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

Object oriented programming

Exercises 3

Version 1.0, 24 February, 2017

Table of Contents

1. Classes

1.1. Defining `class`

Syntax

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

Example `oop_av3_ex1.cpp`

```
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;
    }
};
```

1.2. Creating (instantiating) objects

Syntax

```
class_name object_name;
```

Example `oop_av3_ex1.cpp`

```
    Cuboid c(10, 15, 20);
```

1.3. Objects lifecycle

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

2. Problems

2.1. Triangle

Write a class for geometric figure triangle. In the class implement methods for computing the area and perimeter of the triangle.

Then, 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.

Solution `oop_av31_en.cpp`

```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));
    }
};
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;
}
```

2.2. Employee

Write a class that will represent an employee with following attributes:

- name
- salary
- working position (employee, manager or 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 salary.

Solution `oop_av32_en.cpp`

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

enum position {
    employee, manager, owner
};

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

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

    void setName(char const *name) {
        strcpy(this->name, name);
    }
    void setSalary(int const salary) {
        this->salary = salary;
    }
    void setPosition(position p) {
        pos = p;
    }
    char *getName() const {
        return name;
    }
    int getSalary() const {
        return salary;
    }
    position getPosition() const {
        return pos;
    }
};

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].getSalary() < a[j].getSalary()) {
                p = a[j];
                a[j] = a[i];
                a[i] = p;
            }
}

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

2.3. E-mail

Write a class that defines 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 with existence of the `@` character.

Solution `oop_av33_en.cpp`

```
#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() {
    }
    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);

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);
}
```

3. Source code of the examples and problems

https://github.com/finki-mk/SP/

Source code ZIP