

**PROVA SCRITTA DEL MODULO DI
ELEMENTI DI INFORMATICA
CORSO DI LAUREA IN INGEGNERIA BIOMEDICA
7 febbraio 2019**

MOTIVARE IN MANIERA CHIARA LE SOLUZIONI PROPOSTE A CIASCUNO DEGLI ESERCIZI SVOLTI

ESERCIZIO 1 (4 punti)

Convertire in notazione binaria il valore 289 utilizzando al massimo 8 bit.

ESERCIZIO 2 (4 punti)

Descrivere in modo chiaro e sintetico l'architettura di un moderno sistema operativo.

ESERCIZIO 3 (25 punti)

(8 punti) All'esame del Prof. Marcialis si sono presentati molti studenti e per passare lo scritto hanno dovuto risolvere tre quesiti. La lista è stata memorizzata in un file "iscritti.txt" formattato nel modo seguente. Ciascuno studente è rappresentato dal suo numero di matricola, un valore intero, seguito da tre interi che rappresentano il voto conseguito su ciascuno dei tre esercizi.

Un esempio di file "iscritti.txt" è il seguente:

```
708721334 1 1 15
707523111 2 2 10
707578921 3 3 18
```

Come si nota, il file non presenta gli iscritti ordinati per matricola.

Essendo necessario pubblicare gli esiti nel più breve tempo possibile, il Prof. Marcialis vi incarica di scrivere un programma C che, leggendo il file degli iscritti, produca un analogo file "esiti.txt" formattato nel seguente modo:

```
707523111 14 Prova scritta non superata
707578921 24 Orale facoltativo
708721334 17 Orale obbligatorio
```

Come si può notare, il file degli esiti presenta gli iscritti ordinati per matricola; per ciascuna riga, segue la somma dei parziali ed infine abbiamo una stringa esplicativa sull'esito del voto. Secondo le note regole, sarà "Prova scritta non superata" se il voto è strettamente inferiore a 16, "Orale obbligatorio" se il voto è compreso tra 16 e 20, "Orale facoltativo" in tutti gli altri casi.

Nello scrivere il programma, decidete di definire una struttura dati chiamata `studente` composta dagli slot seguenti:

- Un intero chiamato `matricola`, dove immettere la matricola dello studente;
- Un vettore di tre interi chiamato `parziali`, dove immettere i voti dei tre esercizi;
- Un intero chiamato `voto`, dove immettere il punteggio complessivo;
- Una stringa chiamata `esito`, allocata dinamicamente, che corrisponderà all'esito della prova.

In un vettore di strutture così definite, memorizzate i dati letti dal file e il risultato delle operazioni intermedie.

Implementate poi le seguenti funzioni:

- 1) (5 punti) `leggiRiga(f)`: legge una riga del file `f` aperto in lettura e restituisce un puntatore a variabile di tipo `studente` con gli slot appropriati aggiornati secondo la riga letta. Restituisce `NULL` se si è arrivati a fine file.
- 2) (2 punti) `somma(v, n)`: restituisce la somma di tutti gli elementi memorizzati nel vettore di `n` interi `v`.
- 3) (3 punti) `esito(x)`: restituisce una delle tre possibili stringhe "Orale obbligatorio", "Orale facoltativo", "Prova scritta non superata" in funzione del valore dell'intero `x`.
- 4) (7 punti) `scriviEsiti(nomefile, l, indici, n)`: stampa su file di nome `nomefile` gli slot appropriati, da ciascun elemento del vettore di strutture di tipo `studente`, `l`, formattando la stampa secondo le indicazioni fornite. Gli elementi vengono stampati secondo l'ordine di indici inserito nel vettore di interi `indici`. I vettori `l` e `indici` hanno la stessa dimensione `n`.

Oltre a queste quattro funzioni, utilizzate anche la funzione `ordina(v, n)` che, dato un vettore `v` di `n` interi, restituisce un vettore costituito dagli indici di `v` corrispondenti ai valori ordinati in ordine crescente. Per esempio, se `v={708721334, 707523111, 707578921}`, restituisce il vettore `{1, 2, 0}`.

Nota: Sebbene non sia necessario implementare `ordina()` chi lo farà **correttamente** avrà un bonus di 10 punti.

Soluzione dell'esercizio 1.

