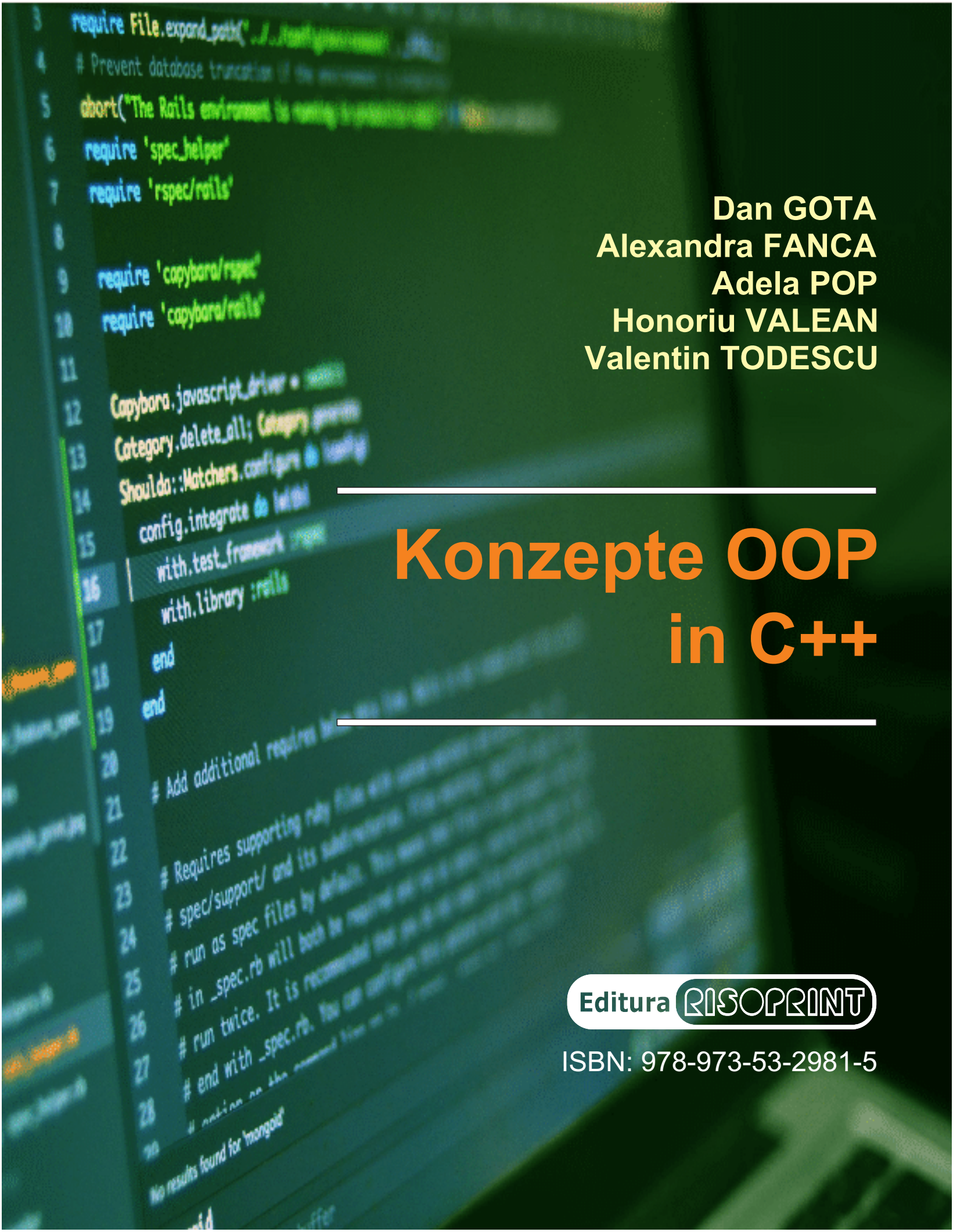

Dan GOTA
Alexandra FANCA
Adela POP
Honoriu VALEAN
Valentin TODESCU

Konzepte OOP in C++

Editura RISOPRINT

ISBN: 978-973-53-2981-5

*Editura **RISOPRINT** este recunoscută de **C.N.C.S.***
***(Consiliul Naţional al Cercetării Ştiinţifice)**.*
www.risoprint.ro** **www.cncs-uefiscdi.ro

