Inhalt

- Vorstellung
- Einleitung, Geschichte
- Grundlagen
- Klassen
- **Vererbung**
- Zusätzliche Themen
- Die Standardbibliothek

Vererbung

- Einfache Vererbung
- Zugriffskontrolle & Vererbungsart
- Vererbung verbieten
- Aufrufreihenfolge & Vererbung von Konstruktoren
- Mehrfache Vererbung
- Abstrakte Basisklassen

Vererbung
Einfache Vererbung

- Person ist die Basisklasse
- Student ist die abgeleitete Klasse
- Jede Instanz von Student kann als Person eingesetzt werden
  - ➔ Jeder Student ist eine Person
- Student erbt alle (public und protected) Eigenschaften (Membervariablen und Methoden) von Person

```
class Person
{
protected: // abgeleite Funktionen haben Zugriff
   string _name;
   int _geburtsjahr;

public:
   Person(string name, int geburtsjahr)
     : _name(name), _geburtsjahr(geburtsjahr) {}
   void ausgabe()
   {
      cout << "Person: " << _name << endl;
   }
};
```

```
class Student : public Person
{
   int _matrikel_nr;

public:
   Student(string n, int j, int m) : Person(n, j)
   {
      _matrikel_nr = m;
   }
   void ausgabe()
   {
      cout << "Student:\nName: " << _name <<
      "\tmatrikel_nr: " << _matrikel_nr << endl;
   }
};
```

Vererbung
Zugriffskontrolle & Vererbungsart

- Die Vererbungsart zeigt an, ob beim Vererben der Zugriff auf Elemente der Basisklasse eingeschränkt wird.
  - Sie wird vor dem Namen der Basisklasse angegeben.
  - Wie bei Memberdeklarationen gibt es die Schlüsselwörter public, protected und private (Standard-Vererbungsart).
- „friend“-Beziehungen und private Variablen oder Methode werden nicht vererbt.

```
class Person { /* ... */ };
class Student : public Person { /* ... */ };
class StudentPro : protected Person { /* ... */ };
class StudentPriv : private Person { /* ... */ }; // := class StudentPriv : Person { /*
... */ };
```

Vererbung
Zugriffskontrolle & Vererbungsart

Ist ein Element in Person	public	protected	private
... wird es in Student	public	protected	nicht übergeben
... wird es in StudentPro	protected	protected	nicht übergeben
... wird es in StudenPriv	private	private	nicht übergeben

```
class Person { /* ... */ };
class Student : public Person { /* ... */ };
class StudentPro : protected Person { /* ... */ };
class StudentPriv : private Person { /* ... */ }; // := class StudentPriv : Person { /*
... */ };
```

Vererbung
Vererbung verbieten

- Das Schlüsselwort final sorgt dafür, dass …
  - von einer Basisklasse nicht geerbt werden kann

```
class NoInheritance final {};

class Derived: NoInheritance {};    // Error !
```

  - eine Klasse der Endpunkt einer Ableitungshierarchie ist

```
class Base {};
class LastClass final: Base {};
class LastLastClass: LastClass {}; // Error !
```

Vererbung
Aufrufreihenfolge Konstruktoren

- Immer, wenn ein Objekt einer abgeleiteten Klasse deklariert wird, wird eine Kette von Konstruktoren ausgeführt.
- Dadurch ist gewährleistet, dass jedes Attribut der Ableitungskette initialisiert wird.
- Die Abarbeitung einer Kette von Konstruktoren beginnt mit der Basisklasse und endet mit der Klasse, von der nicht mehr weiter abgeleitet wird.

```
struct A{};
struct B:A{};
struct C:B{};

C c;                    // A -> B -> C
```

Vererbung
Vererbung von Konstruktoren

- Durch die using-Deklaration erbt eine Klasse alle Konstruktoren ihrer direkten Basisklasse
- Der Default-Konstruktor, der Copy- und Move-Konstruktor wird nicht vererbt
- Die abgeleitete Klasse erbt alle Charakteristiken des Konstruktors (public, private, etc.)
- Default-Argumente für Paramter eines Basisklassenkonstruktors werden nicht vererbt
- Konstruktoren mit denselben Parametern wie die abgeleitete Klasse, werden nicht vererbt.
- **Achtung**: Das Vererben von Konstruktoren birgt die Gefahr, dass ein Attribut in der erbenden Klasse nicht initialisiert wird

