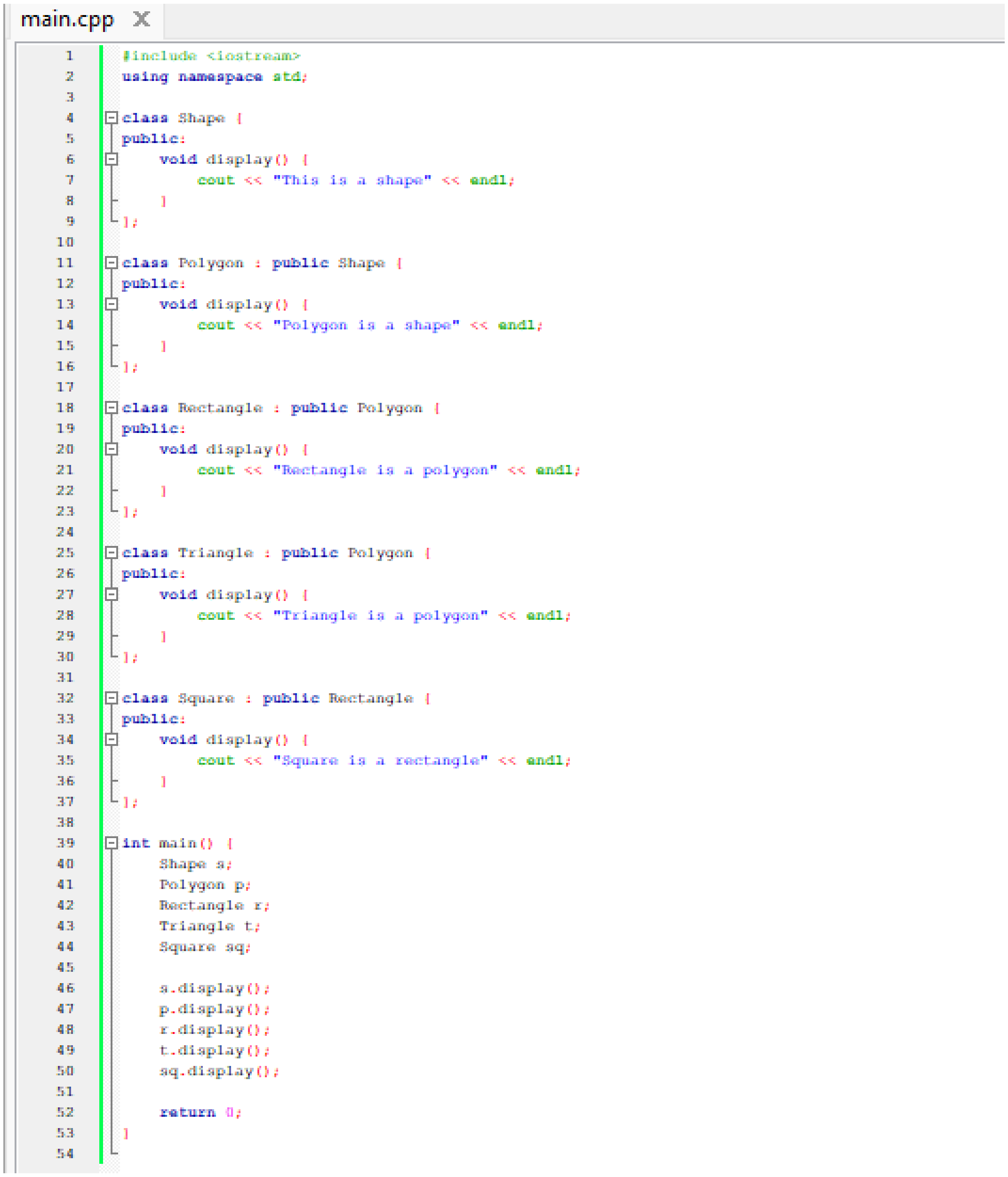
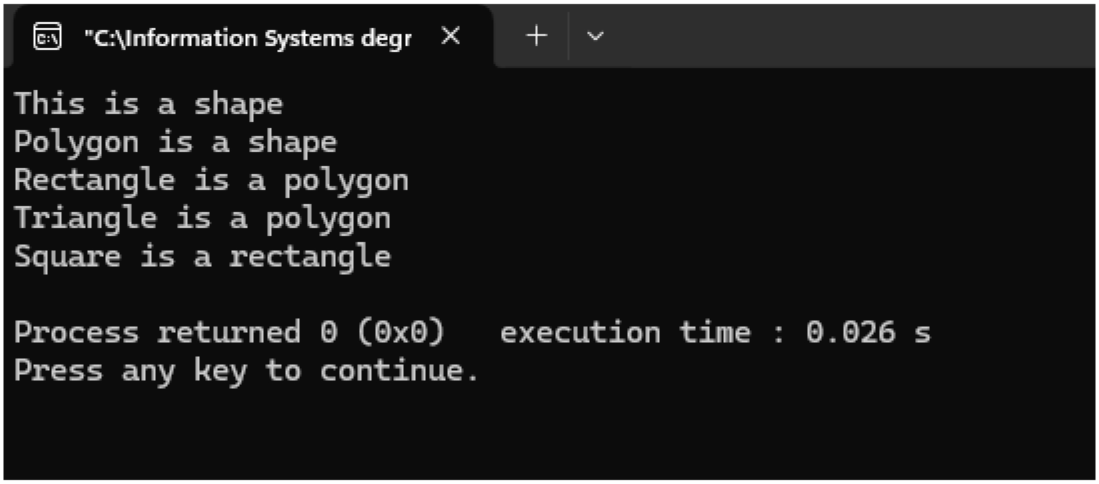
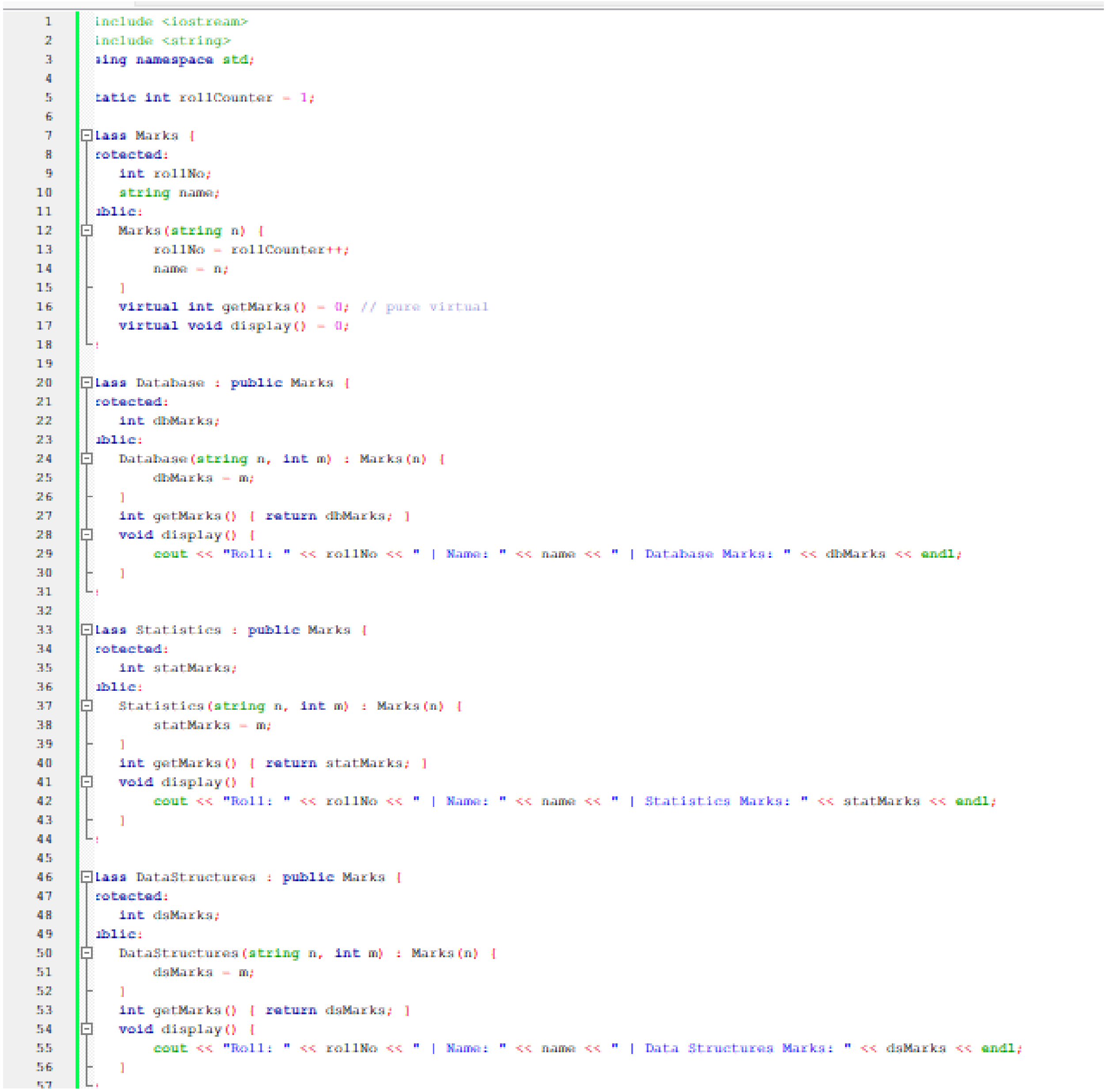
1.



