C: simulazione compito e note

Dal compito del 21.07.2017

Esercizio 5 (max 12 punti)

Siano dati due file di testo (esiti_compito1.txt e esiti_compito2.txt) contenenti gli esiti di due successivi compiti. Ogni file contiene la matricola e il voto in trentesimi di N studenti. Ogni riga riporta matricola e voto di uno studente separati da uno spazio. Le matricole contenute nei due file sono uguali e rispettano lo stesso ordine.

Dal compito del 21.07.2017

Esercizio 5 (max 12 punti)

Scrivere un programma in C che preso in input (da tastiera) un numero di matricola permetta (tramite apposita funzione) di valutare se lo studente indicato ha avuto un andamento stabile (il voto delle due prove è uguale), crescente (il voto del compito2 è maggiore del voto del compito1) o decrescente (il voto del compito1 è maggiore del voto del compito2). Il programma deve stampare a video l'andamento dello studente selezionato.

Dal compito del 21.07.2017

Soluzione

```c
#include <stdio.h>

int variazione(int matricola);

int main() {
    int ris;
    int i=0, matricola;

    printf("Inserisci matricola:\n");
    scanf("%d",&matricola);
    ris=variazione(matricola);
```

```
    switch (ris) {
      case 0:
        printf("La matricola %d non e' presente\n",matricola);
        break;
      case 1:
        printf("La matricola %d
        ha un andamento in diminuzione\n",matricola);
        break;
      case 2:
        printf("La matricola %d
        ha un andamento in aumento\n",matricola);
        break;
      case 3:
        printf("La matricola %d
        ha un andamento stabile\n",matricola);
        break;
    }
}
```

```c
int variazione(int matricola) {
    int m1,v1,m2,v2,ris=0;
    FILE *fp1,*fp2;
    fp1=fopen("esiti_compito1.txt","r");
    fp2=fopen("esiti_compito2.txt","r");

    if(fp1!=NULL && fp2!=NULL) {
        while(feof(fp1)==0) {
            fscanf(fp1,"%d %d",&m1,&v1);
            fscanf(fp2,"%d %d",&m2,&v2);

            if(m1==matricola) {
                if(v1 > v2)
                    ris=1;
                else if(v1 < v2)
                    ris=2;
                else if(v1 == v2)
                    ris=3;
                break;
            } } }
    return ris;
}
```

Alcuni considerazioni sulla lettura da file

Contenuto del file numbers.txt: `12345`

```
#include <stdio.h>
int main() {
  int n;
  FILE *fp1;
  fp1=fopen("numbers.txt","r");

  if(fp1!=NULL) {
      while(feof(fp1)==0) {
          fscanf(fp1,"%d",&n);
          printf("%d ",n);
  }
}
}
```

In questo caso leggo solo un numero: `12345`

Se volessimo leggere i numeri intesi come singola cifra tra 0 e 9?

```c
#include <stdio.h>
int main() {
  int n;
  FILE *fp1;
  fp1=fopen("numbers.txt","r");

  if(fp1!=NULL) {
      while(fscanf(fp1,"%1d",&n)==1) {
          printf("%d ",n);
  }
}
}
```

Attenzione all'uso del `%1d`

Se volessimo inserire i valori in un array?

```c
#include <stdio.h>
int main() {
  int n,v[100],i=0,j;
  FILE *fp1;
  fp1=fopen("numbers.txt","r");

  if(fp1!=NULL) {
      while(fscanf(fp1,"%1d",&v[i])==1) {
          i++;
  }
}

for(j=0;j<i;j++)
  printf("%d ",v[j]);

}
```

Se gli elementi del file sono separati da uno spazio: `1 2 3 4 5`

```c
#include <stdio.h>
int main() {
  int n,v[100],i=0,j;
  FILE *fp1;
  fp1=fopen("/Users/matteo/Downloads/numbers.txt","r");

  if(fp1!=NULL) {
      while(fscanf(fp1,"%d",&v[i])==1) {
          i++;
  }
}

for(j=0;j<i;j++)
  printf("%d ",v[j]);

}
```

Non serve usare `%1d` !

Dal compito del 19.07.2018

Esercizio 5 (max 30 punti)

Scrivere un programma in C che:

1. permetta all'utente di inserire da tastiera n numeri interi (max 2 punti)

```
printf("Quanti numeri vuoi inserire da tastiera?\n");
  scanf("%d",&n);
  while(n>DIM || n<1) {
    printf("ATTENZIONE. Inserisci un valore compreso
    tra 1 e %d\n",DIM);
    scanf("%d",&n);
  }
```

2. carichi tali numeri in un array evitando di inserire duplicati al suo interno. Prima di caricare un numero nell'array sarà pertanto necessario verificare (utilizzando una specifica funzione) la sua presenza all'interno dell'array (max 10 punti)

```c
int check_duplicati(int , int *, int );
```

```c
  for(i=0;i<n;++i) {
    printf("Inserisci elemento %d\n",i+1);
    scanf("%d",&num);
    check=check_duplicati(num,arr,ind);
    if(check==0) {
      printf("Num inserito nell'array.\n");
      arr[ind]=num;
      ++ind;
    }
  }
```

```c
int check_duplicati(int num, int *v, int ind) {
  int i,check=0;
  for(i=0;i<ind;++i)
    if(num==*(v+i)) check=1;
  return check;
}
```

3. carichi in una lista concatenata (utilizzando una specifica funzione) i duplicati, cioè tutti quei numeri che essendo già contenuti nell'array sono stati identificati come duplicati al punto 2 e che quindi non sono stati inseriti nell'array (max 10 punti)

Esempio:

2 (numero inserito nell'array)

4 (numero inserito nell'array)

2 (numero duplicato – inserito nella lista)

3 (numero inserito nell'array)

```c
int check_duplicati(int , int *, int );
```

```c
struct elemento *inserisci(struct elemento *p, int num);
```

```c
  for(i=0;i<n;++i) {
    printf("Inserisci elemento %d\n",i+1);
    scanf("%d",&num);
    check=check_duplicati(num,arr,ind);
    if(check==0) {
      printf("Num inserito nell'array.\n");
      arr[ind]=num;
      ++ind;
    }
    else {
      printf("Num duplicato – inserito nella lista.\n");
      lista=inserisci(lista,num);
    }
  }
```

4. stampi a video (utilizzando una specifica funzione) i valori contenuti nella lista concatenata (max 2 punti)

```c
void visualizza(struct elemento *p);
```

```c
  visualizza(lista); //check if vuota!
```

```c
void visualizza(struct elemento *p) {
    while (p != NULL) {
      printf("%d\n",p->numero);
      p = p->next;
    }
}
```

5. verifichi se la lista concatenata contenga anch'essa numeri duplicati al suo interno (max 6 punti)

```c
int check_duplicati_lista(struct elemento *p);
```

```c
  if(check_duplicati_lista(lista)==1)
  printf("La lista contiene duplicati\n");
```

```c
int check_duplicati_lista(struct elemento *p) {
  int check=0;
  struct elemento *p1=p,*p2;

  while (p1 != NULL) {
    p2=p1->next;
    while (p2 != NULL) {
      if(p1->numero==p2->numero) {
        check=1;
        return check;
      }
      p2=p2->next;
    }
    p1=p1->next;
  }
  return check;
}
```