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

Објектно ориентирано програмирање

Аудиториски вежби 3

Верзија 1.0, 21 Февруари, 2017

Содржина

1. Класи

1.1. Дефинирање класа

Синтакса

```
class ime_na_klasata {
private:
/* deklaracija na promenlivi i metodi koi ne se vidlivi nadvor od klasata */
public :
/* deklaracija na promenlivi i metodi koi se vidlivi nadvor od klasata */
};
```

Пример `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. Креирање (инстанцирање) објекти

Синтакса

```
ime_na_klasa ime_na_objekt;
```

Пример `oop_av3_ex1.cpp`

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

1.3. Животен циклус на објектите

- Конструктор (Constructor) - посебен метод на секоја класа кој се повикува секогаш кога се креира (инстанцира) објект од класата
- Деструктор (Destructor) - посебен метод на секоја класа кој се повикува кога треба да се ослободи меморијата која ја опфатил конструираниот објект

2. Задачи

2.1. Триаголник

Да се напише класа за опишување на геометриско тело триаголник. Во класата да се напишат методи за пресметување на плоштината и периметарот на триаголникот.

Потоа да се напише главна програма во која ќе се инстнацира еден објект од оваа класа со вредности за страните кои претходно ќе се прочитаат од стандарден влез. На овој објект да се повикат соодветните методи за пресметување на плоштината и периметарот.

Пример `oop_av31.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. Вработен

Да се напише класа во која ќе се чуваат основни податоци за вработен:

- име
- плата
- работна позиција (работната позиција може да биде вработен, директор или шеф).

Напишете главна програма во која се читаат од стандарден влез податоци за N вработени, а потоа се печати листа на вработените сортирани според висината на платата во опаѓачки редослед.

Пример `oop_av32.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 const *getName() {
        return name;
    }
    int const getSalary() {
        return salary;
    }
    position const getPosition() {
        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

Да се напише класа која опишува една e-mail порака. Во класата треба да се имплементира метод за прикажување на целокупната порака на екран. Потоа да се напише главна програма во која се внесуваат параметрите на пораката, се инстанцира објект од оваа класа и се печати на екран неговата содржина. За проверување на валидноста на e-mail пораката (постоење на знакот @ во адресата) да се напише соодветна функција.

Пример `oop_av33.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. Изворен код од примери и задачи

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

Source code ZIP