PROGRAMMIERUNG 2

MULTIPLE INHERITANCE

Mehrfachvererbung - Einführung

In C++ k eine Klasse kann von mehreren Basisklassen erben:

```
class A
{
};

class B
{
};

class C : public A, public B
{
};
```

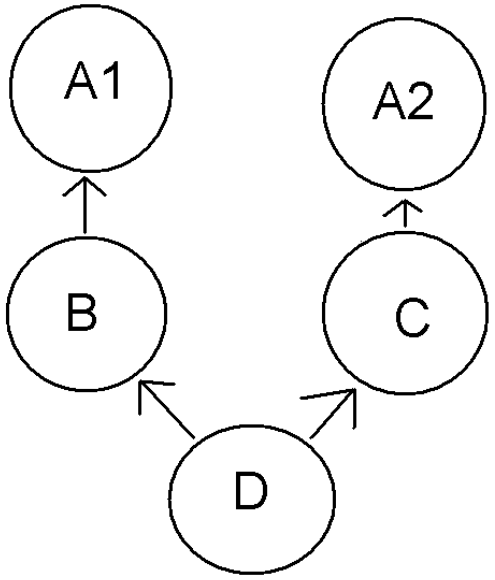
Die Klasse C vereint die Funktionalitäten von A und B

dynamic_cast

Typumwandlung:

Sind zwei Klassen durch eine Vererbungsbeziehung verbundene Klassen,

So kann man in eine Richtung immer casten



Typumwandlung:

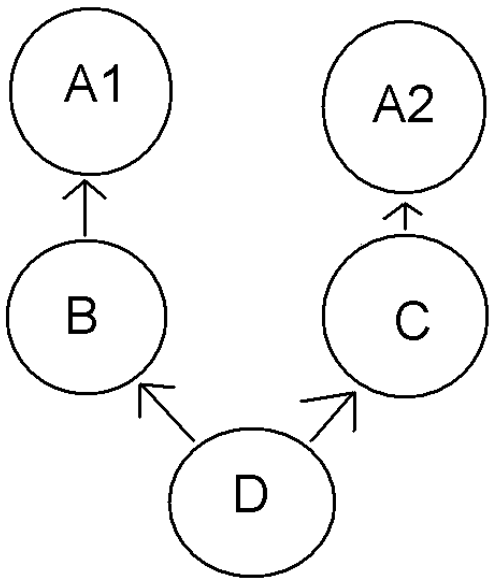
Nach oben casten geht immer

dynamic_cast

Typumwandlung:

Sind zwei Klassen durch eine Vererbungsbeziehung verbundene Klassen,

So kann man in eine Richtung immer casten



Typumwandlung:

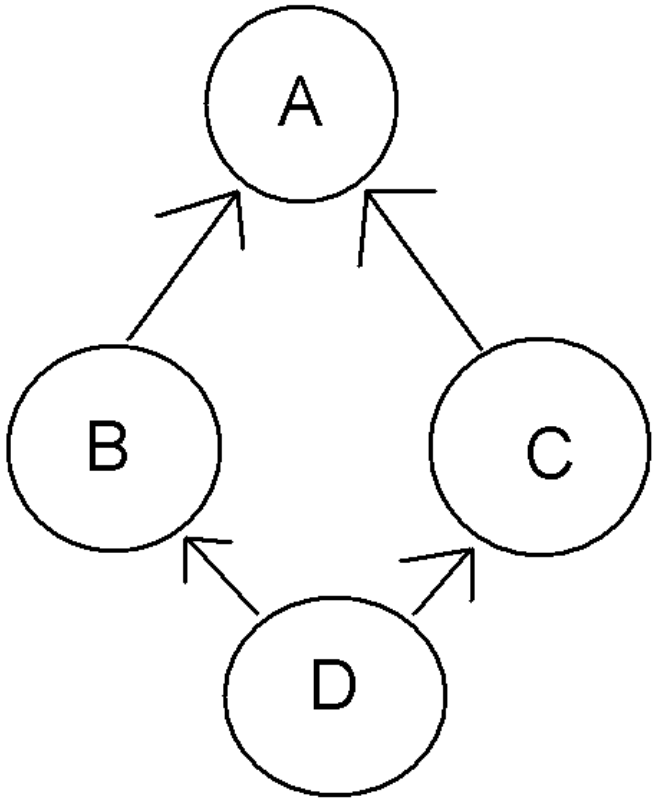
Nach unten casten geht nur bis die tatsächliche Klasse erreicht ist.

Übung 1

a) Recherchiert wie ein dynamic cast in c++ funktioniert.
b) Erstellt eine Pinguin Unterklasse (AlaskaPinguin).
c) Erstellt einen AlaskaPinguin packt ihn in einen Basisklassenpointer.
d) Nutzt den dynmic cast um den AlsakaPinguin in einen normalenPinguin zu umzuwandeln.

