Andere Programmiersprachen - php

Allgemeines zu php

- php ist eine serverbasierte Skriptsprache, die auf dem Server interpretiert wird. php lehnt sich formal an C++/C# und Java an
- Der Quellcode ist für die Clients nicht sichtbar
- Der Quellcode wird mit einem <?php … ?> - Tag in die HTML-Seite eingebunden und an ebensolcher Stelle ausgeführt
- Mit Hilfe des echo-Befehls können Ausgaben in die HTML-Seite erfolgen, die an der Stelle des php-Tags eingefügt werden

```
<p>
<?php
  echo "Hallo";
  $i = 7;
  echo "<BR/>" . $i;
?>
</p>
```

```
<p>
<?php
  echo "Hallo";
    $i = 7;
?>
  <BR/><?php echo $i; ?>
</p>
```

Möglichkeiten php zu nutzen

- Auf der für jeden User mit htw-Login verfügbaren Webseite (siehe http://portal.rz.htw-berlin.de/anleitungen/web/homepage/homepage_einrichten/) wird php-Unterstützung angeboten
  - Erstellen des php-Codes und Hochladen über ein Tool wie winscp
  - Bearbeiten des Codes direkt auf der Homepage mit winscp oder anderen sftp-Clients
  - Testen ohne Debug-Möglichkeiten

- Installation einer xampp-auf dem eigenen Rechner
  - Erstellen der php-Webseite in entsprechenden Verzeichnissen
  - Bearbeiten des Codes direkt auf der eigenen Festplatte
  - Möglichkeit des Debuggens mit eclipse

php und html

- php wird serverseitig in html-Dateien mit der Endung php implementiert. Diese werden nicht wie bei C# kompiliert und ausgeführt, sondern der darin enthaltene Quellcode wird direkt auf dem Server beim Aufruf interpretiert, in der auf dem Client angezeigten Seite kann der php-Quelltext nicht mehr eingesehen werden, es sei denn, der php-Server funktioniert nicht
- Mit dem php-Befehl "echo" oder auch "printf" erfolgt eine Ausgabe aus php in die html-Dokumente an gleicher Stelle. Ohne Ausgaben in php sieht es auf dem Client so aus, als gäbe es den php-Code gar nicht
- Die Erweiterung einer Quellcode-Datei kann dabei php sein, aber auch html, xhtml, … ist möglich, sofern der Server entsprechend konfiguriert ist
- Eine php-Datei muß keine html-Inhalte beinhalten, es kann sich auch nur um php-Code handeln. php-Quellcode-Dateien können über include-Statements zusammen geladen (und damit wiederverwendet) werden. Um Funktionalität aus anderen php-Dateien überhaupt erst nutzen zu können, ist ein include oder require zwingend erforderlich
- Durch die fast beliebige Verknüpfung von html und php können dynamische html-Inhalte durch das php-Skript gestaltet werden oder Programme auf dem Server Aufgaben aus den Anforderungen und Eingaben der User erledigen

Datentypen

- In php gibt es 8 Variablentypen:
  - boolean
  - integer
  - float (Gleitkomma-Zahl mit 8byte, vgl. dem C#-double)
  - string
  - array (Feld)
  - object (eine Instanz einer Klasse)
  - resource (z.B. die Verbindung zu einer Datenbank)
  - NULL (Variable wurde noch nicht oder mit NULL initialisiert oder mit unset wieder zurückgesetzt
- is_int() – Prüft, on eine Variable vom Typ int ist
- is_bool() – Prüft, ob eine Variable vom Typ boolean ist
- is_float() – Prüft, ob eine Variable vom Typ float ist
- is_numeric() – Prüft, ob eine Variable eine Zahl oder ein numerischer String ist
- is_string() – Prüft, ob Variable vom Typ string ist
- is_array() – Prüft, ob die Variable ein Array ist
- is_object() – Prüft, ob eine Variable vom Typ object ist
- is_resource() – Prüft, ob Variable vom Typ resource ist

Variablen in php

- Variablen in php bedürfen keiner gesonderten Deklaration, durch Voranstellung eines $ wird ein Symbol als Variable gekennzeichnet
- Variablen können, wie in C#, in Funktionen oder Klassen verwendet werden, wobei die Gültigkeit der in C# entspricht. Mit public, private und protected kann die Sichtbarkeit von Klassen-Variablen gesteuert werden
- Da php im Gegensatz zu C# keinen stringenten objektorientierten Ansatz verfolgt, können Variablen auch ohne Zugehörigkeit zu einer Klasse oder Funktion definiert werden. Mit dem Schlüsselwort global wird eine global verwendete Variable in einer Funktion nicht erneut definiert, sondern die bereits bestehende globale Variable benutzt (in C# ist dies in Funktionen nicht möglich, in C++ schon)

```php
$a = 1;
$b = 2;

function Summe()
{
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}
```

