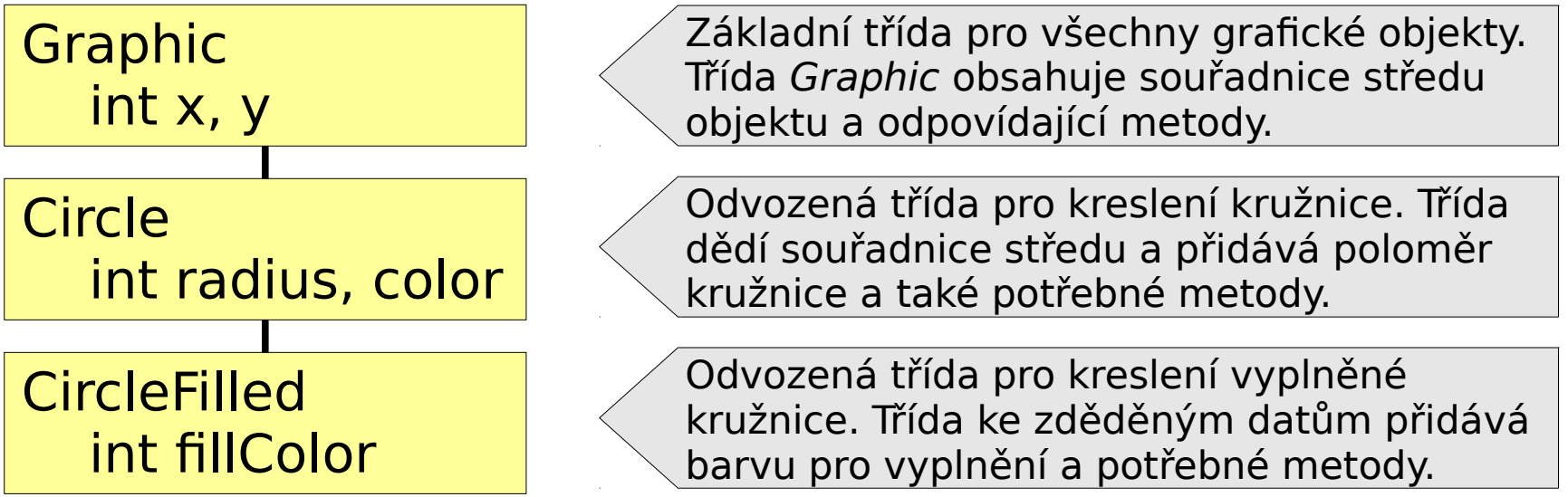
Pokročilé programování v jazyce C pro chemiky

(C3220)

Dědičnost tříd v C++

Dědičnost tříd

- Dědičnost umožňuje vytvářet nové třídy z tříd existujících tak, že odvozené třídy (tzv. potomci) dědí vlastnosti základních tříd (tzv. předek nebo rodič)
- Potomci mají přístup k datovým členům a metodám předků, navíc k ním přidávají vlastní členy třídy
- Děděním je vytvářena hiearchie tříd
- Dědičnost přináší úsporu programátorské práce, protože při vytváření potomka můžeme využívat již existujících metod předka



Definice třídy potomka

- Základní třídu definujeme standardním způsobem
- Odvozená třída obsahuje následující záhlaví:

```
class Odvozena : public Zakladni
```

```
// Zakladni trida
class Graphic
{
  public:
    Graphic();
    void setCentre(int ax, int ay);
  private:
    int x, y;    // Souradnice stredu grafickeho objektu
};

// Odvozena trida dedi data a metody zakladni tridy
class Circle : public Graphic
{
  public:
    Circle();
    void setRadius(int r);
  private:
    int radius;    // Polomer kruznice
    int color;     // Barva kruznice
};
```

Přístup ke členům předka

- Členy třídy (data a metody) mohou bý specifikovány v sekcích **`public`**, **`protected`** nebo **`private`**, přístup ke členům v jednotlivých sekcích je následující:

Přístup	`public`	`protected`	`private`
metody téže třídy	ANO	ANO	ANO
metody potomků	ANO	ANO	NE
metody ostatních tříd a globálních funkcí	ANO	NE	NE

- Členy deklarované jako **`protected`** jsou přístupné v metodách vlastní třídy a metodách potomka, nikoliv však v metodách jiných tříd nebo nečlenských funkcích (pro ně jsou přístupné pouze členy deklarované jako **`public`**)
- Potomek dědí data i metody nejen od přímých předků ale též od prapředků (a jejich předků atd...)

