Programmazione I (Tucci/Distasi)

PR1 MT/RD 15/01/2021

Modello: 1

Cognome: _____

Nome: _____

Matricola: _____

Email: _____

Regole del gioco: Compilare i dati personali prima d'incominciare. Alla fine della prova, inviare un singolo file pdf (non immagini separate, non un link web) all'indirizzo email

prog1unisa.aula1@gmail.com

Stiamo progettando il firmware di un registratore di cassa. Consideriamo la seguente struttura dati che definisce un oggetto di tipo `Acquisto`.

```
typedef struct acquisto
{
  char descrizione[30];
  double quantita;
  double prezzo_unitario;
} Acquisto;
```

Consideriamo poi un file di testo: la prima riga contiene il numero `n` di oggetti `Acquisto` presenti nel resto del file. Le righe successive contengono i dati relativi agli `n` acquisti, sempre un dato per riga. Per esempio, un file con il seguente contenuto:

```
2
Matita
10
1.10
Pomodori_Sorrento
3.5
2.50
```

specifica 2 acquisti: 10 matite (€1.10 l'una) e 3.5 kg di pomodori (€2.50 al kg).

1. Realizzare una funzione

   ```
   Acquisto *leggi_acquisti(FILE *fin, int *n_acquisti)
   ```

   che riceve come parametro un file di testo `fin` già aperto per la lettura e restituisce un array allocato dinamicamente contenente i dati di tipo `Acquisto` letti dal file. Il parametro output `n_acquisti` viene usato dalla funzione per comunicare il numero di oggetti letti, ossia la dimensione dell'array creato. Per semplicità, assumiamo che le descrizioni non contengano mai spazi.

2. Realizzare una funzione

   ```
   double spesa_totale(Acquisto scontrino[], int n)
   ```

   che riceve come parametro un array `scontrino[]` di oggetti `Accquisto` con la sua taglia `n`, e restituisce l'importo totale degli acquisti, ovvero la somma di tutti i subtotali ottenuti moltiplicando i prezzi unitari per le rispettive quantità.

   Facoltativo: definire una funzione `subtotale(Acquisto *articolo)` per risolvere il sottoproblema del subtotale relativo all'acquisto di un solo articolo; usarla in `spesa_totale()`. Notare che in questo caso, la struttura `articolo` è passata per riferimento.

Risposte per il modello 1

Stiamo progettando il firmware di un registratore di cassa. Consideriamo la seguente struttura dati che definisce un oggetto di tipo `Acquisto`.

```
typedef struct acquisto
{
  char descrizione[30];
  double quantita;
  double prezzo_unitario;
} Acquisto;
```

Consideriamo poi un file di testo: la prima riga contiene il numero `n` di oggetti `Acquisto` presenti nel resto del file. Le righe successive contengono i dati relativi agli `n` acquisti, sempre un dato per riga. Per esempio, un file con il seguente contenuto:

```
2
Matita
10
1.10
Pomodori_Sorrento
3.5
2.50
```

specifica 2 acquisti: 10 matite (€1.10 l'una) e 3.5 kg di pomodori (€2.50 al kg).

1. Realizzare una funzione

   ```
   Acquisto *leggi_acquisti(FILE *fin, int *n_acquisti)
   ```

   che riceve come parametro un file di testo `fin` già aperto per la lettura e restituisce un array allocato dinamicamente contenente i dati di tipo `Acquisto` letti dal file. Il parametro output `n_acquisti` viene usato dalla funzione per comunicare il numero di oggetti letti, ossia la dimensione dell'array creato. Per semplicità, assumiamo che le descrizioni non contengano mai spazi.

2. Realizzare una funzione

   ```
   double spesa_totale(Acquisto scontrino[], int n)
   ```

   che riceve come parametro un array `scontrino[]` di oggetti `Accquisto` con la sua taglia `n`, e restituisce l'importo totale degli acquisti, ovvero la somma di tutti i subtotali ottenuti moltiplicando i prezzi unitari per le rispettive quantità.

   Facoltativo: definire una funzione `subtotale(Acquisto *articolo)` per risolvere il sottoproblema del subtotale relativo all'acquisto di un solo articolo; usarla in `spesa_totale()`. Notare che in questo caso, la struttura `articolo` è passata per riferimento.

   Risposta
   La pagina seguente mostra una possibile soluzione.

```c
/* xmalloc.c */
#include <stdlib.h>
#include <stdio.h>

void *xmalloc(size_t nbytes)    // malloc() con controllo errore
{
  void *result;

  if ((result = malloc(nbytes)) == NULL)
    {
      fprintf(stderr, "malloc(%lu) failed. Exiting.\n", nbytes);
      exit(-1);
    }
  return result;
}

/* leggi_acquisti.c */
#include <stdio.h>
#include "acquisto.h"
#include "xmalloc.h"

Acquisto *leggi_acquisti(FILE *fin, int *n_acquisti)
{
  int n;                        // numero acquisti nel file
  int i;
  Acquisto *result;

  fscanf(fin, "%d", &n);
  result = xmalloc(sizeof(Acquisto) * n);
  for (i = 0; i < n; i++)
    {
      fscanf(fin, "%s", result[i].descrizione);
      fscanf(fin, "%lf", &result[i].quantita);
      fscanf(fin, "%lf", &result[i].prezzo_unitario);
    }
  *n_acquisti = n;
  return result;
}

/* spesa_totale.c */
#include "acquisto.h"

double subtotale(Acquisto * p)
{
  return p->quantita * p->prezzo_unitario;
}

double spesa_totale(Acquisto scontrino[], int n)
{
  int i;
  double tot;

  tot = 0.0;
  for (i = 0; i < n; i++)
    {
      tot += subtotale(&scontrino[i]);
    }
  return tot;
}
```

