Laboratorio di Calcolo (GdA, 2018)

Raccolta di codici R

1 Programmi e funzioni C in ordine (circa) cronologico

1.1 Primi passi

```
// -------------------------
// hello_world.c
#include <stdio.h>
int main() {
    printf("Hello World");

    return 0;
}

// -------------------------
// puts_2l.c
#include <stdio.h>
int main() {
    puts("Ciao a tutti");
    puts("Arrivederci");
    return 0;
}

// -------------------------
#include <stdio.h>
int main() {
    puts("Ciao a tutti");
    puts("Arrivederci");
    return 0;
}

// -------------------------
// printf_2l.c
#include <stdio.h>
int main() {
    printf("Ciao a tutti");
    printf("Arrivederci");
    return 0;
}

// -------------------------
// montalbano.c
#include <stdio.h>
int main() {
    printf("Sono Montalbano\n");
    printf("\"Montalbano\" sono");
    return 0;
}

// -------------------------
// montalbano_a.c
#include <stdio.h>
int main() {
```

```
    printf("Sono Montalbano\n");
    printf("'Montalbano' sono");
    return 0;
}

// ----------------------------
// GuardaMamma.c
#include <stdio.h>
int main() {
  int a, b, c;
  a = 2;
  b = 2;
  printf("Guarda mamma come mi diverto\n");
  c = a + b;
  printf("E sono pure bravo: %d + %d = %d\n", a, b, c);
  return 0;
}

// ----------------------------------------------------
// percento.c
#include <stdio.h>
int main() {
   printf("Ho raggiunto il %d%%\n", 75);
   return 0;
}

// -----------------------------------------------------
// radice_quadrata.c
#include <stdio.h>
#include <math.h>
int main() {
  float x = 0.5;
  float result = sqrt(x);
  printf("The square root of %.3f is %.3f\n", x, result);
  return 0;
}

// -----------------------------------------------------
// somma_n_interi.c
#include <stdio.h>
int main() {
  int i, s=0, n=10;

  for (i=1 ; i<=n ; i++) {
    s += i;
    printf("i = %d,  s = %d\n", i, s) ;
  }

  return 0;
}
```

1.2 lanci_spazio_ds_ossatura.c

```
#include <stdio.h>
#include <math.h>

const float h = 0.90, d=0.96;  // dati (m)
const float g = 9.80;          // m/s^2

const int n = 10 + 1;          // numero di punti di quantizzazione
                               // (n - 1) è il nr di intervallini temporali

int main() {

  float t=0.0, s=0.0;
  float x, y, v, ds, xOld, yOld;

  printf("h = %.2f m;  d = %.2f m\n", h, d);

  float tv = ........ ;              // tempo di volo
  float vx = ........ ;              // velocità orizzontale
  float dt = tv / (float) (n-1);     // ampiezza di dt

  printf("tv = %f s  vx = %f m/s;  dt = %.2e s\n", tv, vx, dt);


  for (int i=0; i < n; i++) {

    // coordinate al tempo t
    x = .......  ;
    y = .......  ;

    // dal secondo passo calcoliamo gli incrementi di s
    // da x, y, xOld e yOld (vedi in fondo)
    if (i>0) {
      ds = ........  ;
      s = s + ds;
    }
    printf("i = %d,  t = %f s \n",  i, t);
    printf("      x = %f m,  y = %f m, ds = %f m, s = %f m\n", x, y, ds, s);

    // memorizziamo le coordinate di questo passo
    xOld = x;
    yOld = y;

    // nuovo tempo
    t = t + dt;
  }

  return 0;
}
```

1.3 lanci_spazio_vdt_ossatura.c

```
#include <stdio.h>
#include <math.h>

const float h = 0.90, d=0.96;  // dati (m)
const float g = 9.80;          // m/s^2
```

```
const int n = 10;              // nr di intervallini temporali
                               // Attenzione! ha un significato diverso di
                               // quello che aveva in lanci_spazio_ds_ossatura.c

int main() {

  float tm, ds, s=0.0;
  float vym, vm;          // velocità medie

  printf("h = %.2f m;  d = %.2f m\n", h, d);

  float tv = ..... ;        // tempo di volo
  float vx = ..... ;              // velocità orizzontale
  float dt = tv / (float) n;

  printf("tv = %f s  vx = %f m/s;  dt = %.2e s\n", tv, vx, dt);

  for (int i=0; i < n; i++) {

    tm = ( (float) i + 0.5) * dt;    // tempo medio dell'i-mo intervallino
    vym = ..... ;                    // vy al tempo tm
    vm  = ..... ;                    // modulo di v al tempo tm
    ds  = ..... ;                    // incremento spaziale in dt
    s = s + ds;

    printf("i = %d,  vym = %f m/s,  vm = %f m/s\n", i, vym, vm);
    printf("     ds = %f m, s = %f m\n", ds, s);
  }

  return 0;
}
```

1.4 area_triangolo.c

```
#include <stdio.h>
#include <math.h>

float triangolo(float x, float x1, float x2, float xc, float h, int dbg) {
  float m1, m2, y;

  if ( ( xc < x1 ) || ( xc > x2 ) ) {
    printf("xc (=%f) deve esse compreso fra x1 e x2\n", xc);
    return 0.;
  }

  if (dbg) printf("Bene, ora cominciamo a fare i conti\n");

  m1 = h / (xc-x1);
  m2 = h / (xc-x2);
  if (dbg) printf("Pendenze:  m1 = %f;  m2 = %f\n", m1, m2);

  if( (x < x1) || (x > x2) ) {
    return(0);
  }

  if (x <= xc) {
    y = m1 * (x - x1);
```

```
  } else {
    y = h + m2 * (x - xc);
  }

 if (dbg) printf("y = %f \n", y);

 return(y);
}

int main() {
  float x1=3, x2=10, xc=4.5, h = 5;
  float x, y, dx, area;
  int N;

  printf("Inserisci il numero di intervalli N\n");
  scanf("%d", &N);
  printf("ok, N=%d\n\n", N);

  area = (x2-x1) * h / 2.;
  printf("Area (formula) : %f\n", area);

  dx = (x2-x1) / (float) N;
  printf("x1=%f, x2=%f, xc=%f;  N=%d, dx=%f\n", x1, x2, xc, N, dx);

  area = 0.;
  for( int i = 0; i < (N+1); i++) {
    x = x1 + i * dx;
    y = triangolo(x, x1, x2, xc, h, 0);
    // printf("x = %f,  y = %f\n", x, y);
    area += y * dx;
  }

  printf("\nArea calcolata numericamente: %f\n", area);

  return 0;
}
```