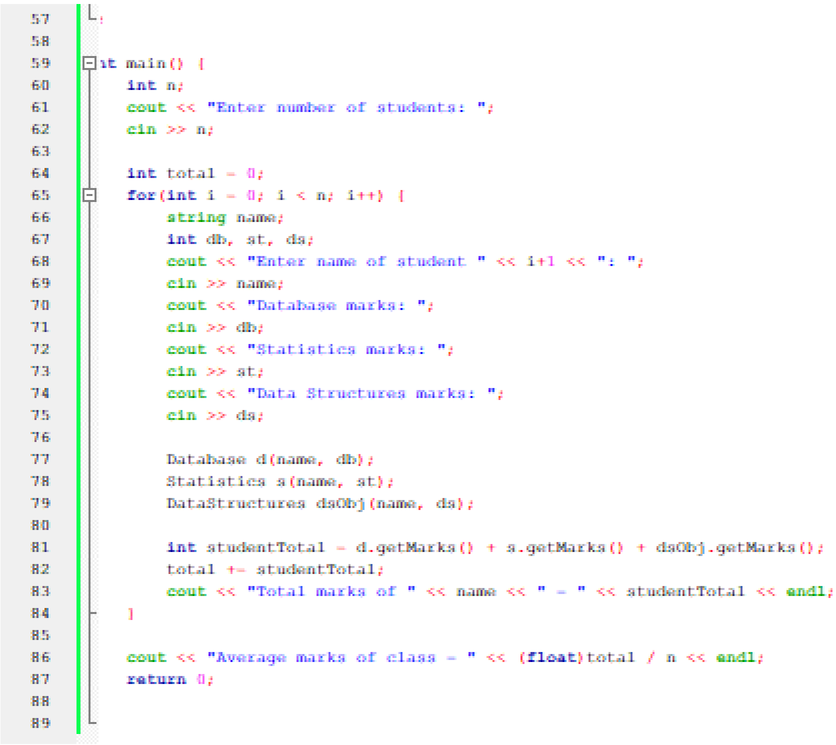
```cpp
#include <iostream>
using namespace std;

class Shape {
public:
    void display() {
        cout << "This is a shape" << endl;
    }
};

class Polygon : public Shape {
public:
    void display() {
        cout << "Polygon is a shape" << endl;
    }
};

class Rectangle : public Polygon {
public:
    void display() {
        cout << "Rectangle is a polygon" << endl;
    }
};

class Triangle : public Polygon {
public:
    void display() {
        cout << "Triangle is a polygon" << endl;
    }
};

class Square : public Rectangle {
public:
    void display() {
        cout << "Square is a rectangle" << endl;
    }
};

int main() {
    Shape s;
    Polygon p;
    Rectangle r;
    Triangle t;
    Square sq;

    s.display();
    p.display();
    r.display();
    t.display();
    sq.display();

    return 0;
}
```



