

Solutions

Only the solutions we discussed on the practical will be here. All the other tasks are perfect to practice for the written exam.

Complex with operators

```
#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 operator+(Complex other);
    Complex operator*(Complex other);
    float abs();

    void print();

    friend ostream& operator<<(ostream& os, Complex& c);

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

Complex Complex::operator*(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);
}

ostream& operator<<(ostream& os, Complex& c) {
    os << c.re << " + " << c.im << "i";
    return os;
}

Complex::~~Complex() {
}

int main(void) {
    Complex a;
    Complex b = Complex(1,2);
    Complex c = a * b;
    Complex d = b + b + b;
    Complex e = b * d;
    //Complex e = b.operator*(d);

    // 5 / 6 / 7;
    // /(5, 6), 7);

    cout << "a: " << a << endl;

    cout << "b: " << b << endl;
    cout << "c: " << c << endl;
    cout << "d: " << d << endl;
    cout << "e: " << e << endl;

    //cout << (c + d) << endl;

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

    return 0;
}

```

String class

```

#include<iostream>

using namespace std;

class String {
private:
    char *str;
    int length;
    int c_string_length(const char* s);
public:
    String();
    String(const char* s);
    String(const String& other);

    void print();
    int getLength();

    String operator+(String other);

    friend String operator+(const char* left, String right);
}

```

```

    ~String();
};

int String::c_string_length(const char* s) {
    int i;
    for(i = 0; s[i] != '\0'; i++) {}
    return i;
}

String::String() {
    str = new char[1];
    str[0] = '\0';
    length = 0;
}

String::String(const char* s) {
    length = c_string_length(s);
    str = new char[length + 1];
    for(int i = 0; s[i] != '\0'; i++) {
        str[i] = s[i];
    }
    str[length] = '\0';
}

String::String(const String& other) {
    this->length = other.length;
    str = new char[this->length + 1];
    for(int i = 0; other.str[i] != '\0'; i++) {
        this->str[i] = other.str[i];
    }
    this->str[this->length] = '\0';
}

void String::print() {
    cout << str << endl;
}

int String::getLength() {
    return length;
}

String String::operator+(String other) {
    int new_length = this->length + other.length + 1;
    char s[new_length];
    for(int i = 0; i < this->length; i++) {
        s[i] = this->str[i];
    }
    for(int i = 0; i < other.length; i++) {
        s[i + this->length] = other.str[i];
    }
    s[new_length] = '\0';
    return String(s);
}

String::~~String() {
    delete[] str;
}

String operator+(const char* left, String right) {
    int j;
    for(j = 0; left[j] != '\0'; j++) {}
    int new_length = j + right.length + 1;
    char s[new_length];

```

```
for(int i = 0; i < j; i++) {
    s[i] = left[i];
}
for(int i = 0; i < right.length; i++) {
    s[i + j] = right.str[i];
}
s[new_length] = '\\0';
return String(s);
}

int main(void) {
    String a;
    String b("batman");
    String c(b);
    String d("catman");
    String e = "ratman" + b;
    String f = b + d;
    // String f = b.operator+(d);
    c.print();
    e.print();
    f.print(); // batmancatman
    cout << c.getLength() << endl;
    return 0;
}
```