Přístup ke členům předka - příklad

```
class Graphic   // Zakladni trida
{
  public:
    Graphic();
    void setCentre(int ax, int ay);
  protected:
    void printCentre();
  private:
    int x, y;    // Souradnice stredu grafickeho objektu
};

class Circle : public Graphic    // Circle je odvozena trida
{
  public:
    void printValues()
      { printCentre(); /* Vola se zdedena metoda */ };
};

int main()
{
  Circle circ;
  circ.setCentre(10, 30); // Metoda setCentre() je zdedena
  // Nasledujici by nefungovalo, protoze se jedna o protected metodu:
  circ.printCentre();
  circ.printValues();   // Tohle bude vporadku
}
```

Překrytí metod předka

- V odvozené třídě můžeme deklarovat metodu se stejným jménem jako v předkovi
- Pokud takovou metodu zavoláme pro objekt odvozené třídy, zavolá se metoda odvozené třídy (tato metoda tedy překryje metodu v předkovi)
- Pokud chceme zavolat překrytou metodu předka musíme před jejím názvem uvést název třídy předka oddělený pomocí dvou dvojteček (např. `Predek::metoda_predka()`).

Překrytí metod předka – příklad 1

```
class Graphic
{
  public:
    void printValues()
      { cout << "Centre: " << x << ", " << y << endl; };
  private:
    int x, y;
};

class Circle : public Graphic
{
  public:
      // Tato metoda prekryva metodu predka printValues()
    void printValues()
      { cout << "Radius, color: " << radius << "," << color << endl;};
  private:
    int radius, color;
};

int main()
{
  Graphic gra;
  Circle circ;
  gra.printValues();     // Vola se metoda ve tride Graphic
  circ.printValues();    // Vola se metoda ve tride Circle
  circ.Graphic::printValues();   // Vola se metoda ve tride Graphic
}
```

Překrytí metod předka – příklad 2

```
class Graphic
{
  public:
    void printValues()
      { cout << "Center: " << x << ", " << y << endl; };
  private:
    int x, y;
};

class Circle : public Graphic
{
  public:
      // Nasledujici metoda prekryva metodu predka se stejnym jmenem
    void printValues()
    {
      Graphic::printValues(); // Volame metodu predka
      // pokud bychom volali pouze printValues(), zavolala
      // by se meto metoda Circle::printValues()
      cout << "Radius: " << radius << endl;
      cout << "Color: " << color << endl;
    };

  private:
    int radius, color;
};
```

Konstruktory a dědičnost

- Při vytvoření objektu odvozené třídy je nejdříve zavolán konstruktor základní třídy, teprve potom konstruktor odvozené třídy – tímto je zajištěno, že v okamžiku kdy se začne vykonávat kód konstruktoru potomka, jsou již datové členy předka inicializovány
- Do konstruktrou základní třídy lze předat argumenty které byly předány konstruktoru odvozené třídy (viz. následující příklad)

