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

# Објектно ориентирано програмирање

Аудиториски вежби 1

# Содржина

1. Структури . . . . . 1
    1.1. Date . . . . . 1
    1.2. Vector . . . . . 2
    1.3. Комплексни броеви . . . . . 3
    1.4. Студенти . . . . . 4
    1.5. Држави . . . . . 6
2. Изворен код од примери и задачи . . . . . 8

# 1. Структури

## 1.1. Date

Да се напише програма која ќе споредува два датуми (ден, месец, година) и ќе ја пресмета разликата во денови од едниот до другиот датум. Пресметките да се реализираат во посебни функции. За датумот да се дефинира посебна структура `date`.

*Решение* `oop_av11.c`

```c
#include <stdio.h>

struct date {
    int year;
    int month;
    int day;
};
typedef struct date date;
// 1 d1 > d2
// 0 d1 == d2
// -1 d1 < d2
int compare(date d1, date d2) {
    if (d1.year > d2.year) return 1;
    else if (d1.year < d2.year) return -1;
    else { // d1.year == d2.year
        if (d1.month > d2.month) return 1;
        else if (d1.month < d2.month) return -1;
        else { // d1.month == d2.month
            if (d1.day > d2.day) return 1;
            else if (d1.day < d2.day) return -1;
            else return 0;
        }
    }
}
/**
* This function is aproximation
*/
int days(date d1, date d2) {
    int days = d1.day - d2.day;
    days += (d1.month - d2.month) * 30;
    days += (d1.year - d2.year) * 365;
    return days;
}

void read(date *d) {
    scanf("%d.%d.%d", &d->day, &d->month, &(*d).year);
}

void print(date *d) {
    printf("%02d.%02d.%d\n", d->day, d->month, d->year);
}

int main() {
    date d1;
    date d2;
    read(&d1);
    read(&d2);

    int res = compare(d1, d2);
    if (res == 0) {
        printf("Dates are equal\n");
    } else if (res > 0) {
        printf("The difference in days is %d days.\n", days(d1, d2));
    } else {
        printf("The difference in days is %d days.\n", days(d2, d1));
    }
    return 0;
}
```

## 1.2. Vector

Да се напише програма која ќе го пресметува векторскиот и скаларниот производ на два вектори. Векторите се претставени со координати во тродимензионален координатен систем. Скаларниот и векторскиот производ

да се пресметуваат со посебни функции. За вектор да се дефинира посебна структура `vector`.

*Решение* `oop_av12.c`

```c
#include <stdio.h>

struct vector {
    float x;
    float y;
    float z;
};

typedef struct vector vector;

float scalar_product(vector v1, vector v2) {
    return v1.x * v2.x + v1.y + v2.y + v1.z * v2.z;
}

vector vector_product(vector v1, vector v2) {
    vector v;
    v.x = v1.y * v2.z - v1.z * v2.y;
    v.y = v1.z * v2.x - v1.x * v2.z;
    v.z = v1.x * v2.y - v1.y * v2.x;
    return v;
}


int main () {
    vector v1 = { 2, 4, 6 };
    vector v2 = { 3, 5, 9 };
    vector v = vector_product(v1, v2);
    printf("v1 * v2 = %.2f\n", scalar_product(v1, v2));
    printf("v1 x v2 = [%.2f, %.2f, %.2f]\n", v.x, v.y, v. z);
    return 0;
}
```

## 1.3. Комплексни броеви

Да се напише програма во која ќе се дефинира структура за претставување комплексни броеви. Потоа да се напишат функции за собирање, одземање и множење на два комплексни броја. Програмата да се тестира во главна програма во која се вчитуваат два комплексни броја од стандарден влез.

