Programmazione I (Tucci/Distasi)

PR1 MT/RD 10/09/2021-PRES

Modello: 1

Cognome: _____

Nome: _____

Matricola: _____

Email: _____

Regole del gioco: Compilare i dati personali prima d'incominciare. Una volta iniziata la prova, non è consentito lasciare l'aula. Usare questi stessi fogli (compreso il retro, dove necessario) per rispondere. *Buon lavoro!*

1. Consideriamo la seguente struttura dati, che rappresenta una vendita. L'importo totale di una vendita è il prodotto del prezzo unitario per la quantità.

```
typedef struct
{
  char description[50];         // what's being sold?
  double price1;                // unit price (for 1 unit)
  double quantity;
} Sale;
```

   - Scrivere una funzione C

     `int sale_cmp(Sale *s1, Sale *s2)`

     che riceve come parametri gli indirizzi di due vendite e segnala quale delle due corrisponde all'importo totale più grande, restituendo: -1 se è la prima che ha il totale più grande; 1 se invece è la seconda; 0 se i due importi sono equivalenti.

   - Scrivere una funzione C

     `Sale * sale_larger(Sale *A, int n, Sale *small, int *newsize)`

     che riceve come parametri un array `A[]` di `n` vendite e un puntatore `small` a una vendita. La funzione restituisce un array allocato dinamicamente contenente copie di tutte le vendite in `A[]` che hanno un importo totale maggiore rispetto alla vendita puntata da `small`. Il parametro output `newsize` sarà impostato alla dimensione dell'array risultante. Nel caso nessuna vendita in `A[]` soddisfi la condizione, `sale_larger()` restituirà `NULL` e imposterà `newsize` a zero.

2. Scrivere una funzione C

```
int sale_save_larger(FILE *savefile, Sale *A, int n, Sale *small)
```

che riceve come parametri un file di testo già aperto, un array di vendite `A[]` con la sua dimensione `n`, e un puntatore a vendita `small`. La funzione scrive nel file una riga per ogni vendita in `A[]` che abbia un importo totale maggiore di quello di `small`. La riga relativa a una vendita contiene la sua descrizione, la quantità, il prezzo unitario e l'importo totale. I dati sono separati da una virgola e uno spazio. Ecco un possibile esempio di output:

```
Matite da disegno, 0.5, 10, 5.0
Peperoni, 1.3, 6.5, 8.45
```

La funzione restituisce il numero di righe scritte nel file.

Risposte per il modello 1

1. Consideriamo la seguente struttura dati, che rappresenta una vendita. L'importo totale di una vendita è il prodotto del prezzo unitario per la quantità.

```
typedef struct
{
  char description[50];         // what's being sold?
  double price1;                // unit price (for 1 unit)
  double quantity;
} Sale;
```

   - Scrivere una funzione C

     ```
     int sale_cmp(Sale *s1, Sale *s2)
     ```

     che riceve come parametri gli indirizzi di due vendite e segnala quale delle due corrisponde all'importo totale più grande, restituendo: -1 se è la prima che ha il totale più grande; 1 se invece è la seconda; 0 se i due importi sono equivalenti.

     Risposta
     Ecco una possibile soluzione.

```
#include <string.h>
#include "sale.h"

double total_amount(Sale * s)
{
  return s->price1 * s->quantity;
}

int sale_cmp(Sale * s1, Sale * s2)
{
  double diff;

  diff = total_amount(s2) - total_amount(s1);
  if (diff > 0)                 // s2 larger
    return 1;
  else if (diff < 0)            // s1 larger
    return -1;
  else
    return 0;                   // equal
}

Sale *sale_cpy(Sale * dest, Sale * src)
{
  strcpy(dest->description, src->description);
  dest->price1 = src->price1;
  dest->quantity = src->quantity;
  return dest;
}
```

- Scrivere una funzione C

  ```
  Sale * sale_larger(Sale *A, int n, Sale *small, int *newsize)
  ```

  che riceve come parametri un array `A[]` di `n` vendite e un puntatore `small` a una vendita. La funzione restituisce un array allocato dinamicamente contenente copie di tutte le vendite in `A[]` che hanno un importo totale maggiore rispetto alla vendita puntata da `small`. Il parametro output `newsize` sarà impostato alla dimensione dell'array risultante. Nel caso nessuna vendita in `A[]` soddisfi la condizione, `sale_larger()` restituirà `NULL` e imposterà `newsize` a zero.

Risposta
Ecco una possibile soluzione.

```
#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;
}

#include "sale.h"
#include "prototypes.h"

Sale *sale_larger(Sale * A, int n, Sale * small, int *newsize)
{
  int i, j, count = 0;
  Sale *result;

  for (i = 0; i < n; i++)       // count "large" sales
    {
      if (sale_cmp(small, &A[i]) == 1)
        {
          count++;
        }
    }
  *newsize = count;             // set output parameter
  if (count == 0)               // no "large" sale to be copied
    {
      return NULL;
    }
  // else, allocate result array and copy large sales
  result = xmalloc(sizeof(Sale) * count);
  for (i = 0, j = 0; i < n; i++)
    {
      if (sale_cmp(small, &A[i]) == 1)
        {
          sale_cpy(&result[j], &A[i]);
          j++;
        }
    }
  return result;
}
```

2. Scrivere una funzione C

```
int sale_save_larger(FILE *savefile, Sale *A, int n, Sale *small)
```

che riceve come parametri un file di testo già aperto, un array di vendite `A[]` con la sua dimensione `n`, e un puntatore a vendita `small`. La funzione scrive nel file una riga per ogni vendita in `A[]` che abbia un importo totale maggiore di quello di `small`. La riga relativa a una vendita contiene la sua descrizione, la quantità, il prezzo unitario e l'importo totale. I dati sono separati da una virgola e uno spazio. Ecco un possibile esempio di output:

```
Matite da disegno, 0.5, 10, 5.0
Peperoni, 1.3, 6.5, 8.45
```

La funzione restituisce il numero di righe scritte nel file.

Risposta
Ecco una possibile soluzione.

```
#include <stdio.h>
#include <stdlib.h>
#include "sale.h"
#include "prototypes.h"

int sale_save_larger(FILE * savefile, Sale * A, int n, Sale * small)
{
  Sale *saveus;
  int i, count;

  saveus = sale_larger(A, n, small, &count);        // build array to be saved
  for (i = 0; i < count; i++)
    {
      fprintf(savefile, "%s, %lf, %lf, %lf\n",
              saveus[i].description,
              saveus[i].price1,
              saveus[i].quantity,
              total_amount(&saveus[i]));
    }
  free(saveus);                 // not needed anymore
  return count;
}
```