zdefiniujemy, wówczas kompilator postara się o wygenerowanie swojego – przypisującego na zasadzie „składnik po składniku".

Słowa „postara się" dobrze oddają tu sytuację. Mianowicie czasem to generowanie może się okazać niemożliwe.

- Jeżeli klasa ma składnik const, to operator nie będzie wygenerowany. Jest to oczywiste, bo skoro w klasie jest jakiś składnik ustanowiony jako const to znaczy, że dopuszczamy jego inicjalizację, ale potem nie wolno już go zmieniać, czyli nic do niego przypisywać.
- Podobnie w przypadku obecności składnika będącego referencją. Jak wiemy, referencję (przezwisko) tylko się inicjalizuje, a potem już przepadło – nie można rozmyślić się i przerzucić przezwisko na inny obiekt.
- Jeśli klasa (np. radio) ma składnik będący obiektem innej klasy (np. tranzystor) i w tej innej klasie (tranzystor) operator przypisania określony jest jako private – to wówczas nie będzie generowany operator przypisania dla klasy go zawierającej (klasa radio).

  To zrozumiałe – jeśli ktoś określa operator= jako private, to znaczy, że nie chce, aby przypisywanie do obiektu odbywało się spoza tej klasy. Klasa zawierająca nie może go użyć.

Dla wtajemniczonych:

- analogicznie jest w przypadku, gdy klasa ma klasę podstawową, w której operator= jest typu private.

Dla wtajemniczonych: przypomnienie o dziedziczeniu operatorów

Jeżeli operator jest funkcją składową klasy, to może być dziedziczony. Dotyczy to wszystkich operatorów oprócz operatora przypisania '='. (Zob. nast. rozdział, str. 803).

19.12 Operator [ ]

Kontynuujemy naszą opowieść o czterech operatorach, które mogą być tylko niestatycznymi funkcjami składowymi odpowiedniej klasy.

Operator [ ] – "odwołanie się do elementu tablicy" – jest operatorem dwuargumentowym. Jeśli chcemy przeładować ten operator, to funkcja operatorowa musi być niestatyczną (czyli zwykłą) funkcją składową klasy.

Jeśli mamy obiekt klasy K i dokonamy przeładowania tego operatora dla tej klasy, to wówczas wyrażenie

```
K obiekt;

obiekt[argument]
```

odpowiada wyrażeniu

```
obiekt.operator[ ](argument)
```

Przypomnę po raz kolejny, że – także i tutaj – treść operatora nie musi mieć nic wspólnego ze znaczeniem, jakie ma on dla typów wbudowanych. Oczywiście lepiej jednak, jeśli służy nam do podobnych celów.

Godny zwrócenia uwagi jest fakt, że argument nie musi być wcale typu int

Było to nie do pomyślenia dla typów wbudowanych. Pisaliśmy przecież wyrażenia:

```
double tablica[30];
```