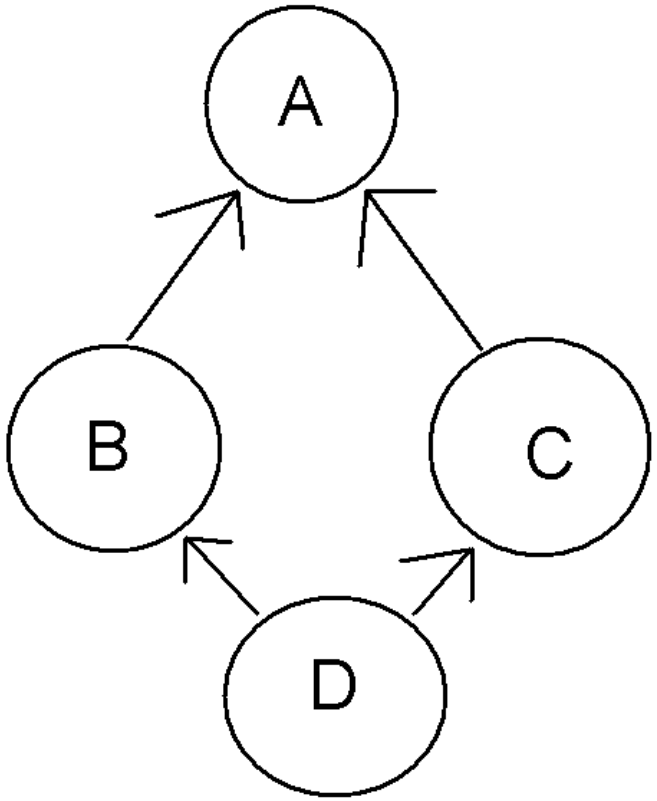
e) Erstellt auf die gleiche Weise einen Pinguin (also Pinguin instanzieren und in einen Basisklassenpointer packen) und versucht ihn in einen AlaskaPinguin umzuwandeln.

The diamond problem



Was ist hier das Problem?

The diamond problem



Mehrdeutigkeit:

Unterschiedliche Implementierungen für dieselbe Methode

B und C erben von A.

Die Klasse D erbt sowohl von B als auch von C.

Wenn es eine Methode in A gibt, die von B und C überschrieben wurde und D sie nicht überschreibt, welche Version der Methode erbt D dann: die von B oder die von C?

The diamond problem

In C++ wird standardmäßig jeder Vererbungspfad separat behandelt.

Das bedeutet, ein D-Objekt würde tatsächlich zwei separate A-Objekte enthalten.

Der Zugriff auf A-Methoden muss korrekt qualifiziert werden.

In C++ muss explizit angegeben werden, aus welcher Elternklasse das zu verwendende Merkmal aufgerufen wird, z.B. Worker::Human.Age.

The diamond problem

```
class Datei {
public:
    void display() { std::cout << "Base class display()" << std::endl;}};

class DateiZumLesen : public Datei {
public:
    void display() { std::cout << "DateiZumLesen class display()" << std::endl;}};

class DateiZumSchreiben : public Datei {
public:
    void display() { std::cout << "DateiZumSchreiben class display()" << std::endl;}};

class DateiZumLesenUndSchreiben : public DateiZumLesen, public DateiZumSchreiben {
public:
    void display() {
        std::cout << "DateiZumLesenUndSchreiben class display()" << std::endl;}};
```

The diamond problem – gelöst!?

```
int main() {
    DateiZumLesenUndSchreiben obj;

    // Ambiguity without qualified member access
    // obj.display();  // Error: ambiguous call to 'display'

    // Resolving ambiguity using qualified member access
    obj.DateiZumLesen::display();
    obj.DateiZumSchreiben::display();

    return 0;
}
```

Virtuelle Vererbung

Jede Instanz der Klasse DateiZumLesenUndSchreiben hat zwei Teilobjekte der Basisklasse Datei.

Das ist hier ein sinnvoller Ansatz, damit Lese- und Schreibzeiger an verschiedenen Positionen stehen können.

Man will aber nicht unbedingt das jedesmal 2 Teilobjekte erzeugt werden (braucht auch 2 vTables)

Sollen die Teilobjekte verschmolzen werden, kennzeichnen Sie die Vererbung mit dem Schlüsselwort virtual

→ das bringt uns zu „Virtueller Vererbung“

```cpp
class Person {
public:
    std::string name;
    virtual void display() = 0;
};

class Mitarbeiter : public virtual Person { /* ... */ };
class Kunde : public virtual Person { /* ... */ };

class MitarbeiterUndKunde : public Mitarbeiter, public Kunde {
public:
    void display() override { std::cout << "DateiZumLesenUndSchreiben class display()"
<< std::endl; }

};



int main()
{
    MitarbeiterUndKunde obj;
    obj.display();
}
```

Virtuelle Vererbung

Besonderheiten:

Wenn die Vererbung von A nach B und die Vererbung von A nach C beide als "virtual" gekennzeichnet sind (Siehe Bsp.),

kümmert sich C++ darum, nur ein A-Objekt zu erstellen, und der Zugriff auf A-Mitglieder funktioniert korrekt.

Der Compiler stellt sicher, dass nur ein Objekt erstellt wird (Siehe nicht virtual – dort werden zwei Objekte erstellt)

Anmerkungen

C++ unterstützt keine explizite wiederholte mehrfache Vererbung

(d.h. eine Klasse mehrmals in einer einzigen Vererbungsliste [class Dog : public Animal, Animal] zu haben).

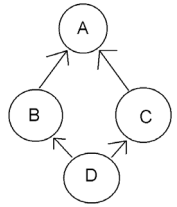
C++ ermöglicht das Erstellen einer einzigen Instanz der mehrfachen Klasse, über den Mechanismus der virtuellen Vererbung (d.h. Worker::Human und Musician::Human verweisen auf dasselbe Objekt).

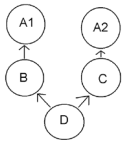
Übung 2

Zusammen:

Refactoring der Bird Klassen

Ein Vogel der fliegen und nicht fliegen kann





Übung 3

Zusammen:

Refactoring der Bird Klassen

Bewegungsinterface wird von der Vogelklasse getrennt (seperate Klassenstruktur).

Selbstständig:

Refactoring der Bird Klasse

Refactoring der Char Klassen