

Fondamenti di Informatica A-L – (Prof. P. Zingaretti)

PROVA SCRITTA – 18 marzo 2003

Avvertenze:

- * Consegnare **solo fogli formato A4**
- * **Scrivere su un solo lato** (no fronte-retro)
- * In ordine di preferenza usare: 1) **inchiostro nero**; 2) matita; 3) inchiostro rosso; 4) inchiostro blu
- * **In testa a ciascun foglio** scrivere: cognome, nome, numero progressivo di pagina rispetto al totale
esempio per il secondo foglio di 3 consegnati: Giuseppe Russo 2/3
- * Mantenere sul banco il **libretto o altro documento di riconoscimento** fino a controllo avvenuto
- * Nient'altro deve trovarsi sul banco: **non è consentito consultare libri, dispense, appunti, ecc.**
- * La **correzione** di riferimento per l'autovalutazione verrà effettuata in questa stessa aula alle ore **17:45**
- * La consegna delle **fotocopie** dei compiti avverrà presso l'Istituto di Informatica al termine della correzione
- * La **prova orale** si terrà **martedì 25 marzo** alle ore **9:00** presso l'Istituto di Informatica



Si ricorda che chi si presenterà all'orale **DEVE** portare l'implementazione al computer della propria soluzione, eventualmente corretta, **corredata di tutto quanto necessario alla verifica** del corretto funzionamento.

1. (7 punti)

Si definisca una funzione C (Pascal) che riceve in input un array di N (variabile) interi e restituisce il numero di elementi dell'array con valore minore del valore medio di tutti gli elementi.

2. (9 punti)

Un prodotto sia caratterizzato dai seguenti attributi: codice, numero pezzi, prezzo di acquisto medio. Si definiscano in linguaggio C (Pascal) una rappresentazione per un insieme di prodotti e una funzione che li ordina in base al numero (crescente) di pezzi secondo l'algoritmo del bubblesort.

3. (14 punti)

Si definisca 'valore' di una stringa la somma dei codici ASCII dei caratteri costituenti la stringa stessa. Si scriva una funzione C (Pascal) che inserisce una stringa su una lista dinamica (rappresentazione collegata mediante puntatore) di stringhe di lunghezza variabile in modo ordinato in base al valore della stringhe.

1. (7 punti)

Si definisca una funzione C (Pascal) che riceve in input un array di N (variabile) interi e restituisce il numero di elementi dell'array con valore minore del valore medio di tutti gli elementi.

```
int less_ave(int *V, int N);

int less_ave(int *V, int N) {
    int i,num=0;
    float media=0;
    for (i=0; i<N; i++) media+=V[i];
    media/=N;
    for (i=0; i<N; i++) if (V[i] < media) num++;
    return num;
};
```

2. (9 punti)

Un prodotto sia caratterizzato dai seguenti attributi: codice, numero pezzi, prezzo di acquisto medio. Si definiscano in linguaggio C (Pascal) una rappresentazione per un insieme di prodotti e una funzione che li ordina in base al numero (crescente) di pezzi secondo l'algoritmo del bubblesort.

```
#define TRUE 1
#define FALSE 0
typedef int boolean;

struct record {
    int codice;
    int num_pezzi;
    int prezzo;
};

void sort(struct record *V, int N);
void swap(struct record *V, int index_1, int index_2);

void sort(struct record *V, int N) {
    int iter=0, count;
    boolean iterazione_a_vuoto=FALSE;

    while (iter < N-1 && iterazione_a_vuoto==FALSE) {
        iterazione_a_vuoto=TRUE;
        for (count=0; count < N-1-iter; count++)
            if (V[count].num_pezzi > V[count+1].num_pezzi) {
                swap(V, count, count+1);
                iterazione_a_vuoto=FALSE;
            }
        iter++;
    }
}

void swap(struct record *V, int index_1, int index_2) {
    struct record tmp;
    tmp=V[index_1];
    V[index_1]=V[index_2];
    V[index_2]=tmp;
}
```

3. (14 punti)

Si definisca 'valore' di una stringa la somma dei codici ASCII dei caratteri costituenti la stringa stessa. Si scriva una funzione C (Pascal) che inserisce una stringa su una lista dinamica (rappresentazione collegata mediante puntatore) di stringhe di lunghezza variabile in modo ordinato in base al valore della stringhe.

```
typedef struct list_tag {
    char *str;
    struct list_tag *next;
} list;

int str_value(char *str) {
    int value=0, i=0;
    while (str[i]!='\0') value+=str[i++];
    return value;
}

void ins_ordI(list **lpp, char *str) {
    list *tmp_lp;
    int i, value=str_value(str);

    while (*lpp!=NULL && str_value((*lpp)->str) < value)
        lpp=&((*lpp)->next); //posizionamento
    //inserimento
    tmp_lp=*lpp; //salvo resto della lista
    *lpp=(list *)malloc(sizeof(list));
    (*lpp)->str=(char *)malloc(sizeof(str)+1);
    for (i=0; i<=sizeof(str); i++) (*lpp)->str[i]=str[i];
    //oppure strcpy((*lpp)->str,str);
    (*lpp)->next=tmp_lp; //ricollego
}
```

```

int less_ave(int *V, int N) {
    int i,num=0;
    float media=0;
    for (i=0; i<N; i++) media+=V[i]; //accumulo valori
    media/=N; //possibile errore se media non è float o double!!!
    for (i=0; i<N; i++) if (V[i] < media) num++;
    return num;
};
*****
struct record {
    int codice;
    int num_pezzi;
    int prezzo;
};

void sort(struct record *V, int N) {
    int iter=0, count;
    boolean iterazione_a_vuoto=FALSE; //controllo array già ordinato!!!

    while (iter < N-1 && iterazione_a_vuoto==FALSE) { //bubblesort
        iterazione_a_vuoto=TRUE;
        for (count=0; count < N-1-iter; count++)
            if (V[count].num_pezzi > V[count+1].num_pezzi) {
                swap(V, count, count+1);
                iterazione_a_vuoto=FALSE;
            }
        iter++;
    }
}

void swap(struct record *V, int index_1, int index_2) {
    struct record tmp;
    tmp=V[index_1];
    V[index_1]=V[index_2];
    V[index_2]=tmp;
}
*****
typedef struct list_tag {
    char *str; //bisogna allocare lo spazio necessario!!!
    struct list_tag *next;
} list;

int str_value(char *str) {
    int value=0, i=0;
    while (str[i]!='\0') value+=str[i++];
    return value;
}

void ins_ordI(list **lpp, char *str) {
    list *tmp_lp;
    int i, value=str_value(str);

    while (*lpp!=NULL && str_value((*lpp)->str) < value)
        lpp=&((*lpp)->next); //posizionamento
    //inserimento
    tmp_lp=*lpp; //salvo resto della lista
    *lpp=(list *)malloc(sizeof(list));
    (*lpp)->str=(char *)malloc(sizeof(str)+1);
    for (i=0; i<=sizeof(str); i++) (*lpp)->str[i]=str[i];
    //oppure strcpy((*lpp)->str,str);
    (*lpp)->next=tmp_lp; //ricollego
}

```