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

Lesson 4

Classes 2

Object-oriented programming

Problem 1

Define a class `Boy` that keeps information for name, surname and years. Define the needed constructors and method for printing in format:
Boy: [Name] [Surname] [Years].
Define a class `Girl` with the same attributes as Boy but with different printing format:
Girl: [Name] [Surname] [Years].
Define a class `Date` that contains one Girl and one Boy. Implement the function `print()` that will print the info of the date in the following format:
Date: Boy: [Name] [Surname] [Years], Girl: [Name] [Surname] [Years].

Write a function `areMatch()` that will return true if their age difference is less than 10 and the sum of the length of their names is even number, otherwise false.

Problem 1
Solution 1/3

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

class Mate {
private:
    char name[100];
    int age;
    bool isFemale;
public:
    Mate(const char *name = "noname", const int age = 0,
         const bool isFemale = true) {
        strcpy(this->name, name);
        this->age = age;
        this->isFemale = isFemale;
    }

    void print() {
        cout << (isFemale ? "Girl: " : "Boy: ") << " " << name;
        cout << " " << age << endl;
    }

    int getAge() {
        return age;
    }

    char *getName() {
        return name;
    }
```

Problem 1
Solution 2/3

```
class Date {
private:
    Mate mate1, mate2;
public:
    Date(const Mate mate1 = Mate(), const Mate mate2 = Mate()) {
        this->mate1 = mate1;
        this->mate2 = mate2;
    }
    void print() {
        cout << "Date: " << endl;
        mate1.print();
        mate2.print();
    }

    bool areMatch() {
        return abs(mate1.getAge() - mate2.getAge()) <= 10 &&
               (strlen(mate1.getName()) + strlen(mate2.getName())) % 2 == 0;
    }
};
```

Problem 1
Solution 3/3

```
int main() {
    char name[100];
    int age;
    int isFemale;
    cin >> name;
    cin >> age;
    cin >> isFemale;
    Mate mate1(name, age, isFemale == 1);
    cin >> name;
    cin >> age;
    cin >> isFemale;
    Mate mate2(name, age, isFemale == 1);
    Date d(mate1, mate2);
    d.print();
    if (d.areMatch()) {
        cout << "Good match." << endl;
    } else {
        cout << "Ooops, not a good match..." << endl;
    }
    return 0;
}
```

Problem 2

Define a class `Team` that keeps information about the name, year formed and the host city. Define a class `Game` that contains info for the home and guest teams (objects from class `Team`), goals scored by the home team, and goals scored by the guest team. Define separate function that accepts as arguments two objects from the class `Game` and checks if the second game is return match (the both teams are the same with reverse order). The the both games are return match, than should return the team that is winner on agregate, otherwise should print message that the games are not return match and return NULL.

Problem 2
Solution 1/3

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

class Team {
private:
    char name[100];
    int yearFormed;
    char city[100];
public:
    Team(const char *name = "", const int yearFormed = 0,
         const char *city = "") {
        strcpy(this->name, name);
        this->yearFormed = yearFormed;
        strcpy(this->city, city);
    }

    char *getName() {
        return name;
    }

    void print() {
        cout << name << " " << yearFormed << endl;
    }

};
```

Problem 2
Solution 2/3

```
class Game {
private:
    Team homeTeam;
    Team awayTeam;
    int homeGoals;
    int awayGoals;
public:
    Game(const Team homeTeam = Team(), const Team awayTeam = Team(),
         const int homeGoals = 0, const int awayGoals = 0) {
        this->homeTeam = homeTeam;
        this->awayTeam = awayTeam;
        this->homeGoals = homeGoals;
        this->awayGoals = awayGoals;
    }

    Team getHome() const {
        return homeTeam;
    }
    Team getAway() const {
        return awayTeam;
    }

    int getHomeGoals() const {
        return homeGoals;
    }

    int getAwayGoals() const {
        return awayGoals;
    }
```

Problem 2
Solution 3/3

```
Team winner(const Game game1, const Game game2) {
    if (strcmp(game1.getHome().getName(), game2.getAway().getName()) == 0 &&
            strcmp(game2.getHome().getName(), game1.getAway().getName()) == 0) {
        int goalsHome = game1.getHomeGoals() + game2.getAwayGoals();
        int goalsAway = game1.getAwayGoals() + game2.getHomeGoals();
        if (goalsHome >= goalsAway) {
            return game1.getHome();
            // return game2.getAway();
        } else {
            return game1.getAway();
        }
    }
    return Team();
}

int main() {
    Team t1("Barcelona", 1900, "Barcelona");
    Team t2("Real Madrid", 1890, "Madrid");
    Team t3("Elche", 1950, "Elche");
    Game g1(t1, t2, 5, 0);
    Game g2(t2, t1, 2, 2);
    Game g3(t1, t3, 6, 2);
    Team w = winner(g1, g3);
    w.print();
    return 0;
}
```

Problem 3

Define a class `Date` that will keep day, month and year (int). Than implement a class `Employee` that has name (max 20 chars), sallary and date of birth (object from class Date). Implement two functions that accept arguments array of Employee objects and their number. First function should return the employee with largest sallary, and the second one should return the youngest employee.

Write a main function that will test these classes and functions.

Problem 3
Solution 1/3

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

class Date {
private:
    int day, month, year;
public:
    Date (const int d = 0, const int m = 0, const int y = 0) {
        day = d;
        month = m;
        year = y;
    }

    int compare(const Date date) {
        if (year > date.year) return 1;
        else if (year < date.year) return -1;
        else if (month > date.month) return 1;
        else if (month < date.month) return -1;
        else if (day > date.day) return 1;
        else if (day < date.day) return -1;
        else return 0;
    }

};
```

Problem 3
Solution 2/3

```
class Employee {
private:
    char name[20];
    float salary;
    Date date_of_birth;
public:

    Employee(const char *n = "", const float s = 0, const Date d = Date()) {
        strcpy(name, n);
        salary = s;
        date_of_birth = d;
    }
    float get_salary() {
        return salary;
    }

    Date get_dateOfBirth() {
        return date_of_birth;
    }

    void print() {
        cout << name << " " << salary << endl;
    }

};
```

Problem 3
Solution 3/3

```
Employee highestsalary (Employee *e, int n) {
    Employee max = e[0];
    for (int i = 1; i < n; i++) {
        if (e[i].get_salary() > max.get_salary())
            max = e[i];
    }
    return max;
}

Employee youngest (Employee *e, int n) {
    Employee min = e[0];
    for (int i = 1; i < n; i++) {
        if (e[i].get_dateOfBirth().compare(min.get_dateOfBirth()) == 1)
            min = e[i];
    }
    return min;
}

int main() {
    Employee employee[10];
    employee[0] = Employee("Jack", 1234, Date(12, 12, 2014));
    employee[1] = Employee("John", 2457, Date(11, 10, 2013));
    employee[2] = Employee("Peter", 3076, Date(7, 5, 1980));
    Employee x = highestsalary(employee, 3);
    Employee y = youngest(employee, 3);
    x.print();
    y.print();
    return 0;
}
```

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