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

Object oriented programming

Exercises 5

Version 1.0, 16 March, 2017

Table of Contents

1. Friend functions, dynamic memory allocation

1.1. Example 1

Define a class `Team` with information for the name of the team, the stadium that they play and their home city.

In the main function create two pointers to objects of class `Team`. Then print the information for the created objects.

Solution `oop_av51a_en.cpp`

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

class Team {
private:
    char name[20];
    char city[20];
    char stadium[30];
public:
    Team(char *name = "", char *city = "", char *stadium = "") {
        strcpy(this->name, name);
        strcpy(this->city, city);
        strcpy(this->stadium, stadium);
    }
    Team(const Team &e) {
        strcpy(name, e.name);
        strcpy(city, e.city);
        strcpy(stadium, e.stadium);
    }
    const char *getName() {
        return name;
    }
    const char *getCity() {
        return city;
    }
    const char *getStadium() {
        return stadium;
    }
    void setName(char *name) {
        strcpy(this->name, name);
    }
    ~Team() {}
};

int main() {

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

    cout << "Teams are: ";
    cout << e1->getName();
    cout << "-";
    cout << e2->getName();

    //e1->getName()->setName("Barselona"); // error
    e1->setName("Barselona");

    cout << "\nAfter the change teams are: ";
    cout << e1->getName();
    cout << "-";
    cout << e2->getName();

    delete e1;
    delete e2;

    return 0;
}
```

1.2. Example 2

Define class `Game` with information for home team and away team (pointers to objects of class `Team`), goals scored by the home team and goals scored by the away team.

Define a global functions `isPick` that as arguments accepts single object of the class

Game and a pick (single character: 1, 2 or X) and returns if the selected pick holds for the game.

In the main function create an object of class Game and check if the pick 1 holds for the created game.

Solution oop_av51b_en.cpp

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

class Team {
private:
    char name[20];
    char city[20];
    char stadium[30];
public:
    Team(char *name = "", char *city = "", char *stadium = "") {
        strcpy(this->name, name);
        strcpy(this->city, city);
        strcpy(this->stadium, stadium);
    }
    Team(const Team &e) {
        strcpy(name, e.name);
        strcpy(city, e.city);
        strcpy(stadium, e.stadium);
    }
    // get methods as const functions
    // they do not mutate the state of the class
    const char *getName() const {
        return name;
    }
    const char *getCity() const {
        return city;
    }
    const char *getStadium() const {
        return stadium;
    }
    void setName(char *name) {
        strcpy(this->name, name);
    }
    ~Team() {}
};

class Game {
private:
    Team *home, *away;
    int goalsHome, goalsAway;

public:
    Game(const Team &d, const Team &g, int gHome, int gAway) {
        home = new Team(d);
        away = new Team(g);
        goalsHome = gHome;
        goalsAway = gAway;
    }
    Game(const Game& n) {
        home = new Team(*n.home);
        away = new Team(*n.away);

        goalsHome = n.goalsHome;
        goalsAway = n.goalsAway;
    }
    // returns pointer of the home team
    Team* getHome() {
        return home;
    }
    // returns const pointer
    const Team* getAway() {
        return away;
    }
    int getGoalsHome() {
        return goalsHome;
    }
    int getGoalsAway() {
        return goalsAway;
    }
    ~Game() {
        cout << "\ndestructor" << endl;
        delete home;
        delete away;
    }
    // friend function
    friend bool isPick(Game n, char tip);
};

bool  isPick(Game n, char tip) {
    if (n.goalsHome == n.goalsAway && tip == 'X') return true;
    else if (n.goalsHome > n.goalsAway && tip == '1') return true;
    else if (n.goalsHome < n.goalsAway && tip == '2') return true;
    else return false;
}

int main() {

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

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

    cout << "Enter pick for the game: ";
    cout << first.getHome()->getName(); //getName - const function
    cout << "-";
    cout << first.getAway()->getName();
    cout << endl;


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

    if (isPick(first, tip)) cout << "You won!";
    else cout << "You lost!";

    first.getHome()->setName("RLM"); // possible
    //first.getAway().setName("BAR"); // not possible :getAway returns const pointer

    cout << "\nGame between: ";
    cout << first.getHome()->getName();
    cout << "-";
    cout << first.getAway()->getName();

    return 0;
}
```

1.3. Example 3

In the main function create pointer to dynamically allocated array of objects of the class `Team`. Enter N teams from SI, and sort by their name and print on SO.

