

ukazatel this (o)

- základ mechanismu volání metod – metodám se předá ukazatel na aktuální objekt, který metodu zavolal (zajistí překladač)
- this - klíčové slovo
- **T* const this; // „součást“ každého objektu**
- předán implicitně do každé metody (skrytý parametr-překladač)
- při použití metody **aa.Metoda()**; se vlastně „volá“ **Metoda(&aa)** – objekt, který chce „svou“ metodu zavolat je do ní překladačem skrytě předán jako parametr.
- Prototyp metody je napsán v hlavičkovém souboru **void Metoda(void)** ale je „přeložen“ se skrytým parametrem jako **void Metoda(T *const this)** a potom v těle lze používat členské metody a data aktuálního objektu {**this->data1 = 4; this->metodax();**}

Použití this při psaní programu:

- přístup k datům a metodám aktuálního objektu (this je možné vynechat, proměnná třídy je první v rozsahu prohledávání – třída tvoří jmenný prostor)
this->data = 5, b = this->metoda(a)
- objekt vrací sám sebe – **return *this;**
- kontrola parametru s aktuálním prvkem **if (this==¶m)**

ukazatel this

Napište třídu Komplex a v ní metodu, která vrátí maximum ze dvou proměnných typu Komplex.

```

class CComplex {
double Re,Im;
public:
Komplex& Max(Komplex &param) // & reference
{ // this je předán implicitně při překladu
// Komplex& Max(Komplex*const this,Komplex &p...
// rozdíl funkce x metoda

if (this == &param) // & adresa
    return *this;// oba parametry totožné a tak
// nepočítám, => rychlý návrat.
if ( this->Re < param.Re ) return param;
else                return *this;
// param i this existují vně -> lze reference
} // konec funkce
}; // konec třídy

```

volání:

```

Komplex aa,b, c ;
c = aa.Max(b); // neboli Max(&aa,b). Překladem
// se aa uvnitř metody mění v this
c = b.Max(b); // mezivýsledek c = b; this v Max je &b

```

Alternativní hlavičky pro metodu Max. Vysvětlete rozdíly při předávání (kde vznikají dočasné proměnné).

```
Komplex Max(Komplex param)  
Komplex & Max(Komplex const &param)
```

```
c = a.Max(b);
```

c = a.Max(b);

společné volání pro obě hlavičky. Z volání není vidět rozdíl pro předávání hodnotou a referencí.

Komplex Max(Komplex param)

předaný parametr b z volání se stane předlohou pro lokální proměnnou param, která vznikne jako (plnohodnotná lokální) kopie – musí tedy vzniknout a zaniknout objekt. Změní-li se param, nemění se původní objekt b.

Vracený parametr je vrácen hodnotou – musí tedy vzniknout (jako plnohodnotná kopie prvku z return xxx;) a zaniknout.

Komplex & Max(Komplex const ¶m)

pokud se předává objekt pomocí reference, nevytváří se žádný nový objekt, odkazujeme se na objekt původní. Manipulací s prvkem param pracujeme přímo s objektem b. Aby nedošlo k nechtěné změně b, můžeme param označit jako const a potom ho nelze změnit. Vracet referenci můžeme, pouze pokud proměnná existuje ve funkci volající i volané.

Jelikož je předávání parametrů pomocí reference úspornější (časově i paměťově), dáváme mu přednost (v situacích kdy to jde!).