Variablen in php

- Wird eine Variable (oder Klasse, Funktion) in einem <?php – Abschnitt global definiert, steht Sie innerhalb der aufgerufenen Seite auch in weiteren <?php – Abschnitten zur Verfügung.
- Das gilt auch, wenn mit include eine php-Quelldatei eingebunden wird - auch dort sind bereits vorher definierte Variablen verfügbar, bzw. umgekehrt. Außerhalb von Funktionen und Klassen definierte Variablen sind damit immer global verfügbar
- Variablen müssen nicht einem festen Datentyp entsprechen, sie entsprechen in ihrem Verhalten dem object-Datentyp von C#, der ebenfalls alle möglichen Datentypen halten kann, nur daß in php ein cast auf den gewünschten Datentyp überflüssig ist
  - Wurde einer Variablen zunächst beispielsweise ein int-Wert von 7 zugeordnet, so ist es durchaus möglich, diese Variable später mit einem string-Wert zu überschreiben. Dies ist zwar komfortabel, eröffnet aber einiges Fehlerpotential
- Werden Variablen im Klassenkontext deklariert, so ist ein Zugriff nur über den Variablennamen wie in C# ohne Verwendung von this nicht möglich. this selber heißt in php gemäß der Variablennamenskonvention $this, der Punktoperator von C# wird in php zum Pfeiloperator ->, damit ähnlich zu C++, bei Zugriff auf die Member-Variablen wird dann aber das $-Zeichen weggelassen, also z.B. $this->wert
- php beinhaltet wie auch C# ein eigenes Speichermanagement, damit ist das Löschen erzeugter Objekte nicht erforderlich

string – Operationen, Math – Funktionen

- Gibt es beim Rechnen mit Zahlen bzgl. der Operatoren keine Unterschiede, so ist das Nutzen von mathematischen Funktionen wie z.B. sin mit php einfacher.
  - Da kein absolut objektorientiertes Vorgehen gefordert wird, lassen sich neben Variablen auch Funktionen ohne Klassenbezug definieren
  - Damit kann das Konzept statischer Methoden verlassen und einfache Bibliotheksfunktionalität direkt angeboten werden, die Funktion sin wird also ohne Klassennamen direkt aufgerufen
  - Statische Funktionen oder Eigenschaften werden ebenfalls nicht mit dem Punktoperator an der Klasse aufgerufen, sondern mit dem :: Operator
- Da ein string in php nicht dem string in C# bzw. .NET entspricht, gibt es hier anders benannte, aber ähnlich arbeitende Funktionen: so wird die Länge eines strings nicht über Eigenschaft Length ermittelt, sondern mit der Funktion strlen, die auch nicht an einem string-Objekt aufgerufen wird, sondern ähnlich wie sin global definiert ist. Durch die fehlende Typisierung würde ein objektorientierter Aufruf problematisch werden
- Ein String wird in php mit dem . verkettet, den +-Operator wie in C# gibt es nicht
- Im Internet besteht das Problem verschiedener Zeichensätze, aber auch die Verwendung von Umlauten mit ä - Schreibweise in html muß unterstützt werden. Dazu bietet php verschiedene Methoden an (z.B. htmlentities)

Funktionen

- Funktionen in php können einer Klasse zugeordnet sein oder auch allein stehen, also vollkommen ohne Klassenbezug (wie in C++)
- Im Unterschied zu C# wird eine Funktion lediglich durch den Kennzeichner function als Funktion deklariert, die Definition eines Rückgabetyps entfällt, da dieser ohnehin wie schon bei den Variablen nicht typisiert werden kann
- Lediglich die Übergabeparameter einer Funktion müssen wie in C# genannt werden, jedoch fehlt auch hier die Angabe eines Datentyps
- Funktionen in Klassen können wie in C# mit den Schlüsselwörtern für den Zugriff wie public, private oder protected versehen werden
- Funktionen können mit return beliebige Werte zurückgeben, an unterschiedlicher Stelle auch unterschiedliche Werte hinsichtlich des Datentyps
- Das Überladen von Funktionen (Funktionen gleichen Namens, aber mit unterschiedlichen Übergabeparametern) ist möglich, wenngleich aufgrund der fehlenden Unterscheidungsmöglichkeiten hinsichtlich eines Datentyps nicht so mächtig wie in C#
- Funktionen lassen sich auch über Variablen aufrufen, hier wird die Funktion Login über eine Variable varFunction aufgerufen. Das kann u.U. gefährlich werden, wenn die Werte vom Nutzer eingebbar sind

```
$varFunction = "Login";
$varFunction();
```

```php
<?php
  function writeXML()
  {
    $writer = new XMLWriter;
    $writer->openURI('output.xml');
    $writer->startDocument('1.0', 'UTF-8');

    $writer->startElement('Eltern');
      $writer->startElement('Kind');
        $writer->startAttribute('Name');
          $writer->text('Willi');
        $writer->endAttribute();
        $writer->startAttribute('Alter');
          $writer->text('5');
        $writer->endAttribute();
      $writer->endElement();
      $writer->startElement('Kind');
        $writer->startAttribute('Name');
          $writer->text('Susi');
        $writer->endAttribute();
        $writer->startAttribute('Alter');
          $writer->text('8');
        $writer->endAttribute();
      $writer->endElement();
    $writer->endElement();
    $writer->endDocument();
  }
?>
```

