

Programowanie obiektowe

10.04.2017

Przykład wykorzystania obiektu

Obiekt X jest instancją klasy Y

Obiekt = tożsamość + stan + zachowanie

```
vector<int> v;
```

```
v.push_back(5);
```

```
v.push_back(6);
```

```
v.push_back(3);
```

```
cout << "Aktualnie w wektorze jest " << v.size()  
      << " elementow" << endl;
```

```
cout << "Pierwszym elementem jest" << v.front() << endl;
```

Obiekt = dane + metody

Przykład

```
class A{
private:
    int x, y;
public:
    void setValueX(int a)
    {
        x = a;
    }
    void setValueY(int b);
};
```

```
int main() {
A obj;
obj.x = 3;
obj.setValueY(3);
...
}
```

← niedozwolone

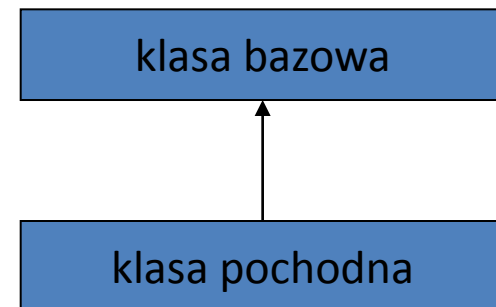
Koncepcja	Realizacja w C++
Abstrakcja	Klasy
Enkapsulacja	Klasy
Ochrona informacji	<i>public/protected/private</i>
Polimorfizm	Przeładowanie operatorów, wzorce, metody wirtualne
Dziedziczenie	Dziedziczenie klas

Dziedziczenie

- Dziedziczenie pozwala stworzyć nową klasę przy wykorzystaniu już istniejącej klasy.
- Dziedziczenie to modyfikacja typu polegająca na jego przystosowaniu do określonych warunków – jest to więc rodzaj specjalizacji.

Dziedziczenie

- Nomenklatura:
 - klasa bazowa (podstawowa albo nadklasa) to klasa, z której dziedziczą inne klasy;
 - klasa pochodna (podklasa) to nowa klasa, która dziedziczy strukturę informacyjną i funkcjonalność z innej klasy.
- Rysunek schematyczny:



Przykład

- Klasa bazowa – Macierz
- Klasa pochodna - MacierzKwadratowa

```
class Macierz {
public:
    Macierz(int n, int m);

    int liczba_wierszy();
    int liczba_kolumn();

    void wypisz();

    void ustaw_wartosc(int wiersz, int kolumna, double wartosc);
    double wartosc(int wiersz, int kolumna);

    void pomnoz_przez(double r);

private:
    int w, k;
    vector< vector<double> > dane;
};
```



```
class MacierzKwadratowa : public Macierz {  
public:  
    MacierzKwadratowa(int n);  
  
    double wyznacznik();  
};
```

Zasada podstawienia Liskov

Funkcje które używają wskaźników lub referencji do klas bazowych, muszą być w stanie używać również obiektów klas dziedziczących po klasach bazowych, bez dokładnej znajomości tych obiektów

Kontrprzykład:

```
void przetwarzajFigureę(Figura& iFigura)
{
    if(typeid(iFigura) == typeid(Prostokąt))
        przetwarzajProstokąt(static_cast<Prostokąt&>(iFigura));
    else if(typeid(iFigura) == typeid(Okrag))
        przetwarzajOkrag(static_cast<Okrag&>(iFigura));

    else if(typeid(iFigura) == typeid(Kwadrat))
        przetwarzajKwadrat(static_cast<Kwadrat&>(iFigura));
}
```

