

$\text{virtual} \text{ tipo } \text{metodo}() = 0; \Rightarrow \text{virtuale puro}$

Si assuma che `Abs` sia una classe base astratta fissata. Definire un template di funzione `bool Fun(T1*, T2&)`, dove `T1` e `T2` sono parametri di tipo, con il seguente comportamento. Si consideri una istanziazione implicita `Fun(ptr, ref)` dove si suppone che i parametri di tipo `T1` e `T2` siano istanziati a tipi polimorfi. Allora `Fun(ptr, ref)` ritorna `true` se e soltanto se valgono le seguenti due condizioni:

1. I parametri di tipo `T1` e `T2` sono istanziati allo stesso tipo:  $\rightarrow \text{typeid} \Rightarrow \text{stesso tipo dinamico}$
2. Siano `D1*` il tipo dinamico di `ptr` e `D2&` il tipo dinamico di `ref`; allora: (i) `D1` e `D2` sono lo stesso tipo e (ii) questo tipo è un sottotipo proprio della classe `Abs`.  $\rightarrow \text{DYNAMIC\_CAST! (sommo)}$

Scrivere la risposta nel riquadro sotto.

$\left[ \text{CONVERSIONE} \right] \rightarrow \text{DYNAMIC\_CAST!}$

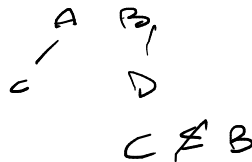
```
template <class T1, class T2>
bool Fun (T1* ptr, T2& ref){
    // (1) if(typeid(T1) == typeid(T2) &&
    // (2.1) (typeid(*ptr) == typeid(&ref) &&
    // (2.2) dynamic_cast<Abs*>(ptr)      {
    return true;
    }
}
```

## Esercizio 2

Siano `A`, `B`, `C` e `D` distinte classi polimorfe e si considerino le seguenti definizioni.

```
template<class A>
A* fun(A& ref) { return &ref; } CEA

int main() {
    B b;
    fun<A>(b);
    B* p = new D();
    C c;
    if (dynamic_cast<B*>(fun<A>(c))) cout << "bianco ";
    else cout << "nero ";
    if( !(dynamic_cast<D*>(new B())) ) cout << "rosso ";
}
```



Si supponga che:

1. il `main()` compili correttamente ed esegua senza provocare errori a run-time;
2. l'esecuzione del `main()` provochi in output su `cout` la stampa nero rosso.