Konstruktory a dědičnost – příklad 1

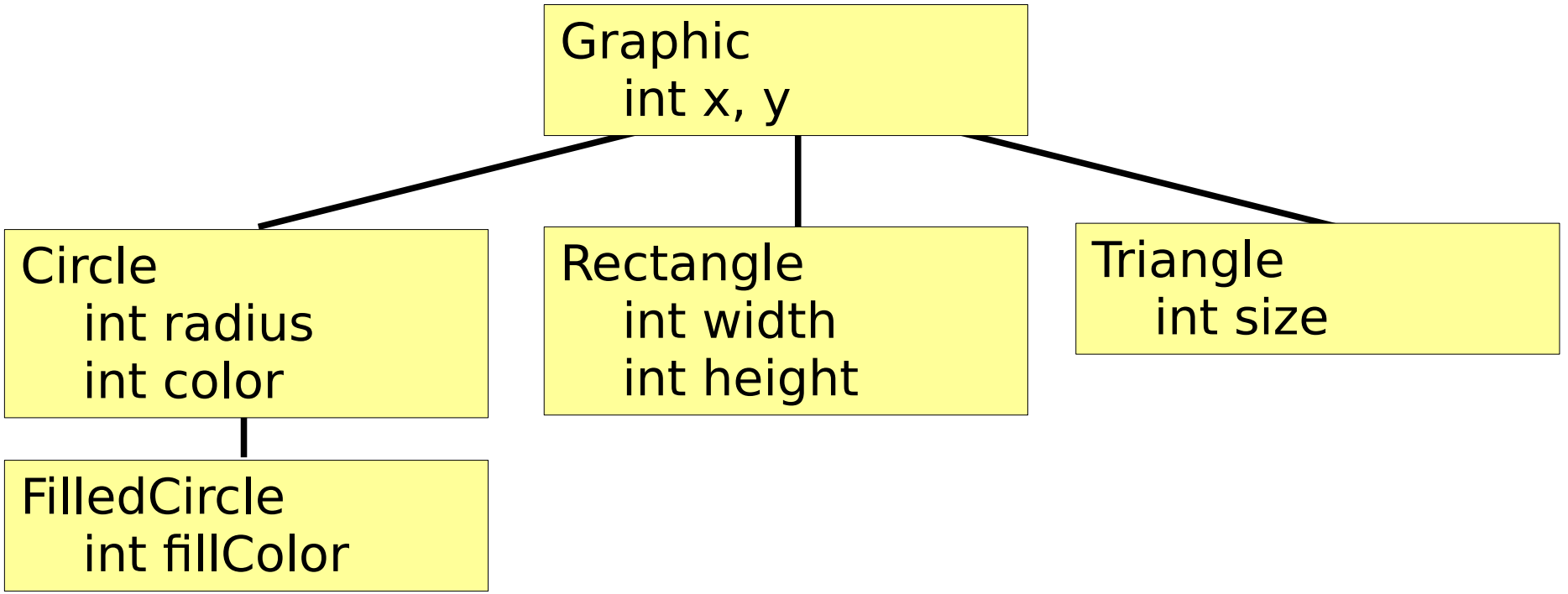
```
class Graphic
{
  public:
      // Konstruktor zakladni tridy se vola jako prvni
    Graphic() { x = 0; y = 0; };
  private:
    int x, y;
};

class Circle : public Graphic
{
  public:
      // Konstruktor odvozene tridy se zavola az po vykonani kodu
      // konstruktoru zakladni tridy
    Circle() { radius = 0; color = 0; };
  private:
    int radius, color;
};


int main()
{
  Circle circ;    // Nejdrive se zavola konstruktor ve tride Graphic,
                  // potom konstruktor ve tride Circle
}
```

Konstruktory a dědičnost – příklad 2

```
class Graphic
{
  public:
    Graphic(int ax, int ay)  { x = ax; y = ay; };
  private:
    int x, y;
};

class Circle : public Graphic
{
  public:
    Circle(int ax, int ay, int r, int c);
  private:
    int radius, color;
};

Circle::Circle(int ax, int ay, int r, int c) : Graphic(ax, ay)
{ radius = r; color = c; }

int main()
{
  Circle circ(20, 30, 5, 1); //Argumenty se predaji konstruktoru ve tride
        // Circle ten ale nejdrive preda nektere z nich konstruktoru v
        // tride Graphic, pote se vykona kod v konstruktoru Graphic()
        // a pak teprve kod v konstruktoru Circle()
}
```

