

7

Polymorfizmus (virtualita)

Polymorfizmus

- Mechanizmus, ktorý umožňuje zmeniť spôsob fungovania metód základnej triedy
- Implementovaný prostredníctvom virtuálnych funkcií
- Virtuálna funkcia dovoľuje jednému typu vyjadriť svoje odlišnosti od iného, podobného typu, pokiaľ sú obidva typy odvodené z tej istej základnej triedy. Tento rozdiel je vyjadrený prostredníctvom rozdielov v správaní funkcií, ktoré voláme prostredníctvom základnej triedy.
- Dovoľuje vylepšovať organizáciu a čitateľnosť kódu
- Umožňuje tvorbu rozširovateľných programov, ktoré sa dajú rozširovať nielen počas počiatočného vývoja projektu, ale i v prípade požiadaviek na nové vlastnosti.

Väzba volania funkcie

- Prepojenie volania funkcie s telom funkcie sa nazýva **väzba**.
 - **včasná väzba** - vytvára sa pred odštartovaním programu (kompilátorom a spojovacím programom-linkerom)
 - **neskorá väzba** - vytvorí sa až za chodu programu, pričom bude založená na **type objektu** (*dynamická väzba, väzba za chodu programu*)

Virtuálne metódy

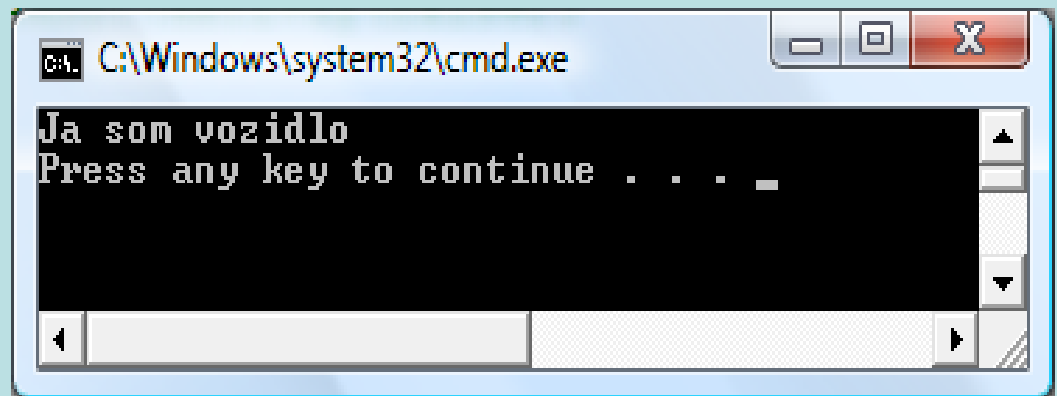
- Ak chceme C++ donútiť, aby konkrétne metódy používali neskorú väzbu, **musíme** pri deklarácií metódy v základnej triede použiť kľúčové slovo **virtual**
- Kľúčové slovo **virtual** sa zadáva **len v deklarácií metódy, nie v definícií**
- Ak je metóda deklarovaná ako **virtual** v základnej triede, potom bude **virtual** vo všetkých odvodených triedach. Všetky metódy v odvodených triedach, ktoré majú zhodnú signatúru s deklaráciou v základnej triede, sa budú volať použitím virtuálneho mechanizmu.
- Predefinovanie virtuálnej metódy v odvodenej triede sa zvyčajne nazýva **prevažovanie**

