



C++ - Modul 06

C++-Würfe

Zusammenfassung:

Dieses Dokument enthält die Übungen des Moduls 06 aus den C++ Modulen.

Version: 6.2

Inhalt

I	Einführung	2
II	Allgemeine Regeln	3
III	Zusätzliche Regel	5
IV	Übung 00: Umwandlung von Skalartypen	6
V	Übung 01: Serialisierung	9
VI	Übung 02: Identifizieren Sie den realen Typ	10
VII	Einreichung und Peer-Evaluierung	11

Kapitel I

Einleitung

C++ ist eine allgemeine Programmiersprache, die von Bjarne Stroustrup als Weiterentwicklung der Programmiersprache C, oder "C mit Klassen", entwickelt wurde (Quelle: [Wikipedia](#)).

Das Ziel dieser Module ist es, Sie in die **objektorientierte Programmierung** einzuführen. Dies wird der Ausgangspunkt für Ihre C++-Reise sein. Viele Sprachen werden empfohlen, um OOP zu lernen. Wir haben uns für C++ entschieden, da es von Ihrem alten Freund C abgeleitet ist. Da es sich um eine komplexe Sprache handelt, und um die Dinge einfach zu halten, wird Ihr Code dem Standard C++98 entsprechen.

Wir sind uns bewusst, dass modernes C++ in vielerlei Hinsicht anders ist. Wenn Sie also ein kompetenter C++-Entwickler werden wollen, liegt es an Ihnen, nach den 42 Common Core weiterzugehen!

Kapitel II

Allgemeine

Vorschriften

Kompilieren

- Kompilieren Sie Ihren Code mit C++ und den Flags -Wall -Wextra -Werror
- Ihr Code sollte sich trotzdem kompilieren lassen, wenn Sie das Flag -std=c++98 hinzufügen

Formatierungs- und Benennungskonventionen

- Die Übungsverzeichnisse werden folgendermaßen benannt: ex00, ex01, ... , exn
- Benennen Sie Ihre Dateien, Klassen, Funktionen, Mitgliedsfunktionen und Attribute wie in den Richtlinien gefordert.
- Schreiben Sie Klassennamen im Format **UpperCamelCase**. Dateien, die Klassencode enthalten, werden immer nach dem Klassennamen benannt. Zum Beispiel: Klassenname.hpp/Klassenname.h, Klassenname.cpp, oder Klassenname.tpp. Wenn Sie also eine Header-Datei haben, die die Definition einer Klasse "BrickWall" enthält, die für eine Ziegelmauer steht, lautet ihr Name BrickWall.hpp.
- Wenn nicht anders angegeben, müssen alle Ausgabemeldungen mit einem Zeilenumbruch abgeschlossen und auf der Standardausgabe ausgegeben werden.
- *Auf Wiedersehen Norminette!* In den C++-Modulen ist kein Kodierungsstil vorgeschrieben. Sie können Ihrem Lieblingsstil folgen. Aber denken Sie daran, dass ein Code, den Ihre Mitbewerber nicht verstehen, auch nicht benotet werden kann. Geben Sie Ihr Bestes, um einen sauberen und lesbaren Code zu schreiben.

Erlaubt/Verboten

Sie programmieren nicht mehr in C. Zeit für C++! Deshalb:

- Sie dürfen fast alles aus der Standardbibliothek verwenden. Anstatt sich also an das zu halten, was Sie bereits kennen, wäre es klug, so viel wie möglich die C++-ähnlichen Versionen der C-Funktionen zu verwenden, an die Sie gewöhnt sind.
- Sie können jedoch keine andere externe Bibliothek verwenden. Das bedeutet, dass C++11 (und abgeleitete Formen) und Boost-Bibliotheken verboten sind. Die

folgenden Funktionen sind ebenfalls verboten: `*printf()`, `*alloc()` und `free()`. Wenn Sie sie verwenden, wird Ihre Note 0 sein und das war's.

- Beachten Sie, dass, sofern nicht ausdrücklich anders angegeben, der using-Namensraum <ns_name> und Freundschaftswörter sind verboten. Andernfalls wird Ihre Note -42 sein.
- **Du darfst die STL nur in den Modulen 08 und 09 verwenden.** Das bedeutet: keine **Container** (Vektor/Liste/Map/usw.) und keine **Algorithmen** (alles, was den <algorithm>-Header erfordert) bis dahin. Andernfalls wird Ihre Note -42 sein.