ISBN: 978-973-53-2981-5

Konzepte OOP in C++

Dan GOTA
Alexandra FANCA
Adela POP
Honoriu VALEAN
Valentin TODESCU

Director editură: GHEORGHE POP

Dan GOTA
Alexandra FANCA
Adela POP
Honoriu VALEAN
Valentin TODESCU

Konzepte OOP in C++

EDITURA RISOPRINT

CLUJ-NAPOCA, 2023

Kapitel 0: Eine kurze Einführung in C++

C++ ist eine Programmiersprache der mittleren Stufe, die von Bjarne Stroustrup seit 1979 in den Bell Labs entwickelt wurde. C++ läuft auf einer Vielzahl von Plattformen wie Windows, Mac OS und verschiedenen Versionen von UNIX. Dieses C++-Tutorial beschreibt auf einfache und praktische Weise C++-Konzepte für Anfänger und fortgeschrittene Softwareentwickler.

C++ ist ein MUST für Studenten und Berufstätige, die eine Karriere als Softwareentwickler anstreben. Nachstehend sind einige der wichtigsten Vorteile des Erlernens von C++ aufgeführt:

a) Die Programmiersprache C++ ist eng mit der Hardware verbunden, so dass Sie die Möglichkeit haben, auf einer niedrigen Ebene zu programmieren, was Ihnen viel Kontrolle in Bezug auf die Speicherverwaltung, eine bessere Leistung und schließlich eine robuste Softwareentwicklung bietet;

b) Die C++-Programmierung vermittelt Ihnen ein klares Verständnis der objektorientierten Programmierung;

c) C++ ist die in der Anwendungs- und Systemprogrammierung am häufigsten verwendete Programmiersprache.

Wozu ist C++ gut?

Wie hier bereits erwähnt wurde, ist C++ eine der am häufigsten verwendeten Programmiersprachen. Sie ist in fast allen Bereichen der Softwareentwicklung vertreten. Wir werden hier einige von ihnen auflisten:

a) Entwicklung von Anwendungssoftware - Die C++-Programmierung wurde bei der Entwicklung fast aller wichtigen Betriebssysteme wie Windows, Mac OSX und Linux eingesetzt. Neben Betriebssystemen wurde auch der Kern vieler Browser wie Mozilla Firefox und Chrome in C++ geschrieben. C++ wurde auch bei der Entwicklung des populärsten Datenbanksystems, MySQL, verwendet.

b) Entwicklung von Programmiersprachen - C++ wurde in großem Umfang bei der Entwicklung neuer Programmiersprachen wie C#, Java, JavaScript, Perl, C Shell UNIX, PHP und Python sowie Verilog usw. verwendet.

c) Computergestützte Programmierung - C++ ist der beste Freund der Wissenschaftler aufgrund seiner hohen Geschwindigkeit und Rechneneffizienz.

d) Spieleentwicklung - C++ ist extrem schnell, was den Programmierern die Möglichkeit gibt, prozedurale Programmierung für CPU-intensive Funktionen durchzuführen und eine bessere Kontrolle über die Hardware zu haben, weshalb es in der Entwicklung von Spiel-Engines weit verbreitet ist.

e) Eingebettete Systeme - C++ wird in großem Umfang bei der Entwicklung von medizinischen und technischen Anwendungen eingesetzt, z. B. bei Software für MRT-Geräte, moderne CAD/CAM-Systeme usw.

