#include <iostream>

using namespace std;

const int MAX = 100;

class Polygon {

public:

Polygon(int numOfSides = 0);

void set(int sideNum, int value);

int get(int sideNum);

virtual int perimeter() = 0;

virtual double area() = 0;

private:

int numOfSides;

int sides[MAX];

};

class Rectangle : public Polygon {

public:

Rectangle(int width, int length);

int perimeter();

double area();

};

class Square : public Rectangle {

public:

Square(int length);

int perimeter();

double area();

};

class RightTri : public Polygon {

public:

RightTri(int s1, int s2, int hypo);

int perimeter();

double area();

};

class RectSolid : public Rectangle {

public:

RectSolid(int width, int length, int height);

double volume();

int getHeight();

void setHeight(int h);

private:

int height;

};

Polygon::Polygon(int numOfSides) : numOfSides(numOfSides) {

for (int i = 0; i < MAX; i++) {

sides[i] = 0;

}

}

void Polygon::set(int sideNum, int value) {

sides[sideNum] = value;

}

int Polygon::get(int sideNum) {

return sides[sideNum];

}

Rectangle::Rectangle(int width, int length) : Polygon(4) {

set(0, width);

set(1, length);

set(2, width);

set(3, length);

}

int Rectangle::perimeter() {

return get(0) + get(1) + get(2) + get(3);

}

double Rectangle::area() {

return get(0) \* get(1);

}

Square::Square(int length) : Rectangle(length, length) {

}

int Square::perimeter() {

return get(0) \* 4;

}

double Square::area() {

return get(0) \* get(0);

}

RightTri::RightTri(int s1, int s2, int hypo) : Polygon(3) {

set(0, s1);

set(1, s2);

set(2, hypo);

}

int RightTri::perimeter() {

return get(0) + get(1) + get(2);

}

double RightTri::area() {

return (double)(get(0) \* get(1)) / (double)2;

}

RectSolid::RectSolid(int width, int length, int height) : Rectangle(width, length), height(height) {

}

double RectSolid::volume() {

return height \* get(0) \* get(1);

}

int main() {

Polygon \*\* polygon = new Polygon\*[4];

polygon[0] = new Rectangle(4, 10);

polygon[1] = new Square(5);

polygon[2] = new RightTri(3, 4, 5);

polygon[3] = new RectSolid(6, 3, 5);

dynamic\_cast<Rectangle\*>(polygon[0]);

cout << "Rectangle" << endl;

cout << "Perimeter: " << polygon[0]->perimeter() << endl;

cout << "Area: " << polygon[0]->area() << endl;

cout << endl;

dynamic\_cast<Square\*>(polygon[1]);

cout << "Square" << endl;

cout << "Perimeter: " << polygon[1]->perimeter() << endl;

cout << "Area: " << polygon[1]->area() << endl;

cout << endl;

dynamic\_cast<RightTri\*>(polygon[2]);

cout << "Right Triangle" << endl;

cout << "Perimeter: " << polygon[2]->perimeter() << endl;

cout << "Area: " << polygon[2]->area() << endl;

cout << endl;

dynamic\_cast<RectSolid\*>(polygon[3]);

cout << "Solid Rectangle" << endl;

cout << "Perimeter: " << polygon[3]->perimeter() << endl;

cout << "Area: " << polygon[3]->area() << endl;

RectSolid \* solidRectangle = dynamic\_cast<RectSolid\*>(polygon[3]);

cout << "Volume: " << solidRectangle->volume() << endl;

cout << endl;

return 0;

}

**Executable module instructions:**

1. **Compile**
2. **Run**

**Test data and expected results:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Shapes** | **Input Data** | **Expected Output** |
| 1 | Rectangle | Width = 4  Length = 10 | Perimeter = 28  Area = 40**40** |
| 2 | Square | Length = 5 | Perimeter = 20  Area = 25 |
| 3 | Right Triangle | s1 = 3  s2 = 4  s3 = 5 | Perimeter = 12  Area = 6 |
| 4 | Solid Rectangle | Width = 6  Length = 3  Height = 5 | Perimeter = 18  Area = 18  Volume = 90 |

**Output:**