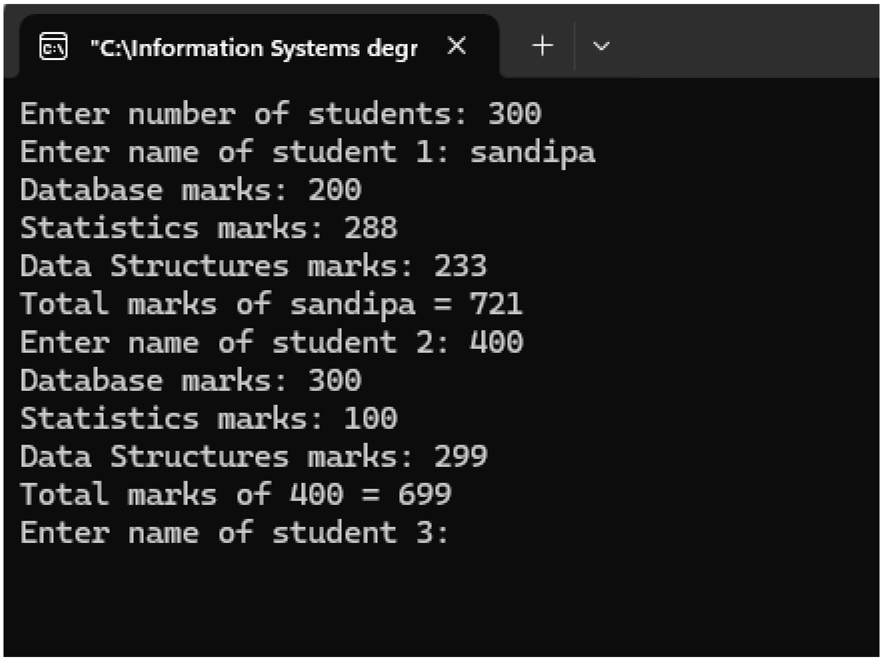
```
This is a shape
Polygon is a shape
Rectangle is a polygon
Triangle is a polygon
Square is a rectangle

Process returned 0 (0x0)   execution time : 0.026 s
Press any key to continue.
```



