Cpp Tricks'n'Quirks #1 [triggering ctors-VS-asgns] fix (wide. 80)
RKr (edit. 5)

CASE **[A1]**

```
int x = 7;
```

W przykładzie **[A1]** deklaracja zmiennej x jest połączona z inicjalizacją.

CASE **[A2]**

```
int x;
x = 7;
```

W przykładzie **[A2]** deklaracja zmiennej x jest oddzielona od inicjalizacji.
Jednak semantyka **[A1]** i **[A2]** jest prawie identyczna, prawie,
dlatego, że x będzie miał chwilowo przypadkową wartość.

CASE **[B1]**

```
ClassName c2(__);
ClassName c1 = c2;
```

W przypadku **[B1]**, kopiowanie c2 do c1 realizowane jest konstruktorem kopiującym,
o sygnaturze takiej jak poniżej.

```
ClassName(const ClassName&);
```

CASE **[B2]**

```
ClassName c2(__);
ClassName c1;
c1 = c2;
```

W przypadku **[B2]**, kopiowanie c2 do c1 realizowane jest nie poprzez konstruktor,
a poprzez operator przypisania kopiującego o sygnaturze takiej jak poniżej.

```
ClassName& ClassName::operator=(const ClassName&);
```

Pomimo tego, że przykłady **[B*]** wyglądają bardzo podobnie do przykładów **[A*]**,
i wydaje się, że powinny być ekwiwalentne,
to pomiędzy **[B1]** i **[B2]** jest znacząca różnica w semantyce wykonania programu.
Jeśli **[*1]** usuniemy konstruktor kopiujący (np. stosując modyfikator "=delete"),
[B1] zgłosi błąd podczas kompilacji, ale **[B2]** raczej zadziała.
Określenie "raczej" jest motywowane tym,
że "used-defined asgn" może zawierać inwokacje konstruktorów kopiujących.
Jeśli **[*1]** usuniemy operator przypisania kopiującego,
to **[B2]** odmówi kompilacji, ale **[B1]** skompiluje się i będzie działał.

ad**[*1]** Usunięcie może się zaistnieć wskutek różnych okoliczności.
Jednym scenariuszem jest usunięcie niejawne,
czyli "implicit deletion of implicitly-declared constructor/operator".
Scenariusz taki zachodzi jeśli spełnione są pewne warunki,
które zawarte są w regułach języka dotyczących obsługi konstruktorów/przypisań.
Innym scenariuszem jest użycie modyfikatora "=delete",
które dokonuje jawnej prewencji/supresji generowania domyślnych elementów,
czyli "implicitly-declared constructor / assignment".

Przykład kodu, w którym dokonano jawnego usunięcia pewnych elementów klasy CC.
Wiersze zawierające tworzenie/kopiowanie obiektów udekorowane są komentarzami,
które sygnalizują jakich wydarzeń oczekujemy.
Wydarzenia te *być może* mogą się zmieniać względem specyfikacji języka,
implementacji kompilatora oraz wybranego poziomu optymalizacji,
jednak (używając www.onlinegdb.com) zgadzają się z moją wiedzą/intuicjami.

Warto zwrócić uwagę na to, że komunikaty kompilatora sygnalizują braki rzeczy,
które są nadmienione w komentarzach, kompilator wyświetla nawet ich kopię.

```cpp
#include <stdio.h> // assuming this line to be line 1

class CC {
  public: int v;

  CC(int value) : v(value) { printf("%d\n", v); }

//__/// play with empty-ctor(s)
  CC() =default;
//  CC() =delete;

//__/// play with copy-ctor(s)
//  CC(const CC& cIn) =default;
  CC(const CC& cIn) =delete;

//__/// play with move-ctor(s)
//  CC(CC&& cIn) =default; /// constexpr CC(CC&& cIn) =default;
  CC(CC&& cIn) =delete; /// constexpr CC(CC&& cIn) =delete;

//__/// play with copy-asgn(s)
//  CC& operator=(const CC& cIn) =default;
  CC& operator=(const CC& cIn) =delete;

//__/// play with move-asgn(s)
//  CC& operator=(const CC&& cIn) =default;
  CC& operator=(const CC&& cIn) =delete;
};

int main()
{
  CC c1; // calling empty-ctor
  CC c2 = c1; // calling copy-ctor (contrary to = being used)

  CC c3; // calling empty-ctor
  CC c4; // calling empty-ctor
  c4 = c3; // calling copy-asgn (accordingly to = being used)

  CC c5 = CC(7); // calling user-ctor + move-ctor afterwards
                 //  (contrary to = being used, asgn isn't called)
                 //   (may depend on optimization and C++Version though)
                 //
                 // The user-ctor will make a local (stackframe-placed) object
                 // that will eventually undergo move-semantics via move-ctor.

  printf("Tests punched-through!!");
  return 0;
}
```

```
================================================================================
 COMPILER ERRORS (C++14) \ Kompilator (C++17) nie zgłasza main.cpp:38:15
================================================================================
main.cpp: In function ‘int main()’:
main.cpp:32:11: error: use of deleted function ‘CC::CC(const CC&)’
   CC c2 = c1; // calling copy-ctor (contrary to = being used)
           ^~
main.cpp:14:3: note: declared here
   CC(const CC& cIn) =delete;
   ^~
main.cpp:36:8: error: use of deleted function ‘CC& CC::operator=(const CC&)’
   c4 = c3; // calling copy-asgn (accordingly to = being used)
        ^~
main.cpp:22:7: note: declared here
   CC& operator=(const CC& cIn) =delete;
       ^~~~~~~~
main.cpp:38:15: error: use of deleted function ‘CC::CC(CC&&)’
   CC c5 = CC(7); // calling user-ctor + move-ctor afterwards
               ^~
main.cpp:18:3: note: declared here
   CC(CC&& cIn) = delete; /// constexpr CC(CC&& cIn) =delete;
   ^~
================================================================================
```