- Nach Aufruf der Funktion writeXML gibt es eine Datei im gleichen Verzeichnis des php-Files mit folgendem Inhalt:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Eltern>
  <Kind Name="Willi" Alter="5"/>
  <Kind Name="Susi" Alter="8"/>
</Eltern>
```

Kontrollfluss in php

- Die Syntax von if, while, foreach und for sind in php gleich wie in C#, lediglich die Deklaration der Steuerungsvariablen (bei for und foreach) entfällt
- Auch die Klammersetzung und das Abschließen einer Zeile mit ; ist genau wie in C#
- Der Kontrollfluß kann php-mäßig unterbrochen werden, um z.B. html-Code einzuschieben, der entsprechende Block wird dann nicht mit einer geschweiften Klammer, sondern mit einem Doppelpunkt eingeleitet

```
<?php if ($this->details): ?>
<p>…</p>
<?php else: ?>
<p>…</p>
<?php endif; ?>
```

- Der nicht betroffene if-Teil wird dann auch html-mäßig nicht durchlaufen, die betroffenen Elemente erscheinen damit nicht auf Client Seite
- Mit php-Abfragen können damit z.B. Einstellungen abhängig vom Browser oder sonstigen Dingen vorgenommen werden, Java-Skripte in verschiedenen Versionen eingebaut werden, …
- Es gibt keine void main in php, der Quelltext einer Seite wird direkt geparst. Befehle in oberster Ebene werden dabei direkt ausgeführt

Klassen

- Klassen in php sind ähnlich wie in C# definierbar
- Konstruktoren vor php 5 hießen wie in C# wie die Klasse selbst, nun dient dazu eine function __construct(), ggfs. mit Parametern
- Das Ableiten erfolgt nicht mit einem : und dem Namen der Basisklasse, sondern wie in Java mit dem Schlüsselwort extends
- parent meint die Basisklasse, $this das aktuelle Objekt wie this in C#, self meint die Klasse, z.B. für den Aufruf einer statischen Funktion

```
class BaseClass
{
    function __construct()
    {
        print "Im BaseClass Konstruktor\n";
    }
}

class SubClass extends BaseClass
{
    function __construct()
    {
        parent::__construct();
        print "Im SubClass Konstruktor\n";
    }
}
```

Klassen

```php
class A
{
    function foo()
    {
        if (isset($this))
        {
            echo '$this ist definiert (';
            echo get_class($this);
            echo ")\n";
        } else
        {
            echo "\$this ist nicht definiert.\n";
        }
    }
}
// Hinweis: die folgende Zeile führt zu einer Warnung, wenn
// E_STRICT aktiviert ist
A::foo();
$a = new A();
$a->foo();
```

- Es ist möglich, Funktionen statisch oder nicht statisch aufzurufen, was aber abgeschaltet werden kann

Quelle: php.net

Klassen

- php unterstützt das Konzept des Polymorphismus, d.h. Methoden und Membervariablen lassen sich in abgeleiteten Klassen überschreiben und werden entsprechend der aufrufenden Objekte ausgeführt
- Es werden wie in C# nur Eigenschaften und Funktionen mit der Sichtbarkeit protected oder public weiter vererbt
- Klassen müssen vor ihrer Verwendung geladen bzw. definiert worden sein. Um lange include-Listen zu vermeiden, kann mit der __autoload-Funktion wie im Bild dargestellt das Laden externer Klassen-Dateien automatisch erfolgen und damit die Handhabung erleichtern
- Das Erstellen eines neuen Objektes gleicht dem Verfahren in C#

```php
function __autoload($class_name)
{
    include $class_name . '.php';
}

$obj  = new MyClass1();
$obj2 = new MyClass2();
```

Quelle: php.net

Formulare

- Beim Formular-Attribut “method” besteht die Wahl zwischen “get” und “post”, hier ein Formular in der "get"-Variante in html-Syntax

```
 <form action="welcome.php" method="get">
Name: <input type="text" name="fname">
Alter: <input type="text" name="alter">
<input type="submit">
</form>
```

- Bei Drücken des Senden-Button wird die Seite welcome.php aufgerufen, auf der die Formulardaten "verarbeitet" werden können, php bietet dafür die Variablen “$_GET” oder “$_POST” unter Verwendung der "name"-Werte der Formularelemente an
- Im Quelltext von welcome.php könnte folgender Code stehen:

