17.05.2022

PROGRAMARE PROCEDURALĂ
– LABORATOR NR. 13 –

1. Fișierul text *test.txt* conține cuvinte despărțite prin semnele de punctuație uzuale, pe mai multe linii. Scrieți un program care să afișeze pe ecran cuvintele din fișier în ordinea descrescătoare a frecvențelor lor. Considerăm faptul că fișierul text conține cel mult 1000 de cuvinte distincte, iar lungimea maximă a unui cuvânt este de 30 de caractere.

Rezolvare:

Vom memora cuvintele distincte din fișier într-un tablou cu structuri de forma {*cuvânt*, *frecvență*}.

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

typedef struct
{
    char cuv[31];
    int frecv;
} Cuvant;

int cmpf(const void * a, const void * b)
{
    Cuvant va = *(Cuvant*)a;
    Cuvant vb = *(Cuvant*)b;

    if(va.frecv < vb.frecv) return 1;
    if(va.frecv > vb.frecv) return -1;

    //cazul va.frecv == vb.frecv
    return strcmp(va.cuv, vb.cuv);
}

int main()
{
    int i, nd = 0;
    Cuvant dist[1000];
    char str[1001];
    FILE *fp;
    char* t;

    fp = fopen("test.txt", "r");
    while(fgets(str, 1000, fp) != NULL)
    {
```

```
        t = strtok(str, " ,.:;?!\n");
        while(t != NULL)
        {
            for(i = 0; i < nd; i++)
                if(strcmp(dist[i].cuv, t) == 0)
                {
                    dist[i].frecv++;
                    break;
                }

            if(i == nd)
            {
                strcpy(dist[nd].cuv, t);
                dist[nd].frecv = 1;
                nd++;
            }

            t = strtok(NULL," ,.:;?!\n");
        }
    }

    fclose(fp);

    qsort(dist,nd,sizeof(Cuvant),cmpf);

    printf("Cuvintele distincte in ordinea descrescatoare a frecventelor:\n");
    for(i = 0; i < nd; i++)
        printf("%s --> %d\n",dist[i].cuv, dist[i].frecv);

    return 0;
}
```

2. Definiți o structură numită *Angajat* care să permită memorarea numelui, vârstei și salariului unui angajat. Fișierul *angajati.txt* conține pe prima linie un număr natural nenul *n* și pe următoarele *n* linii informații despre câte un angajat, despărțite prin virgule. Scrieți un program care să încarce datele din fișierul text într-un tablou unidimensional alocat dinamic și să afișeze următoarele informații pe ecran:

a) informațiile despre un angajat al cărui nume se citește de la tastatură;
b) salariul maxim și numele angajaților care au salariul respectiv;
c) salariul mediu din firmă;
d) angajații sortați alfabetic după nume;
e) angajații sortați descrescător după salarii, la salarii egale crescător după vârstă și la vârste egale alfabetic.

Rezolvare:

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

```c
#include <string.h>

typedef struct{
    unsigned int varsta;
    float salariu;
    char nume[50];
} Angajat;

float salariuMaxim(const Angajat *angajati, const unsigned int nr_ang){
    float sal_max = 0;
    unsigned int index;
    for(index = 0; index < nr_ang; index++)
        if(sal_max < angajati[index].salariu)
            sal_max = angajati[index].salariu;
    return sal_max;
}

float salariuMediu(const Angajat *angajati, const unsigned int nr_ang){
    float sum = 0;
    unsigned int index;
    for(index = 0; index < nr_ang; index++)
        sum += angajati[index].salariu;
    return sum / nr_ang;
}

int comparatorAlfabeticAngajati(const void* angajat_1, const void* angajat_2){
    return strcmp(((Angajat *)angajat_1)->nume, ((Angajat *)angajat_2)->nume);
}

int megaComparator(const void* angajat_1, const void* angajat_2){
    Angajat *ang_1 = (Angajat *)angajat_1;
    Angajat *ang_2 = (Angajat *)angajat_2;

    if(ang_1->salariu < ang_2->salariu)
        return 1;
    if(ang_1->salariu > ang_2->salariu)
        return -1;

    if(ang_1->varsta < ang_2->varsta)
        return -1;
    if(ang_1->varsta > ang_2->varsta)
        return 1;

    return strcmp(ang_1->nume, ang_2->nume);
}

int main()
{
    FILE *f_ang = fopen("angajati.txt", "r");

    unsigned int nr_ang, index;
    fscanf(f_ang, "%u%*c", &nr_ang);