La rappresentazione di 289 con soli 8 bit non è possibile, essendo il massimo intero rappresentabile pari a 255. Potrebbe essere possibile solo utilizzando un eccesso pari a -34, tale cioè da rappresentare il minimo valore, 34, con la configurazione ad otto zeri, e il valore 289 con la configurazione ad otto uni. Sarebbero esclusi però tutti i valori inferiori a 34.

Soluzione dell'esercizio 2.

V. dispense del corso.

Soluzione dell'esercizio 3.

```
/* In questa soluzione sono state inserite delle "bandierine" attraverso printf
   per aiutare gli studenti a vedere quel che accade passo-passo durante
   l'esecuzione del codice. Queste bandierine possono essere rimosse una volta
   chiaro il funzionamento. Premere INVIO per procedere quando l'esecuzione si
   ferma. */
```

```
#include <stdio.h>
#include <stdlib.h>
#define N 200

typedef struct
{
    int matricola, voto;
    int parziali[3];
    char *esito;
} studente;

studente* leggiRiga(FILE* f)
{
    studente* s;
    int i;

    if (feof(f))
        return NULL;

    s=(studente*)malloc(sizeof(studente));

    fscanf(f,"%d", &s->matricola);
    printf("%d\n", s->matricola); //bandierina
    for (i=0; i<3; i++)
        fscanf(f,"%d",&s->parziali[i]);

    return s;
}

int somma(int *v, int n)
{
    int i, s;
    for (i=0, s=0; i<n; i++)
        s=s+v[i];
    return s;
}
```

```

char* esito(int x)
{
    if (x<16)
        return "Prova scritta non superata";
    else
        if (x<21)
            return "Orale obbligatorio";
        return "Orale facoltativo";
}

void scriviEsiti(char* nomefile, studente* l, int* indici, int n)
{
    FILE *f;
    int i;
    studente s;

    f=fopen(nomefile,"w");

    for (i=0; i<n; i++)
    {
        s=l[indici[i]];
        fprintf(f,"%d %d %s\n",s.matricola, s.voto, s.esito);
    }

    fclose(f);
}

```

//IMPLEMENTAZIONE DI ordina FUNZIONE FACOLTATIVA

```
int trovato(int i, int* v, int n)
{
```

```
    int j;
```

```
    printf("\n\t\t\tCerco %d nel vettore",i); //bandierina
```

```
    for(j=0; j<n; j++)
```

```
        if (v[j]==i)
```

```
            return 1;
```

```
    return 0;
```

```
}
```

```
int trovaMinimoCondizionato(int* v, int n, int* indici, int m)
```

```
{
```

```
    int i, imin;
```

```
    imin=0;
```

```
    printf("\n\t\tVettore indici con %d elementi",m); //bandierina
```

```
    for(i=0; i<n; i++)
```

```
        if (!trovato(i,indici,m)) //devo escludere gli indici precedenti
```

```
        {
```

```
            if (v[i]<v[imin])
```

```
                imin=i;
```

```
        }
```

```
        else
```

```
            printf(" ---- presente"); //bandierina
```

```
    printf("\n\t\t\timin=%d",imin); //bandierina
```

```
    return imin;
```

```
}
```

```
int *ordina(int *v, int n) //stile selection-sort
```

```
{
```

```
    int *indici, i, imin, m;
```

```
    indici=(int*)malloc(sizeof(int)*n);
```

```
    for (i=0, m=0; i<n; i++, m++)
```

```
    {
```

```
        printf("\n\tTrovo l'indice corrispondente al minimo escludendo quelli  
presenti nel vettore indici - iterazione %d",i); //bandierina
```

```
        imin=trovaMinimoCondizionato(v,n,indici,m);
```

```
        indici[m]=imin;
```

```
        getch(); //bandierina: ferma l'esecuzione - premere INVIO per continuare
```

```
    }
```

```
    return indici;
```

```
}
```

```
////////////////////////////////////
```

```

int main()
{
    FILE *f;
    studente s[N], *st;
    int *indici, matricole[N];
    int n;

    f=fopen("iscritti.txt","r");
    st=leggiRiga(f);
    n=0;
    while(st)
    {
        st->voto=somma(st->parziali, 3);
        st->esito=esito(st->voto);
        matricole[n]=st->matricola;
        s[n]=*st;
        free(st);
        st=leggiRiga(f);
        n++;
    }
    fclose(f);

    printf("\nOrdinamento del vettore matricole\n"); //bandierina
    indici=ordina(matricole, n);
    scriviEsiti("esiti.txt", s, indici, n);

    free(indici);
    return 0;
}

```