"Ss. Cyril and Methodius" University in Skopje
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Lesson 5

Dynamic allocation of memory, Operator overloading

Object-oriented programming

Problem 1

Define a class for complex numbers. Complex number is a number that can be expressed in the form $a + bi$, where a and b are real numbers and i is the imaginary unit ($i^2 = -1$). Overload the the following operators:
`+`, `-`, `*`, `/`, `+=`, `-=`, `/=`.
Also implement the « operator.

Problem 1
Solution 1/3

```
#include <iostream>
using namespace std;

class Complex {
private:
    float a, b;
public:
    Complex(const float a = 0, const float b = 0) {
        this->a = a;
        this->b = b;
    }
    friend ostream &operator<<(ostream &x, const Complex &c) {
        x << c.a;
        if (c.b >= 0) {
            x << " +";
        }
        x << c.b << "j";
        return x;
    }
    Complex operator+(const Complex &c) {
        Complex res(a + c.a, b + c.b);
        return res;
    }

    Complex operator-(const Complex &c) {
        return Complex(a - c.a, b - c.b);
    }

    Complex operator*(const Complex &c) {
        return Complex(a * c.a - b * c.b, b * c.a - a * c.b);
```

Problem 1
Solution 2/3

```
Complex operator/(const Complex &c) {
    float m = c.a * c.a + c.b * c.b;
    float r = (a * c.a - b * c.b) / m;
    return Complex(r, (b * c.a + b * c.b) / m);
}

Complex &operator+=(const Complex &c) {
    a += c.a;
    b += c.b;
    return *this;
}

Complex &operator-=(const Complex &c) {
    a -= c.a;
    b -= c.b;
    return *this;
}
Complex &operator*=(const Complex &c) {
    a = a * c.a - b * c.b;
    b = b * c.a - a * c.b;
    return *this;
}
Complex &operator/=(const Complex &c) {
    *this = *this / c;
    return *this;
}
bool operator==(const Complex &c) {
    return a == c.a && b == c.b;
}
```

Problem 1
Solution 3/3

```
int main() {
    Complex c1(2, -6);
    Complex c2(3, 5);
    Complex c = c1 + c2;
    cout << c1 << " + " << c2 << " = " << c << endl;
    c = c1 - c2;
    cout << c1 << " - " << c2 << " = " << c << endl;
    c = c1 * c2;
    cout << c1 << " * " << c2 << " = " << c << endl;
    c = c1 / c2;
    cout << c1 << " / " << c2 << " = " << c << endl;
    if (c == c1) {
        cout << "Numbers are equal" << endl;
    }
    return 0;
}
```

Problem 2

Implement class **Array** for onedimensional array of integers. The class should store the total capacity and current size. Use dynamic allocation of memory for storing the data. Implement the following operators:

- operator [] for accessing and changing the value of element
- operator += for adding new element in the array (if the capacity is full increase the array for 100%)

Implement **main** function where you will instaciate object from class Array and read N numbers (from SI). Then print the elements from the array, its capacity, and the total number of elements.

Problem 2
Solution 1/4

```
#include <iostream>
using namespace std;

class Array {
private:
    int *x;
    int size;
    int capacity;
public:
    Array(const int capacity = 5) {
        x = new int[capacity];
        size = 0;
        this->capacity = capacity;
    }

    // copy constructor
    Array(const Array &a) {
        size = a.size;
        capacity = a.capacity;
        x = new int[capacity];
        for(int i = 0; i < size; ++i) {
            x[i] = a.x[i];
        }
    }
```

Problem 2
Solution 2/4

```
// asignment operator =
Array& operator=(const Array &a) {
    if(this == &a) return *this;
    size = a.size;
    capacity = a.capacity;
    delete [] x;
    x = new int[capacity];
    for(int i = 0; i < size; ++i) {
        x[i] = a.x[i];
    }
    return *this;
}

// destructor
~Array() {
    delete [] x;
}

friend ostream& operator<<(ostream &o, const Array &a) {
    o << "Capacity: " << a.capacity << endl;
    o << "Size: " << a.size << endl;
    for(int i = 0; i < a.size; ++i) {
        o << a.x[i] << endl;
    }
    return o;
}
```

Problem 2
Solution 3/4

```
    Array& operator+=(const int n) {
        if(capacity == size) {
            int *y = new int[2 * capacity];
            for(int i = 0; i < size; ++i) {
                y[i] = x[i];
            }
            delete [] x;
            x = y;
            capacity = capacity * 2;
        }
        x[size] = n;
        size++;
        return *this;
    }

    int& operator[](int i) {
        return x[i];
    }

    int getAt(int index) {
        return x[index];
    }

    void setAt(int index, int value) {
        x[index] = value;
    }
};
```

Problem 2
Solution 4/4

```
int main() {
    Array a;
    a += 5;
    a += 1;
    a += 9;
    a += 8;
    a += 3;
    a += 10;
    Array b(a);
    a.setAt(0, 20);
    a = b;
    b.setAt(5, 50);
    cout << a;
    cout << b;
    return 0;
}
```

Problem 3

Write a class for **WebServer**. Each web server has:

- name (max 30 chars)
- list of web pages (dynamicly allocated array of objects from class WebPage).