1.5 cerchio_ossatura.c

```
#include <stdio.h>
#include <math.h>

double semicerchio(double x, double R) {
  float y;

  if( (x < -2.*R) || (x > 2. * R) ) {
    return(0);
  }

  y = ..... ;

  return(y);

}

int main() {
  double R=1;
  double x, y, dx, area;
```

```
  int N;
  double pi = acos(-1.);   // il C non ha pi greco predefinita

  printf("Inserisci il numero di intervalli N\n");
  scanf("%d", &N);
  printf("ok, N=%d\n\n", N);

  area = pi*R*R / 2.;
  printf("Area (formula) : %f\n", area);

  dx = 2. * R / (double) N;
  printf("R=%f, N=%d, dx=%f\n", R, N, dx);

  area = 0.;
  for( int i = 0; i < (N+1); i++) {
    x = ....... ;
    y = ....... ;
    // printf("x = %f,  y = %f\n", x, y);   //scommentare per controllare
    area += y * dx;
  }

  printf("\nArea calcolata numericamente: %.10f\n", area);

  double piNum = 2*area/(R*R);
  printf("\nDa cui, stima 'numerica' di pi greco: %.12f\n", piNum);

  return 0;
}
```

1.6 quarto_cerchio_numeric.c

```
#include <stdio.h>
#include <math.h>

#define NPMAX 100001

float fun( float x ) {
  float y;
  if( x*x > 1 ) {
    return(0);
  } else {
    y = pow( 1 - x*x, 0.5);
    return(y);
  }
}

int main() {
  float x[NPMAX];
  float xc, dx, dy, A=0, l=0;
  int N;

  printf("Dai il nr di intervalli ( <= %d)\n", NPMAX-1);
  scanf("%d", &N);
  printf("N = %d\n", N);

  if ( N > (NPMAX-1)) {
     printf("Massimo valore consentito: %d\nRiprova!\n", NPMAX-1);
     return 0;
```

```
  }

  dx = 1. / (float) N;
  printf("dx = %f\n", dx);

  for(int i=0; i < (N+1); i++) {
    x[i] = dx * i;              // si capisce che è uno spreco di memoria
                                // in quanto la conoscenza simultanea di
                                // tutti gli x[i] non serve: meglio il trucco
                                // di memorizzare il vecchio valore ('xOld')
    if (i > 1) {
        xc = x[i] - dx/2.;
        A +=  dx * fun(xc);
        dy = fun(x[i]) - fun(x[i-1]);
        l += sqrt(dx*dx + dy*dy);
    }
  }

  printf("Area = %f (esatta %f)\n", A, acos(-1)/4.);
  printf("Arco = %f (esatto %f)\n", l, acos(-1)/2.);

  puts("\nStime di pi greco");

  printf("Dall'area: %f\n", 4 * A);
  printf("Dall'arco: %f\n", 2 * l);

  return 0;
}
```

1.7 quarto_cerchio_sampling.c

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int main() {
  float x, y, A;
  int N;

  printf("Dai il nr di punti\n");
  scanf("%d", &N);
  printf("N = %d\n", N);

  int nIN = 0;
  for (int i=0; i < (N); i++) {
    x = (float) rand() / (float) RAND_MAX;
    y = (float) rand() / (float) RAND_MAX;
    if( (x*x + y*y) <= 1. ) nIN += 1;
  }
  printf("nIN = %d \n", nIN);

  A = (float) nIN / N;

  printf("Area = %f (esatta %f)\n", A, acos(-1)/4.);
  puts("\nStima di pi greco");
  printf("Dall'area: %f\n", 4 * A);

  return 0;
```

```
}
```

1.8 quarto_cerchio_sampling_seed.c

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>      // per il seed

int main() {
  float x, y, A;
  int N;

  time_t t;
  unsigned RandomSeed;

  printf("Dai il nr di punti\n");
  scanf("%d", &N);
  printf("N = %d\n", N);

  //   RandomSeed = 12345;
  RandomSeed = (unsigned) time(&t);
  srand( RandomSeed );

  int nIN = 0;
  for (int i=0; i < (N); i++) {
    x = (float) rand() / (float) RAND_MAX;
    y = (float) rand() / (float) RAND_MAX;
    if( (x*x + y*y) <= 1. ) nIN += 1;
  }
  printf("nIN = %d \n", nIN);

  A = (float) nIN / N;

  printf("Area = %f (esatta %f)\n", A, acos(-1)/4.);
  puts("\nStima di pi greco");
  printf("Dall'area: %f\n", 4 * A);

  return 0;
}
```