Dostęp do składników

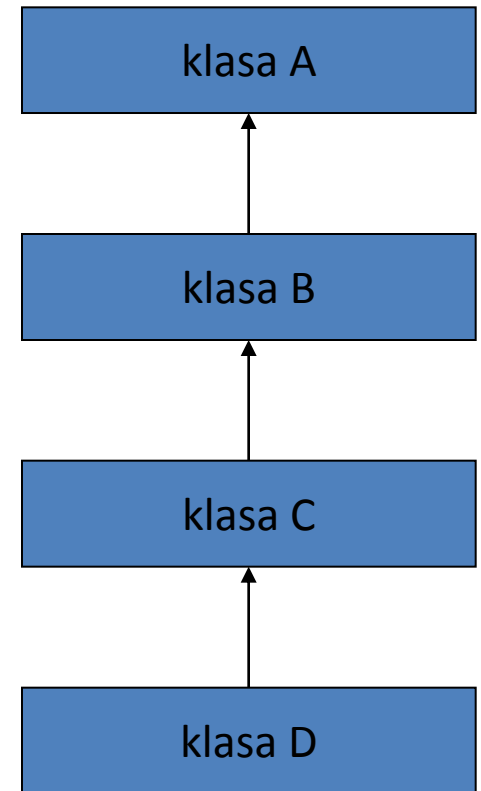
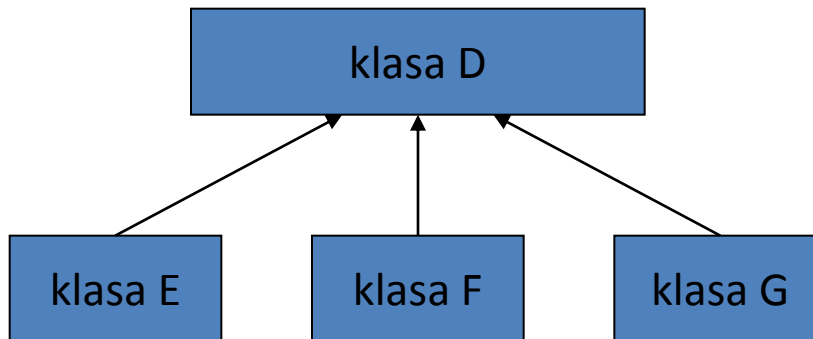
- W klasie pochodnej nie ma dostępu do odziedziczonych składników prywatnych (czyli `private`).
- W klasie pochodnej jest dostęp do odziedziczonych składników nieprywatnych (czyli `protected` i `public`).
- Składniki chronione (czyli `protected`) są dostępne tylko w klasie bieżącej i w klasach pochodnych ale nie na zewnątrz.

Dostęp do składników

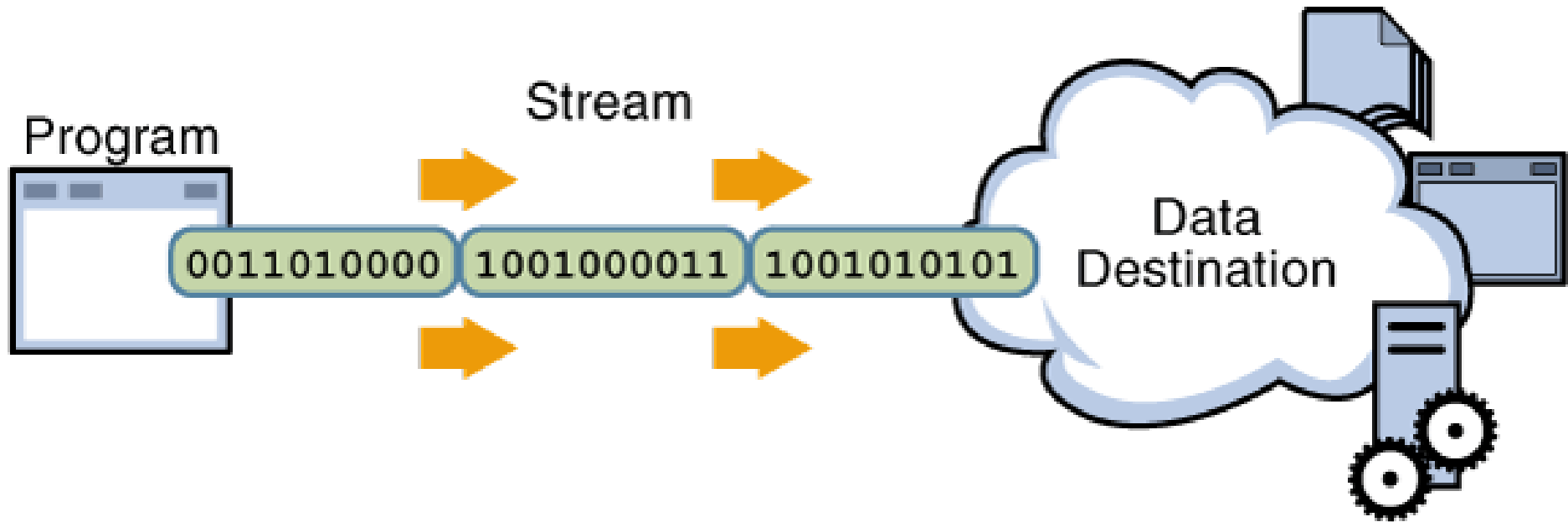
- Klasa pochodna też decyduje o zakresie widoczności odziedziczonych składników nieprywatnych poprzez sposób dziedziczenia (`public`, `protected`, `private`):
 - przy dziedziczeniu publicznym odziedziczone składniki nieprywatne zachowują swój zakres widoczności;
 - przy dziedziczeniu chronionym odziedziczone składniki nieprywatne stają się chronione;
 - przy dziedziczeniu prywatnym odziedziczone składniki nieprywatne stają się prywatne.
- Domyślny sposób dziedziczenia to `private`.

