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

Објектно ориентирано програмирање

Аудиториски вежби 5

Верзија 1.0, 27 Февруари, 2017

Содржина

1. Пријателски функции, динамичка алокација на меморија

1.1. Пример 1

Да се дефинира класа `Ekipa` што содржи информации за име на екипата, назив на стадионот на кој игра и градот од каде потекнува.

Во главната програма да се креираат два покажувачи кон објект од класата `Ekipa`. Потоа отпечати ги информациите за креираните објекти.

Решение `oop_av51a.cpp`

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

class Ekipa {
private:
    char ime[20];
    char grad[20];
    char stadion[30];
public:
    Ekipa(char *ime = "", char *grad = "", char *stadion = "") {
        strcpy(this->ime, ime);
        strcpy(this->grad, grad);
        strcpy(this->stadion, stadion);
    }
    Ekipa(const Ekipa &e) {
        strcpy(ime, e.ime);
        strcpy(grad, e.grad);
        strcpy(stadion, e.stadion);
    }
    const char *getIme() {
        return ime;
    }
    const char *getGrad() {
        return grad;
    }
    const char *getStadion() {
        return stadion;
    }
    void setIme(char *ime) {
        strcpy(this->ime, ime);
    }
    ~Ekipa() {}
};

int main() {

    Ekipa *e1 = new Ekipa("Real Madrid", "Madrid", "Santiago Bernabeu");
    Ekipa *e2 = new Ekipa(*e1);

    cout << "Ekipite se e: ";
    cout << e1->getIme();
    cout << "-";
    cout << e2->getIme();

    //e1->getIme()->setIme("Barselona"); //GRESHKA
    e1->setIme("Barselona");

    cout << "\nPo promenata ekipite se: ";
    cout << e1->getIme();
    cout << "-";
    cout << e2->getIme();

    delete e1;
    delete e2;

    return 0;
}
```

1.2. Пример 2

Да се дефинира класа `Ekipa` што содржи информации за име на екипата, назив на стадионот на кој игра и градот од каде потекнува.

Да се дефинира класа `Natprevar` што содржи информации за домаќин, гостин

(покажувачи кон објекти од класата `Ekipa`), голови кои ги постигнал домаќинот и голови кои ги постигнал гостинот.

Да се дефинира глобална функција `isTip` која како аргумент добива еден објект од класата `Natprevar` и тип за натпреварот (еден знак: 1,2 или X) и враќа дали дадениот тип е точен за натпреварот.

Во главната програма да се креира објект од класата Natprevar и да се провери дали типот 1 е точен тип за креираниот натпревар.

Решение `oop_av51b.cpp`

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

class Ekipa {
private:
    char ime[20];
    char grad[20];
    char stadion[30];
public:
    Ekipa(char *ime = "", char *grad = "", char *stadion = "") {
        strcpy(this->ime, ime);
        strcpy(this->grad, grad);
        strcpy(this->stadion, stadion);
    }
    Ekipa(const Ekipa &e) {
        strcpy(ime, e.ime);
        strcpy(grad, e.grad);
        strcpy(stadion, e.stadion);
    }
    //get funkciite kako konstantni funkcii
    //vo niv ne se menuvaat podatocite od klasata
    const char *getIme() const {
        return ime;
    }
    const char *getGrad() const {
        return grad;
    }
    const char *getStadion() const {
        return stadion;
    }
    void setIme(char *ime) {
        strcpy(this->ime, ime);
    }
    ~Ekipa() {}
};

class Natprevar {
private:
    Ekipa *domakin, *gostin;
    int goloviDomakin, goloviGostin;

public:
    Natprevar(const Ekipa &d, const Ekipa &g, int gDom, int gGost) {
        domakin = new Ekipa(d);
        gostin = new Ekipa(g);
        goloviDomakin = gDom;
        goloviGostin = gGost;
    }
    Natprevar(const Natprevar& n) {
        domakin = new Ekipa(*n.domakin);
        gostin = new Ekipa(*n.gostin);

        goloviDomakin = n.goloviDomakin;
```

```
        goloviGostin = n.goloviGostin;
    }
    //vrakja pokazuvac kon prvata ekipag
    Ekipa* getDomakin() {
        return domakin;
    }
    //vrakja konstanten pokazuvac
    const Ekipa* getGostin() {
        return gostin;
    }
    int getGoloviDomakin() {
        return goloviDomakin;
    }
    int getGoloviGostin() {
        return goloviGostin;
    }
    ~Natprevar() {
        cout << "\nVo destruktor" << endl;
        delete domakin;
        delete gostin;
    }
    //friend funkcija
    friend bool isTip(Natprevar n, char tip);
};

bool  isTip(Natprevar n, char tip) {
    if (n.goloviDomakin == n.goloviGostin && tip == 'X') return true;
    else if (n.goloviDomakin > n.goloviGostin && tip == '1') return true;
    else if (n.goloviDomakin < n.goloviGostin && tip == '2') return true;
    else return false;
}

int main() {

    Ekipa e1("Real Madrid", "Madrid", "Santiago Bernabeu");
    Ekipa e2("FC Barcelona", "Barcelona", "Camp Nou");

    Natprevar first(e1, e2, 1, 3);

    cout << "Vnesi tip za natprevarot: ";
    cout << first.getDomakin()->getIme(); //getIme - const funkcija
    cout << "-";
    cout << first.getGostin()->getIme();
    cout << endl;


    char tip; //1, 2 ili X
    cin >> tip;

    if (isTip(first, tip)) cout << "Tipot e pogoden";
    else cout << "Tipot ne e pogoden";

    first.getDomakin()->setIme("RLM"); //mozno e
    //first.getGostin().setIme("BAR"); //ne e mozno:getGostin vrakja konstanten pokazuvac

    cout << "\nNatprevarot beshe megju: ";
    cout << first.getDomakin()->getIme();
    cout << "-";
    cout << first.getGostin()->getIme();

    return 0;
}
```