```
Hallo <?php echo $_GET["fname"]; ?>!<br>
Du bist <?php echo $_GET["alter"]; ?> Jahre alt.
```

- Der Zugriff auf die Formulardaten erfolgt immer auf der "action"-Seite unter Einbeziehung von $_GET oder $_POST, je nachdem, welche Methode im Formular gewählt wurde. Die "action"-Seite kann die gleiche Seite sein, das php-Skript sollte dann so gestattet sein, daß nur bei $_GET/$_POST-Werten etwas passiert

Formulare - Vor- und Nachteile von post und get

- Bei GET sieht der User, welche Daten übergeben werden (kann Vorteil oder als Nachteil sein), z.B. www.beispiel.de/welcome?fname=paule&alter=32, hierbei sind fname und alter die Namen der Formularelemente
  - Diese Werte kann der Benutzer manipulieren, allerdings ist auch POST kein Schutz dagegen, aber es sieht in den meisten Fällen “schöner” aus, da die URL-Zeile nicht mit Variablen vollgepumpt ist.
- Bei GET ist die Länge begrenzt, bei POST nicht, Bei einem Formularfeld mit sehr vielen Daten sollte POST verwendet werden, da die get-Methode durch die Url-Zeilenlänge des Browsers begrenzt ist. Je nach Browser steht eine maximale Länge um 1024 Zeichen zur Verfügung
- Die Ergebnisseite eines GET-Formulars lässt sich bookmarken, da alle nötigen Informationen in der URL enthalten sind
- Die Ergebnisseite eines POST-Formulars lassen sich weder bookmarken noch im Browser aktualisieren, da die Daten nicht mehr zur Verfügung stehen

php - Weiteres

- Identifikation der aktuellen Verbindung, Cookies usw.
- Zugriff auf das Dateisystem mit php, Funktion fopen und andere
- Zugriff auf Datenbanken wie mysql
- Senden von Mail mit php-Bordmitteln
- php-Bibliotheken für Bitmapbearbeitung, pdf, …
- Nutzen von php in Verbindung mit einem CMS
- …

Andere Programmiersprachen – C++

Allgemeines zu C++

- C++ ist die objektorientierte Erweiterung von C
- Projekte, die in C geschrieben sind, lassen sich auch mit einem C++-Compiler übersetzen und anschließend linken, allerdings ist der C++-Compiler etwas rigoroser als C-Compiler bei der Überprüfung
- C++-Code kann, aber muß nicht unbedingt in Klassen organisiert werden, im Unterschied zu C# kann in C++ also vollkommen prozedural implementiert werden (wenn in C# nur mit statischen Funktionen gearbeitet wird, ist dies auch nicht objektorientiert im Sinne der "reinen Lehre")
- C++ als Sprache ist als ANSI/ISO Norm genormt, es gibt viele Compiler und Linker, auch im OpenSource-Bereich wie z.B. den GNU-Compiler, jedoch vermutlich auch ebenso viele Dialekte, die sich mehr oder weniger an die Norm halten
- Da C++ auf keinen betriebssystemseitig vorhandenen Programmen basiert, lassen sich mit C++ Treiber entwickeln oder Programme erstellen, die ohne Installation gestartet werden können
- Neben Java ist C++ wohl die am meisten verwendete Hochsprache

Die Sprache C++

- Im Unterschied zu C# ist es in C++ möglich, Funktionen und Variablen außerhalb von Klassen zu definieren
- C++ kennt sowohl managed Code (.NET mit Speicherverwaltung) als auch unmanaged Code (mit selbst zu realisierender Speicherverwaltung). Wie gearbeitet werden soll, muss in den Projekteinstellungen festgelegt werden, bei unmanaged Code müssen Objekte, deren Größe erst zur Laufzeit feststeht speichermäßig implizit dynamisch allokiert (new) und später wieder freigegeben (delete) werden
- In C++ sind andere, im Projekt definierte Klassen, nicht ohne weiteres bekannt. Zur gegenseitigen Verwendung werden in C++/C sogenannte Header-Files definiert. In diesen werden die in einer cpp-Datei implementierten Klassen, ihre Methoden, Eigenschaften, aber auch ohne Klassenkontext definierte Methoden deklariert – in einer Form, die den Interface-Definitionen von C# nicht unähnlich ist. Diese Header-Files müssen per include – Statement in Quellcode- oder anderen Header-Dateien eingebunden werden, um benutzt werden zu können. Beim Kompilieren ist in C++ ohne Header-File die Reihenfolge der Deklaration im cpp-File relevant
- Header-Dateien von System-Klassen und –Funktionen liegen in vordefinierten Verzeichnissen, die in den Suchpfad der Entwicklungsumgebung eingetragen sein müssen
- Bei unmanaged Code werden Bibliotheken (lib-Files) beim Linken statisch gebunden. Der Pfad zu diesen Dateien muß direkt oder über Projekteinstellungen definiert werden

Einfache Datentypen in C++ (unmanaged)

- Wichtige Datentypen:
  - int — Datentyp für Ganzzahlen (integer), bei 32-Bit-Rechnern liegt der Wertebereich von -231 bis 231-1
  - long — in der Regel wie int, Unterschied auf 16-Bit-Systemen
  - unsigned int — Nicht negative Ganzzahlen, Verzicht auf das Vorzeichen-Bit, Wertebereich von 0 bis 232-1
  - char — Zeichen (character), Abbildung durch 8byte, d.h. 28 Möglichkeiten, wegen Kompatibilität mit int mit Wertebereich von -128 bis 127, Darstellung des ASCII-Zeichensatzes
  - unsigned char — länderspez. character wegen zu wenig Zeichen in den 127 pos. Werten, damit stehen 256 Zeichen zur Verfügung
  - double — 64-Bit-Gleitpunkt-Zahl, Wertebereich im Header climits
  - long * — long double mit 80-Bit, long long mit 64-Bit, sofern unterstützt
  - float — 32-Bit-Gleitpunkt Zahl, Wertebereich im Header cfloat
  - short int — 2 byte Integer, Wertebereich von -32768 bis 32767, keine Meldung des Kompilers beim Überschreiten!!

Pointer

- Ein wichtiges Element der Sprache C++ und auch in C ist der Zeiger (Pointer)
- Ein Pointer zeigt nur auf den Speicher eines bestimmten Elementes, das heißt den Bereich des Adressraums, in dem das Element physikalisch abgebildet ist
- Notation:
  - int x; //Integer-Variable X
    int *p; //Pointer-Variable p auf einen Speicherbereich, der eine Ganzzahl enthält
- Es gilt:
  - p ist der Verweis (Pointer) auf einen Speicherbereich
  - Auch der Zeiger hat einen Typ, hier zum Beispiel der Pointer auf eine Ganzzahl
  - *p ist der Inhalt des Speicherbereichs, das "*" damit ein Dereferenzierungoperator, der den Speicherinhalt ausliest und den Inhalt anhand des Datentyps interpretiert
  - Das "&" vor einer Variablen stellt einen Adressoperator dar, der die Adresse der folgenden Variablen ermittelt
- Nun könnte folgendes gesetzt werden
  - p = &x; //p zeigt nun auf die Adresse der Variablen x, &=Adressoperator
- Frage: Was wird ausgegeben?
  - ```
    int *p, x;
    x=7;
    p=&x;
    *p++;
    cout << x;
    ```
    ```

