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

Lesson 3

Classes

Object-oriented programming

Defining class and object

Difining class

```
class Class_name {
private:
    /* variables and methods not visible outside of class*/
public:
    /* variables and methods visible outside of class*/
};
```

Instantiating object

```
Class_name object_name;
```

Defining class
Example

```
#include <iostream>
using namespace std;
class Cuboid {
private:
    int a, b, c;
public:
    // Default constructor
    Cuboid(){
        cout << "Default constructor" << endl;
    }
    // Constructor with arguments
    Cuboid(int aa, int bb, int cc) {
        a = aa;
        b = bb;
        c = cc;
        cout << "Constructor" << endl;
    }
    ~Cuboid() { cout << "Destructor" << endl; }
    // Method for computing area
    int area() {
        return 2 * (a * b + a * c + b * c);
    }
    // Method for computing volume
    int volume() {
        return a * b * c;
    }
};
int main() {
    Cuboid c(10, 15, 20);
    cout << "Area is: " << c.area() << endl;
    cout << "Volumenot e: " << c.volume() << endl;
    return 0;
}
```

Objects lifecycle

1. Constructor - special method of each class which is called always when an object of the class is created (instantiated)
2. Destructor - special method of each class which is called to free the memory that the instantiated object allocated

Problem 1

Triangle class

Write a class for geometric figure triangle. In the class implement methods for computing the area and perimeter of the triangle. Than write a main function where you will instantiate one object of this class with values read from SI. Call the methods of this object for computing area and perimeter.

Problem 1

Solution 1/2 (class)

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

class Triangle {
private:
    int a, b, c;
public:
    // Constructor
    Triangle(int x, int y, int z) {
        a = x;
        b = y;
        c = z;
    }
    // Destructor
    ~Triangle() {
    }

    int permeter() {
        return a + b + c;
    }

    float area() {
        float s = (a + b + c) / 2;
        return sqrt(s * (s - a) * (s - b) * (s - c));
    }
};
```

Problem 1
Solution 2/2 (main)

```
int main() {
    int a, b, c;
    cin >> a >> b >> c;
    Triangle t(a, b, c);
    cout << "Area: " << t.area() << endl;
    cout << "Permeter: " << t.permeter() << endl;
    return 0;
}
```

Problem 2

Employee class

Write a class that will keep data for one employee:

- name
- sallary
- position (position can be employee, manager, owner)

Write a main function where from SI you will read data for N employees, and then print a list of employees sorted in descending order by the sallary.

Problem 2
Solution 1/4

```
#include <iostream>
#include <string.h>
using namespace std;

enum position {
    employee, manager, owner
};

class Person {
private:
    char name[100];
    int sallary;
    position pos;
public:
    // Constructors
    Person() {
    }

    Person(char *name, int sallary, position pos) {
        strcpy(this->name, name);
        this->sallary = sallary;
        this->pos = pos;
    }
    // Destruktor
    ~Person() {
    }
```

Problem 2
Solution 2/4

```
    void setName(char const *name) {
        strcpy(this->name, name);
    }
    void setSallary(int const sallary) {
        this->sallary = sallary;
    }
    void setPosition(position p) {
        pos = p;
    }
    char const *getName() {
        return name;
    }
    int const getSallary() {
        return sallary;
    }
    position const getPosition() {
        return pos;
    }
};
```

Problem 2
Solution 3/4

```
void sort(Person a[], int n) {
    int i, j;
    Person p;
    for (i = 0; i < n - 1; i++)
        for (j = i; j < n; j++)
            if (a[i].getSallary() < a[j].getSallary()) {
                p = a[j];
                a[j] = a[i];
                a[i] = p;
            }
}
```

Problem 2
Solution 4/4

```
int main() {
    Person employees[100];
    float sallary;
    int n, pos;
    char name[100];
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> name;
        cin >> sallary;
        cin >> pos;
        employees[i].setName(name);
        employees[i].setSallary(sallary);
        employees[i].setPosition((position) pos);
    }
    sort(employees, n);
    for (int i = 0; i < n; i++) {
        cout << i + 1 << ". " << employees[i].getName() << "\t"
             << employees[i].getSallary() << "\t"
             << employees[i].getPosition() << endl;
    }
    return 0;
}
```

Problem 3

E-mail class

Write a class that describes an e-mail message. The class should implement method for showing the contents of the message on the screen.
Then write a main function where the parameters of the message are read from SI and an object of this class is instantiated. Then call the method for printing the message. The validity of the e-mail address should be checked the existance of the @ character.

Problem 3
Solution 1/4

```
#include <iostream>
#include <cstring>
using namespace std;
class EMail {
private:
    enum {
        AddrLen = 100, SubLen = 200, BodyLen = 1000
    };
    char to[AddrLen];
    char from[AddrLen];
    char subject[SubLen];
    char body[BodyLen];
public:

    EMail(char *to, char *from, char *subject, char *body) {
        strncpy(this->to, to, AddrLen - 1);
        strncpy(this->from, from, AddrLen - 1);
        strncpy(this->subject, subject, SubLen - 1);
        strncpy(this->body, body, BodyLen - 1);
        to[AddrLen - 1] = subject[SubLen - 1] = 0;
        from[AddrLen - 1] = subject[SubLen - 1] = body[BodyLen - 1] = 0;
    }
    ~EMail() {
    }
```

Problem 3
Solution 2/3

```
    void setTo(char const *n) {
        strncpy(to, n, AddrLen - 1);
        to[AddrLen - 1] = 0;
    }
    void setFrom(char const *n) {
        strncpy(from, n, AddrLen - 1);
        from[AddrLen - 1] = 0;
    }
    void setSubject(char const *n) {
        strncpy(subject, n, SubLen - 1);
        subject[SubLen - 1] = 0;
    }
    void setBody(char const *n) {
        strncpy(body, n, BodyLen - 1);
        body[BodyLen - 1] = 0;
    }
    const char* getTo() { return to;    }
    const char* getFrom() { return from; }
    const char* getSubject() { return subject; }
    const char* getBody() { return body; }
    void print() {
        cout << "To: " << to << endl;
        cout << "From: " << from << endl;
        cout << "Subject: " << subject << endl;
        cout << "Body: " << body << endl;
    }
};
bool checkEmail(char* address);
```

Problem 3
Solution 3/3

```
int main() {
    char to[100], from[100], subject[200], body[1000];
    cout << "To: " << endl;
    cin >> to;
    if (checkEmail(to)) {
        cout << "From: " << endl;
        cin >> from;
        cout << "Subject: " << endl;
        cin >> subject;
        cout << "Body: " << endl;
        cin >> body;
        EMail poraka(to, from, subject, body);
        cout << "Sent:" << endl;
        poraka.print();
    } else {
        cout << "Invalid email address!" << endl;
    }
    return 0;
}
bool checkEmail(char *address) {
    int count = 0;
    while (*address != 0)
        if ('@' == *address++)
            count++;
    return (1 == count);
}
```

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