Více potomků odvozených z jednoho předka

- Z jednoho předka lze odvozovat libovolný počet potomků

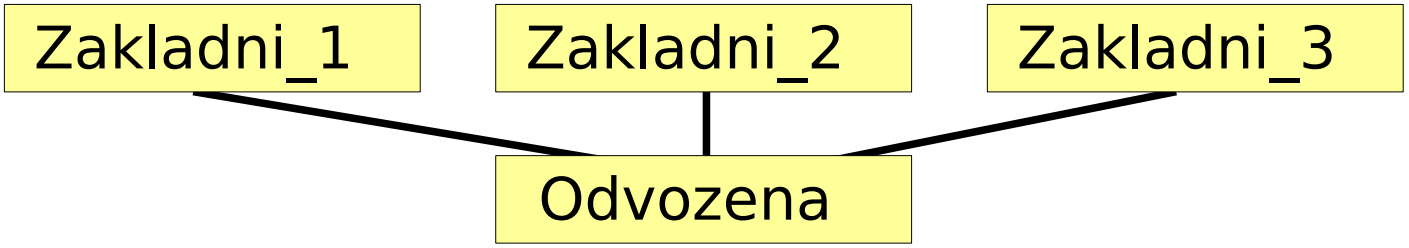


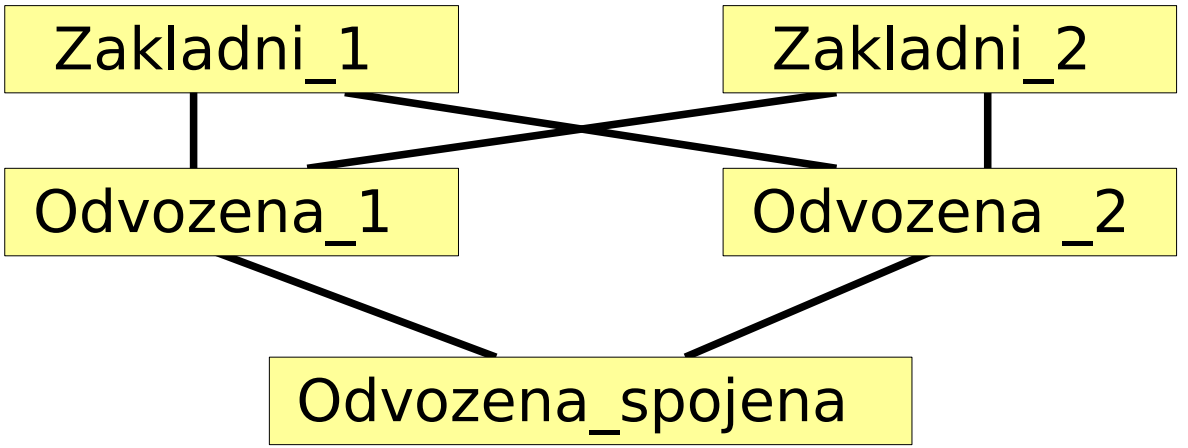
Vícenásobná dědičnost

- Potomky lze odvodit z více než jedné základní třídy – mluvíme o vícenásobné dědičnosti
- Deklarace třídy s vícenásobnou dědičností:

```
class Odvozena : public Zakladni_1, public Zakladni_2
```

- Vícenásobná dědičnost se používá v některých standardních knihovnách, v praxi se však její používání nedoporučuje

