```cpp
// publication.cpp
#include <iostream>
#include <string>
using namespace std;

class Publication {
protected:
    string title;
    float price;
public:
    Publication(): title(""), price(0.0f) {}
    Publication(const string &t, float p): title(t), price(p) {}
    virtual void getData() {
        cout << "Enter title and price: ";
        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // clear
        getline(cin, title);
        cin >> price;
    }
    virtual void putData() const {
        cout << "Title: " << title << ", Price: " << price;
    }
    virtual ~Publication() {}
};

class Book : public Publication {
private:
    int pages;
public:
    Book(): Publication(), pages(0) {}
    void getData() override {
        cout << "Enter book title: ";
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        getline(cin, title);
        cout << "Enter price: "; cin >> price;
        cout << "Enter pages: "; cin >> pages;
    }
    void putData() const override {
        cout << "Book -> ";
        Publication::putData();
        cout << ", Pages: " << pages << "\n";
    }
};

class Tape : public Publication {
private:
    float playTime;
public:
    Tape(): Publication(), playTime(0.0f) {}
    void getData() override {
        cout << "Enter tape title: ";
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        getline(cin, title);
        cout << "Enter price: "; cin >> price;
        cout << "Enter play time (minutes): "; cin >> playTime;
    }
    void putData() const override {
        cout << "Tape -> ";
        Publication::putData();
        cout << ", Play time: " << playTime << " mins\n";
    }
};

int main() {
    cout << "Publication Inheritance demo\n";
    Book b;
    cout << "Using sample data for Book\n";
    // sample fill without interactive input for quick demo
    // b.getData(); // uncomment to take input
    // instead:
    b = Book(); // default
    // set via direct access (for demo) - but members are protected so set via constructor would be
better
    // We'll just use getData for interactive run:
    b.getData();
    b.putData();

    Tape t;
    t.getData();
    t.putData();

    // Polymorphism demo
    Publication *p = &b;
    p->putData(); // should call Book::putData due to virtual
    return 0;
}
```