



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

## Аудиторски вежби 3

Класи

Објектно-ориентирано програмирање

# Дефинирање класа и објект

## Дефиниција на 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 */  
};
```

## Инстанцирање објект

```
Ime_na_klasata ime_na_objektot;
```

# Дефинирање класа

## Пример

```
#include <iostream>
using namespace std;
class Square {
private:
    int a, b, c;
public:
    // Default constructor
    Square(){
        cout << "Default constructor" << endl;
    }
    // Constructor with arguments
    Square(int aa, int bb, int cc) {
        a = aa;
        b = bb;
        c = cc;
        cout << "Constructor" << endl;
    }
    ~Kvadar() { 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;
    }
};

int main() {
    Square s(10, 15, 20);
    cout << "Area is: " << s.area() << endl;
    cout << "Volumenot e: " << s.volume() << 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 стандарден влез. На овој објект да се повикат соодветните методи за пресметување на плоштината и периметарот.

# Задача 1

## Решение 1/2 (Класа)

```
#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 perimeter() {
        return a + b + c;
    }

    float area() {
        float s = (a + b + c) / 2;
        return sqrt(s * (s - a) * (s - b) * (s - c));
    }
};
```

# Задача 1

## Решение 2/2 (main)

---

```
int main() {  
    int a, b, c;  
    cin >> a >> b >> c;  
    Triangle t(a, b, c);  
    cout << "Area: " << t.area() << endl;  
    cout << "Perimeter: " << t.perimeter() << endl;  
    return 0;  
}
```

---

## Задача 2

### Класа за 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 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 position {
    employee, manager, owner
};

class Person {
private:
    char name[100];
    int sallary;
    position pos;
public:
    // Constructors
    Person() {
    }

    Person(char *name, int sallary, position pos) {
        strcpy(this->name, name);
        this->sallary = sallary;
        this->pos = pos;
    }

    // Деструктор
    ~Person() {
    }
}
```

## Задача 2

### Решение 2/4

---

```
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;
}
};
```

---

## Задача 2

### Решение 3/4

---

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

---

# Задача 2

## Решение 4/4

```
int main() {
    Person employees[100];
    float sallary;
    int n, pos;
    char name[100];
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> name;
        cin >> sallary;
        cin >> pos;
        employees[i].setName(name);
        employees[i].setSallary(sallary);
        employees[i].setPosition((position) pos);
    }
    sort(employees, n);
    for (int i = 0; i < n; i++) {
        cout << i + 1 << ". " << employees[i].getName() << "\t"
            << employees[i].getSallary() << "\t"
            << employees[i].getPosition() << endl;
    }
    return 0;
}
```

## 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 print() {
    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 << "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);
}
```



# Материјали и прашања

Предавања, аудиториски вежби, соопштенија  
[courses.finki.ukim.mk](https://courses.finki.ukim.mk)

Изворен код на сите примери и задачи  
<https://github.com/tdelev/OOP/tree/master/latex/src>

Прашања и дискусија  
[forum.finki.ukim.mk](https://forum.finki.ukim.mk)