Pointer und Felder

- Felddefinition
  ```
  char [] = {'H', 'a', 'l', 'l', 'o', '\0'};
  ```
  Hier wird ein Feld von mehreren char definiert, wobei das letzte Element ein Nullzeichen ('\0') ist
  Ein Stringliteral wie z.B. "Hallo" ist also ein Feld von const char, abgeschlossen vom Nullzeichen
- Ein Feld ist immer eine Abfolge von Zeigern, der *-Operator verweist immer auf das erste Element des Feldes. Was erzeugt folgender Code für eine Ausgabe?
  ```
  char pp [] = {'F', 'a', 'l', 'l', 'e', '\0'};
  *pp = 'H';
  std::cout << pp << std::endl;
  ```
- Auf was zeigt folgender Pointer bzw. was bildet er ab?
  ```
  char *pTxt;
  ```
- Welche Ähnlichkeiten haben damit Zeiger und Felder, betrachten Sie die Definitionen von:
  ```
  char *text;
  char andererText[30];
  ```

Übergabeparameter (der main-Funktion)

```
int main (int argc, char* argv[])
```

- Die Hauptfunktion eines jeden C++-Programms kann zwei Übergabeparameter besitzen:
  - `int argc` (Count Of Arguments)
    Gibt an, mit wievielen Übergabeparametern die Exe-Datei gestartet wurde, dabei ist der erste Übergabeparameter immer der Name der Exe-Datei selber
  - `char * argv[]`
    Feld mit den Texten der Übergabeparameter, jeder einzelne Text als char-Pointer übergeben,
    Schreibweise nur in Parameterliste zulässig, ansonsten bedeutet dies char** (Pointer auf Pointer)
- Variablenübergabe an Funktionen
  - Variablenwert als Kopie
  - Variable als Referenz
  - Pointer auf Variable