Pretypovanie smerom nahor (upcasting)

```
class Vozidlo {  
public:  
    void Identifikuj() const { printf("Ja som vozidlo\n"); }  
};  
// Objekt triedy Autobus je Vozidlo - majú rovnaké rozhranie:  
class Autobus : public Vozidlo {  
public:  
    // Predefinovanie funkcie rozhrania:  
    void Identifikuj() const { printf("Ja som autobus\n"); }  
};
```

```
void KtoSom(Vozidlo &i)  
{  
    i.Identifikuj();  
}  
int main()  
{
```

```
    Autobus zaa100;  
    KtoSom(zaa100); // Pretypovanie nahor
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The first line of output is "Ja som vozidlo". The second line is "Press any key to continue . . . _", where the underscore indicates a key has been pressed. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

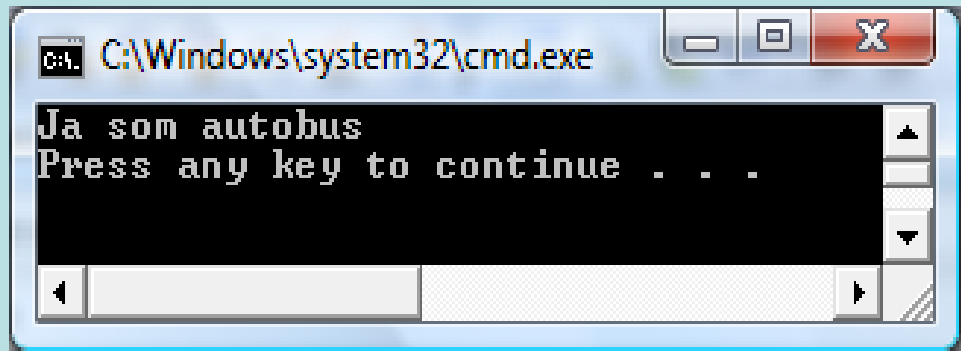
Pretypovanie smerom nahor (upcasting)

```
class Vozidlo {
public:
    virtual void Identifikuj() const { printf("Ja som vozidlo\n"); }
};

// Objekt triedy Autobus je Vozidlo - majú rovnaké rozhranie:
class Autobus : public Vozidlo {
public:
    // Predefinovanie funkcie rozhrania:
    virtual void Identifikuj() const { printf("Ja som autobus\n"); }
};

void KtoSom(Vozidlo &i)
{
    i.Identifikuj();
}

int main()
{
    Autobus zaa100;
    KtoSom(zaa100); // Pretypovanie nahor
}
```



Rozšiřovatelnost

```
class Vozidlo {  
public:  
    virtual void Identifikuj() const { printf("Ja som vozidlo"); }  
    virtual char* Druh() const { return "Vozidlo"; }  
    virtual void Nastav(int) {}  
};
```

```
class Motocykel : public Vozidlo {  
public:  
    void Identifikuj() const {printf("Ja som motocykel"); }  
    char* Druh() const { return "Motocykel"; }  
    void Nastav(int) {}  
};
```

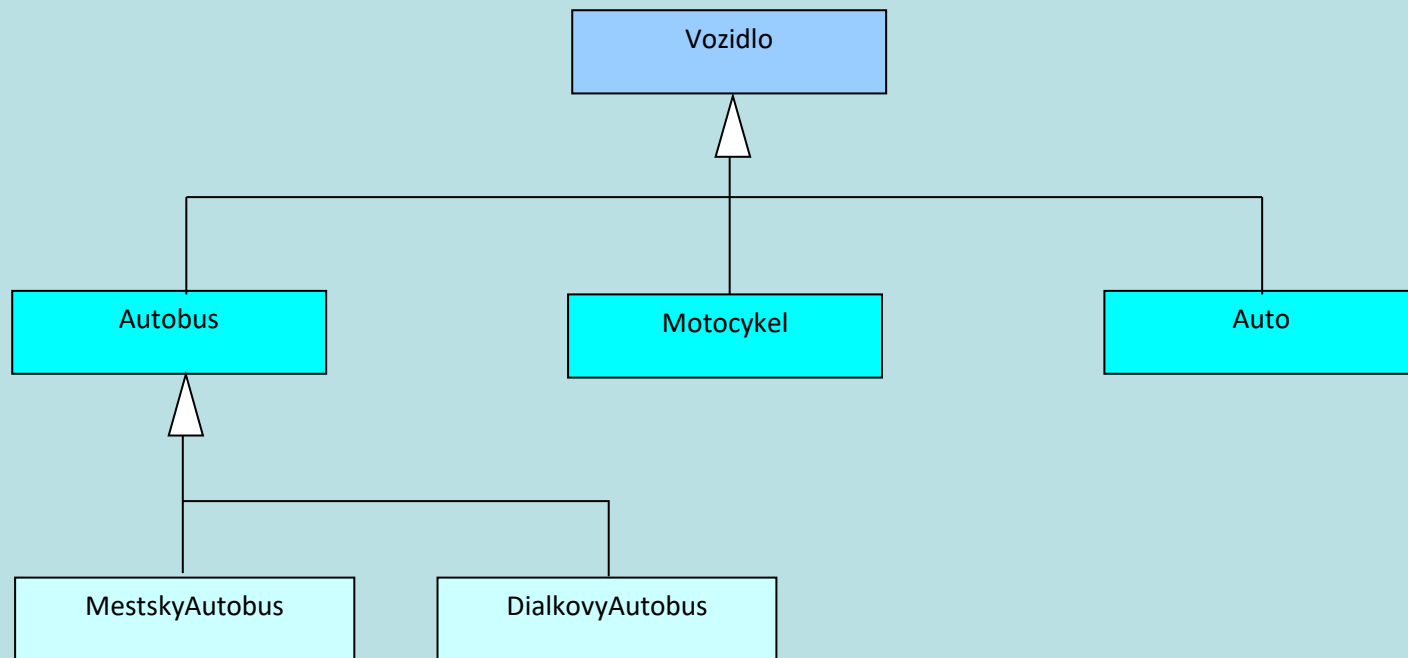
```
class Auto : public Vozidlo {  
public:  
    void Identifikuj() const { printf("Ja som auto"); }  
    char* Druh() const { return "Auto"; }  
    void Nastav(int) {}  
};
```

Rozšiřovatelnost