1.3. Пример 3

Да се дефинира класа `Ekipa` што содржи информации за име на екипата, назив на стадионот на кој игра и градот од каде потекнува.

Во главната програма да се креира покажувач кон динамичко алоцирано поле

од објекти кон класата `Ekipa`. Да се внесат N екипи од тастатура, да се сортираат по име и да се прикажат на екран.

Решение `oop_av51c.cpp`

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

class Ekipa {
private:
    char ime[20];
    char grad[20];
    char stadion[30];
public:
    Ekipa(char *ime = "", char *grad = "", char *stadion = "") {
        strcpy(this->ime, ime);
        strcpy(this->grad, grad);
        strcpy(this->stadion, stadion);
    }
    Ekipa(const Ekipa &e) {
        strcpy(ime, e.ime);
        strcpy(grad, e.grad);
        strcpy(stadion, e.stadion);
    }
    const char *getIme() {
        return ime;
    }
    const char *getGrad() {
        return grad;
    }
    const char *getStadion() {
        return stadion;
    }
    void setIme(char *ime) {
        strcpy(this->ime, ime);
    }
    ~Ekipa() {}
};

void sort(Ekipa *p, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (strcmp(p[i].getIme(), p[j].getIme()) > 0) {
                Ekipa pom = p[i];
                p[i] = p[j];
                p[j] = pom;
            }
        }

    }

}

int main() {

    int n;
    cin >> n;

    //pokazvac kon dinamicko alocirano pole od objekt od Ekipa
    Ekipa *prvenstvo = new Ekipa[n];

    char ime[20], grad[20], stadion[30];

    for (int i = 0; i < n; i++)  {
        cin >> ime >> grad;
        cin.getline(stadion, 29);
        prvenstvo[i] = Ekipa(ime, grad, stadion);
    }
    sort(prvenstvo, n);
    cout << "Ekipite od prvenstvoto se:\n";
    for (int i = 0; i < n; i++)  {
```

```
        cout << i + 1 << " " << prvenstvo[i].getIme() << " (" << prvenstvo[i].getGrad() <<
", " << prvenstvo[i].getStadion() << ")" << endl;
    }

    delete [] prvenstvo;

    return 0;
}
```

2. Задачи

2.1. Array

Да се напише класа `Array` за работа со еднодимензионални полиња од целобројни елементи. За полето се чуваат информации за неговиот вкупен капцитет, тековниот број на елементи. Резервацијата на меморијата да се врши динамички. Во класата да се имплементираат следните функции:

- `add` за додавање нови броеви во полето и притоа ако е исполнет капацитетот на полето (низата) да се зголеми за 100%.
- `change` која има два целобројни аргументи A и B и ги заменува сите броеви A од полето во B.
- `deleteAll` која ги брише сите појавувања на целобројниот аргумент во полето, а притоа капацитетот да не се промени.
- `print` за печатење на елементите од полето.

Да се тестира класата во main функција.

Решение `oop_av52.cpp`

```
#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];
        }
    }

    // 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;
    }

    void print () {
        for (int i = 0; i < size; ++i) {
            cout << x[i] << " ";
        }
        for (int i = size; i < capacity; ++i) {
            cout << "- ";
        }

        cout << endl;
    }
    void change(int n, int m) {
        for (int i = 0; i < size; ++i) {
            if (x[i] == n) x[i] = m;
        }
    }

    void deleteAll(int n) {
        int newSize = 0;
        for (int i = 0, j = 0; i < size; ++i)
            if (x[i] != n) {
                x[j++] = x[i];
                newSize++;
            }
        size = newSize;
    }

    void add(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++;
    }
};

int main() {
    Array a;
    a.add(6);
    a.add(4);
    a.add(3);
    a.add(2);
    a.add(1);