    Angajat *angajati = (Angajat *)malloc(nr_ang * sizeof(Angajat));
    char linie[200], *camp;

    for(index = 0; index < nr_ang; index++){
```

```
        fgets(linie, 200, f_ang);
        camp = strtok(linie, ",");
        strcpy(angajati[index].nume, camp);
        camp = strtok(NULL, ",");
        sscanf(camp, "%u", &angajati[index].varsta);
        camp = strtok(NULL, ",\n");
        sscanf(camp, "%f", &angajati[index].salariu);
    }

    printf("\nAngajatii inregistrati sunt:\n");
    for(index = 0; index < nr_ang; index++)
        printf("%s - %u - %.2f\n", angajati[index].nume, angajati[index].varsta,
                                   angajati[index].salariu);

    float sal_max = salariuMaxim(angajati, nr_ang);
    printf("\nAngajatii cu salariu maxim sunt:\n");
    for(index = 0; index < nr_ang; index++)
        if(angajati[index].salariu == sal_max)
            printf("%s - %u - %.2f\n", angajati[index].nume, angajati[index].varsta,
                                       angajati[index].salariu);

    printf("\nSalariul mediu din firma este: %.2f\n", salariuMediu(angajati, nr_ang));

    qsort(angajati, nr_ang, sizeof(Angajat), comparatorAlfabeticAngajati);

    printf("\nAngajatii sortati alfabetic sunt:\n");
    for(index = 0; index < nr_ang; index++)
        printf("%s - %u - %.2f\n", angajati[index].nume, angajati[index].varsta,
                                   angajati[index].salariu);

    qsort(angajati, nr_ang, sizeof(Angajat), megaComparator);

    printf("\nAngajatii mega sortati sunt:\n");
    for(index = 0; index < nr_ang; index++)
        printf("%s - %u - %.2f\n", angajati[index].nume, angajati[index].varsta,
                                   angajati[index].salariu);

    fclose(f_ang);
    return 0;
}
```

3. Scrieți o funcție cu număr variabil de parametri care să concateneze mai multe șiruri de caractere.

Rezolvare:

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

//n = numarul parametrilor variabili
char* concatenare(int n, ...)
{
    int i, total_lungimi;
    char *rez, *sir;

    va_list lparam, copie_lparam;
```

```
    va_start(lparam, n);
    va_copy(copie_lparam, lparam);

    //parcurg lista pentru a calcula suma lungimilor
    //tuturor sirurilor si a spatiilor dintre ele
    total_lungimi = n-1;
    for(i = 0; i < n; i++)
    {
        sir = va_arg(lparam, char*);
        total_lungimi += strlen(sir);
    }

    //aloc dinamic sirul rezultat
    rez = (char*)malloc(total_lungimi + 1);

    //parcurg copia listei initiale pentru a
    //concatena sirurile cu cate un spatiu intre ele
    strcpy(rez, "");
    for(i = 0; i < n; i++)
    {
        sir = va_arg(copie_lparam, char*);
        strcat(rez, sir);
        strcat(rez, " ");
    }

    rez[total_lungimi] = '\0';

    va_end(lparam);
    va_end(copie_lparam);

    return rez;
}

int main()
{
    char *r;

    r = concatenare(3, "Popa", "Gh.", "Ion");
    printf("Concatenare cu 3 siruri: %s\n", r);

    r = concatenare(5, "Popescu", "R.E.", "Maria", "Ioana", "Gabriela");
    printf("Concatenare cu 5 siruri: %s\n", r);

    free(r);
    return 0;
}
```