*Решение* `oop_av13.c`

```c
#include <stdio.h>
#include <math.h>

typedef struct complex_number {
    float real;
    float imag;
} comp;

comp add(comp a, comp b) {
    comp c = a;
    c.real += b.real;
    c.imag += b.imag;
    return c;
}

comp subtract(comp *pok1, comp *pok2) {
    comp c = *pok1;
    c.real -= (*pok2).real;
    c.imag -= (*pok2).imag;
    return c;
}

void multiply(comp a, comp b, comp *c) {
    c->real = a.real * b.real - a.imag * b.imag;
    c->imag = a.real * b.imag + a.imag * b.real;
}

void print(comp *pok) {
    printf("%.2f", pok->real);
    if (pok->imag >= 0)
        printf("+j%.2f\n", pok->imag);
    else
        printf("-j%.2f\n", fabs(pok->imag));
}

int main() {
    comp a, b, c;
    scanf("%f %f", &a.real, &a.imag);
    scanf("%f %f", &b.real, &b.imag);
    print(&a);
    print(&b);
    printf("a + b\n");
    c = add(a, b);
    print(&c);
    printf("a - b\n");
    c = subtract(&a, &b);
    print(&c);
    printf("a * b\n");
    multiply(a, b, &c);
    print(&c);
    return 0;
}
```

## 1.4. Студенти

Од стандарден влез се читаат податоци за непознат број студенти (не повеќе од 100). Податоците се внесуваат така што во секој ред се состои од:

- името
- презимето
- бројот на индекс (формат xxyzzzz)

- четири броја (поени од секоја задача)

со произволен број празни места или табулатори меѓу нив.

Да се напише програма која ќе испечати список на студенти, каде во секој ред ќе има: презиме, име, број на индекс, вкупен број на бодови сортиран според бројот на бодови. При тоа имињата и презимињата да се напишат со голема почетна буква.

*Решение* `oop_av14.c`

```
#include <stdio.h>
#include <string.h>

struct student {
    char first_name[15];
    char last_name[20];
    int number;
    int points;
};

void norm(char *s) {
    // First letter uppercase, others lowercase
    *s = toupper(*s);
    while (*(++s) != '\0')
        *s = tolower(*s);
}

void sort(struct student a[], int n) {
    int i, j;
    struct student s;
    for (i = 0; i < n; i++)
        for (j = 0; j < n - i - 1; j++)
            if (a[j].points < a[j + 1].points) {
                s = a[j];
                a[j] = a[j + 1];
                a[j + 1] = s;
            }
}

int main() {
    struct student st[50];
    int i, n;
    scanf("%d", &n);
    for (i = 0; i < n; ++i) {
        scanf("%s", &st[i].first_name);
        scanf("%s", &st[i].last_name);
        scanf("%d", &st[i].number);
        int j, zadaca;
        st[i].points = 0;
        for(j = 0; j < 4; j++) {
            scanf("%d", &zadaca);
            st[i].points += zadaca;
        }
        norm(st[i].first_name);
        norm(st[i].last_name);
    }
    sort(st, n);
    for (i = 0; i < n; i++) {
        printf("%d. %s %s\t%d\t%d\n", i + 1, st[i].first_name, st[i].last_name, st[i]
.number, st[i].points);
    }
    return 0;
}
```

## 1.5. Држави

Да се напише програма која од стандарден влез ќе чита податоци за држави и на екран ќе го отпечати името и презимето на претседателот на државата чиј што главен град има најмногу жители. Податоци за држава:

- име
- претседател
- главен град
- број на жители.

Податоци за град:

- име
- број на жители.

Податоци за претседател:

- име
- презиме
- политичка партија.

*Решение* `oop_av15.c`

```c
#include<stdio.h>

typedef struct city {
    char name[30];
    long population;
} city;

typedef struct president {
    char name[20];
    char party[20];
} pres;

typedef struct country {
    char name[30];
    pres president;
    long population;
    city capital;
} country;

int main() {
    country d[20];
    int n, i, maxi, max;
    scanf("%d", &n);
    for (i = 0; i < n; ++i) {
        scanf("%s", &d[i].name);
        printf("president:\n");
        scanf("%s", &d[i].president.name);
        scanf("%s", &d[i].president.party);
        scanf("%d", &d[i].population);
        scanf("%s", &d[i].capital.name);
        scanf("%d", &d[i].capital.population);
    }
    maxi = 0;
    max = d[maxi].capital.population;
    for (i = 0; i < n; ++i)
        if (d[i].capital.population > max) {
            max = d[i].capital.population;
            maxi = i;
        }
    printf(
            "Name of the president of the country with the largest capital is: %s\n",
            d[maxi].president.name);
    return 0;
}
```

# 2. Изворен код од примери и задачи

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

Source code ZIP