    Array b(a);
    Array c;
    c = a;

    b.add(2);
```

```
    b.change(2, 6);
    c.deleteAll(6);

    cout << " a: ";
    a.print();
    cout << " b: ";
    b.print();
    cout << " c: ";
    c.print();
    return 0;
}
```

2.2. Веб сервери

Да се напише класа за работа со веб сервери (`WebServer`). За секој веб сервер се чува: неговото име (max 30 знаци) и - листа од веб страници (динамички алоцирана низа од објекти од класата `WebPage`).

За секоја веб страница се чува: url (max 100 знаци) и содржина (динмички алоцирана низа од знаци).

За класата `WebPage` да се имплементираат функциите:

- Функција `daliseIsti` (`WebPage` p) за споредба на веб страници според url

За класата `WebServer` да се имплементираат функциите:

- `addPage`(`WebPage` p) за додавање нова веб страница во серверот ако таа не е во серверот. Во тој случак да се зголеми големината на низата од веб страниците во сервервот за 1.
- `deletePage`(`WebPage` p) за бришење на веб страница од веб серверот ако таа постои. Во тој случај големината на низата од веб страниците треба да се намали за 1.

Решение `oop_av53.cpp`

```
#include <iostream>
#include <cstring>
using namespace std;
class WebServer; //deklaracija na klasata WebServer

class WebPage {
private :
    char  url [100];
    char* sodrzina;
public :
    WebPage ( char* url = "", char* sodrzina = "" ) {
        strcpy(this -> url, url);
        this -> sodrzina = new char [strlen(sodrzina) + 1];
        strcpy(this -> sodrzina, sodrzina);
    }
```

```
    WebPage (const WebPage& wp) {
        strcpy(this->url , wp.url );
        this-> sodrzina = new char [strlen (wp.sodrzina) + 1];
        strcpy(this->sodrzina, wp.sodrzina );
    }

    ~WebPage () {
        delete [] sodrzina ;
    }
    bool daliIsti(WebPage& wp) {
        return strcmp(url, wp.url) == 0;
    }

    WebPage& operator= (WebPage& wp) {
        if (this != &wp) {
            strcpy (this -> url , wp.url);
            delete [] sodrzina ;
            this -> sodrzina = new char [strlen(wp.sodrzina) + 1];
            strcpy(this -> sodrzina, wp.sodrzina);
        }
        return *this ;
    }

    friend class WebServer; //prijatelska klasa

};

class WebServer {

private:
    char ime [30];
    int count ;
    WebPage* wp;

public:
    WebServer (const char * ime = "", int count = 0, WebPage *wp = 0) {
        strcpy(this ->ime, ime);
        this-> count = count ;
        this->wp = new WebPage [count];
        for (int i = 0; i < count ; i++)
            this->wp[i] = wp[i];
    }

    WebServer(const WebServer &ws) {
        strcpy (this->ime, ws.ime );
        this -> count = ws.count ;
        this -> wp = new WebPage[count];
        for (int i = 0; i < count ; i++)
            this -> wp[i] = ws.wp[i];
    }

    WebServer& operator= (const WebServer &ws) {
        if (this != &ws) {
            strcpy (this ->ime, ws.ime );
            this -> count = ws.count ;
            delete [] this -> wp;
            this -> wp = new WebPage[count];
            for (int i = 0; i < count; i++)
                this ->wp[i] = ws.wp[i];
        }
        return *this ;
    }

    ~ WebServer () {
        delete [] wp;
    }

    WebServer& addPage(WebPage webPage) {
        WebPage * tmp = new WebPage [count + 1]; //alociraj nova memorija
        //so kapacitet za eden povekje od prethodno
        for (int i = 0; i < count; i++)
            tmp [i] = wp[i];

        tmp [count++] = webPage ; //vmetni ja novata veb strana (webPage)
        delete [] wp;
        wp = tmp;
```

```
        return *this ;
    }

    WebServer& deletePage(WebPage webPage) {
        int newCount = 0;
        for (int i = 0; i < count; i++) {
            if (!wp[i].daliIsti(webPage)) {
                newCount++;
            }
        }
        //po brishenjeto kje ima newCount elementi
        WebPage* tmp = new WebPage[newCount];
        newCount = 0;
        for (int i = 0; i < count; i++) {
            if (!wp[i].daliIsti(webPage)) {
                tmp[newCount++] = wp[i];
            }
        }
        delete [] wp;
        wp = tmp;
        count = newCount ;
        return *this ;
    }

    void listPages () {
        cout << "Number: " << count << endl;
        for (int i = 0; i < count; i++)
            cout << wp[i].sodrzina << "- " << wp[i].url << endl ; //direkten pristap do
sodrzina i url
    }
};

int main () {
    WebPage wp1 ("http://www.google.com", "The search engine");
    WebPage wp2 ("http://www.finki.ukim.mk", "FINKI");
    WebPage wp3 ("http://www.time.mk", "Site vesti");

    WebServer ws(" Server ");

    ws.addPage(wp1) ;
    ws.addPage(wp2);
    ws.addPage(wp3) ;

    ws.listPages ();

    cout << "\nAfter delete: \n";
    ws.deletePage(wp3);

    ws.listPages ();

    return 0;
}
```

3. Изворен код од примери и задачи

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

Source code ZIP