1.9 intro_vettori.tex

```
//---------------------------------------------------------------------
// vettori_01.c
#include <stdio.h>

int main() {
  int v1[] = {1, 2, 3, 4, 5};
  int v2[] = {2, 4, 6, 8, 10};
  int v3[5];

  int sv3 = 0;
  int i;         /* Attenzione: gcc del laboratorio non accetta la
                    dichiarazione come primo argomento del for() !! */
```

```
  printf("\nValori iniziali\n");
  for (i=0; i<5; i++) {
    printf("i = %d;  v1, v2, v3: %d  %d  %d\n", i, v1[i], v2[i], v3[i]);
  }

  printf("\nOra inizializziamo v3[] e calcoliamo sv3\n");
  for (i=0; i<5; i++) {
    v3[i] = v1[i] + v2[i];
    printf("i = %d;  v1, v2, v3: %d  %d  %d\n", i, v1[i], v2[i], v3[i]);
    sv3 += v3[i];
  }

  printf("sv3 = %d\n", sv3);

  return 0;
}

//------------------------------------------------------------------------
// vettori_02.c
// Contenuto iniziale dei vettori?
// Attenzione! dipende da dove l'eseguibile risiede nella RAM!

#include <stdio.h>

int main() {
  int v1[] = {1, 2, 3, 4, 5};
  int v2[] = {2, 4, 6, 8, 10};
  int v3[5];

  int sv3 = 0;
  int i;         /* Attenzione: gcc del laboratorio non accetta la
                    dichiarazione come primo argomento del for() !! */

  puts("contenuto iniziale di v3");

  for (int i=0; i<5; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u;  float: %f; exp: %e\n",
           i, v3[i], v3[i], (float) v3[i], (float) v3[i]);
  }


  printf("\nOra inizializziamo v3[]\n");
  for (i=0; i<5; i++) {
    v3[i] = v1[i] + v2[i];
  }
  puts("contenuto finale di v3");

  for (i=0; i<5; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u;  float: %f; exp: %e\n",
           i, v3[i], v3[i], (float) v3[i], (float) v3[i]);
  }

  return 0;
}
//------------------------------------------------------------------------
// vettori_03.c
// Attenzione: in C si può anche sconfinare!!
//   Primo esempio
```

```
#include <stdio.h>

int main() {

  int v3[5];
  int i;            /* Attenzione: gcc del laboratorio non accetta la
                     dichiarazione come primo argomento del for() !! */

  puts("Ora proviamo a 'sconfinare'");

  for (i=0; i<20; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u;  float: %f; exp: %e\n",
           i, v3[i], v3[i], (float) v3[i], (float) v3[i]);
  }

  return 0;
}

//------------------------------------------------------------------------
// vettori_04.c
// Attenzione: in C si può anche sconfinare!!
// Secondo esempio: sconfinamento da un vettore all'altro!

#include <stdio.h>

int main() {

  int v3[10], v4[10];
  int i;

  puts("Contenuto iniziale dei due vettori");
  puts("(ci limitiamo a int e unsigned)");

  puts("v3[]");
  for (i=0; i<10; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v3[i], v3[i]);
  }
  puts("v4[]");
  for (i=0; i<10; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v4[i], v4[i]);
  }

  puts("\nE ora sconfiniamo da v3[]");
  for (i=0; i<20; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v3[i], v3[i]);
  }

  return 0;
}

//------------------------------------------------------------------------
// vettori_05.c
// Attenzione: in C si può anche sconfinare!!
// Altro esempio di sconfinamento

#include <stdio.h>

int main() {
```

```
  int v3[10], v4[10];
  int i;

  puts("Contenuto iniziale dei due vettori");
  puts("(ci limitiamo a int e unsigned)");

  puts("v3[]");
  for (i=0; i<10; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v3[i], v3[i]);
  }
  puts("v4[]");
  for (i=0; i<10; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v4[i], v4[i]);
  }

  puts("\nE ora sconfiniamo da v3[] SCRIVENDOCI degli 0'");
  for (i=0; i<20; i++) {
    v3[i] = 0;
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v3[i], v3[i]);
  }

  puts("\nE vediamo cosa è successo a v4[]'");
    puts("v4[]");
  for (i=0; i<10; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v4[i], v4[i]);
  }

  return 0;
}

//---------------------------------------------------------------------
// vettori_06.c
// Attenzione: in C si può anche sconfinare!!
//   Terzo esempio: sconfinamento ESAGERATO da un vettore: crash!!

#include <stdio.h>

int main() {

  int v3[10], v4[10];
  int i;

  puts("Contenuto iniziale dei due vettori");
  puts("(ci limitiamo a int e unsigned)");

  puts("v3[]");
  for (i=0; i<10; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v3[i], v3[i]);
  }
  puts("v4[]");
  for (i=0; i<10; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v4[i], v4[i]);
  }

  puts("\nE ora sconfiniamo DI MOLTO su v3[] SCRIVENDOCI degli 0'");
  for (i=0; i<100; i++) {
    v3[i] = 0;
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v3[i], v3[i]);
  }
```

```
  puts("\nE vediamo cosa è successo a v4[]");
    puts("v4[]");
  for (i=0; i<10; i++) {
    printf("i = %d:  int: %d ;  unsigned: %u\n", i, v4[i], v4[i]);
  }

  return 0;

/*

Messaggio di errore _di_esecuzione_

*** stack smashing detected ***: <unknown> terminated
Aborted (core dumped)

 => Morale: BISOGNA STARE ATTENTI!!
                                                          */
}
```

1.10 media_voti.c

```
#include <stdio.h>
#include <stdlib.h>    // per la random
#include <time.h>      // per il seed

#define NVOTI 25

float CalcolaMedia(int n, float valori[]) {
  int i;
  float sum = 0;
  for(i=0; i<n; i++){
    sum  += valori[i];
  }
  return( sum / n );
}

int main() {
  time_t t;               // vedi quarto_cerchio_sampling_seed.c
  unsigned RandomSeed;
  float voti[NVOTI];
  int i;

  // inizializziamo la random 'a caso'
  RandomSeed = (unsigned) time(&t);
  srand( RandomSeed );

  // generiamo numeri 'a caso' fra 18 e 30
  for (i=0; i<NVOTI; i++) {
    voti[i] = 18. + rand() % 13;
    printf(" %.0f\n", voti[i]);
  }

  float media = CalcolaMedia( NVOTI, voti );
  printf("voto medio %.1f\n", media);
  return 0;
}
```

1.11 esempio_define.c

```
#include <stdio.h>

#define NEGATIVO < 0
#define CICLA10VOLTE for(i=0; i<10; i++)

int main() {
  int i;
  for (i=-5; i<10; i++){
    printf(" %d", i);
    if(i NEGATIVO) printf(": negativo");
    printf("\n");
  }

  CICLA10VOLTE printf(" i = %d\n", i);

  return 0;
}
```

1.12 Variabili char. . . e caratteri

```
// -------------------------
// char_01.c
#include <stdio.h>
int main() {
  int i;
  char n;
  n = 0;      // una variabile char accetta un valore numerico
              // e lo tratta correttamente
  for (i=0; i<20; i++) {
    n++;
    printf(" %2d %2u %2x %2X %2o \n", n, n, n, n, n);
  }
  return 0;
}

// ------------------------------------------------------------
// char_02.c
// variabili char e differenza fra %d e %u (integer e unsigned)
// vediamo cosa succede al limite della 'capacità'
#include <stdio.h>
int main() {
  int i, ind;
  char n;
  n = 0;      // una variabile char accetta un valore numerico
                // e lo tratta correttamente

  ind = 0;    // portiamoci dietro un indice intero
  for (i=0; i<260; i++) {
    n++;
    ind++;
    printf("%3d: %3d %3u %4x %4X %5o \n", ind, n, n, n, n, n);
  }
  return 0;
}
```

```
// ------------------------------------------------------------
// char_03.c
// variabili char e differenza fra char e 'unsigned char'
//
#include <stdio.h>
int main() {
  int i;
  char n=0;
  unsigned char un=0;
  int ind = 0;

  for (i=0; i<260; i++) {
    n++;
    un++;
    ind++;
    printf("%3d: %3d %3u %3X\n", ind, n, un, un);
  }
  return 0;
}

// ------------------------------------------------------------
// char_04.c
// uso di 'char' per veri caratteri (con operazioni, più codice ASCII)
#include <stdio.h>
int main() {
  int i;
  char lm = 'a';
  char lM = 'A';

  printf("%c   %c\n", lm, lM);
  for (i=0; i<25; i++) {
    lm += 1;
    lM += 1;
    printf("%c   %c\n", lm, lM);
  }

  char c1 = 'A';
  char c2 = 'L';

  printf("\n%c - %c = %d\n", c2, c1, c2 - c1);
  printf("\n%c + %d = %c\n", 'M', 10, 'M' + 10);
  printf("\nCodice ASCII di A: %d\n", 'A');

  return 0;
}

// ------------------------------------------------------------
// char_05.c
/*
    Program to Print ASCII Value
   (https://www.programiz.com/c-programming/examples/ASCII-value-character)
 */
#include <stdio.h>
int main()
{
    char c;
    printf("Enter a character: ");
```

```
    // Reads character input from the user
    scanf("%c", &c);

    // %d displays the integer value of a character
    // %c displays the actual character
    printf("ASCII value of %c : %d\n", c, c);
    return 0;
}
```