Einige Designanforderungen

- Speicherlecks treten auch in C++ auf. Wenn Sie Speicher zuweisen (mit der Funktion new Schlüsselwort), müssen Sie **Speicherlecks** vermeiden.
- Von Modul 02 bis Modul 09 muss der Unterricht in der **orthodoxen kanonischen Form** gestaltet werden, es sei denn, es ist ausdrücklich etwas anderes angegeben.
- Jede Funktionsimplementierung in einer Header-Datei (mit Ausnahme von Funktionsschablonen) bedeutet 0 für die Übung.
- Sie sollten in der Lage sein, jeden Ihrer Header unabhängig von anderen zu verwenden. Daher müssen sie alle Abhängigkeiten einschließen, die sie benötigen. Allerdings müssen Sie das Problem der doppelten Einbindung vermeiden, indem Sie **Include-Guards** hinzufügen. Andernfalls wird Ihre Note 0 sein.

Lies mich

- Sie können bei Bedarf zusätzliche Dateien hinzufügen (z. B. um Ihren Code aufzuteilen). Da diese Aufgaben nicht von einem Programm überprüft werden, können Sie dies gerne tun, solange Sie die vorgeschriebenen Dateien einreichen.
- Manchmal sehen die Richtlinien einer Übung kurz aus, aber die Beispiele können Anforderungen aufzeigen, die nicht ausdrücklich in den Anweisungen stehen.
- Lesen Sie jedes Modul vollständig durch, bevor Sie beginnen! Wirklich, tun Sie es.
- Bei Odin, bei Thor! Benutze deinen Verstand!!!



Sie werden eine Menge Klassen implementieren müssen. Das kann mühsam sein, es sei denn, Sie können in Ihrem Lieblings-Texteditor Skripte schreiben.



Sie haben einen gewissen Spielraum bei der Durchführung der Übungen. Halten Sie sich jedoch an die vorgeschriebenen Regeln und seien Sie nicht faul. Sie würden eine Menge nützlicher Informationen verpassen! Zögern Sie nicht, sich über theoretische Konzepte zu informieren.

Kapitel III

Zusätzliche

Regelung

Die folgende Regel gilt für das gesamte Modul und ist nicht optional.

Für jede Übung muss die Typumwandlung mit einer bestimmten Art von Guss gelöst werden.

Ihre Wahl wird während der Verteidigung überprüft.

Kapitel IV

Übung 00: Umwandlung von Sklartypen

	Übung 00
Umwandlung von skalaren Typen	
Turn-in-Verzeichnis : <i>ex00/</i>	
Einzureichende Dateien: Makefile, *.cpp, *.{h, hpp}	
Erlaubte Funktionen: Jede Funktion, die einen String in einen int, einen float oder einen double umwandelt. Diese Funktion ist hilfreich, erfüllt aber nicht alle Aufgaben.	

Schreiben Sie eine Klasse `ScalarConverter`, die nur eine statische Methode "convert" enthält, die als Parameter eine String-Repräsentation eines C++-Literal in seiner gebräuchlichsten Form annimmt und seinen Wert in der folgenden Reihe von Sklartypen ausgibt:

- char
- int
- Schwimmer
- doppelt

Da diese Klasse überhaupt nichts speichern muss, darf sie von den Benutzern nicht instanzierbar sein.

Außer für char-Parameter wird nur die dezimale Schreibweise verwendet.

Beispiele für Char-Literale: 'c', 'a', ...

Der Einfachheit halber sei darauf hingewiesen, dass nicht anzeigbare Zeichen nicht als Eingaben verwendet werden sollten. Wenn eine Umwandlung in ein Zeichen nicht anzeigbar ist, wird eine informative Meldung ausgegeben.

Beispiele für int-Literale: 0, -42, 42...

Beispiele für Float-Literale: 0.0f, -4.2f, 4.2f...

Sie müssen auch diese Pseudoliterale behandeln (Sie wissen schon, für die Wissenschaft): -inff, +inff

und nanf.

Beispiele für Doppel-Literale: 0.0, -4.2, 4.2...

Sie müssen auch diese Pseudoliterale behandeln (Sie wissen schon, zum Spaß): -inf, +inf und nan.

Schreiben Sie ein Programm, um zu testen, ob Ihre Klasse wie erwartet funktioniert.