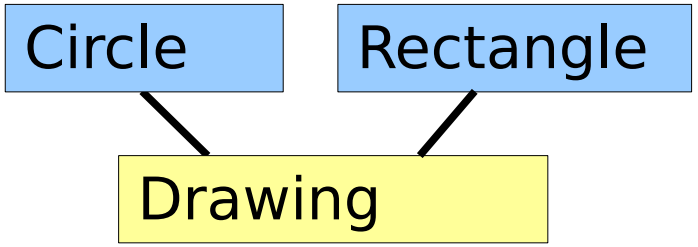




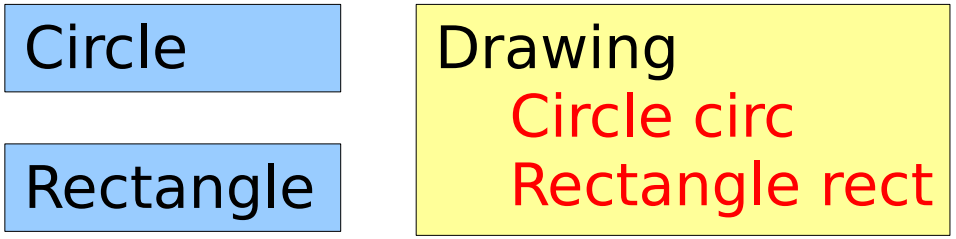
Dědičnost vs. člen třídy

- Při tvorbě programů je někdy potřeba rozhodnout, zda-li použijeme k propojení tříd dědičnost, nebo do třídy vložíme objekt jako její člen
- Pokud si nejsme jisti který přístup je v daném případě lepší, upřednostňujeme v praxi druhou možnost
- Dědičnost používáme pouze v dobře odůvodněných případech
- Objekt odvozujeme pomocí dědičnosti pouze v případě, že jejich vztah odpovídá větě: objekt_odvozeny **je** objekt_zakladni

Nevhodný způsob použití dědičnosti:



Vhodný přístup bez použití dědičnosti:




```
class Drawing : public Circle,
                public Rectangle
{
};
```

```
class Drawing
{
  private:
    Circle circ;
    Rectangle rect;
};
```

Dodržujte následující pravidla

- Při tvorbě programu postupujte pomalu. Nejdříve vytvořte základní třídou teprve později přidejte odvozenou třídu. Vytvořte vždy nejdříve základ třídy (proměnné a konstruktor bez parametrů) a funkci `main()` a program přeložte, opravte chyby. Potom postupně přidávejte jednotlivé metody, pokaždé přeložte (překladač nesmí hlásit chyby). Pak přidejte odvozenou třídu.
- V každé třídě vytvořte konstruktor a v něm inicializujte členská data (na základě předaných parametrů nebo vhodnými implicitními hodnotami). Pokud by třída neměla konstruktor, členská data zůstala neinicializovaná. **Členská data objektového typu není třeba inicializovat, protože jejich konstruktor se zavolá automaticky a v něm budou inicializována.**
- Při načítání dat od uživatele a při výpisu hodnot vždy vypisujte vhodný informační text, aby uživatel věděl, jaká data má zadat a jaké informace program vypsal.

Cvičení – 1. část

1. Vytvořte program obsahující základní třídu `Graphic` a z ní odvozenou `Circle` s následujícími vlastnostmi:
   - Třída `Graphic` bude obsahovat:
     - souřadnice středu grafického objektu (`int x, y;`)
     - konstruktor s parametry pro inicializaci souřadnic středu:
       `Graphic(int ax, int ay);`
     - metodu `setValues(int ax, int ay)` pro nastavení souřadnic středu
     - metody `getCentreX()` a `getCentreY()` pro získání souřadnic středu
     - metodu `printValues()` pro výpis hodnot datových položek třídy (tj. hodnot `x` a `y`).
   - Třída `Circle` bude obsahovat:
     - hodnotu poloměru kružnice (`int radius;`) a číslo barvy kružnice (`int color;`)
     - konstruktor s parametry pro inicializaci souřadnic středu, poloměru a čísla barvy (souřadnice středu bude předávat konstruktoru předka):
       `Circle(int ax, int ay, int r, int c);`
     - metodu `setValues(int ax, int ay, int r, int c)` pro nastavení souřadnic středu, poloměru a čísla barvy (pro nastavení souřadnic středu bude tato metoda volat metodu předka).
     - metodu `getRadius()` pro získání hodnoty poloměru kružnice a `getColor()` pro získání čísla barvy
     - metodu `printValues()` pro výpis hodnot datových položek třídy (pro výpis souřadnic středu bude tato metoda volat metodu předka)
     - metodu `draw()` pro vykreslení kružnice pomocí knihovny g2
   - Ve funkci `main()` vytvořte objekt typu `Circle` a inicializujte jeho souřadnice a poloměr vhodnými hodnotami. Vypište na výstup hodnoty objektu zavoláním jeho metody `printValues()`, pak ještě zavolejte přímo jeho metodu `printValues()` zděděnou z předka `Graphic` (vypíše jen souřadnice středu). Potom zavolejte metodu `setValues()` která nastaví nové hodnoty (souřadnic středu a poloměru) a opět se vypíší pomocí `printValue()` na výstup. Nakonec zavolá metodu `draw()`.