1.13 Operazioni sui bit

```
// --------------------------
// attenzione_AND_OR.c
/*
    ATTENZIONE a non confondere  '&&' con '&' e '||' con '|'
*/
#include <stdio.h>
int main() {
  unsigned char due=2, tre=3, otto=8, dieci=10;

  printf(" due & tre : %d\n", due & tre );
  printf(" due && tre : %d\n", due && tre );

  printf(" due & otto : %d\n", due & otto );
  printf(" due && otto : %d\n", due && otto );

  printf(" due & dieci : %d\n", due & dieci );
  printf(" due && dieci : %d\n", due && dieci );

  printf(" due | tre : %d\n", due | tre );
  printf(" due || tre : %d\n", due || tre );

  printf(" due | otto : %d\n", due | otto );
  printf(" due || otto : %d\n", due || otto );

  printf(" due | dieci : %d\n", due | dieci );
  printf(" due || dieci : %d\n", due || dieci );

  return 0;
}
// --------------------------
// bits_ops.c
/*
    operazioni sui singoli bit
*/
#include <stdio.h>
int main() {
  int i;
  unsigned char a=0, b, c, d;

  printf("a = %u  [0x%X] \n", a, a);

  puts("\n Ci mettiamo 1");
  a = 1;
  printf("a = %u  [0x%X] \n", a, a);

  a = 1;
```

```
  puts("\n spostiamo il primo bit a sinistra 7 volte");
  for (i=0; i<7; i++) {
    a = a << 1;
    printf("a = %3u  [0x%2X] \n", a, a);
  }

  //-----------------------------------------------

  puts("\n Settiamo tutti i bit a 1");
  a = 0b11111111;
  printf("a = %3u  [0x%X] \n", a, a);

  puts("\n definiamo altre due variabili con 4 bit ON");
  b = 0b00001111;
  printf("b = %3u  [0x%2X] \n", b, b);
  c = 0b11110000;
  printf("c = %3u  [0x%2X] \n", c, c);

  puts("\nOra facciamo AND e OR fra le variabili");
  printf("b | c = %3u  [0x%2X] \n", b|c, b|c);
  printf("b & c = %3u  [0x%2X] \n", b&c, b&c);
  printf("a & b = %3u  [0x%2X] \n", a&b, a&b);
  printf("a & c = %3u  [0x%2X] \n", a&c, a&c);
  printf("a | b = %3u  [0x%2X] \n", a|b, a|b);
  printf("a | c = %3u  [0x%2X] \n", a|c, a|c);

  puts("\nFacciamo anche qualche XOR");
  printf("b ^ c = %3u  [0x%2X] \n", b^c, b^c);
  printf("a ^ b = %3u  [0x%2X] \n", a^b, a^b);
  printf("a ^ c = %3u  [0x%2X] \n", a^c, a^c);


  puts("\nNegazioni");
  printf("~a = %3u  [0x%2X] \n", (unsigned char) ~a, (unsigned char) ~a);
  printf("~b = %3u  [0x%2X] \n", (unsigned char) ~b, (unsigned char) ~b);
  printf("~c = %3u  [0x%2X] \n", (unsigned char) ~c, (unsigned char) ~c);
  printf("~(~a) = %3u  [0x%2X] \n",
    (unsigned char) ~(~a), (unsigned char) ~(~a));

  printf("~0b10000001 = %3u  [0x%2X] \n",
    (unsigned char) ~0b10000001, (unsigned char) ~0b10000001);

  puts("\nAltri esempi di shift");
  printf("b << 4; %3u  [0x%2X] \n",  (unsigned char)b<<4,  (unsigned char)b<<4);
  printf("c >> 4; %3u  [0x%2X] \n",  (unsigned char)c>>4,  (unsigned char)c>>4);
  d = b << 6;
  printf("d = (b << 6); %3u  [0x%2X] \n",  (unsigned char) d,  (unsigned char) d);
  printf("d >> 6; %3u  [0x%2X] \n",  (unsigned char) d>>6,  (unsigned char) d>>6);
  puts("\n(i bit persi... sono persi)");

  return 0;
}
```

