

# Funkcje wirtualne, dziedziczenie wielokrotne

---

Wykład 7

# Konwersje standardowe przy dziedziczeniu

- Wskaźnik lub referencja do obiektu klasy pochodnej może być niejawnie przekształcony na wskaźnik (referencję) dostępnej jednoznacznie klasy podstawowej (tylko przy dziedziczeniu publicznym)
- Konwersje zachodzą
  - Przy przesyłaniu argumentów do funkcji - funkcja która powinna otrzymać wskaźnik (referencję) do obiektu klasy podstawowej może otrzymać wskaźnik (referencję) do obiektu klasy pochodnej
  - Przy zwracaniu przez funkcję rezultatu - funkcja, która powinna zwrócić wskaźnik (referencję) do obiektu klasy podstawowej może zwrócić wskaźnik (referencję) do klasy pochodnej

# Konwersje standardowe przy dziedziczeniu...

- Konwersje zachodzą...
  - Przy przeładowanych operatorach - analogicznie jak w przypadku funkcji
  - W wyrażeniach inicjalizujących - do konstruktora kopiującego spodziewającego się referencji do obiektu klasy podstawowej można wysłać referencję do obiektu klasy pochodnej
- Obiekty klas pochodnych mogą być traktowane jak obiekty swych klas podstawowych tylko wtedy, gdy pracujemy na ich adresach (referencjach)
- **UWAGA:** Do funkcji spodziewającej się adresu tablicy obiektów klasy podstawowej nie można wysłać adresu tablicy obiektów klasy pochodnej
- Przykład `cpp_7.1`

# Funkcje wirtualne

- Dopiero przy funkcjach wirtualnych pojawia się możliwość faktycznego programowania (z)orientowanego obiektowo
- Realizacja klas podstawowych i pochodnych w zasadzie jest taka sama ale
  - Pojawia się słowo **virtual** przy nazwach funkcji składowych (niekoniecznie wszystkich), które umożliwia wykonywanie różnych funkcji w zależności od typu obiektu na rzecz, którego chcemy taką funkcję wywołać
  - Wykorzystanie funkcji wirtualnych może w znakomity sposób ułatwić i uprościć nam pracę na projektem

# Posługiwanie się funkcjami wirtualnymi

- Możliwe jest ustawienie wskaźnika (referencji) typu klasy podstawowej tak, żeby pokazywał na obiekt klasy pochodnej
  - Wynika to z uogólnienia tzn. klasa pochodna jest szczególnym (bardziej wyspecjalizowanym) typem klasy podstawowej
    - Np. mając wskaźnik do pojazdów możemy nim pokazywać na samochód lub nawet na „malucha” i nie jest to niezgodne z naszym wyobrażeniem o rzeczywistości (samochód jest rodzajem pojazdu)
- Sytuacja odwrotna nie jest już prawdziwa
  - Np. mając wskaźnik do samochodu nie możemy nim (w ogólności) pokazywać na pojazd, gdyż pojazdem może np. być rower

# Różnica między funkcją wirtualną a zwykłą

- Mając wskaźnik klasy podstawowej pokazujący na obiekt klasy pochodnej wywołanie metody
  - Niewirtualnej (zwykłej) spowoduje wywołanie odpowiedniej funkcji składowej w klasie podstawowej - zupełnie normalna sytuacja
    - Przykład cpp\_7.2
  - Wirtualnej spowoduje wywołanie odpowiedniej funkcji składowej uzależnionej od typu obiektu, na który w danym momencie pokazuje wskaźnik
    - Przykład cpp\_7.3
- Dodanie przymiotnika **virtual** przy funkcji składowej w klasie podstawowej mówi, że od tego momentu wszystkie dalsze pokolenia będą tą funkcję mieć wirtualną
  - Tylko wtedy kiedy taka funkcja jest identyczna tzn. posiada taką samą nazwę, przyjmuje takie same parametry oraz zwraca taki sam typ

# Polimorfizm

- Polimorfizm - wielość form
  - Jest to wykazywanie przez metodę różnych form działania w zależności od tego jaki typ obiektu aktualnie jest wskazywany przez wskaźnik lub referencję
  - **Shape→Rys () ;** oznacza w zależności od kontekstu
    - **Shape→Shape::Rys () ;**
    - **Shape→Circ::Rys () ;**
    - **Shape→Rec::Rys () ;**
  - Sama funkcja wirtualna polimorfizmu nie wykazuje
  - Funkcja nie jest polimorficzna, ale tylko jej wywołanie jest

# Jaki jest pożytek z polimorfizmu

- Program jest rozszerzalny o nowe obiekty (typy), a ich dodanie nie wymaga zmian w już istniejącym kodzie
  - W szczególności w miejscach, gdzie decyduje się jakiej klasy jest obiekt pokazywany przez wskaźnik lub nazywany referencją
  - Nie wymaga instrukcji wyboru
- Jednak są również wady
  - „Inteligentne” zachowanie kompilatora w stosunku do funkcji wirtualnych okupione jest dłuższym czasem ich wywoływania
  - Oraz tym, iż obiekty danej klasy są większe od odpowiadających im obiektów klasy, która nie zawiera funkcji wirtualnych
  - Dlatego funkcje nie są z założenia wirtualne (w C++)
- Przykład `cpp_7.4`



# Wczesne i późne wiązanie

- Wczesne wiązanie następuje w sytuacji kiedy wywoływane są zwykłe funkcje i na etapie kompilacji wywołania funkcji powiązane zostają z adresami, pod którymi te funkcje się znajdują
  - Inaczej wiązanie w czasie kompilacji
- Późne wiązanie występuje w sytuacji kiedy posługujemy się funkcjami wirtualnymi. Kiedy kompilator widzi funkcję wirtualną to nie podstawia określonego adresu, ale generuje odpowiedni kod pozwalający na wybór określonej wersji funkcji na etapie wykonania programu
- W wywołaniu funkcji wirtualnych może wystąpić wczesne wiązanie jeżeli już na etapie kompilacji wiadomo dokładnie, która wersja funkcji ma zostać wywołana
  - Wywołanie na rzecz obiektu (`obiekt.funkcja()`)
  - Jawne użycie kwalifikatora zakresu (`wskaźnik->klasa::funkcja()`)
    - Należy stosować jeżeli sięgamy do składników klasy podstawowej z funkcji składowych klasy pochodnej
  - Wywołanie funkcji z konstruktora lub destruktora klasy podstawowej