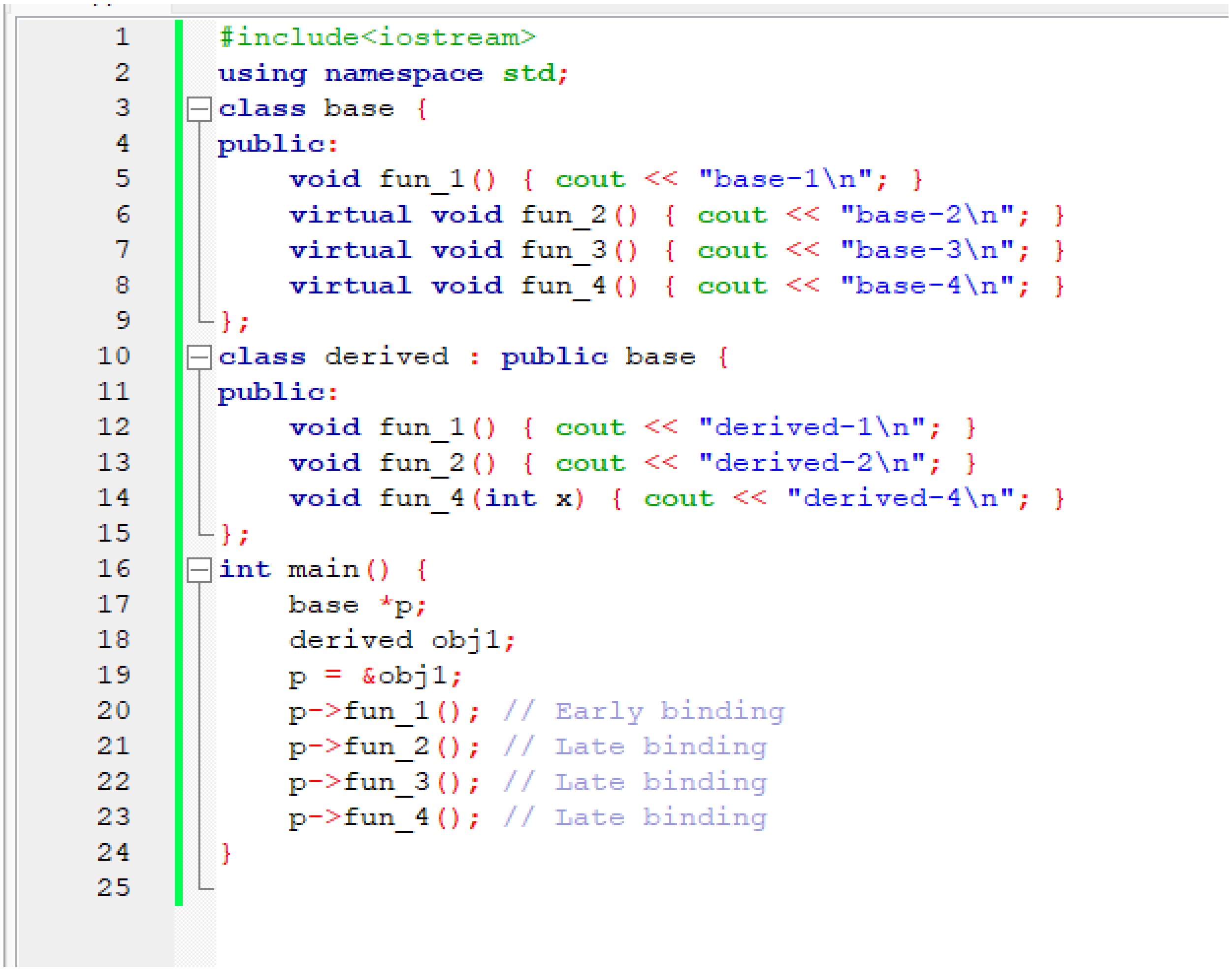
```cpp
include <iostream>
include <string>
sing namespace std;

tatic int rollCounter = 1;

lass Marks {
rotected:
    int rollNo;
    string name;
ublic:
    Marks(string n) {
        rollNo = rollCounter++;
        name = n;
    }
    virtual int getMarks() = 0; // pure virtual
    virtual void display() = 0;
;

lass Database : public Marks {
rotected:
    int dbMarks;
ublic:
    Database(string n, int m) : Marks(n) {
        dbMarks = m;
    }
    int getMarks() { return dbMarks; }
    void display() {
        cout << "Roll: " << rollNo << " | Name: " << name << " | Database Marks: " << dbMarks << endl;
    }
;

lass Statistics : public Marks {
rotected:
    int statMarks;
ublic:
    Statistics(string n, int m) : Marks(n) {
        statMarks = m;
    }
    int getMarks() { return statMarks; }
    void display() {
        cout << "Roll: " << rollNo << " | Name: " << name << " | Statistics Marks: " << statMarks << endl;
    }
;

lass DataStructures : public Marks {
rotected:
    int dsMarks;
ublic:
    DataStructures(string n, int m) : Marks(n) {
        dsMarks = m;
    }
    int getMarks() { return dsMarks; }
    void display() {
        cout << "Roll: " << rollNo << " | Name: " << name << " | Data Structures Marks: " << dsMarks << endl;
    }
;
```