Each web page has:

- url (max 100 chars)
- content (dynamicly allocated array of chars).

For the class WebServer overload the following operators:

- += adding new page on the server
- -= removing web page from the server.

Problem 3
Solution 1/6

```
#include <iostream>
#include <cstring>
using namespace std;

class WebPage {
private:
    char url[100];
    char *content;
public:
    WebPage(const char *url = "", const char *content = "") {
        strcpy(this->url, url);
        this->content = new char[strlen(content) + 1];
        strcpy(this->content, content);
    }
    WebPage(const WebPage &wp) {
        strcpy(url, wp.url);
        content = new char[strlen(wp.content) + 1];
        strcpy(content, wp.content);
    }

    WebPage& operator=(const WebPage &wp) {
        if(this == &wp) return *this;
        strcpy(url, wp.url);
        delete [] content;
        content = new char[strlen(wp.content) + 1];
        strcpy(content, wp.content);
        return *this;
    }
```

Problem 3
Solution 2/6

```
    ~WebPage() {
        delete [] content;
    }

    bool operator==(const WebPage &wp) {
        return strcmp(url, wp.url) == 0;
    }

    friend ostream& operator<<(ostream &out, const WebPage &wp) {
        out << wp.url << endl;
        out << wp.content << endl;
        return out;
    }
};
```

Problem 3
Solution 3/6

```
class WebServer {
private:
    char name[100];
    WebPage *pages;
    int size;
public:
    WebServer(const char *name="") {
        strcpy(this->name, name);
        size = 0;
    }
    WebServer(const WebServer &ws) {
        strcpy(name, ws.name);
        size = ws.size;
        pages = new WebPage[size];
        wp_copy(pages, ws.pages, size);
    }
    void wp_copy(WebPage *p1, WebPage *p2, int n) {
        for(int i = 0; i < n; ++i) {
            p1[i] = p2[i];
        }
    }
```

Problem 3
Solution 4/6

```
WebServer& operator=(const WebServer &ws) {
    if(this == &ws) return *this;
    strcpy(name, ws.name);
    size = ws.size;
    delete [] pages;
    pages = new WebPage[size];
    wp_copy(pages, ws.pages, size);
    return *this;
}
~WebServer() {
    delete [] pages;
}

WebServer& operator+=(const WebPage &wp) {
    WebPage *temp = new WebPage[size + 1];
    wp_copy(temp, pages, size);
    delete [] pages;
    pages = temp;
    pages[size] = wp;
    size++;
    return *this;
}
```

Problem 3
Solution 5/6

```
    WebServer& operator-=(const WebPage &wp) {
        WebPage *temp = new WebPage[size - 1];
        int k = 0;
        for(int i = 0; i < size; ++i) {
            if(!(pages[i] == wp)) {
                temp[k++] = pages[i];
            }
        }
        delete [] pages;
        pages = temp;
        size--;
        return *this;
    }

    friend ostream& operator<<(ostream &out, const WebServer &ws) {
        out << ws.name << endl;
        for(int i = 0; i < ws.size; ++i) {
            out << ws.pages[i];
        }
        return out;
    }
};
```

Problem 3
Solution 6/6

```
int main() {
    WebServer ws("FINKI Server");
    WebPage wp1("www.google.com", "The world");
    WebPage wp2("www.facebook.com", "The people");
    WebPage wp3("www.twitter.com", "The birds");
    ws += wp1;
    ws += wp2;
    ws += wp3;
    cout << ws;
    ws -= wp2;
    cout << "After deleting facebook.com" << endl;
    cout << ws;
    return 0;
}
```

Problem 4

Write a class for **Student**. Each student has:

- name (dynamicly allocated char array)
- average (decimal number)
- year (int).

For the class implement:

- constructors and destructors
- operator++ increasing the year of the student
- operator « for printing the info of the student
- operator > for comparing two students by their average.

Then implement class **Group** with dynamic array of students. For this class implement:

- constructrs and destructors
- operator += adding new student in the group
- operator ++ increasing the year of all the students in the group
- operator « printing all the students
- method **reward** that prints only the students with larger average grade than 9.5.
- method **highestAverage** that prints the highest average grade of the group.

Materials and Questions

Lectures, exsercises and announcements
courses.finki.ukim.mk

Source code of all examples and problems
https://github.com/tdelev/SP/tree/master/latex/src

Questions and discussion
forum.finki.ukim.mk