C++ wurde entwickelt, um die Objektorientierung in die Programmiersprache C zu bringen, die bereits eine der anspruchsvollsten Programmiersprachen auf dem Markt ist. Bei der rein objektorientierten Programmierung geht es darum, Code zu schreiben, die ein Objekt mit bestimmten Eigenschaften und Methoden erzeugt. Bei der Erstellung von C++-Modulen sind wir bestrebt, das gesamte Universum als Objekte zu betrachten. Ein Auto zum Beispiel ist ein Objekt mit bestimmten Merkmalen wie Farbe, Anzahl der Türen, Hubraum usw. Dazu gehören auch Funktionen wie Beschleunigung, Bremsen und so weiter.

Die objektorientierte Programmierung basiert auf einigen grundlegenden Konzepten. Im Folgendem werden wir diese grundlegenden Konzepte kurz vorstellen

Objekt

Dies ist das grundlegende Element der objektorientierten Programmierung. Das heißt, dass sowohl die Daten als auch die Datenoperationsfunktionen als Objekt/Objekte gruppiert sind.

Klasse

Ein Plan/Schema für ein Objekt wird bei der Definition einer Klasse festgelegt. Dieser beschreibt keine Daten, sondern definiert, was der Klassenname bedeutet, d. h. was ein Klassenobjekt ist und welche Operationen mit ihm durchgeführt werden können.

Abstraktion

Datenabstraktion ist definiert als die Bereitstellung wichtiger Informationen für die Außenwelt bei gleichzeitiger Ausblendung von Hintergrunddetails, d. h. die Darstellung der notwendigen Informationen in einem Programm, ohne dass die Einzelheiten angezeigt werden.

Ein Datenbanksystem zum Beispiel verbirgt bestimmte Aspekte der Datenspeicherung, -erstellung und -pflege. In ähnlicher Weise geben C++-Klassen viele Methoden nach

außen hin frei, ohne interne Informationen über diese Funktionen oder Daten preiszugeben.

Verkapselung

Bei der Kapselung werden die Daten und die Funktionen, die mit ihnen arbeiten am selben Ort gespeichert. Bei prozeduralen Sprachen ist es nicht immer offensichtlich, welche Funktionen mit welchen Variablen arbeiten, aber die objektorientierte Programmierung bietet Ihnen einen Rahmen, um die relevanten Daten und Funktionen in einem Objekt zusammenzufassen.

Vererbung

Die Wiederverwendung der Code ist eine der vorteilhaftesten Eigenschaften der objektorientierten Programmierung. Wie der Name schon sagt, ist die Vererbung der Prozess der Erstellung einer neuen Klasse aus einer bestehenden Klasse, die als Basisklasse bekannt ist. Die neue Klasse wird als abgeleitete Klasse bezeichnet. Dies ist ein wichtiger Begriff in der objektorientierten Programmierung, da er dazu beiträgt, die Größe des Codes zu reduzieren.

Polymorphismus

Polymorphismus ist die Fähigkeit, einen Operator oder eine Funktion auf mehr als eine Weise zu verwenden oder Operatoren oder Funktionen unterschiedliche Bedeutungen oder Funktionen zu geben. Der Begriff "poly" bedeutet "viele". Polymorphismus bezieht sich auf eine einzelne Funktion oder einen einzelnen Operator, der je nach Kontext auf verschiedene Weise verwendet werden kann.

Überladung

Zum Polymorphismus gehört auch die Idee des Überladens. Man spricht von Überladung, wenn ein bestehender Operator oder eine Funktion auf einen neuen Datentyp angewendet wird.

Kapitel 1: Klassen und Objekte in C++

Eine Klasse spezifiziert die Form eines Objekts, indem sie die Darstellung von Daten und Methoden zur Veränderung dieser Daten in einem einzigen Paket zusammenfasst. Die Mitglieder einer Klasse sind die Daten und Methoden, aus denen die Klasse besteht.

a) Definition von Klassen in C++

Wenn Sie eine Klasse anlegen, erstellen Sie im Wesentlichen einen Plan für einen Datentyp. Dieser beschreibt keine Daten, sondern legt fest, was der Klassenname bedeutet, d. h. woraus ein Klassenobjekt besteht und welche Aktionen mit ihm durchgeführt werden können.