Hierarchia klas

- Dziedziczenie może mieć wiele poziomów.
- Jedna klasa może być klasą bazową dla wielu innych klas.



Dwa przykłady – ROOT i Qwt

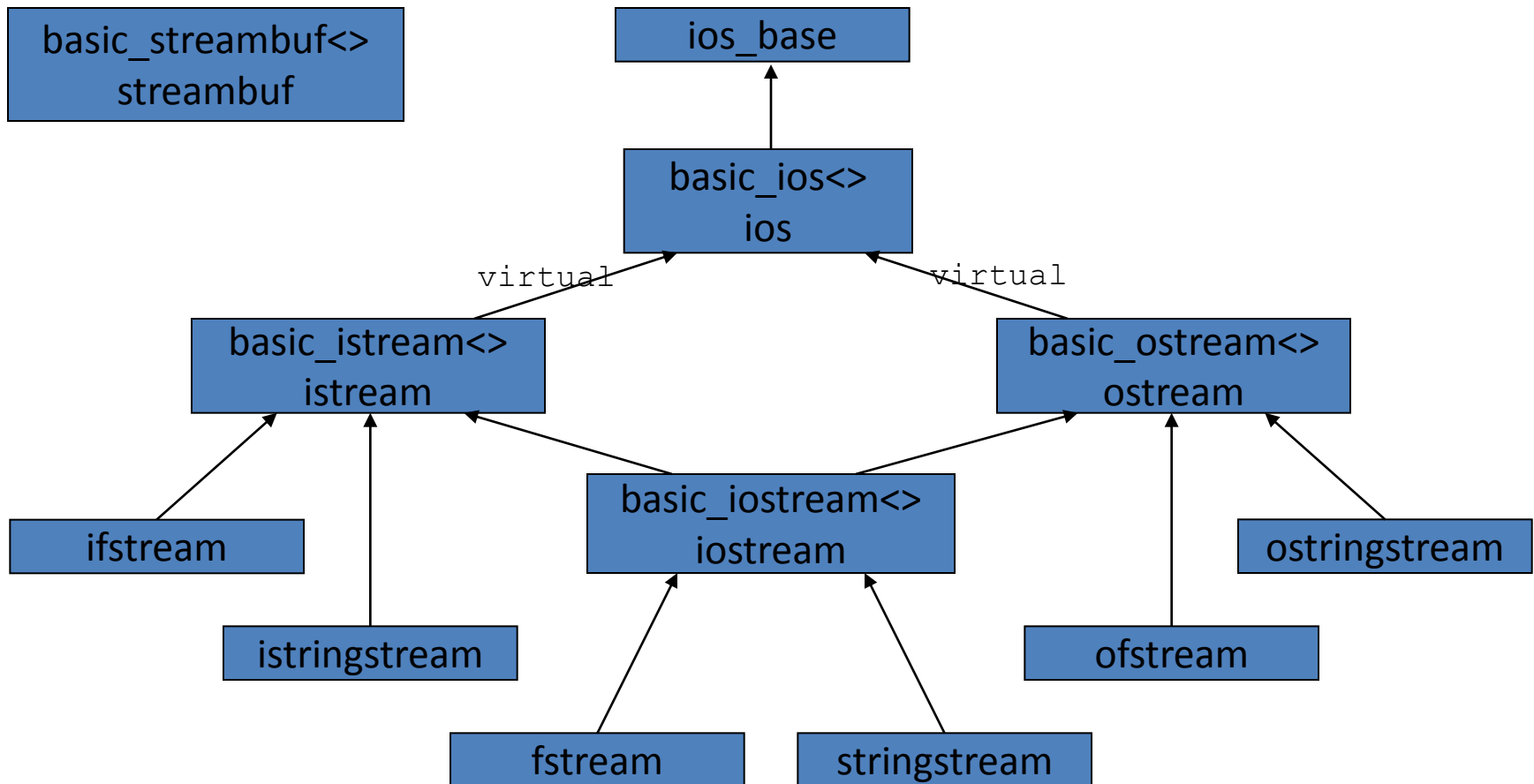


Strumienie

Strumień standardowy
biblioteka:
`<iostream>`



Hierarchia klas strumieniowych realizujących operacje we/wy



Pobieranie danych ze strumienia

operator lub funkcja	Uwagi
>>	Operator >> domyślnie pomija białe znaki. Wczytuje ciąg znaków do pojawiania się kolejnego białego znaku
<code>get(char* gdzie, int ile, char ogran = '\n')</code>	Ogranicznik nie jest wyjmowany ze strumienia
<code>getline(char* gdzie, int ile, char ogran = '\n')</code>	Ogranicznik jest wyjmowany ze strumienia
<code>read(char* gdzie, int ile)</code>	Nie dopisuje na koncu stringu znaku NULL

Zapisywanie danych do strumienia

operator lub funkcja	Uwagi
<<	przekierowanie do strumienia
put(char znak)	Wstawia do strumienia jeden znak
write(char* skąd, int ile)	Wstawia do strumienia określoną ilość danych

```
#include<iostream>
#include<sstream>
#include<string>

using namespace std;

// wypisanie pierwszej kolumny z 2-kol pliku
int main(int argc, char *argv[])
{
    string line;
    double x,y;
    while(getline(cin, line)){
        istringstream stream(line); // zamiana string na strumien

        stream >> x >> y;
        cout << x << endl;
    }
    return 0;
}
```

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
    string a;
    cout << "Nacisnij Enter aby zakonczyc zapis.\n";
    ofstream f("log.txt");
    cin >> a;
