12. LABOR

TÍPUSKONVERZIÓK KEZELÉSE

Általános információk

Az összes feladat hibátlan megoldására 1 iMSc pont kapható.

Kötelező feladatok

1. String osztály: közös áttekintés

Fejleszd tovább a kiadott *String* osztályt (*Conversion* solution), amely a szokásos operátorokon kívül felhasználható C stílusú nullterminált stringként.

Tervezzétek meg közösen az osztályt, főleg konverzió szempontjából!

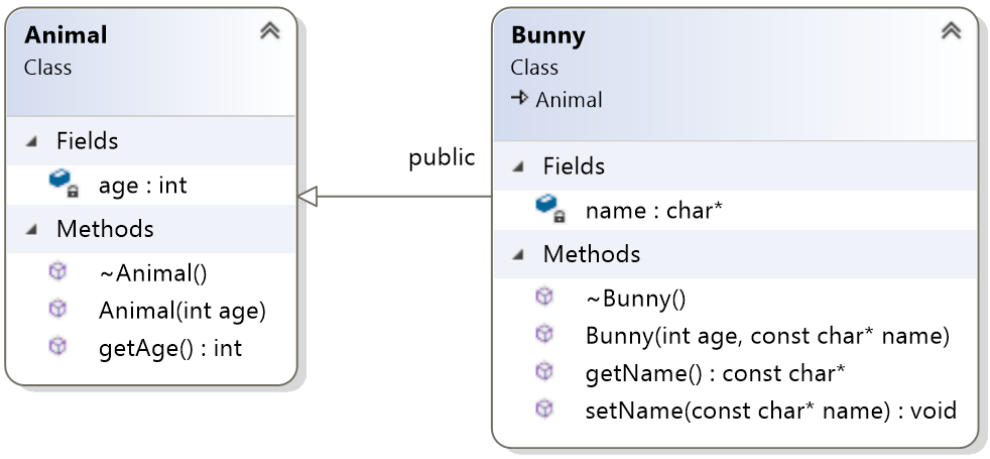
- Miért a *const char** konverziós operátort definiáljuk felül, és miért nem a *char * operátort?
- Miért érdemes reprezentációt váltani az előzőhez képest?

2. String osztály: tesztelés

- **Helyezz el töréspontokat** az összes konstruktor és konverziós operátor elején! Haladj át a *stringTest()* függvény minden során, és indokold meg, miért hívta meg azt a fordító.
- Mi a különbség a következő két sor között?
  - `cout << str1 << endl;`
  - `cout << static_cast<const char*>(str1) << endl;`
- Ha kihagyod az explicit típuskonverziót („*(String)*”) az alábbi kifejezésből, hibaüzenet keletkezik:
  - `str1 == (String)"My name is Bond. James Bond."`
  - Mi okozza? Találd ki a többértelműség okát!
- Mutasd meg, hogy a *operator+=* nem *char**-ot vár, de működik és egy ideiglenes *String* jön létre a "James Bond." -ból, ha a `const char*-t` váró konstruktor nem `explicit`!
  - `str1 += "James Bond.";`

3. Animal és Bunny osztályok UML diagramm alapján

Készíts egy Animal és egy Bunny osztályt a következő UML diagramm alapján:



1. ábra Animal és Bunny

Ügyelj a következőkre:

- Használd az `explicit` és `virtual` kulcsszavakat ott, ahol kell!
- Ne legyen memóriaszivárgás!
- Minden accessor függvény legyen `const` tagfüggvény!

4. Animal és Bunny osztályok tesztje

A test.cpp-ben szedd ki a kommentet a `animalTest();` elől, hogy lefusson az előbb megírt két osztály tesztje.

Összegzésképp tisztázzátok közösen, hogy mikor jöttek elő a következő fogalmak a labor során! Mire érdemes vigyázni velük kapcsolatban? Mik az előnyeik, hátrányaik?

- C típusú castolás
- Implicit típuskonverzió
- Explicit típuskonverzió
- Konverziós operátor
- Konverziós konstruktor
- **`static_cast`** `<` *new_type* `>` `(` *expression* `)`
- **`dynamic_cast`** `<` *new_type* `>` `(` *expression* `)`
- **`const_cast`** `<` *new_type* `>` `(` *expression* `)`
- **`reinterpret_cast`** `<` *new_type* `>` `(` *expression* `)`