Eine Klassendefinition beginnt mit dem Schlüsselwort class, dann folgt der Klassenname und schließlich der Körper der Klasse, der von einem Klammerpaar eingeschlossen wird. Nach einer Klassendefinition muss ein Semikolon oder eine Reihe von Anweisungen stehen.

Zum Beispiel haben wir den Datentyp Box mit dem Schlüsselwort class wie folgt definiert --

b) Definition von Objekten in C++

Eine Klasse dient als Modell für Dinge, daher besteht ein Objekt im Wesentlichen aus einer solchen. Die Deklaration von Objekten einer Klasse erfolgt auf die gleiche Weise wie die Deklaration von Variablen der Grundtypen.

c) Zugriff auf Mitgliederdaten

Der Direktzugriffsoperator "." von Mitgliedern kann verwendet werden, um auf öffentliche Datenmitglieder von Objekten einer Klasse zugreifen zu können. Zur

d) Klassen und Objekte im Detail

Bislang haben wir die Grundlagen von Klassen und Objekten in C++ vorgestellt. Es gibt noch ein paar weitere faszinierende Themen im Zusammenhang mit C++-Klassen und -Objekten, die wir in den folgenden Abschnitten betrachten werden.

d.1) Klassenzugehörigkeitsfunktionen

Eine Funktion als Klassenmitglied ist eine Funktion, die, wie jede andere Variable ihre Definition oder ihren Prototyp in der Klassendeklaration hat. Sie hat Zugriff auf alle Mitglieder der Klasse, für die es Mitglied ist und kann auf jedes Objekt dieser Klasse einwirken.
Lassen Sie uns eine Mitgliedfunktion verwenden, um auf die Mitglieder einer zuvor erstellten Klasse zuzugreifen, anstatt sie direkt anzusprechen.

d.2) Modifikatoren für den Klassenzugang

Ein Klassenmitglied kann öffentlich, privat oder geschützt sein. Einschliesslich werden auch die Mitglieder als privat erklärt. Eines der wichtigsten Merkmale der objektorientierten Programmierung ist das Data Hiding, das es den Programmmethoden ermöglicht, den direkten Zugriff auf die interne Darstellung eines Klassentyps zu vermeiden. Die ausgewiesenen öffentlichen, privaten und geschützten Bereiche innerhalb des Klassenkörpers legen die Zugriffsbeschränkungen für Klassenmitglieder fest. Zugriffsspezifizierer sind die folgenden Begriffe: public, private und protected. Eine Klasse kann zahlreiche Teile haben, die als public, protected oder private gekennzeichnet sind.

d.2.1) Öffentliche Mitglieder

Auf ein öffentlichen Mitglied kann man von überall innerhalb eines Programms zugreifen und nicht nur innerhalb der Klasse. Wie das folgende Beispiel zeigt, können Sie den Wert von öffentlichen Variablen setzen und empfangen, ohne Mitgliedsfunktionen zu verwenden - *<für denZugang zum Code siehe Anhang3

d.2.2) Private Mitglieder

Außerhalb der Klasse kann man auf eine private Mitgliedsvariable oder Funktion nicht zugreifen oder sie sogar einsehen. Auf private Mitglieder kann nur über Klassenfunktionen und Freundesfunktionen zugegriffen werden.
Standardmäßig sind alle Mitglieder einer Klasse privat; zum Beispiel ist „Breite” ein privates Mitglied der Klasse und das bedeutet, dass wenn Sie einen Mitglied nicht kennzeichnen, er als privates Mitglied angenommen wird.

d.2.3) Geschützte Mitglieder

Eine geschützte Member-Variable oder -Funktion ist mit einem privaten Member vergleichbar, hat aber den zusätzlichen Vorteil, dass sie über abgeleitete Klassen, d. h. Unterklassen, zugänglich ist.
Im nächsten Kapitel werden Sie mehr über abgeleitete Klassen und Vererbung erfahren. Sehen Sie sich zunächst das folgende Beispiel an, in dem wir eine untergeordnete Klasse CutieMica aus einer übergeordneten Klasse Cutie erstellt haben.

