Универзитет „Св. Кирил и Методиј“ - Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Објектно ориентирано програмирање

Аудиториски вежби 4

Содржина

1. Композиција, copy-конструктор

1.1. Средба

Да се дефинира класа `Momche` која содржи информации за име, презиме и години. За класата да се дефинираат конструктори, деструктор и метод за печатење на објектот на екран во формат:

`Momche: Ime Prezime Godini.`

Да се дефинира класа `Devojche` со истите атрибути и методи со разлика во форматот на печатење:

`Devojche: Ime prezime godini.`

Креирајте класа `Sredba` која содржи податоци за едно момче и едно девојче.

- Креирајте функција `print()` која ги печати податоците за момчето и девојчето во следниот формат:

`Sredba: Momche: Ime Prezime Godini Devojche: Ime Prezime Godini.`

- Напишете функција `daliSiOdgovaraat()` која печати "Si odgovaraat" доколку разликата на нивните години е помала или еднаква на 5 или "Ne si odgovaraat" во спротивно.

Решение `oop_av41.cpp`

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

class Momche {
private:
    int godini;
    char ime[20];
    char prezime[20];
public:
    Momche(int godini = 0, char *ime = "", char *prezime = "") {
        this->godini = godini;
        strcpy(this->ime, ime);
        strcpy(this->prezime, prezime);
    }
    Momche(const Momche &m) {
        godini = m.godini;
        strcpy(ime, m.ime);
        strcpy(prezime, m.prezime);
    }
    ~Momche() {}
    int getGodini() {
```

```
        return godini;
    }
    void print() {
        cout << "Momche: " << ime << " " << prezime << " " << godini;
    }
};

class Devojche {
private:
    int godini;
    char ime[20];
    char prezime[20];
public:
    Devojche(int godini = 0, char *ime = "", char *prezime = "") {
        this->godini = godini;
        strcpy(this->ime, ime);
        strcpy(this->prezime, prezime);
    }
    Devojche(const Devojche &d) {
        godini = d.godini;
        strcpy(ime, d.ime);
        strcpy(prezime, d.prezime);
    }
    ~Devojche() {}
    int getGodini() {
        return godini;
    }
    void print() {
        cout << "Devojche: " << ime << " " << prezime << " " << godini;
    }
};

class Sredba {

private:
    Momche momche;
    Devojche devojche;
public:
    Sredba(const Momche m, const Devojche d) {
        momche = m;
        devojche = d;
    }
    ~Sredba() {}

    void print() {
        cout << "Sredba: ";
        momche.print();
        devojche.print();
    }
    void daliSiOdgovaraat() {
        if (abs(momche.getGodini() - devojche.getGodini()) < 5)
            cout << "Si odgovaraat" << endl;
        else
            cout << "Ne si odgovaraat" << endl;

    }
};

int main() {
    int godini;
    char ime[20], prezime[20];

    cout << "Informacii za momche: " << endl;
    cout << "Ime: ";
    cin >> ime;
    cout << "Prezime: ";
    cin >> prezime;
    cout << "Godini: ";
    cin >> godini;
    Momche m(godini, ime, prezime);
    Momche momche(m); //eksplicitno povikuvanje na copy konstruktor za momche

    cout << "Informacii za Devojche: " << endl;
    cout << "Ime: ";
    cin >> ime;
    cout << "Prezime: ";
```

```
    cin >> prezime;
    cout << "Godini: ";
    cin >> godini;
    Devojche d(godini, ime, prezime);
    Devojche devojche = d; //eksplicitno povikuvanje na copy konstruktor za devojche

    Sredba s(momche, devojche);//implicitno povikuvanje na copy konstruktor za momche i
devojche
    s.print();
    s.daliSiOdgovaraat();
    return 0;
}
```

1.2. Екипа

Да се дефинира класа `Ekipa` што содржи информации за име на екипата, година на формирање и градот од каде потекнува.

Да се дефинира класа `Natprevar` што содржи информации за домаќин, гостин (објекти од класата `Ekipa`), голови кои ги постигнал домаќинот и голови кои ги постигнал гостинот.

Да се дефинира посебна функција `revans` што како аргументи прима два објекта од класата `Natprevar` и проверува дали едниот натпревар е реванш на другиот. Еден натпревар е реваш на друг ако гостинот и домаќинот од првиот натпревар се истите екипи со домаќинот и гостинот од вториот натпревар, соодветно.

Да се дефинира функцијата `duel` која што како аргументи прима два објекта од класата `Natprevar` и ако едниот натпревар е ревашн на другиот функцијата треба да ја врати екипата која е подобра во меѓусебниот дуел. Во спротивно да испечати порака дека натпреварите не се совпаѓаат. Во случајот кога е нерешено функцијата враќа 0.

Решение `oop_av42.cpp`

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

class Ekipa {
private:
    int godina;
    char ime[20];
    char grad[20];
public:
    Ekipa(int godina = 0, char *ime = "", char *grad = "") {
        this->godina = godina;
        strcpy(this->ime, ime);
        strcpy(this->grad, grad);
    }
    Ekipa(const Ekipa &e) {
```

```
        godina = e.godina;
        strcpy(ime, e.ime);
        strcpy(grad, e.grad);
    }
    const char *getIme() {
        return ime;
    }
    ~Ekipa() {}
};

class Natprevar {
private:
    Ekipa domakin, gostin;
    int goloviDomakin, goloviGostin;
public:
    Natprevar(const Ekipa &d, const Ekipa &g, int gDom, int gGost) {
        domakin = d;
        gostin = g;
        goloviDomakin = gDom;
        goloviGostin = gGost;
    }
    Ekipa getDomakin() {
        return domakin;
    }
    Ekipa getGostin() {
        return gostin;
    }
    int getGoloviDomakin() {
        return goloviDomakin;
    }
    int getGoloviGostin() {
        return goloviGostin;
    }
    ~Natprevar() {}
};

bool revans(Natprevar n1, Natprevar n2) {
    if (!(strcmp(n1.getDomakin().getIme(), n2.getGostin().getIme()) == 0 && strcmp(n1
.getGostin().getIme(), n2.getDomakin().getIme()) == 0)) {
        return false;
    }
    return true;
}

Ekipa duel(Natprevar n1, Natprevar n2) {
    if (revans(n1, n2)) {
        int eGolovi1 = n1.getGoloviDomakin() + n2.getGoloviGostin();
        int eGolovi2 = n2.getGoloviDomakin() + n1.getGoloviGostin();
        if (eGolovi1 > eGolovi2) return n1.getDomakin();
        else if (eGolovi1 < eGolovi2) return n1.getGostin();
        else if (n1.getGoloviGostin() > n2.getGoloviGostin()) return n1.getGostin();
        else if (n1.getGoloviGostin() < n2.getGoloviGostin()) return n1.getDomakin();
        else return 0;
    }
    else {
        cout << "Ne se sofpagjaat." << endl;
        return 0;
    }
}

int main() {

    Ekipa e1(1880, "Real Madrid", "Madrid");
    Ekipa e2(1880, "FC Barcelona", "Barcelona");

    Natprevar n1(e1, e2, 1, 2);
    Natprevar n2(e2, e1, 1, 0);

    cout << duel(n1, n2).getIme();

    return 0;
}
```