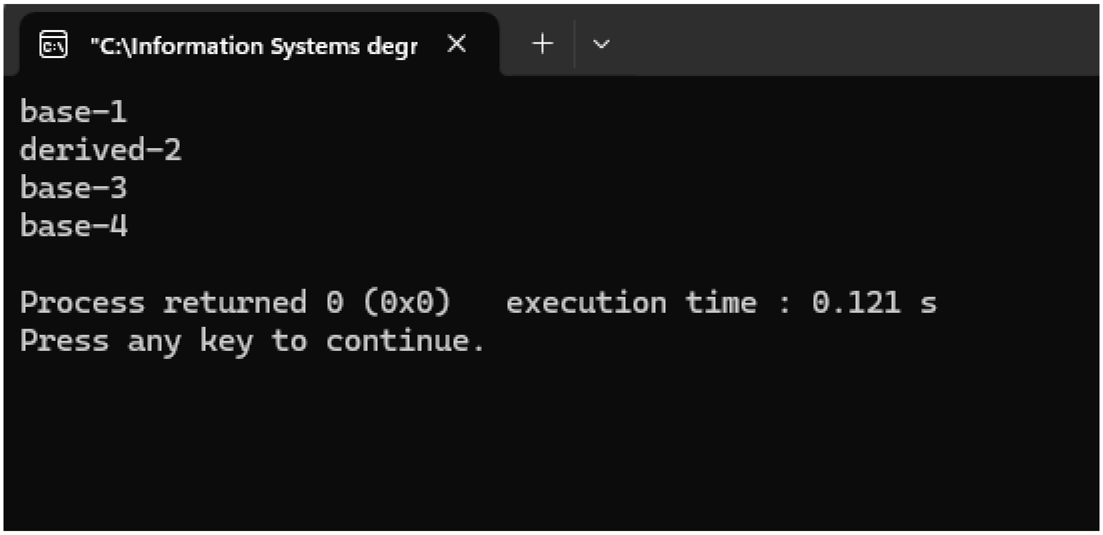
```cpp
;

nt main() {
    int n;
    cout << "Enter number of students: ";
    cin >> n;

    int total = 0;
    for(int i = 0; i < n; i++) {
        string name;
        int db, st, ds;
        cout << "Enter name of student " << i+1 << ": ";
        cin >> name;
        cout << "Database marks: ";
        cin >> db;
        cout << "Statistics marks: ";
        cin >> st;
        cout << "Data Structures marks: ";
        cin >> ds;

        Database d(name, db);
        Statistics s(name, st);
        DataStructures dsObj(name, ds);

        int studentTotal = d.getMarks() + s.getMarks() + dsObj.getMarks();
        total += studentTotal;
        cout << "Total marks of " << name << " = " << studentTotal << endl;
    }

    cout << "Average marks of class = " << (float)total / n << endl;
    return 0;

