



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

Аудиториски вежби 3

Класи и преоптоварување на оператори

Напреден развој на софтвер

Дефинирање класа

```
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 */  
};
```

Дефинирање класа

Пример

```
#include <iostream>
using namespace std;
class Kvadar {
private:
    int a, b, c;
public:
    // Default constructor
    Kvadar(){
        cout << "Default constructor" << endl;
    }
    // Constructor so argumenti
    Kvadar(int aa, int bb, int cc) {
        a = aa;
        b = bb;
        c = cc;
        cout << "Constructor" << endl;
    }
    ~Kvadar() { cout << "Destructor" << endl; }
    // Metod za presmetuvanje plostinata
    int plostinata() {
        return 2 * (a * b + a * c + b * c);
    }
    // Metod za presmetuvanje volumen
    int volumen() {
        return a * b * c;
    }
};

int main() {
    Kvadar k1(10, 15, 20);
    cout << "Plostinata e: " << k1.plostinata() << endl;
    cout << "Volumenot e: " << k1.volumen() << endl;
    return 0;
}
```

Животен циклус на објектите

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

Задача 1

Класа за 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 тастатура. На овој објект да се повикат соодветните методи за пресметување на плоштината и периметарот.

Задача 1

Решение

```
#include <iostream>
using namespace std;
class Triagolnik {
private:
    int a, b, c;
public:
    Triagolnik(int x, int y, int z) {
        a = x;
        b = y;
        c = z;
    }
    ~Triagolnik() {}
    int perimetar() {
        return a + b + c;
    }
};

int main() {
    int p, q, r;
    cout << "Vnesi tri strani:" << endl;
    cin >> p >> q >> r;
    Triagolnik t(p, q, r);
    cout << "Perimetar na triagolnikot:" << t.perimetar() << endl;
    return 0;
}
```

Задача 2

Класа за вработен

Да се напише класа во која ќе се чуваат основни податоци за вработен (име, плата и работна позиција). Работната позиција може да биде вработен, директор или шеф.

Напишете главна програма во која се внесуваат податоци за N вработени од тастатура, а потоа се пачати листа на вработените сортирани според висината на платата во опаѓачки редослед.

Задача 2

Решение 1/4

```
#include <iostream>
#include <string.h>
using namespace std;
enum pozicija {
    vraboten, direktor, sef
};
class Person {
private:
    // podatoci
    char name[100];
    int plata;
    pozicija pos;
public:
    // konstruktori
    Person() {
    }

    Person(char *n, int s, pozicija p) {
        strcpy(name, n);
        plata = s;
        pos = p;
    }
    //destruktor
    ~Person() {
    }
}
```

Задача 2

Решение 2/4

```
// Interfejs funkcii
void setName(char const *n) {
    strcpy(name, n);
}
void setPlata(int p) {
    plata = p;
}
void setPosition(pozicija p) {
    pos = p;
}
char const *getName(void) {
    return name;
}
int const getPlata(void) {
    return plata;
}
pozicija const getPosition(void) {
    return pos;
}
};
```

Задача 2

Решение 3/4

```
void Sort(Person a[], int n);

int main() {
    Person vraboteni[100];
    float plata;
    int n, pos;
    char name[100];
    cout << "Vnesi broj na vraboteni (max 100): ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Vnesi ime: ";
        cin >> name;
        cout << "Vnesi plata: ";
        cin >> plata;
        cout << "Vnesi pozicija (0 - vraboten, 1 - direktor, 2 - shef): ";
        cin >> pos;
        vraboteni[i].setName(name);
        vraboteni[i].setPlata(plata);
        vraboteni[i].setPosition((pozicija) pos);
    }
    Sort(vraboteni, n);
    for (int i = 0; i < n; i++) {
        cout << i + 1 << ". " << vraboteni[i].getName() << "\t"
            << vraboteni[i].getPlata() << "\t"
            << vraboteni[i].getPosition() << endl;
    }
    return 0;
}
```

Задача 2

Решение 4/4

```
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].getPlata() < a[j].getPlata()) {
                p = a[j];
                a[j] = a[i];
                a[i] = p;
            }
}
```

Задача 3

Класа за 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.