Was wird ausgegeben?

```
#include <iostream>

int zeiger(int * uebergabe);
int wert(int uebergabe);
int referenz(int &uebergabe);

int main(int argc, char* argv[])
{
     int intwert = 5;
     zeiger(&intwert);
     std::cout << "Nach Zeiger:   " << intwert << std::endl;
     wert(intwert);
     std::cout << "Nach Wert:     " << intwert << std::endl;
     referenz(intwert);
     std::cout << "Nach Referenz: " << intwert << std::endl;
     return 0;
}

int zeiger(int * uebergabe)
{
     *uebergabe = 75;
     return 0;
}

int wert(int uebergabe)
{
     uebergabe = 33;
     return 0;
}
```

```
int referenz(int &uebergabe)
{
          uebergabe = 22;
          return 0;
}
```

Speichermanagement

- An manchen Stellen muss das automatische Speichermanagement von C++ im Stack außer Kraft gesetzt werden
- Es muss die Möglichkeit geben, Speicher für einen bestimmten Datentyp zu bestimmten Zeiten zu reservieren und bei keiner weiteren Verwendung auch wieder individuell frei zu geben
- In C gibt es dazu die Operatoren/Funktionen malloc (memory allocation) und free
- In C++ werden dazu new und delete verwendet

```
int *getIntegerpointer()
{
    int *pInt = new int;
    *pInt = 34;
    return pInt;
}
```

- Reservierter Speicher (allocated) muss in eigener Verantwortung wieder freigegeben werden, ansonsten droht ein Speicherüberlauf nach entsprechender Programmlaufzeit
- Durch das delete wird der verwiesene Speicher wieder für die Benutzung freigegeben, die Zeigervariable wird ungültig, ein weiterer Zugriff oder ein weiteres delete auf die Variable führt zum Programmabsturz

Verwendung von new und delete in C++

- Die dynamische Erzeugung von Elementen während der Laufzeit ohne automatisches Speichermanagement ermöglicht die Verwaltung von bei Kompilierzeit nicht bekannten Größenverhältnissen, z.B. in Arrays
- Das automatische Speichermanagement erfolgt auf dem Stack (Stapelspeicher), dieser steht nur für diese Art (autom.) des Speichermanagements zur Verfügung, ist schnell und effizient, allerdings von der Größe her begrenzt, die aber häufig im Compiler einstellbar ist
- Beim manuellen Speichermanagement werden die Elemente auf dem Heap (freier Speicher) erzeugt, das ist i.d.R. langsamer, aber für große Datenmengen geeignet
- Beim Erzeugen eines Elementes mit new muß das delete an geeigneter Stelle erfolgen, hier ist i.d.R. eine Betrachtung der Objektlebensdauer erforderlich, um keine Memoryleaks zu erzeugen, auch ein delete zu viel (bei unsauberer Programmierung) würde auf einen ungültigen Zeiger treffen und dann ebenso zum Absturz führen
- New und delete sind die am häufigsten zu Programmabstürzen führenden Elemente, Speicherzugriffsfehler basieren häufig auf frühzeitig zerstörten Zeigern durch mehrfache Verwendung oder schlechten Programmierstil
- Zur Vermeidung dieses Problems werden häufig so genannte Smartpointer eingesetzt, die durch interne Referenzzählung ihre Verwendung protokollieren und dann, wenn kein Pointer mehr den Speicherinhalt referenziert, diesen selbständig löschen

Klassen in C++ – Unterschiede in der Implementierung zu C#

```
class myClass //Deklaration steht häufig in einer Header-Datei
{
public:
    myClass();            //Konstruktor
    ~myClass();           //Destruktor
    void myMethod(int a); //einfache Methode
private:
    …
};

//Implementierung (häufig in C++-Datei)
myClass::myClass() //Konstruktor
{
    …
}
myClass::~myClass() //Destruktor
{
    …
}
myClass::myMethod(int a) //einfache Methode
{
    …
}
```

Klassen in C++

- Gebe es eine Klasse myClass, so kann eine Variable dieses Typs in C++ auf unterschiedliche Weise erstellt werden:

```
myClass var1; //Erzeugung auf dem Stack (ohne new)
myClass *var2 = new myClass; //Erzeugung auf dem Heap
```

- Bei Zerstörung einer mit "new" erzeugten Instanz muss der belegte Speicher wieder freigegeben werden, dazu gehört u.U. auch die Freigabe von Membervariablen, die ebenfalls mit new erzeugt wurden. Dafür steht der sogenannte Destruktor zur Verfügung, in dem diese Aufgaben implementiert werden.
- Der Speicher des oben mit new erzeugten Objektes var2 muss, wenn dieses nicht mehr benötigt wird, wieder freigegeben werden (s.o.):

```
delete var2;
```