```



```
Enter number of students: 300
Enter name of student 1: sandipa
Database marks: 200
Statistics marks: 288
Data Structures marks: 233
Total marks of sandipa = 721
Enter name of student 2: 400
Database marks: 300
Statistics marks: 100
Data Structures marks: 299
Total marks of 400 = 699
Enter name of student 3:
```

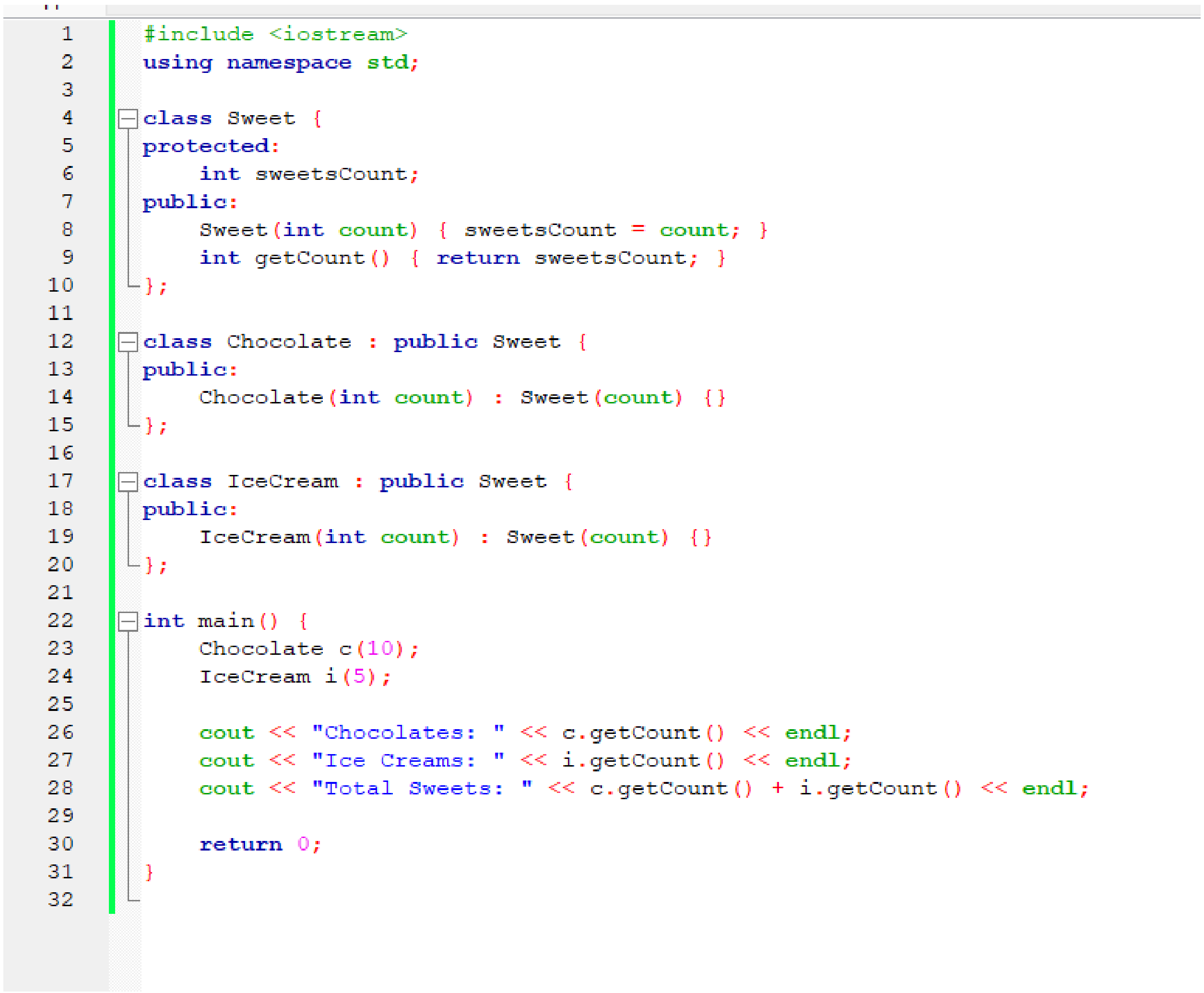


```cpp
#include<iostream>
using namespace std;
class base {
public:
    void fun_1() { cout << "base-1\n"; }
    virtual void fun_2() { cout << "base-2\n"; }
    virtual void fun_3() { cout << "base-3\n"; }
    virtual void fun_4() { cout << "base-4\n"; }
};
class derived : public base {
public:
    void fun_1() { cout << "derived-1\n"; }
    void fun_2() { cout << "derived-2\n"; }
    void fun_4(int x) { cout << "derived-4\n"; }
};
int main() {
    base *p;
    derived obj1;
    p = &obj1;
    p->fun_1(); // Early binding
    p->fun_2(); // Late binding
    p->fun_3(); // Late binding
    p->fun_4(); // Late binding
}