1.14 Introduzione ai puntatori

```
// ------------------------------------------------------------------
// test_cambio_argomento.c
// vediamo cosa succede se si prova a cambiare un argomento dentro
// una funzione
#include <stdio.h>

void cambiaArgomento(int a) {
  printf("da cambiaArgomento(), valore iniziale: a = %d\n", a);
  a = 2*a;
  printf("da cambiaArgomento(), dopo modifica:   a = %d\n", a);
}

int main() {
  int a = 17;
  printf("da main(): a = %d\n", a);
  cambiaArgomento(a);
  printf("da main(): a = %d\n", a);

  return 0;
}

// ------------------------------------------------------------------
// test_cambio_argomento_puntatore.c
// vediamo cosa succede se si prova a cambiare un argomento dentro
// una funzione: uso di puntatori
#include <stdio.h>

void cambiaArgomento(int *a) {
  printf("da cambiaArgomento(), valore iniziale: a = %d\n", *a);
  *a = 2*(*a);
  printf("da cambiaArgomento(), dopo modifica:   a = %d\n", *a);
}

int main() {
  int a = 17;

  printf("da main(): a = %d\n", a);
  cambiaArgomento(&a);
  printf("da main(): a = %d\n", a);

  return 0;
}

// ------------------------------------------------------------------
// test_swap.c
// uso di puntatori per scambiare il contenuto di due variabili

#include <stdio.h>

void swap (int *a, int *b) {
  int tmp;
  tmp = *a;
  *a = *b;
  *b = tmp;
}

int main() {
```

```
  int a, b;

  printf("Dai due numeri interi: ");
  scanf("%d  %d", &a, &b);
  printf("Valori ricevuti %d %d  \n", a, b);
  swap (&a, &b);
  printf("Valori scambiati %d %d \n", a, b);

  return 0;
}

// ------------------------------------------------------------
// test_swap_puntatori.c
// uso di puntatori per scambiare il contenuto di due variabili

#include <stdio.h>

void swap (int *a, int *b) {
  int tmp;
  printf("\nDa swap(): ricevo %d e %d\n", *a, *b);
  printf("           (locazioni %p e %p)\n\n", a, b);
  tmp = *a;
  *a = *b;
  *b = tmp;
}

int main() {
  int a, b;

  printf("Dai due numeri interi: ");
  scanf("%d  %d", &a, &b);
  printf("Valori ricevuti %d %d  \n", a, b);
  printf("Saranno messi nelle locazioni %p e %p\n", &a, &b);
  swap (&a, &b);
  printf("Valori scambiati %d %d \n", a, b);

  return 0;
}

// ------------------------------------------------------------------
// test_swap_puntatori_a.c
// uso di puntatori per scambiare il contenuto di due variabili
// Saltando da una variabile all'altra
//  --  da non fare nei programmi seri !! ---

#include <stdio.h>

void swap (int *a) {
  int tmp;
  printf("\nDa swap(): ricevo %d\n", *a);
  printf("           (locazione %p)\n\n", a);
  printf(" locazione successiva %p\n\n", a+1);
  tmp = *a;
  *a = *(a + 1);
  *(a + 1) = tmp;
}

int main() {
  int a, b;
```

```
  printf("Dai due numeri interi: ");
  scanf("%d  %d", &a, &b);
  printf("Valori ricevuti %d %d  \n", a, b);
  printf("Saranno messi nelle locazioni %p e %p\n", &a, &b);
  swap (&a);
  printf("Valori scambiati %d %d \n", a, b);

  return 0;
}

// ------------------------------------------------------------------
// puntatori_0.c
/*
      puntatori e contenuto delle locazioni di memoria
*/

#include <stdio.h>

int main() {

  short int a=11, b=12, c=13, d=14, e=15;
  int A=21, B=22, C=23, D=24, E=25;
  float f1=3.1, f2=3.2, f3=3.3, f4=3.4, f5=3.5;
  double d1=31., d2=32., d3=33., d4=34., d5=35.;

  printf("a-e : %d, %d, %d, %d, %d\n", a,b,c,d,e);
  printf("A-E : %d, %d, %d, %d, %d\n",A,B,C,D,E);
  printf("d1-d2 : %.1f, %.1f, %.1f, %.1f, %.1f\n",f1,f2,f3,f4,f5);
  printf("d1-d2 : %.1f, %.1f, %.1f, %.1f, %.1f\n",d1,d2,d3,d4,d5);

  printf("Puntatori ad a-e:\n %p\n %p\n %p\n %p\n %p\n", &a,&b,&c,&d,&e);
  printf("Puntatori ad A-E:\n %p\n %p\n %p\n %p\n %p\n", &A,&B,&C,&D,&E);
  printf("Puntatori a f1-f5:\n %p\n %p\n %p\n %p\n %p\n", &f1,&f2,&f3,&f4,&f5);
  printf("Puntatori a d1-d5:\n %p\n %p\n %p\n %p\n %p\n", &d1,&d2,&d3,&d4,&d5);

  puts("\nvalore della variable a cui punta il puntatore");
  printf("a-e : %d, %d, %d, %d, %d\n", *(&a),*(&b),*(&c),*(&d),*(&e));
  puts("etc...");

  return 0;
}
// ------------------------------------------------------------------
// puntatori_1.c
/*
    Uso improprio di puntatori

    => Segmentation fault (core dumped)

*/


#include <stdio.h>

int main() {

  int *p1, *p2;
  *p1=51;
  *p2=52;
```

```
   printf("*p1: %d,   *p2: %d\n",*p1,*p2);

  return 0;
}

// ------------------------------------------------------------
// puntatori.c    (F. Safai Tehrani))
#include<stdio.h>

typedef unsigned char byte;

void nonCambia(int subvar) {
    printf("[nonCambia] Un intero: %d\n", subvar);
    int* psubvar = &subvar;
    printf("[nonCambia] L'indirizzo dell'intero: %p\n", psubvar);
    subvar = 55;
    printf("[nonCambia] Un intero (diverso): %d\n", subvar);
}

void oraCambia(int* psubvar) {
    printf("[oraCambia] Un intero: %d\n", *psubvar);
    printf("[oraCambia] L'indirizzo dell'intero: %p\n", psubvar);
    *psubvar = 55;
    printf("[oraCambia] Un intero (diverso): %d\n", *psubvar);
}

int main(int argc, char** argv) {

    int var = 42;
    printf("[main] Un intero: %d\n", var);
    int* pvar = &var;
    printf("[main] L'indirizzo dell'intero: %p\n", pvar);

    printf("\n[main] chiamo nonCambia...\n");
    nonCambia(var);
    printf("[main] Un intero (di nuovo): %d\n", var);

    printf("\n[main] chiamo oraCambia...\n");
    oraCambia(pvar);
    printf("[main] Un intero (di nuovo): %d\n", var);

    printf("\n[main] da ricordare:\n");
    printf("[main] *&variabile (%d) == variabile (%d)\n", *&var, var);
    printf("[main] &*puntatore (%p) == puntatore (%p)\n", &*pvar, pvar);

    return 0;
}

// ------------------------------------------------------------------
// aritmetica_puntatori.c
/*
     Variante di puntatori.c

     -> curiosa aritmetica dei puntatori
*/

#include <stdio.h>

int main() {
```

```
  short int a=11, b=12, c=13, d=14, e=15;
  int A=21, B=22, C=23, D=24, E=25;
  float f1=3.1, f2=3.2, f3=3.3, f4=3.4, f5=3.5;
  double d1=31., d2=32., d3=33., d4=34., d5=35.;

  int *pA;
  pA = &A;

  int *pB;
  pB = &B;

  printf("a-e : %d, %d, %d, %d, %d\n", a,b,c,d,e);
  printf("A-E : %d, %d, %d, %d, %d\n",A,B,C,D,E);
  //printf("d1-d2 : %.1f, %.1f, %.1f, %.1f, %.1f\n",f1,f2,f3,f4,f5);
  //printf("d1-d2 : %.1f, %.1f, %.1f, %.1f, %.1f\n",d1,d2,d3,d4,d5);

  printf("Puntatori ad a-e:\n %p\n %p\n %p\n %p\n %p\n", &a,&b,&c,&d,&e);
  printf("Puntatori ad A-E:\n %p\n %p\n %p\n %p\n %p\n", &A,&B,&C,&D,&E);
  // printf("Puntatori a f1-f5:\n %p\n %p\n %p\n %p\n %p\n", &f1,&f2,&f3,&f4,&f5);
  //  printf("Puntatori a d1-d5:\n %p\n %p\n %p\n %p\n %p\n", &d1,&d2,&d3,&d4,&d5);

  puts("\nvalore della variable a cui punta il puntatore");
  printf("A-E : %d, %d, %d, %d, %d\n", *(&A),*(&B),*(&C),*(&D),*(&E));
  puts("etc...");

  puts("\nCuriosa aritmetica dei puntatori\n");
  printf("*pA: %d; *(pA+4): %d; *(pA+1): %d; \n", *pA, *(pA+4), *(pA+1));
  printf("\n     pA: %p\n (pA+4): %p\n (pA+1): %p; \n", pA, (pA+4), (pA+1));

  printf("\n pB : %p\n", pB);
  printf("\n pB - pA : %ld\n", pB - pA);

  return 0;
}

// ------------------------------------------------------------------
// puntatori_puntatori_0.c

// puntatore a puntatore ...


#include <stdio.h>

int main() {

  int a = 13;

  int *p;
  p = &a;
  printf("a = %d ; *p = %d  ( p = %p)\n", a, *p, p);

  int **pp;
  pp = &p;
  printf("**pp = %d; *pp = %p; pp = %p \n", **pp, *pp, pp);

  return 0;
}
/*  Come esercizio provare ad andare a un livello superiore,
     ovvero puntatore→puntatore→puntatore→variabile ...      */
```