2 body

Cvičení – 2. část

2. Vytvořte program který bude vycházet z předchocí úlohy, ale implementujte následující modifikace:
   - Vytvořte třídu `Rectangle` (odvozenou z `Graphic`) pro kreslení obdélníku. Třída bude mít členy `width` a `height` (typu `int`) a související metody podobné jako ve třídě `Circle` (tj. `setValues()`, `getWidth()`, `getHeight()`, `printValues()` a `draw()`). Obdélník se bude vždy vykreslovat černou barvou. Pro vykreslení obdélníku použijte funkci g2_rectangle (int dev, double x1, double y1, double x2, double y2).
   - Vytvořte třídu `FilledCircle` (odvozenou z `Circle`) pro kreslení kružnice vyplněné barvou. Třída bude mít člen `fillColor` pro číslo barvy (typu `int`) a související metody podobné jako ve třídě `Circle` (tj. `setValues()`, `getFillColor()`, `printValues()` a `draw()`) (Pozn.: při vykreslování vyplněné kružnice se musí nejdříve nastavit barva výplně pomocí `g2_pen()` a vykreslit vyplněná kružnice pomocí `g2_filled_circle()`, pak nastavit barva kružnice také pomocí `g2_pen()` a kreslit obrys kružnice pomocí `g2_circle()`).
   - V každé třídě dále implementujte metodu `readValues()`, která načte ty právě ty hodnoty, které daná třída potřebuje (a předtím volá obdobnou metodu předka pro mačtení hodnot vyžadovaných předkem).

   Program nabídne uživateli možnost kreslit kružnici, čtverec nebo vyplněnou kružnici. Potom si od něj vyžádá příslušné hodnoty souřadnic, poloměru, barvy atd. (voláním metody `readValues()`). Na konec na obrazovku terminálu vypíše načtené hodnoty (voláním `printValues()`) a vykreslí příslušný objekt. **2 body**

Cvičení – 3. část

3. Vytvořte program který vykreslí obrazec se třemi vyplněnými kružnicemi ohraničenými obdélníkem (viz. obrázek níže), barva výplně kružnic bude specifikována uživatelem. Program bude vycházet z předchocí úlohy. V programu implementujte novou třídu `Drawing`, která bude obsahovat 3 objekty vyplněné kružnice (typu `FilledCircle`) a jeden objekt obdélníku (typu `Rectangle`). Dále bude obsahovat:
   - konstruktor `Drawing()`
   - metodu `readValues()` pro načtení barvy výplně
   - metodu `setFillColor(int fc)` pro nastavení barvy výplně všech tří kružnic
   - metodu draw(), která otevře okno a vykreslí do něj tři vyplněné kružnice a ohraničující obdélník.

   Podle potřeby upravte existující třídy (např. budou potřebovat konstruktory bez parametrů, do třídy `FilledCircle` doplňte metodu `setFillColor(int fc)`, okno pro kreslení bude otevřeno v `Drawing::draw()`, a metody `draw()` grafických tříd budou přijímat parametr s číslem okna a budou provádět pouze operace kreslení do tohoto okna). **1 bod**