreditgeber& kreditgeber;
  const int betrag;
};

Kredit::Kredit(Kreditgeber& kreditgeber, const int betrag)
  : kreditgeber {kreditgeber}, betrag {betrag}     {
    kreditgeber.auszahlen(betrag);
}

Kredit::~Kredit() {
  kreditgeber.zurueckzahlen(betrag);
}


void nutze(Kredit kredit) {
  cout << "Kreditnehmer nutzt den Kredit." << endl;
```

```
}
int main(void) {
  Kreditgeber sparkasseHannover {"Sparkasse Hannover"};
  Kredit meinKredit {sparkasseHannover, 5000}; // Bekomme Kredit
  nutze(meinKredit);                           // Nutze Kredit
  // Rueckzahlung automatisch durch Destruktor von meinKredit :-)
  return 0;
}
```

Die Ausgabe lautet:

```
Sparkasse Hannover zahlt Kredit über 5000 EUR aus.
Kreditnehmer nutzt den Kredit.
Sparkasse Hannover bekommt Kredit über 5000 EUR zurückgezahlt.
Sparkasse Hannover bekommt Kredit über 5000 EUR zurückgezahlt.
```

Warum wird der Kredit zweimal zurückgezahlt? Wie sollte man das Programm korrigieren?

7 C++ und Speicherklassen (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 10.b

a) Welche Ausgabe produziert das folgende Programm? Versuchen Sie diese Aufgabe zu lösen, bevor Sie das Programm ausführen. Beachten Sie dabei besonders der unterschiedliche Verhalten der beiden `foo`-Aufrufe.

```
#include <iostream>
using std::cout;
using std::endl;

class MyInt {
public:
  MyInt(int i): i(i) {
    cout << "MyInt(" << i << ") constructed." << endl;
  }
  MyInt(const MyInt& m): i(m.i) {
    cout << "MyInt(" << i << ") constructed." << endl;
  }
  ~MyInt() {
    cout << "MyInt(" << i << ") destructed." << endl;
  }
private:
  int i;
};

void foo(MyInt m) {
  cout << "foo invoked." << endl;
}

MyInt global {1};

int main(void) {
  MyInt* heap;
  cout << "Right before MyInt local" << endl;
  MyInt local {2};
  heap = new MyInt {3};
  foo(local);
```

```
  foo(MyInt{4});
  // foo(new MyInt{5});  // Compiler error - why?
  cout << "Cleaning up now..." << endl;
  delete heap;
  return 0;
}
```

b) Warum wäre die mit "Compiler error" markierte Zeile ein Fehler? Machen Sie sich klar, was der Unterschied zwischen dieser Zeile und der Zeile direkt darüber ist.

8 Verbesserung der Ort-Klasse aus der Vorlesung (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 10.c

a) Die Klasse `Ort` aus der Vorlesung (Folien 10-6/7) verwendet ein Attribut vom Typ `char*`, das sie mit `malloc` und `free` verwaltet – das ist eigentlich C und kein C++. Stellen Sie die Klasse auf die entsprechenden Typen und Mechanismen von C++ um. Inwiefern wird die Klasse dadurch einfacher?
b) Müssen Sie noch weiteren Code hinzufügen, um Objekte dieser Klasse korrekt kopieren zu können (z.B. für die Parameterübergabe mit call-by-value)? Die gleiche Frage für die Klasse aus der Vorlesung?

9 Item-Ids ergänzen (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 10.d

L.10

Lesen Sie zunächst mindestens den Abschnitt L.10.1.2+4 im Dokument PR3_10_L.pdf im Vorlesungsordner.

Ergänzen Sie in `CartItem` ein ganzzahliges Attribut `itemId`. Das Attribut soll direkt bei der Konstruktion mit einem Wert belegt werden, und zwar erhält jede Instanz von `CartItem` eine eindeutige `itemId`. Realisieren Sie diese Funktionalität durch ein Klassenattribut für die letzte vergebene Id.

Ergänzen Sie schließlich in `ShoppingCart` eine Methode `getItemIds`, die ein Array genau der benötigten Größe mit allen Item-Ids dynamisch anlegt und zurückgibt. Geben Sie die Ids gefolgt von den Artikelnamen im Hauptprogramm aus (benutze Sie dazu in `ShoppingCart ShoppingCart::getItem()` und `CartItem::getName()`) und vergessen Sie nicht, das dynamisch erzeugte Array in der `main`-Funktion nach der Ausgabe wieder freizugeben. Verwenden Sie nicht `malloc/free`, sondern die neuen Möglichkeiten von C++.