```

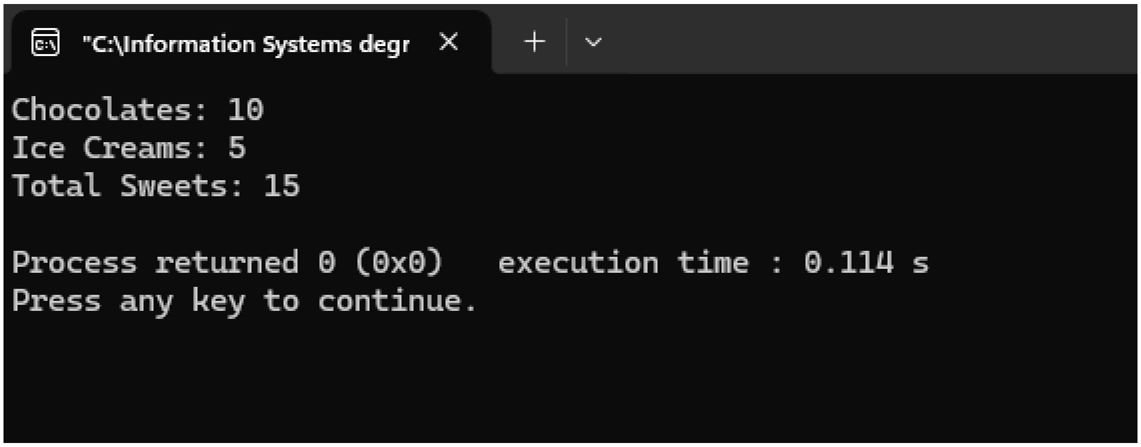


```
base-1
derived-2
base-3
base-4

Process returned 0 (0x0)   execution time : 0.121 s
Press any key to continue.
```

4

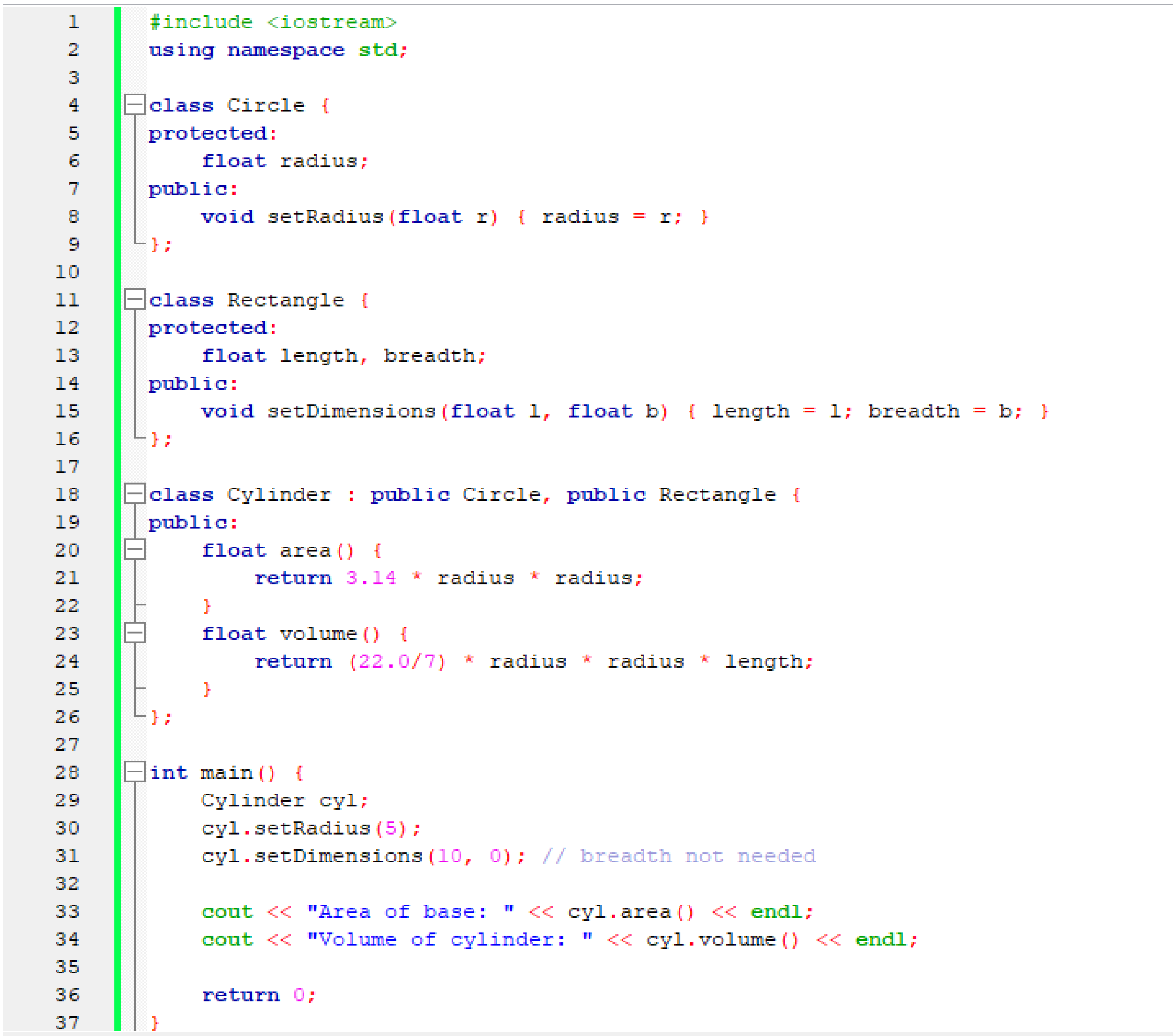


```cpp
#include <iostream>
using namespace std;

class Sweet {
protected:
    int sweetsCount;
public:
    Sweet(int count) { sweetsCount = count; }
    int getCount() { return sweetsCount; }
};

class Chocolate : public Sweet {
public:
    Chocolate(int count) : Sweet(count) {}
};

class IceCream : public Sweet {
public:
    IceCream(int count) : Sweet(count) {}
};

int main() {
    Chocolate c(10);
    IceCream i(5);

    cout << "Chocolates: " << c.getCount() << endl;
    cout << "Ice Creams: " << i.getCount() << endl;
    cout << "Total Sweets: " << c.getCount() + i.getCount() << endl;

    return 0;
}
```