Programmazione I (Tucci/Distasi)

PR1 MT/RD 15/01/2021

Modello: 2

Cognome: _____

Nome: _____

Matricola: _____

Email: _____

Regole del gioco: Compilare i dati personali prima d'incominciare. Alla fine della prova, inviare un singolo file pdf (non immagini separate, non un link web) all'indirizzo email

prog1unisa.aula2@gmail.com

Stiamo progettando il firmware di un registratore di cassa. Consideriamo la seguente struttura dati che definisce un oggetto di tipo `Acquisto`.

```
typedef struct acquisto
{
  char descrizione[30];
  double quantita;
  double prezzo_unitario;
} Acquisto;
```

Consideriamo poi un file di testo: la prima riga contiene il numero `n` di oggetti `Acquisto` presenti nel resto del file. Le righe successive contengono i dati relativi agli `n` acquisti, sempre un dato per riga. Per esempio, un file con il seguente contenuto:

```
2
Matita
10
1.10
Pomodori_Sorrento
3.5
2.50
```

specifica 2 acquisti: 10 matite (€1.10 l'una) e 3.5 kg di pomodori (€2.50 al kg).

1. Realizzare una funzione

   ```
   Acquisto *leggi_acquisti(FILE *fin, int *n_acquisti)
   ```

   che riceve come parametro un file di testo `fin` già aperto per la lettura e restituisce un array allocato dinamicamente contenente i dati di tipo `Acquisto` letti dal file. Il parametro output `n_acquisti` viene usato dalla funzione per comunicare il numero di oggetti letti, ossia la dimensione dell'array creato. Per semplicità, assumiamo che le descrizioni non contengano mai spazi.

2. Realizzare una funzione

   ```
   double spesa_totale(Acquisto scontrino[], int n)
   ```

   che riceve come parametro un array `scontrino[]` di oggetti `Accquisto` con la sua taglia `n`, e restituisce l'importo totale degli acquisti, ovvero la somma di tutti i subtotali ottenuti moltiplicando i prezzi unitari per le rispettive quantità.

   Facoltativo: definire una funzione `subtotale(Acquisto *articolo)` per risolvere il sottoproblema del subtotale relativo all'acquisto di un solo articolo; usarla in `spesa_totale()`. Notare che in questo caso, la struttura `articolo` è passata per riferimento.

Risposte per il modello 2

Stiamo progettando il firmware di un registratore di cassa. Consideriamo la seguente struttura dati che definisce un oggetto di tipo `Acquisto`.

```
typedef struct acquisto
{
  char descrizione[30];
  double quantita;
  double prezzo_unitario;
} Acquisto;
```

Consideriamo poi un file di testo: la prima riga contiene il numero `n` di oggetti `Acquisto` presenti nel resto del file. Le righe successive contengono i dati relativi agli `n` acquisti, sempre un dato per riga. Per esempio, un file con il seguente contenuto:

```
2
Matita
10
1.10
Pomodori_Sorrento
3.5
2.50
```

specifica 2 acquisti: 10 matite (€1.10 l'una) e 3.5 kg di pomodori (€2.50 al kg).

1. Realizzare una funzione

   ```
   Acquisto *leggi_acquisti(FILE *fin, int *n_acquisti)
   ```

   che riceve come parametro un file di testo `fin` già aperto per la lettura e restituisce un array allocato dinamicamente contenente i dati di tipo `Acquisto` letti dal file. Il parametro output `n_acquisti` viene usato dalla funzione per comunicare il numero di oggetti letti, ossia la dimensione dell'array creato. Per semplicità, assumiamo che le descrizioni non contengano mai spazi.

2. Realizzare una funzione

   ```
   double spesa_totale(Acquisto scontrino[], int n)
   ```

   che riceve come parametro un array `scontrino[]` di oggetti `Accquisto` con la sua taglia `n`, e restituisce l'importo totale degli acquisti, ovvero la somma di tutti i subtotali ottenuti moltiplicando i prezzi unitari per le rispettive quantità.

   Facoltativo: definire una funzione `subtotale(Acquisto *articolo)` per risolvere il sottoproblema del subtotale relativo all'acquisto di un solo articolo; usarla in `spesa_totale()`. Notare che in questo caso, la struttura `articolo` è passata per riferimento.

   Risposta
   La pagina seguente mostra una possibile soluzione.

```c
/* xmalloc.c */
#include <stdlib.h>
#include <stdio.h>

void *xmalloc(size_t nbytes)    // malloc() con controllo errore
{
  void *result;

  if ((result = malloc(nbytes)) == NULL)
    {
      fprintf(stderr, "malloc(%lu) failed. Exiting.\n", nbytes);
      exit(-1);
    }
  return result;
}

/* leggi_acquisti.c */
#include <stdio.h>
#include "acquisto.h"
#include "xmalloc.h"

Acquisto *leggi_acquisti(FILE *fin, int *n_acquisti)
{
  int n;                        // numero acquisti nel file
  int i;
  Acquisto *result;

  fscanf(fin, "%d", &n);
  result = xmalloc(sizeof(Acquisto) * n);
  for (i = 0; i < n; i++)
    {
      fscanf(fin, "%s", result[i].descrizione);
      fscanf(fin, "%lf", &result[i].quantita);
      fscanf(fin, "%lf", &result[i].prezzo_unitario);
    }
  *n_acquisti = n;
  return result;
}

/* spesa_totale.c */
#include "acquisto.h"

double subtotale(Acquisto * p)
{
  return p->quantita * p->prezzo_unitario;
}

double spesa_totale(Acquisto scontrino[], int n)
{
  int i;
  double tot;

  tot = 0.0;
  for (i = 0; i < n; i++)
    {
      tot += subtotale(&scontrino[i]);
    }
  return tot;
}
```