1.3. Вработен

Да се напише класа `Datum` во која ќе се чуваат ден, месец и година (цели броеви).

Да се напише класа `Vraboten` во која се чува име на вработениот (не повеќе од 20 знаци), плата и датум на раѓање (објект од класата `Datum`).

Да се напишат две функции кои како аргументи примаат низа од вработени и големина на низата. Едната функција го враќа вработениот со најголема плата, а другата функција го враќа најмладиот вработен во фирмата.

Во главната програма потребно е да се испечатат на екран податоците за најмалдиот и најплатениот вработен. Печатењето на вработениот да биде реализирано со посебна функција `print()` во рамките на класата `Vraboten`.

Решение `oop_av43.cpp`

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

class Datum {
private:
    int den, mesec, godina;
public:
    Datum(int den = 0, int mesec = 0, int godina = 0) {
        this->godina = godina;
        this->mesec = mesec;
        this->den = den;
    }
    Datum(const Datum &d) {
        godina = d.godina;
        mesec = d.mesec;
        den = d.den;
    }
    ~Datum() {}

    int getDen() { return den; }
    int getMesec() { return mesec; }
    int getGodina() { return godina; }
};

class Vraboten {
private:
    char ime[20];
    int plata;
    Datum dataRagjanje;
public:
    Vraboten() {}
    Vraboten(char *ime, int plata, const Datum &data) {
        strcpy(this->ime, ime);
        this->plata = plata;
        dataRagjanje = data;
    }
    ~Vraboten() {}

    int getPlata() {
        return plata;
```

```
    }
    Datum getDataRagjanje() {
        return dataRagjanje;
    }
    void print() {
        cout << "Ime: " << ime << endl;
        cout << "Plata: " << plata << endl;
        cout << "Datum na ragjanje: " << dataRagjanje.getDen() << "." << dataRagjanje
.getMesec() << "." << dataRagjanje.getGodina() << endl;
    }
};
//go vrakja vraboteniot so najgolema plata od nizata v
Vraboten najgolemPlata(Vraboten v[], int n) {
    int max = v[0].getPlata();
    int ind = 0;
    for (int i = 1; i < n; i++) {
        if (v[i].getPlata() > max) {
            max = v[i].getPlata();
            ind = i;
        }
    }
    return v[ind];
}

//0 - isti se, 1 - datumot d1 e po datumot d2, 2 datumot d2 e po datumot d1
int sporedba(Datum d1, Datum d2) {
    if (d1.getGodina() > d2.getGodina()) return 1;
    else if (d1.getGodina() < d2.getGodina()) return 2;
    else if (d1.getMesec() > d2.getMesec()) return 1;
    else if (d1.getMesec() < d2.getMesec()) return 2;
    else if (d1.getDen() > d2.getDen()) return 1;
    else if (d1.getDen() < d2.getDen()) return 2;
    else return 0;
}
//go vrakja najmladiot vraboten od nizata v
Vraboten najmlad(Vraboten v[], int n) {
    Datum data(v[0].getDataRagjanje());
    int ind = 0;
    for (int i = 1; i < n; i++) {
        if (sporedba(v[i].getDataRagjanje(), data) == 1) {
            data = v[i].getDataRagjanje();
            ind = i;
        }
    }
    return v[ind];
}
int main() {
    Datum d1(1, 1, 1980);
    Datum d2(1, 2, 1983);
    Datum d3(11, 12, 1984);

    Vraboten v[3];
    Vraboten v1("Marjan", 40000, d1);
    Vraboten v2("Stefan", 30000, d2);
    Vraboten v3("Marko", 20000, d3);
    v[0] = v1;   v[1] = v2;   v[2] = v3;

    cout << "Najmlad vraboten: " << endl;
    najmlad(v, 3).print();

    cout << "Vraboten so najgolema plata: " << endl;
    najgolemPlata(v, 3).print();

    return 0;
}
```

2. Изворен код од примери и задачи

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

Source code ZIP