Задача 3

Решение 1/4

```
#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() {
    }
```

Задача 3

Решение 2/3

```
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 pecati() {
    cout << "To: " << to << endl;
    cout << "From: " << from << endl;
    cout << "Subject: " << subject << endl;
    cout << "Body: " << body << endl;
}
};
bool checkEmail(char* address);
```

Задача 3

Решение 3/3

```
int main() {
    char to[100], from[100], subject[200], body[1000];
    cout << "Vnesi do kogo e porakata: " << endl;
    cin >> to;
    if (checkEmail(to)) {
        cout << "Vnesi od kogo e porakata: " << endl;
        cin >> from;
        cout << "Vnesi naslov: " << endl;
        cin >> subject;
        cout << "Vnesi soдрzina: " << endl;
        cin >> body;
        EMail poraka(to, from, subject, body);
        cout << "Isprateno:" << endl;
        poraka.pecati();
    } else {
        cout << "Pogresna adresa za isprakjanje!" << endl;
    }
    return 0;
}

bool checkEmail(char *address) {
    int count = 0;
    while (*address != 0)
        if ('@' == *address++)
            count++;
    return (1 == count);
}
```

Задача 4

Класа за комплексни броеви

```
#include <iostream>
#include <cmath>
using namespace std;
class kompleksen {
private:
    float re, im;
public:
    kompleksen(float x = 0.0, float y = 0.0) {
        re = x;
        im = y;
    }
    ~kompleksen() {}
    void print() {
        cout << re;
        if (im < 0)
            cout << "-j";
        else
            cout << "+j";
        cout << fabs(im) << endl;
    }
}
```

Задача 4

Преоптоварување на оператори $+=$, $-=$, $*=$, $/=$

```
kompleksen& operator+=(const kompleksen &a) {
    re += a.re;
    im += a.im;
    return *this;
}
kompleksen& operator-=(const kompleksen &a) {
    re -= a.re;
    im -= a.im;
    return *this;
}
kompleksen& operator*=(const kompleksen &a) {
    re = re * a.re - im * a.im;
    im = im * a.re + re * a.im;
    return *this;
}
kompleksen& operator/=(const kompleksen &a) {
    float m = a.re * a.re + a.im * a.im;
    float r = (re * a.re - im * a.im) / m;
    im = (im * a.re + re * a.im) / m;
    re = r;
    return *this;
}
```

Задача 4

Преоптоварување на оператори +, -, *, /

```
kompleksen operator+(kompleksen a, kompleksen b) {  
    kompleksen c = a;  
    return c += b;  
}  
kompleksen operator-(kompleksen a, kompleksen b) {  
    kompleksen c = a;  
    return c -= b;  
}  
kompleksen operator*(kompleksen a, kompleksen b) {  
    kompleksen c = a;  
    return c *= b;  
}  
kompleksen operator/(kompleksen a, kompleksen b) {  
    kompleksen c = a;  
    return c /= b;  
}
```

Задача 4

Преоптоварување на оператори «, »

```
friend ostream& operator<<(ostream &out, const kompleksen &k) {
    out << k.re;
    if (k.im < 0)
        out << "-j";
    else
        out << "+j";
    out << fabs(k.im) << endl;
    return out;
}

friend istream& operator>>(istream &in, kompleksen &k) {
    cout << "Vnesi realen del: ";
    in >> k.re;
    cout << "Vnesi imaginaren del: ";
    in >> k.im;
    return in;
}
```

Задача 4

Главна функција

```
int main() {  
    kompleksen k1(2.0, -3.5), k2(1.0, 7.5); //2-j3.5, 1+j7.5  
    kompleksen k = k1 + k2;  
    k.print();  
    k = k1 - k2;  
    cin >> k;  
    cout << k;  
    return 0;  
}
```

Материјали и прашања

Предавања, аудиториски вежби, соопштенија
courses.finki.ukim.mk

Изворен код на сите примери и задачи
bitbucket.org/tdelev/finki-nrs

Прашања и одговори
qa.finki.ukim.mk