- Vererbung ist nur in Verbindung mit Klassen, die durch den class-key class oder struct eingeleitet werden möglich; es können struct-Ableitungen von class-Klassen und umgekehrt gebildet werden.
- Vererbung erzeugt einen speziellen Klassentyp auf der Grundlage eines anderen Klassentyps, ein Objekt einer abgeleiteten Klasse ist ein erweitertes/spezialisiertes Objekt der Basisklasse.
- In der abgeleiteten Klasse werden die zusätzlichen Daten beschrieben.
- Jede Instanz der abgeleiteten Klasse enthält alle Member der Basisklasse.
- Private Member der Basisklasse sind für die abgeleiteten Klassen nicht sichtbar, obwohl ein Objekt einer abgleiteten Klasse alle Member der Basisklasse enthält.
- Um die Daten der Basisklasse zu erreichen, müssen public-Memberfunktionen der Basisklasse verwendet werden.
- Die Beziehungen zwischen Basis- und abgeleiteten Klassen lassen sich in einem Klassendiagramm darstellen.
- Für eine Konvertierung von einem Pointer auf ein Objekt der Basisklasse in einen Pointer auf ein Objekt einer abgeleiteten Klasse ist ein cast notwendig. Eine solche Konvertierung ist jedoch gefährlich, weil nicht gesichert ist, daß alle Member der abgeleiteten Klasse existieren.
- Eine Konvertierung eines Pointers auf ein Objekt einer abgeleiteten Klasse auf ein Objekt der Basisklasse ist problemlos möglich..
- Die Benutzung der Basisklasse kann vor allem für Listentechniken sinnvoll sein.

Redefinition von Memberfunktionen

- Memberfunktionen können in abgeleiteten Klassen redefiniert werden.
- Mit Hilfe des scope-Operator können die Memberfunktionen der Basisklasse aufgerufen werden.
- Über den Typ eines Objekts wird automatisch die richtige Funktion ausgewählt.
- Wird eine Memberfunktion über einen Pointer aufgerufen, so bestimmt der Typ des Pointers, welche Funktion ausgeführt wird, nicht der tatsächliche Typ des Objektes.
- Wird auf ein Objekt mit einem Pointer der Basisklasse verwiesen, so sind die Memberfunktionen der Basisklasse verfügbar.

Virtuelle Funktionen

- Eine virtuelle Funktion ist eine Memberfunktion, von der man weiß, daß sie überladen werden wird von Memberfunktionen abgeleiteter Klassen.
- Ruft man eine virtuelle Funktion über den Pointer der Basisklasse, so wird die Memberfunktion der entsprechenden abgeleiteten Klasse, die dem tatsächlichen Typ des Objektes zugehört, ausgeführt.
- Eine virtuelle Funktion wird über das Schlüsselwort 'virtual' in der Klassendeklaration gekennzeichnet.
- Die Möglichkeit, eine Memberfunktion eines Objektes ohne dessen genauen Typ zu kennen, aufrufen zu können, nennt man Polymorphie.
- Als Konsequenz ergibt sich, daß man nicht alle abgeleiteten Klassen einer Basisklasse kennen muß, um mit der Klasse sinnvoll arbeiten zu können.

Dynamische Bindung

- Der Compiler muß einen Code generieren, der es gestattet, die wirklich auszuführende Funktion erst zur Laufzeit des Programms zuzuordnen, diese Technik wird dynamisches Binden genannt.
- Dynamische Bindung erlaubt, das Verhalten bereits übersetzten Codes variabel zu halten.
- Diese Technik gestattet es, weitere Ableitungen einer Basisklasse in extra Quellfiles zu formulieren und nur diese zu übersetzen und zu den anderen Moduln zu linken. Sie sind dann in Verbindung mit der Basisklasse genauso benutzbar, wie die bereits existierenden Ableitungen.