- var1 wird, da auf dem Stack erzeugt, automatisch entfernt. Da der Stack begrenzt ist, können nicht alle Objekte dort gehalten werden.
- Der -> Operator wird bei Pointervariablen für den Zugriff auf Eigenschaften und Methoden benutzt, bei Stackvariablen der . wie in C#
- Jede komplexere Klasse sollte Konstruktor und Destruktor enthalten, C++ erzeugt aber immer auch einen Standardkonstruktor

Vererbung in C++

- Eine Methode kann wie in C# in der abgeleiteten Klasse "überschrieben" werden:

```
class baseClass
{
public:
        virtual void check();
}

class myClass : public baseClass
{
public:
        (virtual) void check(); //virtual hier nicht mehr notwendig
};
```

- Polymorphismus in C++ wird durch das Schlüsselwort virtual wie in C# angezeigt, das override in der abgeleiteten Klasse entfällt

Andere Programmiersprachen - Java

Allgemeines zu Java

- Java ist eine objektorientierte Programmiersprache und eine eingetragene Marke des Unternehmens Sun Microsystems (die 2010 von Oracle übernommen wurde), es besteht eine hohe Affinität zu Unix-Systemen (wie z.B. Linux)
- Um Java-Programme zu erstellen, wird die Java-Entwicklungswerkzeug (JDK – Java Development Toolkit) und die Java-Laufzeitumgebung (JRE – Java Runtime Environment) benötigt. Die Laufzeitumgebung selbst besteht aus der virtuellen Maschine (JVM) sowie den mitgelieferten Bibliotheken der Java-Laufzeitumgebung
- Die Maschine, die den vorkompilierten Bytecode ausführt (interpretiert), ist jedoch typischerweise virtuell – das heißt, der Code wird meist nicht direkt auf der Hardware ausgeführt, sondern durch die JRE auf der Zielplattform
- Durch die Laufzeitumgebung, die für viele Plattformen angeboten wird (verbreiteter als .NET) wird eine Plattformunabhängigkeit erreicht, durch die Java-Programme ohne weitere Änderung auf Rechnerarchitekturen mit vorhandener JRE laufen können
- Hersteller lassen eigene JRE für ihre Plattform zertifizieren. Handys, Autos, HiFi-Anlagen und anderen elektronischen Geräten nutzen Java

Gemeinsamkeiten mit C#

- Das Ergebnis nach dem Kompilieren ist in beiden Sprachen eine system- und maschinenunabhängige Zwischensprache, welche in einer "virtuellen Maschine" ausgeführt wird. Java tut dies durch Kompilieren in Byte-Code und Ausführen in der JVM (Java Virtual Machine). Microsoft kompiliert C# in IL-Code (Intermediate Language) und führt diesen in der CLR (Common Language Runtime) aus. Die JVM interpretiert oder jitted (Just In Time Compiling) den Bytecode. Die CLR setzt ausnahmslos auf JIT und schließt das Interpretieren aus Performance-Gründen aus
- In beiden Sprachen übernimmt eine Garbage-Collection die Freigabe von nicht mehr benötigtem Speicher, so dass sich nicht mehr der Programmierer darum kümmern muss
- Es gibt keine aus C und C++ bekannten Header-Files
- Die Reihenfolge der Deklaration von Klassen ist irrelevant und es gibt keine Probleme bei zirkulären Abhängigkeiten
- Alle Klassen sind von "object" abgeleitet und bekommen über das Keyword "new" Speicher auf dem Heap zugewiesen
- Es werden Threads unterstützt (locked/synchronized)
- Mehrfachvererbung ist nur durch Interfaces möglich - eine Klasse kann von maximal einer anderen Klasse abgeleitet sein

Quelle: Lehr

Gemeinsamkeiten mit C#

- Konzept der Inneren Klassen – auch in Java können Klassen innerhalb von Klassen definiert werden
- Es ist nicht möglich, Funktionen, Variablen oder Konstanten außerhalb einer Klasse zu definieren
- Arrays und Strings beinhalten wie in C# immer ihre Länge und besitzen Funktionalitäten zum Prüfen ihrer Grenzen
- Alle Werte müssen vor der Verwendung initialisiert werden
- Exceptions: Wie in C# dürfen Try-Blöcke einen "finally"-Teil besitzen