```
class MestskyAutobus : public Autobus {
public:
    void Identifikuj() const {
        printf("Ja som mestsky autobus");
    }
    char* Druh() const { return "Mestsky autobus"; }
};

class DialkovyAutobus : public Autobus {
public:
    void Identifikuj() const {
        printf("Ja som dialkovy autobus");
    }
    char* Druh() const { return "Dialkovy autobus"; }
};
```


Hierarchia

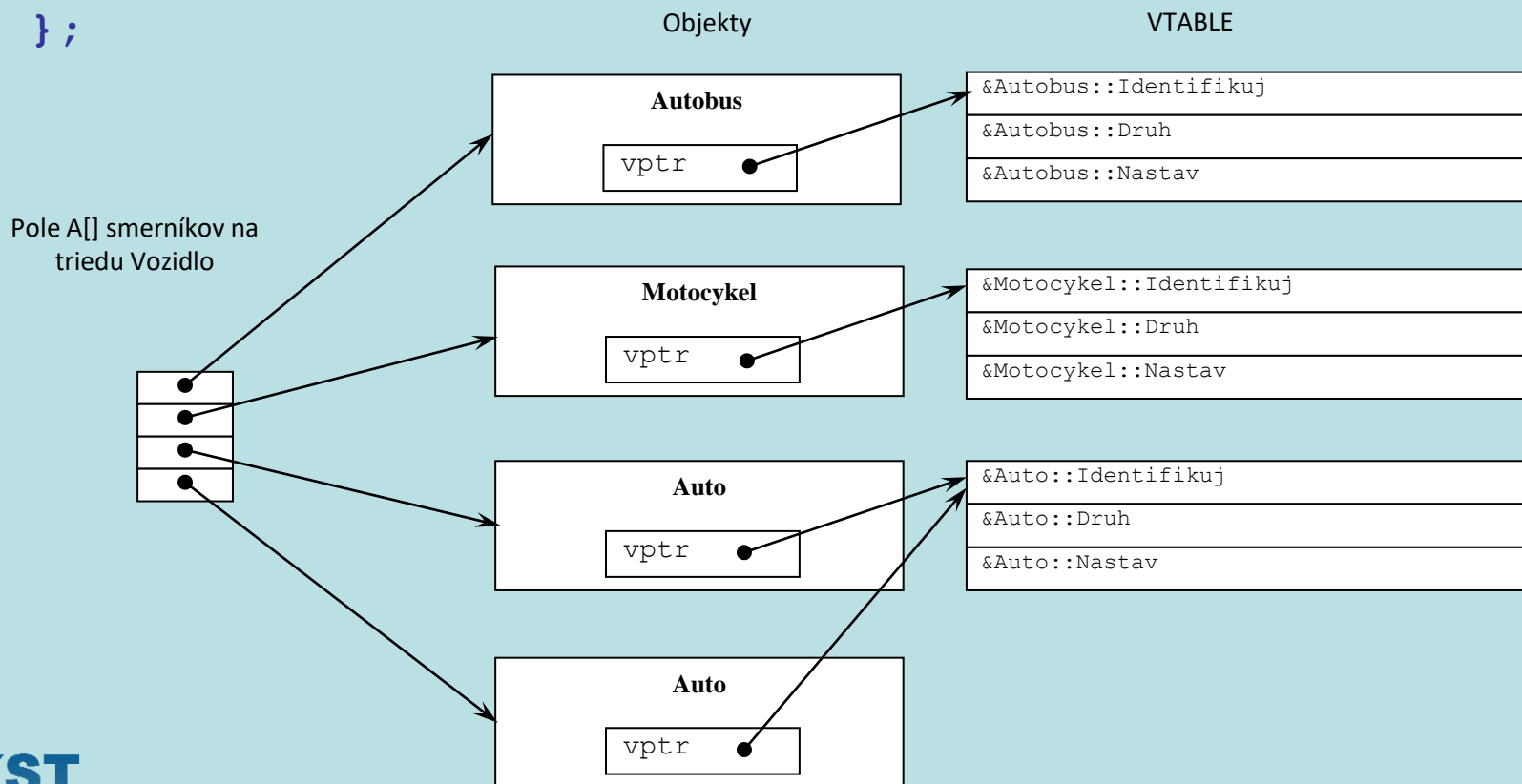


C++ a implementácia neskorej väzby

- Kompilátor vytvára pre každú triedu jednu tabuľku (nazývanú VTABLE), ktorá obsahuje virtuálne funkcie.
- V každej triede, ktorá obsahuje aspoň jednu virtuálnu funkciu, je skryte uložený smerník, nazývaný **vpoiner** (skrátene VPTR), ktorý ukazuje na VTABLE triedy.
- Keď zavoláme virtuálnu funkciu prostredníctvom smerníka na základnú triedu (t.j. keď urobíme tzv. **polymorfné volanie**), kompilátor potichu vygeneruje kód na výber VPTR a vyhľadanie adresy funkcie vo VTABLE, čím sa zavolá správna funkcia a realizuje sa neskorá väzba.

Zobrazenie virtuálnych funkcií

