Preklapanje funkcija

- U skup funkcija kandidata u postupku ADL ulaze samo funkcije članice klase koja predstavlja tip objekta čija se funkcija poziva, ne i one iz osnovne klase. To znači da funkcija sa istim imenom, ukoliko ne redefiniše virtuelnu funkciju osnovne klase, *sakriva* sve funkcije iz osnovne klase sa istim imenom (isto važi i za druge vrste članova). Na primer:

```
struct B {
  void f (int);
};

struct D : B {
  void f ();
};
```

Ova funkcija ne redefiniše, već sakriva funkciju *B::f(int)*

```
int main () {
  D d;
  d.f(1);
}
```

Greška u prevođenju: u skupu kandidata ne postoji funkcija *f* koja može da primi *int*. Moglo bi ovako: *d.B::f(int)*

- Ovakva pojava nije dobra, jer iako izvedena klasa nasleđuje funkcije, one više ne čine njen interfejs (ukoliko se objektu pristupa kao instanci izvedene, a ne osnovne klase), što narušava semantiku nasleđivanja. Svi entiteti iz osnovne klase sa datim imenom mogu se uvesti u opseg pretrage unutar izvedene klase direktivom *using*:

```
struct D : B {
  using B::f;
  void f() {}
};
```

U opseg važenja klase *D* uključene su sve funkcije sa imenom *f* iz klase *B*

```
int main () {
  D d;
  d.f(1);
}
```

Sada je ovo ispravno, poziva se *d.B::f(1)*

Preklapanje funkcija

- Ukoliko je funkcija jednoznačno odabrana ovim postupkom potrage kao ona koja treba da bude pozvana, a ta funkcija je označena kao *obrisana* (*deleted*), specifikatorom *=delete* u deklaraciji, poziv neće biti ispravan, prevodilac će prijaviti grešku i neće tražiti neku drugu preklopljenu funkciju koju bi mogao da pozove uz implicitne konverzije argumenata. Ovako se mogu zabraniti neke implicitne konverzije, odnosno pozivi neke funkcije za neke tipove argumenata koji bi mogli biti konvertovani u tipove parametara postojećih funkcija. Na primer:

```
int f (double) { return 0; }

int f (int) = delete;

int main () {
  f(1);
}
```

- Analogno važi i za proveru prava pristupa: ukoliko odabrana funkcija nije dostupna na mestu poziva, poziv neće biti ispravan, prevodilac će prijaviti grešku i neće tražiti drugu preklopljenu funkciju koju može da pozove, a koja je dostupna; provera prava pristupa vrši se nakon i nezavisno od potrage i odabira pozvane preklopljene funkcije. Na primer:

```
class X {
public:
  int f (double) { return 0; }

private:
  int f (int) { return 1; }
};

int main () {
  X x;
  x.f(1);
}
```