Sie müssen zunächst den Typ des als Parameter übergebenen Literals ermitteln, es von String in seinen eigentlichen Typ konvertieren und es dann **explizit** in die drei anderen Datentypen umwandeln. Zeigen Sie schließlich die Ergebnisse wie unten dargestellt an.

Wenn eine Konvertierung keinen Sinn ergibt oder überläuft, zeigen Sie eine Meldung an, um den Benutzer darüber zu informieren, dass die Typkonvertierung unmöglich ist. Fügen Sie alle Kopfzeilen ein, die Sie benötigen, um numerische Grenzen und spezielle Werte zu behandeln.

```
/konvertieren 0
Zeichen: Nicht
anzeigbar int: 0
Float: 0.0f
Double:
0.0
./convert nan
char: unmöglich
int: unmöglich
float: nanf
double: nan
./convert 42.0f
Zeichen: "42"
int: 42
float: 42.0f
double: 42.0
```


Kapitel V

Übung 01: Serialisierung

	Übung : 01
	Serialisierung
Turn-in-Verzeichnis :	g
anzureichende Dateien:	Makefile, *.cpp,
	*.{h, hpp}

Verbotene Funktionen : Keine

Implementieren Sie eine Klasse Serializer, die vom Benutzer in keiner Weise initialisierbar sein wird, mit den folgenden statischen Methoden:

```
uintptr_t serialize(Data* ptr);
```

Sie nimmt einen Zeiger und konvertiert ihn in den vorzeichenlosen Ganzzahltyp `uintptr_t`.

```
Data* deserialize(uintptr_t raw);
```

Sie nimmt einen vorzeichenlosen Integer-Parameter und konvertiert ihn in einen Zeiger auf Data.

Schreiben Sie ein Programm, um zu testen, ob Ihre Klasse wie erwartet funktioniert.


Sie müssen eine nicht leere (d. h. mit Datenelementen versehene) Datenstruktur erstellen.

Verwenden Sie `serialize()` auf die Adresse des Data-Objekts und übergeben Sie dessen Rückgabewert an `deserialize()`. Stellen Sie dann sicher, dass der Rückgabewert von `deserialize()` mit dem ursprünglichen Zeiger übereinstimmt.

Vergessen Sie nicht, die Dateien Ihrer Datenstruktur einzureichen.

Kapitel VI

Übung 02: Identifizieren Sie den realen Typ

	Übung : 02
Identifizieren Sie den tatsächlichen Typ	
Turn-in-Verzeichnis : <i>ex02/</i>	
Einzureichende Dateien: Makefile, *.cpp, *.{h, hpp}	
Verbotene Funktionen : <code>std::typeinfo</code>	

Implementieren Sie eine Base-Klasse, die nur einen öffentlichen virtuellen Destruktor hat. Erstellen Sie drei leere Klassen **A**, **B** und **C**, die öffentlich von Base erben.



Diese vier Klassen müssen nicht in der orthodoxen kanonischen Form gestaltet werden.

Implementieren Sie die folgenden Funktionen:

```
Base * generate(void);
```

Es instanziiert zufällig A, B oder C und gibt die Instanz als Base-Zeiger zurück. Sie können für die Implementierung der Zufallsauswahl alles verwenden, was Sie möchten.

```
void identify(Base* p);
```

Sie gibt den tatsächlichen Typ des Objekts aus, auf das p zeigt: "A", "B" oder "C".

```
void identify(Base& p);
```

Sie gibt den tatsächlichen Typ des Objekts aus, auf das p zeigt: "A", "B" oder "C". Die Verwendung eines Zeigers innerhalb dieser Funktion ist verboten.

Die Aufnahme des `typeinfo`-Headers ist verboten.

Schreiben Sie ein Programm, um zu testen, ob alles wie erwartet funktioniert.

Kapitel VII

Einreichung und Peer-Evaluierung

Reichen Sie Ihre Arbeit wie gewohnt in Ihrem Git-Repository ein. Nur die Arbeit in Ihrem Repository wird während der Verteidigung bewertet. Zögern Sie nicht, die Namen Ihrer Ordner und Dateien zu überprüfen, um sicherzustellen, dass sie korrekt sind.



16D85ACC441674FBA2DF65190663E136253996A5020347143B460E2CF3A3784D794B
104265933C3BE5B62C4E062601EC8DD1F82FEB73CB17AC57D49054A7C29B5A5C1D8
2027A997A3E24E387