```
Vozidlo* A[] = {  
    new Autobus(),  
    new Motocykel(),  
    new Auto(),  
    new Auto()  
};
```

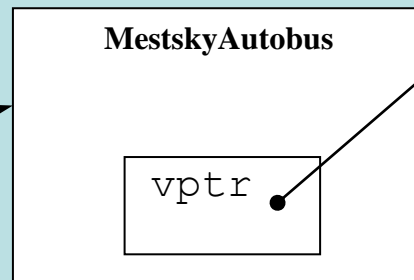
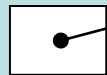


Volanie virtuálnej funkcie

- `p->Nastav(3);`

Smerník na
triedu Vozidlo

Vozidlo* p



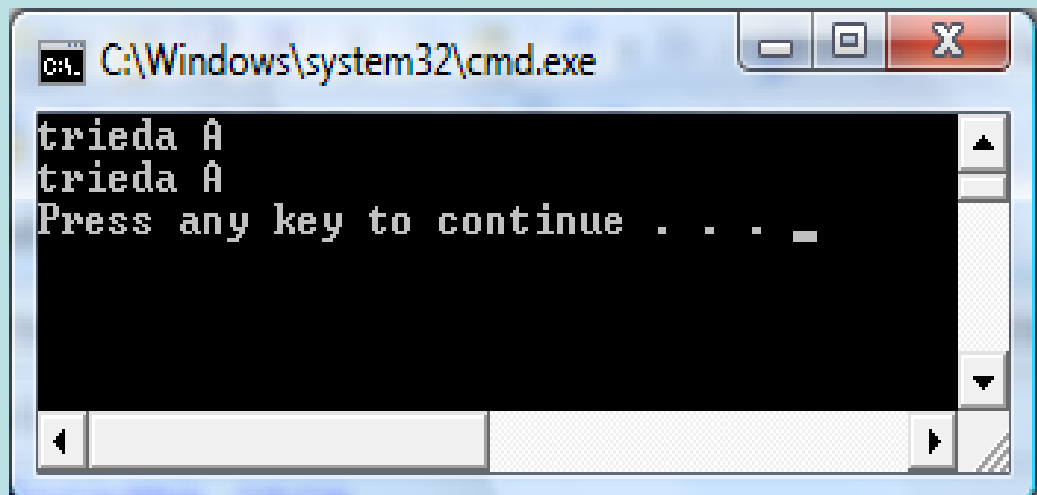
VTABLE triedy

MestskyAutobus

• [0]	<code>&MestskyAutobus::Identifikuj</code>
• [1]	<code>&MestskyAutobus::Druh</code>
• [2]	<code>&MestskyAutobus::Nastav</code>

Príklad

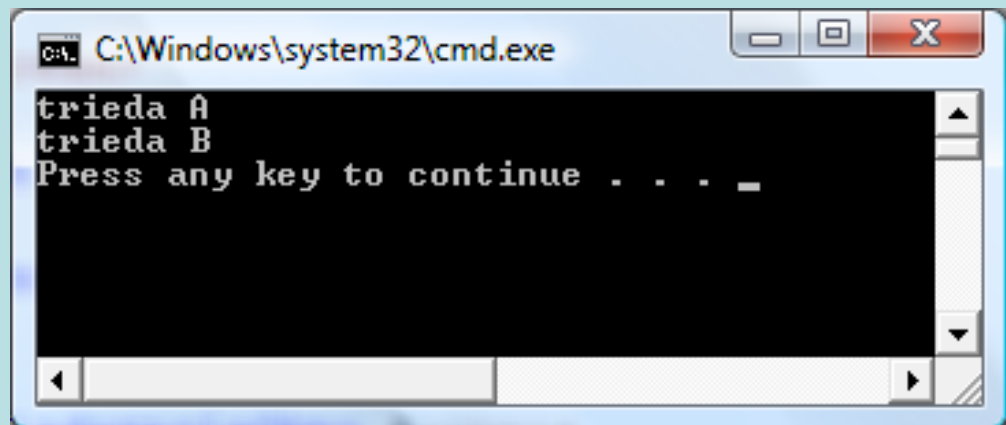
```
class A {  
protected:  
    const char* ToString() { return "trieda A"; }  
public:  
    void Print() { cout << ToString() << '\n'; }  
};  
  
