Esercizio 9.18. Si ricorda che nella gerarchia di classi per l'I/O la classe base astratta `ios` ha il distruttore virtuale. Si definisca una classe `C` che soddisfa le seguenti specifiche.

1. Un oggetto della classe `C` è caratterizzato da un vector di puntatori a `ios` e dal numero massimo di puntatori che questo vector può contenere. Deve essere disponibile un costruttore ad un argomento intero k, con un valore di default positivo, che determina il numero massimo k di puntatori che il vector può contenere.

2. Deve essere disponibile un metodo `void insert(ios&)` con il seguente comportamento: una invocazione `c.insert(s)` inserisce nel vector di `c` un puntatore a `s` quando valgono entrambe le seguenti condizioni (altrimenti lascia inalterato il vector):

   (a) il vector può contenere ancora elementi rispetto al numero massimo possibile;

   (b) se `D&` è il tipo dinamico di `s` allora il tipo `D` è diverso sia da `fstream` che da `stringstream`.

3. Deve essere disponibile un template di metodo `int conta(T&)`, dove `T` è un parametro di tipo, con il seguente comportamento: ogni invocazione `c.conta(t)` ritorna il numero di puntatori del vector di `c` che hanno un tipo dinamico `D*` tale che il tipo `D` è un sottotipo del tipo del parametro attuale `t`.

Ad esempio, il seguente `main()` deve compilare ed eseguire correttamente provocando le stampe indicate:

```
int main() {
  ifstream f("pippo"); ofstream g("mandrake");
  fstream h("pluto"), i("zagor");
  ostream* p = &g;
  stringstream s;
  C c(10);
  c.insert(f); c.insert(g); c.insert(h);
  c.insert(i); c.insert(*p); c.insert(s);
  istream& r=f;
  cout << c.conta(r); // stampa: 1 (è il puntatore all'oggetto f)
}
```

```
class C{

// 1
private:
        std::vector<ios*> v;
        int max;
public:
        C(int k = 10): max(k) {}

// 2

        void insert(ios& s){
                // (a)
                if(v.size() ≤ max &&
                // (b)
                typeid(s) ≠ typeid(fstream) &&
                typeid(s) ≠ typeid(stringstream)){
                        v.push_back(&s);
                }
        }
// 3

        template<class T>
        int conta(T& t){
                int cont = 0;
                for(auto it = v.begin(); it ≠ v.end();
                ++it){
                        if(dynamic_cast<T*>(*it)){
                                cont++;
                        }
                }
                return cont;
        }

};
```