Cvičení 02

Vytvořte třídu **Uctenka**, která obsahuje privátní atributy **int cisloUctenky** (unikátní identifikátor účtenky), **double castka** (částka bez dph)**, double dph** (sazba dph v procentech). Vytvořte metody pro čtení a nastavení atributů.

Vytvořte třídu **Pokladna**, která obsahuje dynamicky alokované pole účtenek o 10 prvcích a atribut **pocetVydanychUctenek**, který představuje počet vydaných účtenek. Dále ve třídě vytvořte privátní statický atribut **citacId**, který slouží pro generování čísel účtenek. Na začátku inicializujte statický atribut na hodnotu 1000. Ve třídě **Pokladna** vytvořte veřejné metody:

- **Uctenka& vystavUctenku(double, double)** – vystaví účtenku, nastaví parametry účtence v poli, inkrementuje *pocetVydanychUctenek* a vrací referenci na vystavenou účtenku.
- **Uctenka& dejUctenku**(**int**) – vyhledá a vrátí účtenku z pole dle *cislaUctenky*. Pokud neexistuje vraťte první účtenku z pole.
- **double dejCastku() const** – vrátí celkovou částku ze všech vystavených účtenek.
- **double dejCastkuVcDph() const** – vrátí celkovou částku vč. DPH ze všech vystavených účtenek (*castkaVcDph* = *castka* * (1 + *dph*).

Ve funkci main() třídy otestujte – vytvořte instanci třídy Pokladna a následně vystavte alespoň 3 různé účtenky. Vyzkoušejte manipulaci s účtenkou vrácenou metodou vystavUctenku() a následně voláním dejUctenku().
Vypočítejte a vypište celkovou částku a celkovou částku bez dph z pokladny.
Program vypracujte do oddělených CPP a H souborů (pro jednotlivé třídy).

Vypracovaný projekt odevzdejte do konce cvičení na STAG! Ve formátu ZIP.

Poznámky:

- deklarace třídy (atributy, úplné funkční prototypy metod) – **H** soubor, definice metod (těla - kód) – **CPP** soubor
- paměť pro **statický** atribut musí být alokována v **CPP** souboru (a také přiřazena výchozí hodnota)
- **`Trida* ptr;`**
  - alokován jako **`new Trida`** nebo **`new Trida[…]`**
  - může být **nullptr** a neukazovat nikam
  - může být platný
    - ukazovat na **jeden objekt**
      `ptr[0] == *ptr`
      `ptr[0].metoda() == ptr->metoda() == (*ptr).metoda()`
    - ukazovat na **pole objektů**
      `typ ptr[0] == Trida`
- **`Trida** ptr;`**
  - Alokován jako **`new Trida*`** nebo **`new Trida*[…]`**
  - může reprezentovat dvourozměrné pole **objektů Trida**
    `typ ptr[0][0] == Trida`
  - může reprezentovat jednorozměrné pole, kde jednotlivé položky jsou **ukazatele na Trida**
    typ ptr[0] == Trida*
- Pomocí reference lze měnit původní objekt. Pokud chci referenci měnit i později musím vytvořit referenční proměnnou – **`Trida& ref = dejReferenci()`**