Das folgende Beispiel ist identisch mit dem vorhergehenden, mit der Ausnahme, dass jede Memberfunktion in der abgeleiteten Klasse CutieMica Zugriff auf den Width Member hat.

d.3) Konstruktore und Destruktore:

Wenn ein neues Objekt einer Klasse erstellt wird, wird ein Klassenkonstruktor aufgerufen. Ein Destruktor ist eine bestimmte Funktion, die ausgeführt wird, wenn ein Objekt erstellt und dann gelöscht wird.
Ein Konstruktor hat den gleichen Namen wie die Klasse und gibt keinen Wert zurück, nicht einmal void. Konstruktore sind nützlich, wenn es darum geht, Anfangswerte für Mitgliedsvariablen zu setzen und werden im folgenden Beispiel erläutert.

d.3.1) Parametrisierte Konstruktore

Ein Standardkonstruktor hat keine Parameter, aber Sie können sie hinzufügen, wenn Sie benötigt werden. Wie im folgenden Beispiel gezeigt wird, können Sie damit einen Anfangswert für ein Objekt festlegen, wenn es erstellt wird.

d.3.2) Verwendung von Initialisierungslisten zur Initialisierung von Feldern

Um Felder in einem parametrisierten Konstruktor zu initialisieren, verwenden Sie die folgende Syntax:

d.3.3) Destruktor einer Klasse

Ein Destruktor ist eine spezifische Mitgliedsfunktion einer Klasse, die immer dann aufgerufen wird, wenn ein Objekt dieser Klasse den Gültigkeitsbereich verlässt oder wenn der delete-Ausdruck auf einen Verweis auf das Objekt dieser Klasse angewendet wird.

Ein Destruktor hat denselben Namen wie die Klasse, trägt jedoch ein Tilde-Symbol (~) vorangestellt und kann keine Werte zurückgeben oder Parameter annehmen. Der Destruktor ist nützlich, um Ressourcen vor dem Beenden eines Programms freizugeben, z. B. um Dateien zu schließen und Speicher freizugeben.
Das folgende Beispiel zeigt, wie man einen Destruktor verwendet.

d.3.4) Kopierkonstruktor

Der Kopierkonstruktor ist ein Konstruktor, der ein Objekt erzeugt, indem er es mit einem zuvor konstruierten Objekt der gleichen Klasse initialisiert wurde.

- Initialisieren eines Objekts aus einem anderen Objekt desselben Typs mit Hilfe des Kopierkonstruktors.
- Um ein Objekt als Parameter an eine Funktion zu übergeben, kopieren Sie es.
- Rückgabe eines Objekts aus einer Funktion durch Kopieren.

d.4) Friend funktionen (friend)

Eine friend Funktion hat vollen Zugriff auf private und geschützte Klassenmitglieder. Die friend Funktion einer Klasse wird außerhalb des Geltungsbereichs der Klasse deklariert. Sie hat aber Zugriff auf alle privaten und geschützte Mitglieder der Klasse. Friends sind keine Mitgliedsfunktionen, obwohl Prototypen für Fiend Funktionen in der Klassenspezifikation erscheinen. Eine Funktion, eine Funktionsvorlage oder eine Mitgliedsfunktion kann ein friend sein, genau wie eine Klasse oder eine Klassenvorlage; in diesem Fall ist die gesamte Klasse und ihre friend Mitglieder. Um eine Funktion als friend einer Klasse zu bezeichnen, verwenden Sie das Schlüsselwort friend vor dem Funktionsprototyp in der Klassendeklaration, wie unten gezeigt wird.

