

**University American College Skopje**

**Course: Object Programming**

# Inheritance 1

## Exercises

Prepared by: Adrijan Božinovski, Ph.D.

# The `string` class

- The `string` class can be used after using the directive `#include <string>`
  - i.e. through this the `string` library gets included in the code
- In C++, `strings` are treated as arrays of characters – you can access any individual character of a string as if it were defined as an array of characters
  - Using the operator `[]` (i.e. square parentheses) is valid for strings
- The main advantage of using strings is that their values can be assigned directly, using the `=` operator
  - i.e. they can be handled like regular variables; no `strcpy()` needed
- The main disadvantage is that the keyboard input of strings is slightly different
  - The `cin.getline()` function can not be used with `strings`

# Assignment 1

- Create a base class **Edition**, with the following fields:
  - Private:
    - string title
  - Public:
    - A get() and a set() function for the title

# Assignment 1

- Create a class **Book**, which inherits from **Edition**, and has the following additional fields:
  - Private:
    - string author
  - Public:
    - A get() and a set() function for the author
- Create a class **Magazine**, which inherits from **Edition**, and has the following additional fields:
  - Private:
    - int copies
  - Public:
    - A get() and a set() function for the copies

# Assignment 1

- In the main() function
  - Create a **Book** and a **Magazine** object
  - Set the titles of both the **Book** and **Magazine** objects
  - Set the author of the **Book** object
  - Set the number of copies of the **Magazine** object
  - Display the title and author of the **Book** object
  - Display the title and number of copies of the **Magazine** object

# Assignment 1

```
The title of the book is Thinking in C++ and the author is Bruce Eckel  
The title of the magazine is Wired and the number of copies 20000  
Press any key to continue
```

# Assignment 2

- Modify Assignment 1, so that you will add a constructor for each class
- The constructors of the derived classes should make use of the constructor of the base class, to initialize the fields belonging to the base class

```
Edition::Edition(string t)
{
    set_title(t);
}
```

```
Book::Book(string t, string a) : Edition(t)
{
    set_author(a);
}
```

# Assignment 3

- Create a base class **Triangle**, with the following fields:
  - Private:
    - double a, b, c
  - Public:
    - get() methods for all three sides
    - a parameterized constructor which takes three double parameters and sets the private fields to the values of the parameters
    - A double baseArea() function, which calculates the area of the triangle (using Heron's formula)



# Assignment 3

- Create a class **Pyramid**, which inherits from **Triangle**, and has the following additional fields:
  - Private:
    - double h
  - Public:
    - a get() method for the h field
    - a parameterized constructor that takes 4 double parameters and sets the fields to the values of the parameters (it makes use of the base class' constructor to set the values of the fields representing the base triangle)
    - a double volume() function, that calculates the volume of the pyramid ( $\text{volume} = \text{base area} * \text{height} / 3$ )



# Assignment 4

- Create a base class **Rectangle**, with the following fields:
  - Private:
    - double a, b
  - Public:
    - get() methods for all three sides
    - a parameterized constructor which takes two double parameters and sets the private fields to the values of the parameters
    - A double area() function, which calculates the area of the rectangle with given sides

# Assignment 4

- Create a class **Cuboid**, which inherits from **Rectangle**, and has the following additional fields:
  - Private:
    - double c
  - Public:
    - a get() method for the c field
    - a parameterized constructor that takes 3 double parameters and sets the fields to the values of the parameters (it makes use of the base class' constructor to set the values of the fields representing the base rectangle)
    - a double area() function, that calculates the area of the cuboid ( $\text{area} = 2*a*b + 2*b*c + 2*a*c$ )
      - a\*b is the base rectangle area; use the *this* pointer to access the base class **Rectangle**'s area() function

# Assignment 4

- In the main() function
  - Input the 3 sides through the keyboard
  - Create a Cuboid object from those data
  - Display the 3 sides, the base area, and the entire cuboid area on the screen

```
Enter the three sides of a cuboid: 1.2 2.3 3.4
This is the cuboid that has been entered:
Side a: 1.2      Side b: 2.3      Side c: 3.4
Area of the base: 2.76
Area of the entire cuboid: 29.32
Press any key to continue_
```