C++; delete Java;

Část 2: RAII

Kennny

srpen 2017

RAII

- Resource Acquisition Is Initialization
- využívá konstruktor a destruktor pro zabrání / uvolnění zdroje
- prakticky vždy staticky alokováno
- to umožňuje vázat RAII objekt na konkrétní scope

RAII

- je to ochrana proti zapomínání
- zpohodlnění práce
- často je navíc využita i explicitní scope pro přehlednost
- na jednoduchých příkladech není moc vidět důležitost a užitečnost principu RAII

Příklad kostry RAII struktury

```
struct WCKeyHolder
{
    public:
        WCKeyHolder()
        {
            acquireKey();
        }

        ~WCKeyHolder()
        {
            releaseKey();
        }
}
```

Příklad použití RAII struktury

```
// sraci scope
{
    // konstruktor zabere zdroj
    WCKeyHolder wckey;

    sit();
    while (!ass.empty())
        ass.pop();
    wipe(ass);
}
// konec explicitni scope zapricini volani destruktoru
// tedy uvolneni zdroje
```

RAII

- spousty věcí v C++ STL nějakým způsobem utilizuje RAII
- souborové streamy - otevírá a zavírá soubor
- mutex a jeho lock_guard - zabírá a uvolňuje mutex
- smart pointery - zvyšují a snižují reference count
- a další...

Příklad

- Prostor pro příklad 02_a_raii

Konec 2. části

```
exit(0);
```