class B : public A {  
protected:  
    const char* ToString() { return "trieda B"; }  
};  
  
int main()  
{  
    A a;  
    B b;  
    a.Print();  
    b.Print();  
    return 1;  
}
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The output of the program is visible: "trieda A" on the first line, "trieda A" on the second line, and "Press any key to continue" on the third line. The cursor is positioned at the end of the third line. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Virtuálne

```
class A {  
protected:  
    virtual const char* ToString() { return "trieda A"; }  
public:  
    void Print() { cout << ToString() << '\n'; }  
};  
  
class B : public A {  
protected:  
    virtual const char* ToString() { return "trieda B"; }  
};  
  
int main()  
{  
    A a;  
    B b;  
    a.Print();  
    b.Print();  
    return 1;  
}
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background and white text. The output of the program is displayed as follows:
trieda A
trieda B
Press any key to continue . . . _
The cursor is positioned at the end of the third line. The window includes standard Windows window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

Abstraktná trieda

- Abstraktná trieda je trieda s aspoň jednou čisto virtuálnou metódou
- Čisto virtuálna metóda je virtuálna metóda, ktorej v prototype priradíme nulu

virtual int metoda(int param) = 0;

- Z abstraktnej triedy nie je možné vytvoriť objekt
- Prinútenie potomka definovať metódu
- Často sa používa pre definovanie **rozhrania** (interface)

Konštruktory a deštruktory

- Virtualita nefunguje v konštruktore a deštruktore – volá lokálnu funkciu
- Konštruktor nemôže byť virtuálny
- Virtuálny deštruktor