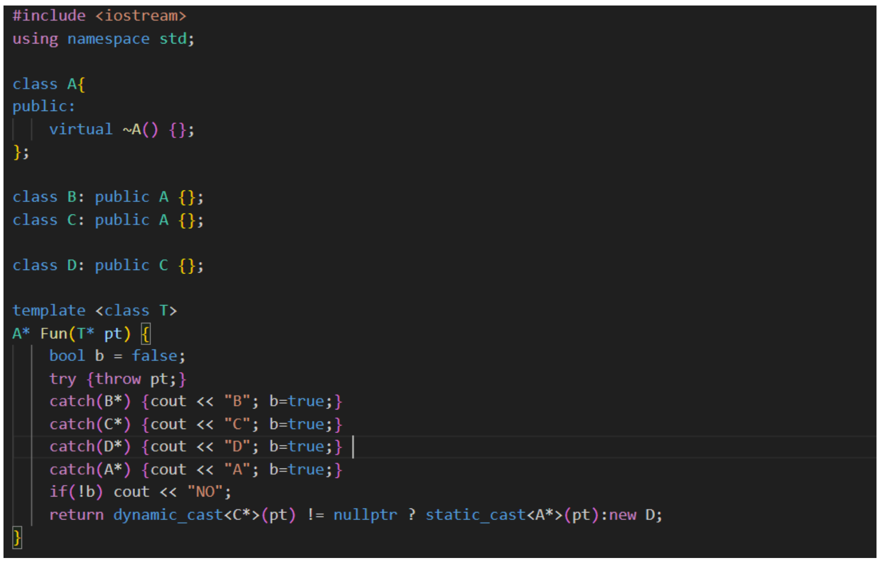
6 × C -> TIPO STATICO -> T

{ T -> A
{ PA2 -> A



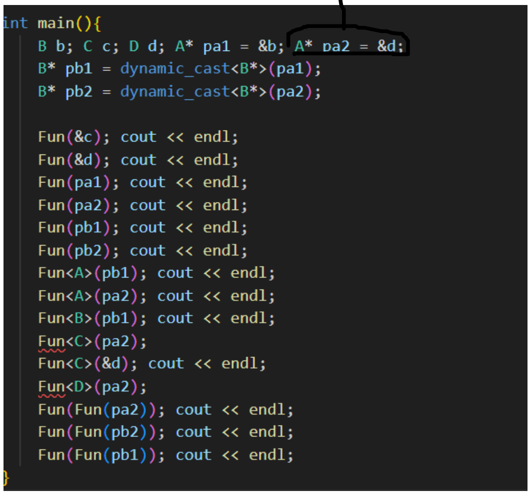
```cpp
#include <iostream>
using namespace std;

class A{
public:
    virtual ~A() {};
};

class B: public A {};
class C: public A {};

class D: public C {};

template <class T>
A* Fun(T* pt) {
    bool b = false;
    try {throw pt;}
    catch(B*) {cout << "B"; b=true;}
    catch(C*) {cout << "C"; b=true;}
    catch(D*) {cout << "D"; b=true;}
    catch(A*) {cout << "A"; b=true;}
    if(!b) cout << "NO";
    return dynamic_cast<C*>(pt) != nullptr ? static_cast<A*>(pt):new D;
}
```



```cpp
int main(){
    B b; C c; D d; A* pa1 = &b; A* pa2 = &d;
    B* pb1 = dynamic_cast<B*>(pa1);
    B* pb2 = dynamic_cast<B*>(pa2);

    Fun(&c); cout << endl;
    Fun(&d); cout << endl;
    Fun(pa1); cout << endl;
    Fun(pa2); cout << endl;
    Fun(pb1); cout << endl;
    Fun(pb2); cout << endl;
    Fun<A>(pb1); cout << endl;
    Fun<A>(pa2); cout << endl;
    Fun<B>(pb1); cout << endl;
    Fun<C>(pa2);
    Fun<C>(&d); cout << endl;
    Fun<D>(pa2);
    Fun(Fun(pa2)); cout << endl;
    Fun(Fun(pb2)); cout << endl;
    Fun(Fun(pb1)); cout << endl;
}
```

FUN(&C) -> C / FUN(&D) -> C / FUN(PA1 / PA2) -> A

FUN(PB1 / PB2) -> B / FUN<A>(PB1) -> A /

FUN<A>(PA2) -> A / FUN<B>(PB1) -> B

C ≠ PA2 -> TO WORK....

FUN<C>(PA2) -> NON COMPILA

FUN<D>(PA2)

D ≠ PA2 -> TO WORK...

FUN<C>(D) -> C

FUN(FUN(PA2)) -> AA

FUN(FUN(PB2)) -> $\textcircled{B}$A -> STAMPA PA2 / PB2

A ② B ①

FUN(FUN(PB1)) -> $\textcircled{B}$A -> STAMPA PA1 / PB1

A ② B ①