Programowanie

Dariusz Wardecki, wyk. XIII

Powtórzenie

Co zostanie wypisane?

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

int main(int argc, char *argv[]){
  char buff[10];
  ifstream plik;
  plik.open(argv[1]);
  plik.seekg(3);
  plik.read(buff, 10);
  cout << buff << endl;
  plik.close();
  return 0;
}
```

```
1194822060  1.2  983.7  83.6567
1194822120  1.2  983.7  83.7717
1194822180  1.22833  983.7  83.6433
1194822240  1.3  983.7  83.585
1194822300  1.3  983.7  83.8283
```

Powtórzenie

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

int main(int argc, char *argv[]){
  char buff[100];
  int i = 0;
  ifstream plik;
  plik.open(argv[1]);
  ofstream out(argv[2]);
  while(plik.getline(buff, 100)) i++;
  out << i << endl;
  plik.close();
  out.close();
  return 0;
}
```

Co robi program?

Funkcja `getline`

Dwie różne funkcję!

`getline(char*,int)`	`getline(stream&,string&)`

`char buf[n];`
`ifstream in;`
`in.open("dane.txt");`
`in.getline(buf,n);`
	`ifstream in;`
	`string str;`
	`in.open("dane.txt");`
	`getline(in, str);`

`char buf[n];` `cin.getline(buf,n);`	`string str;` `getline(cin, str);`

Klasa `istringstream`

`istringstream` - konwersja stringa w strumień

Przykładowy plik

```
1194822060  1.2  983.7  83.6567
1194822120  1.2  983.7  83.7717
1194822180  1.22833  983.7  83.6433
```

```
double x, y, z, v;
string str;
ifstream in("plik");
getline(in, str);                  //zapisanie jednej linii z pliku
                                   //do stringu
istringstream strumien(str);       //zamiana stringu str w strumien
strumien >> x >> y >> z >> v;      //wypisanie danych ze strumienia
                                   //do zmiennych
```

Dziedziczenie

operator widoczności

```
class Zwierze
   {
         public:
               Zwierze();
               void jedz();
   };
```

```
class Ptak : public Zwierze
   {
         public:
               Ptak();
               void lec();
   };
```

```
class Ryba : public Zwierze
   {
         public:
               Ryba();
               void plyn();
   };
```

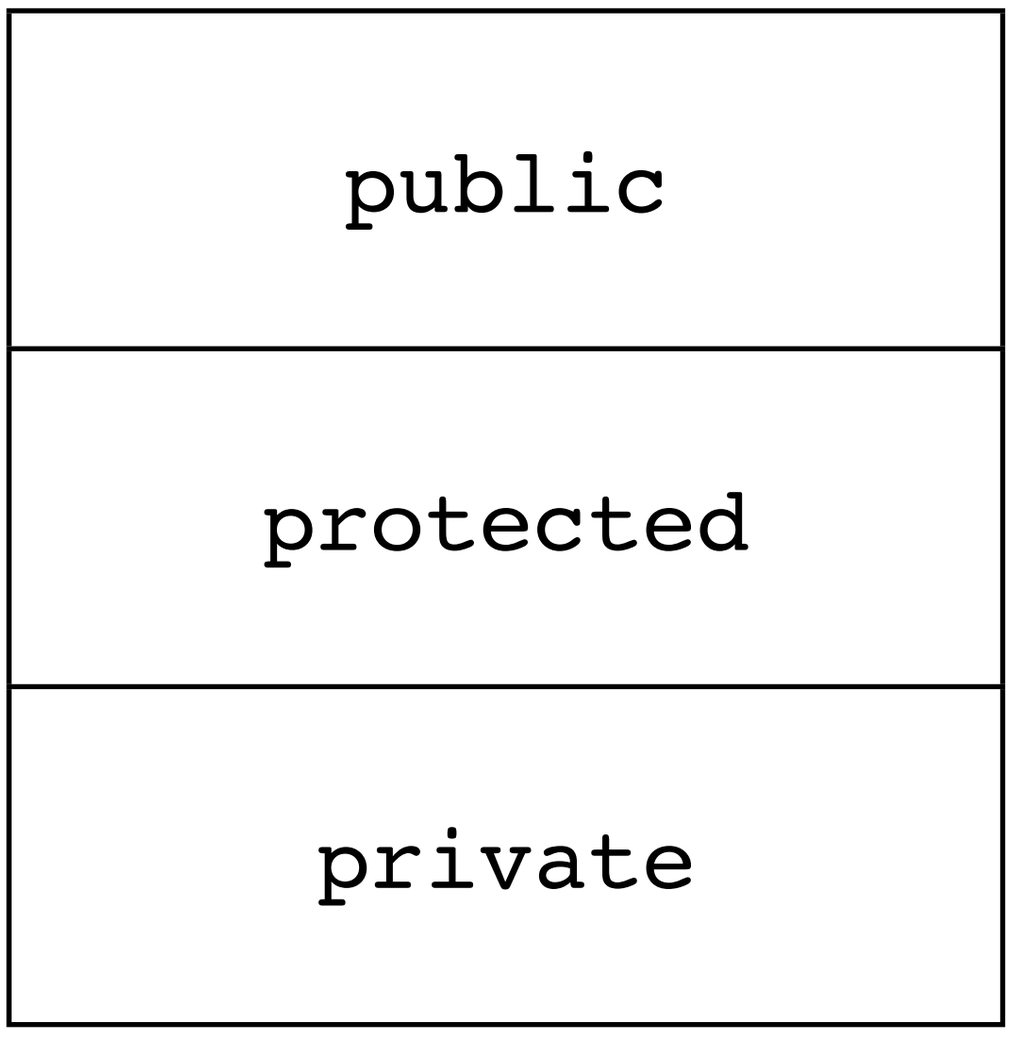
Dziedziczenie

```
Ptak ptak;
ptak.jedz(); //metoda z klasy Zwierze
ptak.lec(); //metoda z klasy Ptak

Ryba *ryba=new Ryba();
ryba->jedz(); //metoda z klasy Zwierze
ryba->plyn(); //metoda z klasy Ryba
```

Zakres dostępu

Zakres dostępu do składników klas



Operator widoczności

- **public** - oznacza, że dziedziczone elementy (np. zmienne lub funkcje) mają taką widoczność jak w klasie bazowej.

public -> public
protected -> protected
private -> brak dostępu w klasie pochodnej

- **protected** - oznacza, że elementy publiczne zmieniają się w chronione.

public -> protected
protected -> protected
private -> brak dostępu w klasie pochodnej

- **private** - oznacza, że wszystkie elementy klasy bazowej zmieniają się w prywatne.

public -> private
protected -> private
private -> brak dostępu w klasie pochodnej

- **brak operatora** - oznacza, że niejawnie (domyślnie) zostanie wybrany operator **private**..

public -> private
protected -> private
private -> brak dostępu w klasie pochodnej