d.5) Der pointer “this”

Jedes Objekt hat einen eindeutigen pointer der sich auf das eigentliche Objekt bezieht.
In C++ hat jedes Objekt Zugriff auf seine eigene Adresse durch einen wesentlichen pointer, der als pointer „this“ bekannt ist. Alle Mitgliedsfunktionen nehmen diesen pointer als Standardargument. Dadurch kann in einer Mitgliedsfunktion auf das aufzurufende Objekt verwiesen werden.
Die friends sind keine Mitglieder einer Klasse, daher gibt es keinen pointer in Friend-Funktionen. Dieses Indikator ist nur für Mitgliederfunktionen verfügbar.
Zum besseren Verständnis des Begriffs pointer „this“ sei folgendes Beispiel angeführt:

* für den Zugang zum Code siehe Anhang 6

d.6) Statische Mitglieder einer Klasse

Datenmitglieder und Funktionsmitglieder einer Klasse können beide als statisch deklariert werden. Das Schlüsselwort *static* kann verwendet werden, um Klassenmitglieder statisch zu machen. Wenn wir ein Klassenmitglied als statisch bezeichnen, geben wir an, dass die statische Komponente nur einmal dupliziert wird, egal wie viele Klassenobjekte erzeugt werden.

Alle Objekte der Klasse haben einen statisches Mitglied. Wenn keine zusätzliche Initialisierung vorhanden ist, werden alle statischen Daten bei der Erstellung des ersten Objekts auf Null gesetzt. Wir können das Schlüsselwort nicht in die Klassendeklaration aufnehmen, aber wir können es außerhalb der Klassendeklaration initialisieren, wie im folgenden Beispiel gezeigt wird, indem wir die statische Variable neu deklarieren und den Auflösungsoperator *scope*: verwenden, um festzustellen, zu welcher Klasse sie gehört.
Versuchen wir das folgende Beispiel, um das Konzept der statischen Datenelemente zu verstehen -
* für den Zugang zum Code siehe Anhang 7

d.7) Statische Mitgliedsfunktionen

Sie können ein Funktionsmitglied als statisch machen, indem Sie es als solches definieren. Dies macht ihn unabhängig von einem bestimmten Objekt in der Klasse. Selbst wenn es keine Instanzen der Klasse gibt, kann eine statische Mitgliedsfunktion aufgerufen werden, und statische Funktionen sind einfach über den Klassennamen und den Domänenauflösungsoperator zugänglich.

Außerhalb der Klasse kann eine statische Mitgliedsfunktion nur auf statische Datenmitglieder, andere statische Mitgliedsfunktionen und alle anderen Funktionen zugreifen.

Statische Mitgliedsfunktionen sind von der Klasse abgegrenzt und haben keinen Zugriff auf diese Klassenreferenz. Sie können eine statische Mitgliedsfunktion verwenden, um zu prüfen, ob bestimmte Klassenobjekte erzeugt wurden oder nicht.

Kapitel 2: Vererbung in C++

Die Idee der Vererbung ist eine der wichtigsten Ideen in der objektorientierten Programmierung. Die Vererbung ermöglicht es uns, eine Klasse in Bezug auf eine

andere Klasse zu definieren, was die Entwicklung und Wartung von Anwendungen erleichtert. Dies ermöglicht auch die Wiederverwendung von Codefunktionen und eine kurze Implementierungszeit.

Anstatt bei der Einrichtung einer Klasse völlig neue Datenelemente und Methoden zu entwickeln, kann der Programmierer angeben, dass die neue Klasse die Elemente einer bestehenden Klasse erben soll. Die alte Klasse wird als Basisklasse bezeichnet, während die neue Klasse als abgeleitete Klasse bezeichnet wird.

