Tartalomjegyzék

Exercises

Open a new project for each exercise or a new file if you're in the command line.

From this point onward use .cpp file extensions!

Complex class from the lecture

Execute the code from the lecture:

```
#include<iostream>
#include<cmath>

using namespace std;


class Complex {
private:
  float re;
  float im;
public:
  Complex();
  Complex(const Complex& other);
  Complex(float r);
  Complex(float r, float i);

  Complex add(Complex other);
  Complex times(Complex other);
  float abs();

  void print();

  ~Complex();
};

Complex::Complex() {
  re = 0;
  im = 0;
}

Complex::Complex(const Complex& other) {
  re = other.re;
  im = other.im;
}

Complex::Complex(float r) {
  re = r;
  im = 0;
}
```

```
Complex::Complex(float r, float i) {
  re = r;
  im = i;
}

Complex Complex::add(Complex other) {
  return Complex(this->re + other.re, this->im + other.im);
}

Complex Complex::times(Complex other) {
  float real = this->re * other.re - this->im * other.im;
  float imag = this->re * other.im + this->im * other.re;
  return Complex(real, imag);
}

float Complex::abs() {
  return sqrt(this->re * this->re + this->im * this->im);
}

void Complex::print() {
  cout << re << " + " << im << "i" << endl;
}

Complex::~Complex() {
}

int main(void) {
  Complex a;
  Complex b = Complex(1,2);
  Complex c = a.times(b);

  a.print();
  b.print();
  c.print();

  (b.add(c)).print();

  cout << b.abs() << endl;

  return 0;
}
```

Rectangle

Write a **Rectangle** class!

- Constructed with the length of its 2 edges.
- Methods to get its area and perimeter.
- It should have a **scale** method that can scale the rectangle (both edge lengths are multiplied by the given number).
- It should be copyable, like so:

```
Rectangle a(2, 5);
Rectangle b(a);
```

Back to the Complex

Let's check back on the Complex class. Feed its code to pythontutor and check when each method/constructor is called.

String class

Write a String class that will ease our use of C strings.

- The default constructor should create an empty string.
- We should be able to construct it from a C string. The constructor should look like this:

```
String(const char* s);
```

- We should be able to retrieve its length.
- Make a **print** method that prints the string using **cout**.
- Make a method that concatenates two Strings.

Linked list

Implement the previously used linked list with classes. The list should store floats and have the following features:

- default constructor: empty list
- copy constructor
- constructor that builds the list from an array
- destructor
- append: new element at the end
- insert: new element at the given index
- size: returns the number of elements stored
- in: true/false whether the given value is stored in the list or not
- index: retrieves the element at the given index
- remove: removes an element at the given index
- where: returns the first index where the given value can be found, returns -1 if the value isn't in the list