

Supplementary Notes on Inheritance

Page 27: Order of Cons/Destruction: Student w/ an Address

```
// init-order.cpp

#include <iostream>
using namespace std;

class Address {
public:
    Address() {                // Step 6
        cout << "Address's constructor" << endl; // Step 7
    }
    ~Address() {               // Step 12
        cout << "Address's destructor" << endl;  // Step 13
    }
};

class UPerson {
public:
    UPerson() {                // Step 3
        cout << "UPerson's constructor" << endl; // Step 4
    }
    ~UPerson() {               // Step 14
        cout << "UPerson's destructor" << endl;  // Step 15
    }
};

class Student : public UPerson {
public:
    Student() : Uperson() {    // Step 2
        cout << "Student's constructor" << endl; // Step 8
    }
    ~Student() {               // Step 10
        cout << "Student's destructor" << endl;  // Step 11
    }
private:
    Address address;           // Step 5
};

int main() {
    Student x;                 // Step 1
    return 0;                  // Step 9
}
```

Output:

```
UPerson's constructor    Step 4
Address's constructor    Step 7
Student's constructor    Step 8
Student's destructor     Step 11
Address's destructor     Step 13
UPerson's destructor     Step 15
```

Page 29: Move Address to UPerson

```
// init-order2.cpp

#include <iostream>
using namespace std;

class Address {
public:
    Address() {                // Step 5
        cout << "Address's constructor" << endl; // Step 6
    }
    ~Address() {               // Step 14
        cout << "Address's destructor" << endl;  // Step 15
    }
};

class UPerson {
public:
    UPerson() {                // Step 3
        cout << "UPerson's constructor" << endl; // Step 7
    }
    ~UPerson() {               // Step 12
        cout << "UPerson's destructor" << endl;  // Step 13
    }
private:
    Address address;           // Step 4
};

class Student : public UPerson {
public:
    Student() : Uperson() {    // Step 2
        cout << "Student's constructor" << endl; // Step 8
    }
    ~Student() {               // Step 10
        cout << "Student's destructor" << endl;  // Step 11
    }
};

int main() {
    Student x;                 // Step 1
    return 0;                  // Step 9
}
```

Output:

```
Address's constructor    Step 6
UPerson's constructor    Step 7
Student's constructor    Step 8
Student's destructor     Step 11
UPerson's destructor     Step 13
Address's destructor     Step 15
```

Page 32: Problem #2: Name Conflicts

```
// name-conflict.cpp

#include <iostream>
using namespace std;

void print(int x, int y) {           // Step 5, 12, 22, 28, 34, 41, 50, 59
    cout << x << " , " << y << '\n'; // Step 6, 13, 23, 29, 35, 42, 51, 60
}

class B {
private:
    int x, y;
public:
    B(int p=1, int q=2) : x(p), y(q) { // Step 2, 9
        cout << "Base class constructor: "; // Step 3, 10
        print(x,y); // Step 4, 11
    }
    void f() const { // Step 19, 31, 38, 47, 56
        cout << "Base class: "; // Step 20, 32, 39, 48, 57
        print(x,y); // Step 21, 33, 40, 49, 58
    }
};

class D : public B {
private:
    float x, y;
public:
    D() : B(), // Step 8
        x(10.0), y(20.0) { // Step 14
        cout << "Derived class constructor\n"; // Step 15
    }
    void f() const { // Step 25
        cout << "Derived class: "; // Step 26
        print(x,y); // Step 27
        B::f(); // Step 30
    }
};

void smart(const B* z) { // Step 44, 53
    cout << "Inside smart(): "; // Step 45, 54
    z->f(); // Step 46, 55
}

int main() {
    B base(5,6); cout << endl; // Step 1
    D derive; cout << endl; // Step 7
    B* b = &base; // Step 16
    D* d = &derive; // Step 17

    b->f(); cout << endl; // Step 18
    d->f(); cout << endl; // Step 24
    b = &derive; // Step 36
    b->f(); cout << endl; // Step 37

    smart(b); cout << endl; // Step 43
    smart(d); cout << endl; // Step 52
    return 0; // Step 61
}
```

Output:

```
Base class constructor:  5 , 6      Step 3 (Base...), 6 (5, 6)

Base class constructor:  1 , 2      Step 10 (Base...), 13 (1,2)
Derived class constructor      Step 15

Base class:  5 , 6                  Step 20 (Base...), 23 (5,6)

Derived class: 10 , 20              Step 26 (Derived...), 29 (10,20)
Base class:  1 , 2                  Step 32 (Base...), 35 (1,2)

Base class:  1 , 2                  Step 39 (Base...), 42 (1,2)

Inside smart():  Base class:  1 , 2 Step 45 (Insid...), 48 (Base...), 51 (1,2)

Inside smart():  Base class:  1 , 2 Step 54 (Insid...), 57 (Base...), 60 (1,2)
```