Die Idee der Vererbung setzt eine Ist-ein-Beziehung um: z. Bsp. Ein Säugetier ist ein Tier, und ein Hund ist ein Säugetier, also ist ein Hund ein Tier usw.

a) **Grundlegende und abgeleitete Klassen**

Kapitel 3: Überladen (Operatoren und Funktionen) in C++

Funktionsüberladung und Operatorenüberladung sind Begriffe, die in C++ verwendet werden, um die Möglichkeit zu beschreiben, mehrere Definitionen für einen Funktionsnamen oder Operator im selben Feld anzugeben.

Eine überladene Anweisung ist eine Anweisung, die denselben Namen hat wie eine zuvor deklarierte Anweisung in derselben Domäne, aber beide Anweisungen haben separate Parameter und deutlich unterschiedliche Definitionen (Implementierung).

Wenn Sie eine überladene Funktion oder einen überladenen Operator aufrufen, vergleicht der Compiler die Argumenttypen, die Sie zum Aufrufen der Funktion oder des Operators verwendet haben und zwar mit den in den Definitionen angegebenen Parametertypen, um festzustellen, welche Definition zu verwenden ist.

Bei der Überladungsauflösung wird die am besten geeignete überladene Funktion oder der am besten geeignete Operator ermittelt.

a) Funktionsüberladung in C++

In ein und demselben Bereich können Sie mehrere Definitionen für denselben Funktionsnamen haben. Die Typen und/oder die Anzahl der Argumente in der Argumentliste müssen in jeder Funktionsdeklaration unterschiedlich sein. Es ist nicht möglich, Funktionsdeklarationen, die sich nur im Rückgabetyp unterscheiden, zu überladen.

Kapitel 4: Polymorphismus in C++

Polymorphismus bezieht sich auf die Tatsache, dass etwas in mehr als einer Form existiert. Polymorphismus tritt in der Regel auf, wenn es eine Hierarchie von Klassen gibt, die durch Vererbung verbunden sind.
Polymorphismus in C++ bezieht sich auf die Tatsache, dass je nach Art des Objekts, das eine Mitgliedsfunktion aufruft, eine andere Funktion ausgeführt wird.
Betrachten Sie das folgende Szenario in dem eine Basisklasse von zwei zusätzlichen Klassen abgeleitet wird.

*für den Zugang zum Code siehe Anhang 19

Kapitel 5: C++-Interfacestellen (Abstrakte Klassen)

Ein Interface erklärt das Verhalten oder die Fähigkeiten einer C++-Klasse, ohne sich auf eine bestimmte Implementierung dieser Klasse festzulegen.
Abstrakte Klassen werden zur Implementierung von C++-Schnittstellen verwendet. Diese abstrakten Klassen sind nicht mit der Datenabstraktion zu verwechseln, einem Konzept, das die Implementierungsdetails von den verbundenen Daten trennt.
Eine Klasse wird abstrakt, wenn mindestens eine ihrer Funktionen als rein virtuelle Funktion erklärt wird. Der Ausdruck "= 0" in der Erklärung einer rein virtuellen Funktion lautet wie folgt:

Kapitel 6: Datenkapselung in C++

Die folgenden zwei wesentlichen Komponenten sind in allen C++-Programmen vorhanden:

- Programmanweisungen (Code) sind die Teile eines Programms, die Aktionen ausführen. Sie werden als Funktionen bezeichnet.
- Programmdaten - sind die Programminformationen, die durch Programmfunktionen beeinflusst werden.

Kapselung ist ein Begriff aus der objektorientierten Programmierung, der die Daten und die Funktionen, die sie verwalten, miteinander verbindet und beide vor externen Eingriffen und Missbrauch schützt. Der entscheidende OOP-Begriff des Data Hiding wurde aus der Datenkapselung geboren.

Anhänge

Anhang 1 – Anhang 20