    if (f.good())
    {
        f << a;
        f.close();
    }
    return 0;
}
```

```
#include <iostream>
#include <sstream>
using namespace std;

int main ()
{
    long x;
    string napis;
    stringstream ss;
    cout << "Podaj dowolna liczbe calkowita: ";
    cin >> x;
    ss << x;
    napis = ss.str();
    cout << "Dlugosc napisu wynosi "
         << napis.size() << " znakow." << endl;
    return 0;
}
```

```
void wypisz_date(ostream &strumien, int dzien, int miesiac, int rok) {  
    strumien << dzien << "." << miesiac << "." << rok << endl;  
}
```

```
#include <iostream>
using namespace std;

class Date
{
    int mo, da, yr;
public:
    Date(int m, int d, int y) {
        mo = m; da = d; yr = y;
    }
    friend ostream& operator<<(ostream& os, const Date& dt);
};

ostream& operator<<(ostream& os, const Date& dt)
{
    os << dt.da << '.' << dt.mo << '.' << dt.yr;
    return os;
}

int main()
{
    Date dt(5, 6, 92);
    cout << dt;
}
```



```
class Box
{
    public:
        // konstruktor
        Box(double l=2.0, double b=2.0, double h=2.0)
        {
            cout <<"Constructor called." << endl;
            length = l;
            breadth = b;
            height = h;
            // każdy obiekt zwiększa licznik
            objectCount++;
        }
        double Volume()
        {
            return length * breadth * height;
        }
    private:
        double length, breadth, height;
        static int objectCount;
};

// Initialize static member of class Box
int Box::objectCount = 0;
```