Quelle: Lehr

Unterschiede zu C#

- Konzeptionelle Unterschiede zu Java bestehen insbesondere in der Umsetzung von Callback-Mechanismen. C# implementiert hierzu die Unterstützung von Delegaten (englisch delegates), einem Konzept, das einem Kommando-Muster vergleichbar ist. In Java kommt hingegen das Beobachter-Entwurfsmuster zum Einsatz.
- Des Weiteren unterstützt C# so genannte Attribute (attributes), die es erlauben, die Funktionalität der Sprache über Metadaten im Code zu erweitern (eine ähnliche Funktionalität wurde in Form der oben beschriebenen Annotations in Java 5.0 übernommen). C# enthält auch Bestandteile der Sprachen VisualBasic, zum Beispiel Eigenschaften (properties), sowie Konzepte aus C++.
- In C# ist es ebenso wie in Java möglich, Ausnahmen (exceptions) zu einer Methode zu deklarieren. In Java können Ausnahmen so deklariert werden, dass sie auch verarbeitet werden müssen (checked exception).
- Systembefehle können in .NET über platform invoke aufgerufen werden. Dies ist in Java von der Syntax her nicht möglich, kann aber über die Klassenbibliothek mittels Runtime.exec() und java.lang.ProcessBuilder beziehungsweise Jakarta Commons Exec bewerkstelligt werden.
- In C# kann mit unsafe auch unmanaged Code implementiert werden, dies ist in Java nicht möglich

Quelle: Wikipedia

Datentypen in Java

- byte — 8 bit vorzeichenbehaftete Ganzzahl
- short — 16 bit vorzeichenbehaftete Ganzzahl
- int — 32 bit vorzeichenbehaftete Ganzzahl
- long — 64 bit vorzeichenbehaftete Ganzzahl
- char — 16 bit Unicode-Zeichen
- boolean — boolean-Wert, Größe abhängig von JVM
- float — 32 bit Fließkommazahl (nach IEEE 754)
- double — 64 bit Fließkommazahl (nach IEEE 754)
- String — Unicode-String

Klassen in Java

```
class Punkt             // Deklaration der Klasse Punkt
{
    int x;    /* Deklaration einer Membervariablen*/
    int y;

    public void machwas()
    {
        …
    }
}
```

- Im Gegensatz zu C#/C++ kann auf Klassen und ihre Methoden bzw. Eigenschaften innerhalb eines Pakets (entspricht in etwa dem Namensraum in C#) zugegriffen werden (internal wie in C# im Paket ist also Standard)
- Mit private, protected und public kann dies erweitert (über die Paketgrenzen hinaus) oder eingeschränkt werden, wobei Klassen selber nicht private oder protected sein können
- Für die Vererbung in C# wird nicht :baseclass, sondern ein extends baseclass geschrieben
- Das casten von Basisklassen- auf abgeleitete Klassen-Typen erfolgt wie in C# mit eine cast-Operator

Klassen in Java

- In Java ist das Verhalten von Methoden in abgeleiteten Klassen standardmäßig polymorph, d.h. es wird ohne das Schlüsselwort virtual zu nutzen standardmäßig dieser Mechanismus wie in C#/C++ verwendet
- Das Versiegeln von Klassen und Methoden, das in C# mit dem Schlüsselwort sealed erfolgt, heißt in Java final, damit wird der virtuelle Mechanismus unterbrochen und ein Verdecken ist möglich.
- Standard in C# ist verdecken (ohne virtual) in Java überschreiben
- Java kennt wie C# keine Mehrfachvererbung
- Interfaces werden wie C# definiert, in der Klasse, in der ein Interface implementert wird, mit implements xxx angezeigt. In C# dürfen unterschiedliche Interfaces, die von einer Klasse implementiert werden, gleiche Namen haben, da der Interfacenamen in der Implementierung verwendet wird. In Java ist das nicht möglich
- Das Verdecken ist in Java mit Membervariablen möglich, die gleichnamigen Variablen der Basisklasse sind dann mit super.Name aber immer noch erreichbar, das funktioniert ähnlich wie in C# mit base
- Abstrakte Klassen lassen sich wie in C# definieren, ebenso abstrakte Methoden
- Properties werden in Java nicht unterstützt, die elegante Schreibweise durch die Getter und Setter muß durch Methoden umschrieben werden

Methodenaufrufe

- In C# sind Aufrufe mit reiner Parameterübergabe immer call by value, ebenso wie in Java
- Variablen werden also immer kopiert, bei Objekten gilt dies auch, allerdings wird der Zeiger auf die Objektdaten kopiert.
- Wird ein übergebenes Objekt verändert, so verändert sich dies auch im aufrufenden Kontext, da ja die lokale Variable nur die Adresse zu den Objektdaten hält und diese über den Punktoperator verändert werden können

```
public static void main(String[] args) {
    Ball meinBall = new Ball();
    System.out.println("Ball vor dem Wandel: " + meinBall.typ);    // Handball
    wandel(meinBall);
    System.out.println("Ball vor dem Wandel: " + meinBall.typ);    // Golfball
}
private static void wandel(Ball iball){
    iball.typ = "Golfball";
    iball = null;
}
```

- iball ist nur ein Verweis auf das in main erzeugte Objekt, das die Objektdaten zwar ändern kann, bei Löschung (=null) aber nur sich selbst nicht mehr auf das Objekt zeigen lässt
- Explizite call by ref wie in C# mit ref oder out gibt es nicht