Datové typy I

Termín odevzdání:	**01.05.2022 23:59:59**	292179.752 sec
Pozdní odevzdání s penalizací:	**15.05.2022 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)	
Hodnocení:	**0.0000**	
Max. hodnocení:	**3.0000** (bez bonusů)	
Odevzdaná řešení:	0 / 20 Volné pokusy + 20 Penalizované pokusy (-2 % penalizace za každé odevzdání)	
Nápovědy:	0 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)	

Úkolem je realizovat sadu tříd, které budou implementovat popis některých datových typů v C/C++.

Kompilátor si během kompilace musí pamatovat vlastnosti datových typů. Některé datové typy jsou v kompilátoru pevně zabudované (např. int, double), jiné si programátor může definovat ve svém programu (odvozené typy: např. enum, struct). Sada implementovaných tříd dokáže zachytit popis takových typů. Konkrétně chceme popisovat pouze některé typy (int, double, enum a struct), navíc pro ně chceme pouze některé operace.

Očekává se, že implementujete následující třídy s následujícím rozhraním:

CDataTypeInt

Třída reprezentuje primitivní typ `int`. Má následující rozhraní:

implicitní konstruktor
: inicializuje instanci třídy,

getSize()
: metoda vrátí velikost typu, zde vždy 4,

operátor ==
: porovná tento typ s jiným typem, vrátí true, pokud jsou oba typy shodné (oba jsou typy int),

operátor !=
: porovná tento typ s jiným typem, vrátí true, pokud jsou typy rozdílné,

operátor <<
: zobrazí název typu do zadaného proudu.

CDataTypeDouble

Třída reprezentuje primitivní typ `double`. Má následující rozhraní:

implicitní konstruktor
: inicializuje instanci třídy,

getSize()
: metoda vrátí velikost typu, zde vždy 8,

operátor ==
: porovná tento typ s jiným typem, vrátí true, pokud jsou oba typy shodné (oba jsou typy double),

operátor !=
: porovná tento typ s jiným typem, vrátí true, pokud jsou typy rozdílné,

operátor <<
: zobrazí název typu do zadaného proudu.

CDataTypeEnum

Třída reprezentuje výčtový typ. Má následující rozhraní:

implicitní konstruktor
: inicializuje instanci třídy,

getSize()
: metoda vrátí velikost typu, zde vždy 4,

add()
: metoda přidá další hodnotu do výčtu. Pokud stejná hodnota ve výčtu již existuje, je vyvolaná výjimka (viz ukázkový běh),

operátor ==

porovná tento typ s jiným typem, vrátí true, pokud jsou oba typy shodné (oba jsou výčtové typy a mají stejný výčet hodnot ve stejném pořadí),

operátor !=
: porovná tento typ s jiným typem, vrátí true, pokud jsou typy rozdílné,

operátor <<
: zobrazí název typu do zadaného proudu. Pozor, hodnoty výčtu musí být zobrazené v pořadí zadávání.

CDataTypeStruct

Třída reprezentuje typ struktura. Má následující rozhraní:

implicitní konstruktor
: inicializuje instanci třídy,

getSize()
: metoda vrátí velikost typu (dle obsažených složek),

addField(name, type)
: metoda přidá další složku zadaného jména a typu (int/double/enum) na konec seznamu složek. Pokud je jméno složky duplicitní, vyhlásí výjimku (viz ukázkový běh),

field(name)
: zpřístupní složku zadaného jména, případně vyhodí výjimku pokud složka takového jména neexistuje (viz ukázkový běh). Složka zpřístupněná touto metodou bude pouze čtena,

operátor ==
: porovná tento typ s jiným typem, vrátí true, pokud jsou oba typy shodné (oba typy jsou struktury, mají stejný počet složek, složky na každé pozici mají stejný typ, jména složek se ale mohou lišit),

operátor !=
: porovná tento typ s jiným typem, vrátí true, pokud jsou typy rozdílné,

operátor <<
: zobrazí název typu do zadaného proudu. Pořadí složek odpovídá pořadí jejich přidání metodou `addField`.

Odevzdávejte zdrojový kód s implementací tříd `CDataTypeInt`, `CDataTypeDouble`, `CDataTypeEnum` a `CDataTypeStruct`. Třídy musejí implementovat rozhraní popsané výše. Do tříd si ale můžete doplnit další metody, které Vám implementaci zjednoduší.

Poznámky:

- Při implementaci můžete využít STL a typ c++ string. Základem implementace je ale dobrý návrh tříd a využití polymorfismu. Vaše řešení nebude uznáno, pokud nebude využívat polymorfismus.
- Při porovnání zápisu typů (výstupy operátoru <<) se nekontrolují bílé znaky. Pro testování si v ukázkovém použití implementujete funkci `whitespaceMatch`), která porovná řetězce bez ohledu na bílé znaky.
- Pro návratovou hodnotu metody `getSize` použijte typ `size_t`.
- Ukázka použití tříd je v přiloženém archivu.
- Řešení této úlohy **nelze** použít pro code review. (Pro code review lze použít správné řešení navazující úlohy - Datové typy II.)
- **Dodatek 9.4.2022:** pokud je vstupním parametrem metody řetězec, pak umožněte použití jak C-řetězece (`const char *`), tak i C++ řetězece (`std::string`).

Vzorová data: Download

Odevzdat: Choose File No file chosen Odevzdat

Referenční řešení