Programmazione I (Tucci/Distasi)

PR1 MT/RD 03/02/2021

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`

Buon lavoro!

1. Scrivere una funzione

   ```
   int * elementi_minori (int A[ ], int n, int val, int *newsize)
   ```

   che riceve un vettore `A` di `n` interi e restituisce un array d'interi allocato dinamicamente e contenente tutti gli elementi di `A` che hanno un valore inferiore a `val`. Il parametro output `newsize` conterrà il numero di elementi del nuovo array. Se l'array da restituire risultasse vuoto, `newsize` sarà impostato di conseguenza e la funzione restituirà `NULL`. Altrimenti, l'array restituito dovrà essere allocato col numero strettamente necessario di elementi.

2. Scrivere una funzione

   ```
   void salva_minori (FILE *fout, int A[ ], int n, int val)
   ```

   che, ricevendo un vettore `A` di `n` interi, scrive in un file binario già aperto in precedenza il valore di `val`, il numero degli elementi di `A` che hanno valore inferiore a `val`, poi tutti i valori stessi.

   Per esempio, se `A = {18, -14, 21, 36, 63, -1}` e `val = 36`, il file puntato da `fout` conterrà i seguenti valori interi: `36 4 18 -14 21 -1`. Se invece `val = -100`, il file conterrà i seguenti valori interi: `-100 0`.

   Potrebbe essere conveniente usare la funzione scritta all'esercizio precedente.

Risposte per il modello 1

1. Scrivere una funzione

   ```
   int * elementi_minori (int A[ ], int n, int val, int *newsize)
   ```

   che riceve un vettore `A` di `n` interi e restituisce un array d'interi allocato dinamicamente e contenente tutti gli elementi di `A` che hanno un valore inferiore a `val`. Il parametro output `newsize` conterrà il numero di elementi del nuovo array. Se l'array da restituire risultasse vuoto, `newsize` sarà impostato di conseguenza e la funzione restituirà `NULL`. Altrimenti, l'array restituito dovrà essere allocato col numero strettamente necessario di elementi.

   Risposta
   Ecco una possibile soluzione.

```
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;
}

int *elementi_minori(int A[], int n, int val, int *newsize)
{
  int i, j, n_minori, *new_array;

  n_minori = 0;
  for (i = 0; i < n; i++)       // contiamo elementi minori di val
    {
      if (A[i] < val)
        {
          n_minori++;
        }
    }
  *newsize = n_minori;
  if (n_minori == 0)            // nessun elemento trovato
    {
      return NULL;
    }
  new_array = xmalloc(n_minori * sizeof(int));
  j = 0;                        // posizione in new_array
  for (i = 0; i < n; i++)
    {
      if (A[i] < val)           // copia elemento minore di val
        {
          new_array[j++] = A[i];
        }
    }
  return new_array;
}
```

2. Scrivere una funzione

   ```
   void salva_minori (FILE *fout, int A[ ], int n, int val)
   ```

   che, ricevendo un vettore `A` di `n` interi, scrive in un file binario già aperto in precedenza il valore di `val`, il numero degli elementi di `A` che hanno valore inferiore a `val`, poi tutti i valori stessi.

   Per esempio, se `A = {18, -14, 21, 36, 63, -1}` e `val = 36`, il file puntato da `fout` conterrà i seguenti valori interi: `36 4 18 -14 21 -1`. Se invece `val = -100`, il file conterrà i seguenti valori interi: `-100 0`.

   Potrebbe essere conveniente usare la funzione scritta all'esercizio precedente.

   Risposta Ecco una possibile soluzione.

```
// print an error msg when fwrite fails
void error_fwrite(int a, int b)
{
  fprintf(stderr, "fwrite() failed in salva_minori() size=%d, nobj=%d\n", a, b);
  exit(-2);
}

void salva_minori(FILE * fout, int A[], int n, int val)
{
  int *minori, size;
  size_t write_status;

  minori = elementi_minori(A, n, val, &size);
  write_status = fwrite(&val, sizeof(int), 1, fout);
  if (write_status != 1)
    {
      error_fwrite(sizeof(int), 1);
    }
  write_status = fwrite(&size, sizeof(int), 1, fout);
  if (write_status != 1)
    {
      error_fwrite(sizeof(int), 1);
    }
  if (minori == NULL)           // array vuoto
    {
      return;
    }
  write_status = fwrite(minori, sizeof(int), size, fout);
  if (write_status != size)
    {
      error_fwrite(sizeof(int), size);
    }
  fclose(fout);
}
```