Laboratorio 1 ◇ algoritmi e array

17-10-2019

Esercizio 15.1: Fattori

Scrivere un programma che acquisisce due numeri interi relativi e visualizza 1 se uno è un divisore dell'altro o viceversa, 0 altrimenti. Dopo il valore visualizzato, mettere un `'a-capo'`.

Ingresso/Uscita:
input: `due numeri interi`
output: `un intero (seguito da un carattere 'a-capo')`

Alcuni casi di test per il collaudo:

input: `5542 18`
output: `0`

input: `5542 17`
output: `1`

input: `13 1950`
output: `1`

```c
#include <stdio.h>

int main(int argc, char * argv[])
{
    int n1, n2;
    int ris;

    scanf("%d", &n1);
    scanf("%d", &n2);

    if(n1 && n2)
        if(n1 % n2 == 0 || n2 % n1 == 0)
            ris = 1;
        else
            ris = 0;
    else
        ris = 0;

    printf("%d\n", ris);
    return 0;
}
```

Esercizio 15.2: Padding

Si vuole rappresentare a video un valore naturale `num` utilizzando un numero a scelta di cifre `k` inserendo 0 nelle posizioni più significative, fino a raggiungere la dimensione desiderata. Per esempio, volendo rappresentare `842` su `5` cifre, si ottiene `00842`.
Scrivere un programma che acquisisce due valori interi entrambi strettamente positivi (e finché non è così richiede il valore che non rispetta il vincolo) num e `k`, quindi rappresenta `num` su `k` cifre. Se `k` e' minore del numero di cifre presenti in `num`, il programma visualizza il valore `num` come è. Dopo il valore visualizzato, mettere un `'a-capo'`.

Ingresso/Uscita:
input: `due numeri interi (da verificare)`

output: `un intero (seguito da un carattere 'a-capo')`

Alcuni casi di test per il collaudo:

input: `11304 9`
output: `000011304`

input: `-4 9000 -5 -2 2`
output: `9000`

input: `1 1`
output: `1`

```c
#include <stdio.h>
#define BASE 10

int main(int argc, char * argv[])
{
    int num, k;
    int val, i, ncifre;

    do
        scanf("%d", &num);
    while(num <= 0);

    do
        scanf("%d", &k);
    while(k <= 0);

    val = num;
    i = 0;
    while(val > 0){
        val = val/BASE;
        i++;
    }
    for(; k > i; i--)
        printf("0");
    printf("%d\n", num);

    return 0;
}
```

Esercizio 15.3: Super Mario

Nella preistoria dei videogiochi in Super Mario della Nintendo, Mario deve saltare da una piramide di blocchi a quella adiacente. Proviamo a ricreare le stesse piramidi in C, in testo, utilizzando il carattere cancelletto (#) come blocco, come riportato di seguito. In realtà il carattere # è più alto che largo, quindi le piramidi saranno un po' più alte.

```
   #  #
  ##  ##
 ###  ###
####  ####
```

Notate che lo spazio tra le due piramidi è sempre costituito da **2** spazi, indipendentemente dall'altezza delle piramidi. Inoltre, alla fine delle piramidi **non ci devono essere spazi**. L'utente inserisce

l'altezza delle piramidi, che deve essere un valore strettamente positivo e non superiore a 16. In caso l'utente inserisca un valore che non rispetta questi vincoli, la richiesta viene ripetuta.

Ingresso/Uscita:
input: `un numero intero (da verificare)`
output: `una sequenza di caratteri`

```c
#include <stdio.h>
#define BLOCK '#'
#define MIN 1
#define MAX 16

int main(void)
{
    int h;    /* altezza */
    int i, j;

    do {
        scanf("%d", &h);
    } while(h < MIN || h > MAX);

    i = 1;
    while(i <= h)
    {
        /* spazi */
        j = 0;
        while(j < h - i){
            printf(" ");
            j++;
        }
        j = 0;
        while(j < i){
            printf("%c", BLOCK);
            j++;
        }
        /* separatore */
        printf("  ");
        j = 0;
        while(j < i){
            printf("%c", BLOCK);
            j++;
        }
        j = 0;
        while(j < h - i){
            printf(" ");
            j++;
        }
        printf("\n");
        i++;
    }

    return 0;
}
```

Esercizio 15.4: Scorrimento a destra – rightshift

Scrivere un programma che acquisita una sequenza di 10 caratteri, modifica la sequenza in modo tale che la sequenza finale sia quella iniziale, fatta scorrere a destra di una posizione, con l'ultimo carattere riportato in testa. Se per esempio la sequenza iniziale è `attraverso`, la sequenza finale sarà `oattravers`. Una volta modificata la sequenza memorizzata, visualizzarla e farla seguire da un carattere `'a-capo'`.

Ingresso/Uscita:
input: `10 caratteri`
output: `10 caratteri (seguiti da un carattere 'a-capo')`

Alcuni casi di test per il collaudo:

input:	`attraverso`
output:	`oattravers`
input:	`capitolino`
output:	`ocapitolin`
input:	`trampolini`
output:	`itrampolin`

```c
#include <stdio.h>

#define N 10

int main(int argc, char * argv[])
{
  char seq[N];
  int tmp, i;

  for(i = 0; i < N; i++)
    scanf("%c", &seq[i]);

  tmp = seq[N-1];
  for(i = N-1; i > 0; i--)
    seq[i] = seq[i-1];
  seq[0] = tmp;

  for(i = 0; i < N; i++)
    printf("%c", seq[i]);
  printf("\n");
  return 0;
}
```

Esercizio 15.5: Troncabile primo a destra

Scrivere un programma che acquisisce un valore intero strettamente positivo, e finché non è tale lo richiede. Il programma analizza il valore intero e visualizza 1 nel caso sia un troncabile primo a destra, 0 altrimenti. Un numero si dice troncabile primo a destra se il numero stesso e tutti i numeri che si ottengono eliminando una alla volta la cifra meno significativa del numero analizzato al passo precedente, sono numeri primi. Per esempio, se il numero iniziale è 719, i numeri che si ottengono "eliminando una alla volta la cifra meno significativa del numero analizzato al passo precedente .." sono 71 e 7. Dopo il valore visualizzato, mettere un `'a-capo'`.

Ingresso/Uscita:
input: `un intero (da verificare)`

output: un intero (seguito da un carattere 'a-capo')

Alcuni casi di test per il collaudo:

input: 719
output: 1

input: 473
output: 0

input: -42 -18 311111
output: 0

input: 3137
output: 1

```c
#include<stdio.h>

#define BASE 10

int main (int argc, char *argv[])
{
  int n;
  int i, isprime;

  do
    scanf("%d", &n);
  while(n <= 0);

  isprime = 1;

  while (n > 1 && isprime)
  {
    /* e' un numero primo ? */
    if (n % 2 == 0 && n != 2)
      isprime = 0;
    else {
      half = n/2;
      for(i = 3; i < half && isprime; i = i+2)
        if(n % i == 0)
          isprime = 0;
    }
    n = n / BASE;
  }

  printf("%d\n", isprime);
  return 0;
}
```