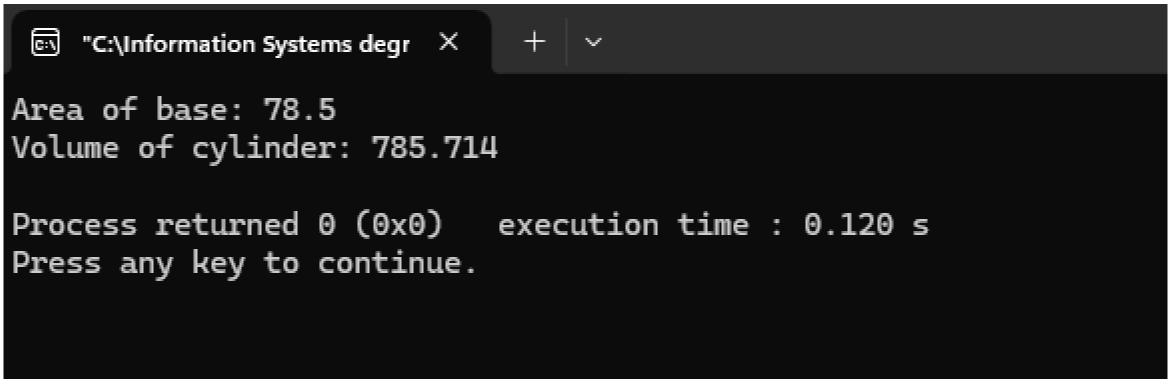


```
Chocolates: 10
Ice Creams: 5
Total Sweets: 15

Process returned 0 (0x0)   execution time : 0.114 s
Press any key to continue.
```

5



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

class Circle {
protected:
    float radius;
public:
    void setRadius(float r) { radius = r; }
};

class Rectangle {
protected:
    float length, breadth;
public:
    void setDimensions(float l, float b) { length = l; breadth = b; }
};

class Cylinder : public Circle, public Rectangle {
public:
    float area() {
        return 3.14 * radius * radius;
    }
    float volume() {
        return (22.0/7) * radius * radius * length;
    }
};

int main() {
    Cylinder cyl;
    cyl.setRadius(5);
    cyl.setDimensions(10, 0); // breadth not needed

    cout << "Area of base: " << cyl.area() << endl;
    cout << "Volume of cylinder: " << cyl.volume() << endl;

    return 0;
}
```



```
Area of base: 78.5
Volume of cylinder: 785.714

Process returned 0 (0x0)   execution time : 0.120 s
Press any key to continue.
```