Lab 5.1.2 Restricting access to object data

Objectives

Familiarize the student with:

- creating setter access methods;
- limiting the range of accepted values in access methods.

Scenario

One important aspect of object-oriented programming is data encapsulation.

This is a complicated term for protecting the data of an object.

But why should you protect the data? Look at the code below.

As we can see, this isn't correct. But you can fix this!

- Hide the area and side members using the keyword **private**;
- add a public method **set_side** to the **Square** class that will update both fields;
- **set_side** should also ignore the change if the argument is less than 0;
- change the print function implementation into a public method of **Square**.

```
#include <iostream>
#include <string>

using namespace std;

class Square
{
public:
  Square(double side);
  double    side;
  double    area;
  // Your code here
};

Square::Square(double side)
{
  this->side = side;
  this->area = side * side;
}

void print(Square* square)
{
  cout << "Square: side=" << square->side << " area=" square->area << endl;
}


int main()
{
  Square s(4);


  print("square);

  s.side = 2.0;
  print("square);

  s.side = -33.0;
  print("square);

  return 0;
}
```