1.15 Vettori e puntatori

```
//-------------------------------------------------------------
// ritorna_vettore.c
/*
   Esempio di un vettore modificato da una funzione
*/

#include <stdio.h>

void ioVec(int n, int v[]) {
  int i;
  for(i=0; i<n; i++) v[i] = -i;
}

int main() {
  int i, v[10], lv;

  lv = (int) sizeof(v) / (int) sizeof(int);

  puts("\nPrima di  ioVec");
  for(i=0; i<lv; i++) {
    v[i] = 3*i;
    printf("i = %d\n", v[i]);
  }

  ioVec(lv, v);

  puts("\nDopo  ioVec");
  for(i=0; i<lv; i++) {
    printf("i = %d\n", v[i]);
  }

  return 0;
}

//-------------------------------------------------------------
// vettore-puntatore.c
/*
   Variante di ritorna_vettore.c per mostrare come
   all'interno della funzione vettore e puntatore sono la stessa cosa.
   Cambia solo il modo di accedere ai vari elementi
         =>   v[]   <-->  *v
         =>   v[i]  <-->  *(v+i)
*/

#include <stdio.h>

void ioVec(int n, int *v) {
  int i;
  for(i=0; i<n; i++) *(v+i) = -i;
}

int main() {
  int i, v[10], lv;

  lv = (int) sizeof(v) / (int) sizeof(int);
```

```
  puts("\nPrima di  ioVec");
  for(i=0; i<lv; i++) {
    v[i] = 3*i;
    printf("i = %d\n", v[i]);
  }

  ioVec(lv, v);

  puts("\nDopo  ioVec");
  for(i=0; i<lv; i++) {
    printf("i = %d\n", v[i]);
  }

  return 0;
}

//-------------------------------------------------------------
// vettore-puntatore_1.c
 /*
   Variante di ritorna_vettore.c per mostrare come
   all'interno della funzione vettore e puntatore sono la stessa cosa.
   Cambia solo il modo di accedere ai vari elementi
         =>   v[]   <-->  *v
         =>   v[i]  <-->  *(v+i)
*/

#include <stdio.h>

void ioVec(int n, int *v) {
  int i;
  for(i=0; i<n; i++) *(v+i) = -i;
}

int main() {
  int i, v[10], lv;

  lv = (int) sizeof(v) / (int) sizeof(int);

  puts("\nVettore originale");
  for(i=0; i<lv; i++) {
    v[i] = 3*i;
    printf("i = %d\n", v[i]);
  }

  puts("\nModifichiamo gli elementi di un vettore\nusando il suo nome come fosse un puntatore");
  for(i=0; i<lv; i++) *(v+i) = -i;

  puts("\nDopo  modifica");
  for(i=0; i<lv; i++) {
    printf("i = %d\n", v[i]);
  }

  return 0;
}

//-------------------------------------------------------------
// inverti_vettore.c
#include <stdio.h>
```

```
void invVec(int n, int v[]) {
  int i, vtemp[n];
  // mettiamo gli elementi, in ordine inverso, nel vettore temporaneo
  for(i=0; i<n; i++) {
    vtemp[i] = v[n-i-1];
  }
  // copiamoli infine su quello originale
  for(i=0; i<n; i++) {
    v[i] = vtemp[i];
  }
}

int main() {
  int i, v[10], lv;
  lv = (int) sizeof(v) / (int) sizeof(int);

  for(i=0; i<lv; i++) {
    v[i] = 2*i;
    printf("i = %d\n", v[i]);
  }

  invVec(lv, v);

  puts("\nDopo  invVec");
  for(i=0; i<lv; i++) {
    printf("i = %d\n", v[i]);
  }

  return 0;
}
```