Solution `oop_av51c_en.cpp`

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

class Team {
private:
    char name[20];
    char city[20];
    char stadium[30];
public:
    Team(char *name = "", char *city = "", char *stadium = "") {
        strcpy(this->name, name);
        strcpy(this->city, city);
        strcpy(this->stadium, stadium);
    }
    Team(const Team &e) {
        strcpy(name, e.name);
        strcpy(city, e.city);
        strcpy(stadium, e.stadium);
    }

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

    const char *getCity() const {
        return city;
    }

    const char *getStadium() const {
        return stadium;
    }

    void setName(char *name) {
        strcpy(this->name, name);
    }
};

void sort(Team *teams, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (strcmp(teams[i].getName(), teams[j].getName()) > 0) {
                Team temp = teams[i];
                teams[i] = teams[j];
                teams[j] = temp;
            }
        }

    }

}

int main() {

    int n;
    cin >> n;

    // pointer to a dynamically allocated array of teams
    Team *league = new Team[n];

    char name[20], city[20], stadium[30];

    for (int i = 0; i < n; i++)  {
        cin >> name >> city;
        cin.getline(stadium, 29);
        league[i] = Team(name, city, stadium);
    }
    sort(league, n);
    cout << "League teams:\n";
    for (int i = 0; i < n; i++)  {
        cout << i + 1 << " " << league[i].getName() << " (" << league[i].getCity() << ", "
<< league[i].getStadium() << ")" << endl;
    }

    delete [] league;

    return 0;
}
```

2. Problems

2.1. Array

Implement class `Array` for working with one-dimensional array of integers. Reserve the memory dynamically. Implements the following functions in the class:

- `add` - adding new elements (integer), and if the capacity is filled it should be expanded for 100%.
- `replaceAll` - that accepts two integer arguments A and B and replaces all occurrences of element A with the element B.
- `deleteAll` - that deletes all occurrences of the passed argument.
- `print` - for printing the array

Test the class in the main function.

Solution `oop_av52_en.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];
        }
    }

    // assignment 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 replaceAll(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.replaceAll(2, 6);
    c.deleteAll(6);

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

2.2. Web Server

Write a class that represents a `WebServer`. For each web server we store:

- name (max 30 chars)
- list of web pages (dynamically allocated array of objects of class `WebPage`.

For each web page we store:

- url (max 100 characters)
- contents (dynamically allocated array of characters).

In the class `WebPage` implement the following functions:

- `equal(WebPage wp)` - for comparing two web pages by their url

For the class `WebServer` implement:

- `addPage(WebPage wp)` - for adding new web page if it doesn't exist in the server already. Increase the capacity for +1.
- `deletePage(WebPage wp)` - for deleting a web page from the server if it exists. The capacity should be decreased for -1.

Solution `oop_av53_en.cpp`

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

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

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

    ~WebPage() {
        delete [] contents ;
    }
    bool equal(WebPage& wp) {
        return strcmp(url, wp.url) == 0;
    }
    WebPage& operator=(WebPage& wp) {
        if(this != &wp) {
            strcpy(this->url , wp.url);
            delete [] contents ;
            this->contents = new char [strlen(wp.contents) + 1];
            strcpy(this->contents, wp.contents);
        }
        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]; // allocate new array
        // with increased capacity for +1
        // copy the contents of the current array
        for(int i = 0; i < count; i++)
            tmp [i] = wp[i];

        tmp [count++] = webPage ; // enter the new webpage
        delete [] wp; // delete the old array
        wp = tmp; // move the pointer to the new array
        return *this ;
    }

    WebServer& deletePage(WebPage webPage) {
        int newCount = 0;
        for(int i = 0; i < count; i++) {
            if(!wp[i].equal(webPage)) {
                newCount++;
            }
        }
        // after the deletion there will be newCount elements
        WebPage* tmp = new WebPage[newCount];
        newCount = 0;
        for(int i = 0; i < count; i++) {
            if(!wp[i].equal(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].contents << "- " << wp[i].url << endl ; // direct access of
contents and 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. Source code of the examples and problems

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

Source code ZIP