Implementation dynamischer Funktionen

- Virtuelle Funktionen benötigen sehr wenig Overhead, kaum mehr, als gewöhnliche Funktionsaufrufe.
- In vielen Fällen kann statische Bindung erzeugt werden, immer dann, wenn der Objekttyp während der Kompilierung bekannt ist.
- Der Kompiler erzeugt eine Funktionspointertabelle (VMT-virtual method table) für jede Klasse, die virtuelle Funktionen enthält.
- In der VMT ist für jede virtuelle Funktion ein Funktionspointer eingetragen, der auf die zur Klasse gehörenden Memberfunktion zeigt. Das Objekt bringt gewissermaßen seine Memberfunktion mit.
- Enthält eine abgeleitete Klasse keine Definition der virtuellen Funktion, so wird die Adresse der virtuellen Funktion der Basisklasse eingetragen.
- Die VMT existiert für jede Klasse genau einmal, jedes Objekt der Klasse wird mit einem Zeiger auf die VMT ausgestattet, der Zeiger trägt das Attribut verborgen (hidden).

rein virtuelle Funktionen

- Rein virtuelle Funktionen werden in einer Klassendeklaration nur deklariert, es folgt keine Definition.
- Virtuelle Funktionen werden durch "virtual r_typ f_name (par_list) =0; " vereinbart.
- Eine solche rein virtuelle Funktion stellt lediglich ein ploymorphes Interface für die abgeleiteten Klassen dar.
- Es ist nicht möglich, Instanzen von Klassen, die rein virtuelle Funktionen enthalten, zu erzeugen.
- Es können nur Instanzen abgeleiteter Klassen, die Memberfunktionen für die rein virtuellen Funktionen der Basisklasse bereitstellen, erzeugt werden.
- Eine Klasse, die rein virtuelle Funktionen enthält, wird auch abstrakte Klasse genannt, weil von ihr keine Instanzen erzeugt werden können, wohl aber kann es Pointer auf abstrakte Klassen geben.
- Enthält eine abgeleitete Klasse keine Memberfunktion zu einer oder mehreren rein virtuellen Funktion(en) der Basisklasse, so ist auch sie eine abstrakte Klasse.
- Eine abstrakte Klasse läßt sich nicht von einer konkreten Klasse ableiten.
- Es kann sinnvoll sein, eine Klasse, die nur Definitionen rein virtueller Funktionen enthält, zu definieren.

Destruktoren in abgeleiteten und Basisklassen

- Ausführung des Destruktors der abgeleiteten Klasse, dann Ausführung des Destruktors der Basisklasse, wenn ein Objekt seine Gültigkeit verliert.
- Wird ein abgeleitetes Objekt, auf das ein Zeiger der Basisklasse zeigt mit delete vernichtet, so wird lediglich der Destruktor der Basisklasse ausgeführt.
- Als Lösung sollte der Destruktor der Basisklasse als virtuelle Funktion vereinbart werden.
- Jede Klasse, die virtuelle Funktionen verwendet, sollte auch einen virtuellen Destruktor besitzen.
- Destruktoren können virtuell sein, Konstruktoren nicht.

Der Zugriffsmodifakator protected

- Protected member unterscheiden sich von private member darin, daß sie auch den Memberfunktionen direkt abgeleiteter Klassen bekannt sind.
- Protected sollte sparsam verwendet werden, weil sonst Änderungen sehr aufwendig werden.

Public und Private Basis-Klassen

- public : Public-Member der Basisklasse sind auch public-Member der abgeleiteten Klasse
- private : Public- und protected-Member der Basisklasse sind private-Member der abgeleiteten Klasse und somit in weiteren Ableitungen nicht bekannt;
  Privateableitungen sind sehr ungewöhnlich die implizite Konvertierung in ein Objekt der Basisklasse ist nicht möglich, auch Polymorphismus kann nicht zur Anwendung kommen.
- Man spricht auch von Durchsichtigkeit/ Undurchsichtigkeit von Klassen in einer Klassenhierarchie.
- Die generelle Undurchsichtigkeit kann für einzelne Klassenmember aufgehoben werden, in dem in der Klasse, die private abgeleitet ist, der Name des Members in Verbindung mit der Klasse, in der es deklariert ist, neu deklariert wird.

Mehrfachvererbung

- Mehrfachvererbung: eine abgeleitete Klasse übernimmt die Eigenschaften (Member) mehrerer Basisklassen.
- Wenn eine Klasse mehrfach als indirekte Basisklasse vorkommt, erfordert die Auswahl von Memberfunktionen die Angabe des Klassennamens, deren Member die Funktion ist, weil der Kompiler keine eindeutige Zuordnung mehr treffen kann.
- Auch unterschiedliche Basisklassen können Member gleichen Namens enthalten, auch dann muß der Klassenname angegeben werden.
- Virtuelle Basisklasse: Virtual wird an den Stellen, an denen die Klasse die die Daten definiert, als direkte Basisklasse auftritt, angewendet.
- Basisinitialisierer der abgeleiteten Klassen werden durch Komma getrennt vor dem Body des Constructors der abgeleiteten Klasse angegeben.