1.16 Dettagli su puntatori e accesso alla mempria (F. Safai Tehrani)

```
//-----------------------------------------------------
// puntatori.c
   (vedi sopra)

//-----------------------------------------------------
// arrays.c
#include<stdio.h>
#include<stdlib.h>

int main(int argc, char** argv) {
    int vSize = 16;
    int index;

    if(argc>1) {
        vSize = atoi(argv[1]);
    }

    printf("Definisco un vettore di interi di lunghezza %d\n", vSize);

    int sData[vSize];

    printf("Dimensione del vettore da sizeof(v)/sizeof(int):     %d\n\n", sizeof(sData)/sizeof(int));
```

```
    printf("Puntatore all'array:                      v     => %p\n", sData);
    printf("Puntatore al primo elemento dell'array:   &v[0] => %p\n", &sData[0]);

    printf("\nPuntatore al secondo elemento dell'array: v+1   => %p\n", sData+1);
    printf("Puntatore al secondo elemento dell'array: &v[1] => %p\n", &sData[1]);

    printf("\nStampo il contenuto dell'array, usando l'accesso diretto ==> v[i]:\n");

    for(index=0; index<vSize; index++) {
        sData[index] = index*index;
        printf("v[%d] => %d -- ", index, sData[index]);
    }

    printf("\n\noppure, usando l'aritmetica dei puntatori ==> *(v+i):\n");

    for(index=0; index<vSize; index++) {
        *(sData+index) = index*index*index;
        printf("*(v+%d) => %d -- ", index, *(sData+index));
    }

   printf("\n");

    return 0;
}

//-------------------------------------------------------
// dataDump.c
#include<stdio.h>
#include<stdlib.h>

typedef unsigned char byte;

void dumpData(int len, void* data) {
    int i;
    for(i=0; i<len; i++) {
        if(i%8==0) { printf("\n[%p] ",data+i); }
        printf("%02x ", *((byte *)(data+i)));
    }
    printf("\n\n");
}

int main(int argc, char** argv) {
    int data[16];
    float data2[16];
    int index;
    for(index=0; index<16; index++) {
        data[index] = index+1;
        data2[index] = (float) index+1;
    }
    dumpData(sizeof(data), data);
    dumpData(sizeof(data2), data2);

    return 0;
}

//-------------------------------------------------------
// cmdLineDump.c
#include<stdio.h>
#include<stdlib.h>
```

```
typedef unsigned char byte;

void dumpData(int len, void* data) {
    int i;
    // +1 funziona dato che so che i miei dati sono stringhe
    // per cui sono null-terminated (ovvero finiscono con \0)
    for(i=0; i<len+1; i++) {
        printf("%02x ", *((byte *)(data+i)));
    }
    printf("\n\n");
}

int main(int argc, char** argv) {
    int idx, idx2;
    if(argc<2) {
        printf("Nope, mi serve qualche argomento...\n");
        exit(0);
    }

    for(idx=1; idx<argc; idx++) {
        printf("Stringa %d: ", idx);

        idx2 = 0;
        while(argv[idx][idx2] != 0) {
            printf("%c", argv[idx][idx2]);
            idx2++;
        }
        printf(" [intero=%d] [float=%f] ==> ", atoi(argv[idx]), atof(argv[idx]));
        dumpData(idx2, argv[idx]);
    }
    return 0;
}

//-----------------------------------------------------
// allineamento.c
#include<stdio.h>

typedef unsigned char byte;

void dumpData(int len, void* data) {
    int i;
    for(i=0; i<len; i++) {
        if(i%8==0) { printf("\n[%p] ",data+i); }
        printf("%02x ", *((byte *)(data+i)));
    }
    printf("\n\n");
}

int main(int argc, char** argv) {
    int idx;
    byte memory[8];

    for(idx=0; idx<8; idx++) {
        memory[idx] = (byte)(idx+1);
    }
    printf("Il contenuto di memory: ");
    dumpData(8, memory);
```

```
    printf("[allineato a %d byte ] byte  : ", sizeof(byte));
    for(idx=0; idx<8/sizeof(byte); idx++)
       printf("[0x%02x] %d ", memory[idx], memory[idx]);

    printf("\n\n[allineato a %d bytes] short : ", sizeof(short));
    short* memory_as_short = (short*)memory;
    for(idx=0; idx<8/sizeof(short); idx++)
       printf("[0x%04x] %d ", memory_as_short[idx], memory_as_short[idx]);

    printf("\n\n[allineato a %d bytes] int   : ", sizeof(int));
    int* memory_as_int = (int*)memory;
    for(idx=0; idx<8/sizeof(int); idx++)
       printf("[0x%08x] %d ", memory_as_int[idx], memory_as_int[idx]);

    printf("\n\n[allineato a %d bytes] long  : ", sizeof(long));
    long* memory_as_long = (long*)memory;
    for(idx=0; idx<8/sizeof(long); idx++)
       printf("[0x%02lx] %ld ", memory_as_long[idx], memory_as_long[idx]);

    printf("\n");

    return 0;
}
```

1.17 Altri dettagli sulle variabili

```
//=========================================================
// info_limiti_variabili.c
#include <stdio.h>
#include <limits.h>
#include <float.h>

int main() {

  printf("INT_MAX: %d\n",  INT_MAX);
  printf("INT_MIN: %d\n",  INT_MIN);
  printf("UINT_MAX: %u\n\n",  UINT_MAX);

  printf("LONG_MAX: %ld\n",  LONG_MAX);
  printf("LONG_MIN: %ld\n\n",  LONG_MIN);

  printf("ULONG_MAX: %lu\n\n",  ULONG_MAX);

  printf("FLT_MAX: %e\n", FLT_MAX);
  printf("FLT_MIN: %e\n", FLT_MIN);
  printf("FLT_DIG: %d\n", FLT_DIG);
  printf("FLT_MANT_DIG: %d\n", FLT_MANT_DIG);
  printf("FLT_MAX_EXP: %d\n", FLT_MAX_EXP);
  printf("FLT_EPSILON: %e\n\n", FLT_EPSILON);

  printf("DBL_MAX: %e\n", DBL_MAX);
  printf("DBL_MIN: %e\n", DBL_MIN);
  printf("DBL_DIG: %d\n", DBL_DIG);
  printf("DBL_MANT_DIG: %d\n", DBL_MANT_DIG);
  printf("BDL_MAX_EXP: %d\n", DBL_MAX_EXP);
  printf("BDL_EPSILON: %e\n\n", DBL_EPSILON);
```

```
  return 0;
}

//=========================================================
// bit_analyser.c (fuori programma, ma a qualcuno potrebbe interessare)

/* mostra la rapresentazione a bit di
    - integer
    - unsigned
    - float

  uso:   ./bit_analyser i -10
         ./bit_analyser u 127
         ./bit_analyser f 0.123

  [https://en.wikipedia.org/wiki/Single_precision]

*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

void printByte(char byte);
void analizzaInt(int *n);
void analizzaUns(unsigned *n);
void analizzaFlo(float *n);

int main(int argc, char *argv[]) {
  int i, n;
  unsigned un;
  float f;
  char *ptr;

  if (argc < 3) {
    printf("\nUso, ad es.: ./bit_analysier i -2345\n\n");
    return 0;
  }

  if ( strcmp("i", argv[1]) == 0 ) {
    printf("Analizzo un integer \n");
    n = atoi(argv[2]);
    analizzaInt(&n);
  } else if ( strcmp("u", argv[1]) == 0 )  {
    printf("Analizzo un unsigned \n");
    un = (unsigned) strtoul(argv[2], &ptr, 10);
    //https://www.tutorialspoint.com/c_standard_library/c_function_strtoul.htm
    analizzaUns(&un);
  } else if ( strcmp("f", argv[1]) == 0 ) {
    printf("Analizzo un float \n");
    f = atof(argv[2]);
    analizzaFlo(&f);
  } else {
    printf("Type %s non riconosciuto\nAccettati: i, u, f\n", argv[1]);
  }

  return 0;
```

```
}

//-----------------------------------------------------------
void printByte(char byte) {
  int j;
  for(j=0; j<8; j++) {
      printf("%d", (byte & (128 >> j)) && 1);
      if(j==3) printf(" ");
    }
  printf(" ");
}

void analizzaInt(int *n) {   // max: 2^31 - 1 = 2147483647
                             // min:           -2147483648
  int i;
  char byte;
  unsigned char *pc = (unsigned char *)n;
  printf("Rappresentazione di %d\nHex:    ", *n);
  for (i = 0; i < sizeof(int); i++) {
    printf("%02X ",  *(pc - i + sizeof(int) - 1) );
  }
  printf("\nBinary: ");
  for (i = 0; i < sizeof(int); i++) {
    printByte( *(pc - i + sizeof(int) - 1 ) );
  }
  puts("");
}

void analizzaUns(unsigned *n) {  // max: 2^32 - 1 =  4294967295
  int i;
  char byte;
  unsigned char *pc = (unsigned char *)n;
  printf("Rappresentazione di %u\nHex:    ", *n);
  for (i = 0; i < sizeof(int); i++) {
    printf("%02X ",  *(pc - i + sizeof(int) - 1) );
  }
  printf("\nBinary: ");
  for (i = 0; i < sizeof(int); i++) {
    printByte( *(pc - i + sizeof(int) - 1 ) );
  }
  puts("");
}

void analizzaFlo(float *n) {
   int i,j;
  unsigned char *pc = (unsigned char *)n;
  // unsigned char *pc0 = (unsigned char *)n;
  unsigned char byte;
  unsigned char bits[sizeof(float)*8];
  unsigned char ibit = sizeof(float)*8;
  char segno;
  char esponente;
  float frazione;
  float ricostruito;

  printf(" %13e: ", *n);
  for (i = 0; i < sizeof(float); i++) {
    printf("%02X ", *(pc - i + sizeof(float) - 1) );
    printf("       ");
```

```
  }
  puts("");
  segno=1;
  printf("                ");
  for (i = 0; i < sizeof(float); i++) {
    byte = *(pc -i + sizeof(float) - 1) ;
    for(j=0; j<8; j++) {
      bits[--ibit] = (byte & (128 >> j)) && 1;
      printf("%d", bits[ibit]);  // non vale la pena di usare printByte()
                                 // visto che dobbiamo riempire bits[]
      if(j==3) printf(" ");
    }
    printf(" ");
  }
  if (bits[sizeof(float)*8 - 1]) {
    segno = -1;
    printf("  [-]");
  }
  puts("");
  esponente = -127;
  for (ibit=23; ibit<31; ibit++) {
    esponente += bits[ibit] * (1 << (ibit-23));
  }
  printf("esponente: %d - 127: %d\n", esponente + 127, esponente);

  frazione = 1;
  for (i=1; i<23; i++) {
    frazione += bits[23-i] * pow(2,-i);
  }
  printf("frazione : %f\n", frazione);

  ricostruito = segno * pow(2, esponente) * frazione;

  printf("valore ricostruito dal contenuto dei bit %e\n\n", ricostruito);
}
```