Vererbung
Virtuelle Methoden

- Um einer Methode in einer Basisklasse in einer abgeleiteten Klasse ein neues Verhalten zuzuordnen, wird sie in der abgeleiteten Klasse überschrieben.
- Zum Überschreiben muss die Methode in der Basisklasse ***virtual*** deklariert werden. Gerne wird die überschriebene Methode aus Dokumentationszwecken als ***virtual*** deklariert.

```
class Person {
    virtual void ausgabe()
    {
        cout << "Person: " << _name << endl;
    }
```

```
class Student : public Person
{
   virtual void ausgabe()
   {
      cout << "Student: Name: " << _name << "\tmatrikel_nr: " << _matrikel_nr << endl;
   }
```

Vererbung
Virtuelle Methoden

- Wird eine virtuelle Methode über einen Basisklassenzeiger oder eine Referenz auf ein Objekt einer abgeleiteten Klasse aufgerufen, wird die Methode der abgeleiteten Klasse ausgeführt.

```
Person *p_ptr1;
Student s1("Frank", 1995, 1234);
p_ptr1 = &s1;        // Pointer Person zeigt auf Student
p_ptr1->ausgabe();   // Student::ausgabe()
```

- **Achtung**: Die Entscheidung, welches Objekt verwendet wird, wird zur Laufzeit getroffen. Wird auch als dynamische oder späte Bindung bezeichnet.

Vererbung
Mehrfache Vererbung

- Ähnlich zur einfachen Vererbung, werden die Namen der Basisklassen in einer kommaseparierten Liste angegeben
- Die Mehrfachvererbung folgt den Regeln der Einfachvererbung:
  - Jeder Basisklasse können ihre Zugriffsrechte vorangestellt werden.
  - Klassen besitzen per Default private-; Strukturen public-Zugriffsrecht
- Enthält eine Instanz einer Klasse mehr als eine Instanz einer Basisklasse, ist der Aufruf ihrer Mitglieder mehrdeutig und führt zu einem Fehler ➔ Diamond-Problem
- Mehrdeutige Aufrufe bei Mehrfachvererbung lassen sich lösen, indem dem mehrdeutigen Mitglied den Namem der Basisklasse vorangestellt wird.

Vererbung
Mehrfache Vererbung: Virtuelle Basisklasse

- Mit einer virtuellen Basisklasse kann das Problem der Mehrfachvererbung behoben werden, bei der Objekte einer abgeleiteten Klasse mehre Instanzen einer Basisklasse besitzen ➔ Diamond-Problem
- Durch die Verwendung des Schlüsselwortes *virtual* bei der Vererbung von einer Basisklasse wird diese virtuell.
- Falls für virtuelle Basisklassen kein Default-Konstruktor verwendet wird, muss dieser explizit in der abgeleiteten Klasse aufgerufen werden.

Vererbung
Abstrakte Klassen

- Eine Klasse die über eine oder mehrere virtuelle Methoden verfügt
- Virtuelle Methode:
  - Wird mit virtuell deklariert
  - =0 an die Klassendeklaration gehangen

```
virtual string getGeschlecht() = 0;
```

- Abstrakte Basisklassen Klassen dienen oft als Schnittstelle (Interface) für Klassenhierarchien, da sie konkret vorschreiben was Klassen implementieren müssen

Vererbung
Abstrakte Klassen

- Regeln:
  - Eine Klasse, die rein virtuelle Methoden enthält, kann nicht instanziiert werden.
  - Eine abgeleitete Klasse muss Definitionen für alle rein virtuellen Methoden bereitstellen, um instanziiert werden zu können.
  - Eine rein virtuelle Methode kann in einer Klasse definiert werden, die als rein virtuell deklariert ist.
  - Wenn der Destruktor einer Klasse als rein virtuell deklariert ist, muss trotzdem eine Definition dieser Methode in derselben Klasse angegeben werden.
    - ➔ Beliebtes Verfahren in C++, um eine Klasse zur abstrakten Basisklassen zu erklären

Vererbung
Übungsprojekt

- Todo Liste als Gruppenarbeit
- Eventuell GitHub Classroom, link folgt …