1.18 random_a.c

```
/*
    generatori random 'umani' per uniforme discreta e continua

    Versione in cui le funzioni definite dopo main()
    -> i 'prototype' delle funzioni vanno dichiarati prima
*/

#include <stdio.h>
#include <stdlib.h>    // per la random
#include <time.h>      // per il seed

// --- dichiariamo le funzioni, il cui sorgente è in fondo ----
void setRandomSeed( unsigned RandomSeed);
int irunif(int min, int max);
float frunif(float min, float max);

int main() {
  int i;
  unsigned seed;
```

```
  printf("Dai un seed (intero >= 0;  se == 0 si usa il tempo) : ");
  scanf("%u", &seed);
  setRandomSeed(seed);

  // proviamo la random uniforme intera
  for(i=0; i<10; i++) {
    printf(" %d \n", irunif(1, 6) );   // per giocarci a dadi ...
  }


  for(i=0; i<10; i++) {
    printf(" %f \n", frunif(0, 10) );
  }

  return 0;
}


//---  funzioni (ricordarsi di dichiarale prima di main!)  ----

void setRandomSeed( unsigned RandomSeed) {
  time_t t;

  if (RandomSeed == 0 ) {     // se == 0 si usa il tempo
     RandomSeed = (unsigned) time(&t);
  }

  printf("seed %u\n", RandomSeed );
  srand( RandomSeed );
}

int irunif(int min, int max) {

  if(min >= max) return(0);

  return( min + rand() % (max-min+1) );
}

float frunif(float min, float max) {
  float rzerouno;

  if(min >= max) return(0);

  rzerouno = (float) rand() / (float) RAND_MAX;

  return( min + rzerouno * (max-min) );
}
```

1.19 Esempio di organizzazine delle function in una 'libreria'

```
//-----------------------------------------------------------------
// random_lib.c
/*
    Generatori random 'umani' per uniforme discreta e continua
    File con sole funzioni
    Compilare con "gcc -c random_lib.c"
```

```
*/

#include <stdio.h>
#include <stdlib.h>    // per la random
#include <time.h>      // per il seed
//--------------------------------------
void setRandomSeed( unsigned RandomSeed) {
  time_t t;

  if (RandomSeed == 0 ) {     // se == 0 si usa il tempo
     RandomSeed = (unsigned) time(&t);
  }

  printf("seed %u\n", RandomSeed );
  srand( RandomSeed );
}
//------------------------------
int irunif(int min, int max) {

  if(min >= max) return(0);

  return( min + rand() % (max-min+1) );
}
//-------------------------------
float frunif(float min, float max) {
  float rzerouno;

  if(min >= max) return(0);

  rzerouno = (float) rand() / (float) RAND_MAX;

  return( min + rzerouno * (max-min) );
}

//---------------------------------------------------------------------
// random_lib.h
void setRandomSeed( unsigned RandomSeed);
int irunif(int min, int max);
float frunif(float min, float max);

//---------------------------------------------------------------------
// random_test.c
/*
    generatori random 'umani' per uniforme discreta e continua

    Versione con le funzioni definite in random_lib.c

    Compilare con "gcc -o random_test random_test.c random_lib.o"

*/

#include <stdio.h>
#include "random_lib.h"    // per random_lib.c
// si noti l'assenza di <stdlib.h> e <time.h>, già inclusi dove servivano

int main() {
  int i;
  unsigned seed;
```

```
  printf("Dai un seed (intero >= 0;  se == 0 si usa il tempo) : ");
  scanf("%u", &seed);
  setRandomSeed(seed);

  // proviamo la random uniforme intera
  for(i=0; i<10; i++) {
    printf(" %d \n", irunif(1, 6) );   // per giocarci a dadi ...
  }


  for(i=0; i<10; i++) {
    printf(" %f \n", frunif(0, 10) );
  }

  return 0;
}
```

1.20 Uso di argc e argv

```
//-------------------------------------------------------------
// uso_argc-argv1.c
/*
    uso di argc e argv per passare dei parametri
    al programma da riga di comando

    Eseguire con il comando
       ./uso_argc-argv1 argomenti

    ad esempio
       ./uso_argc-argv1  45 ciao 7 3.14
*/

#include<stdio.h>
int main(int argc, char* argv[]) {
  int i;

  printf("argc = %d \n", argc);
  for(i=0; i<argc; i++) {
    printf("i = %d, argv[%d] = %s  \n", i, i, argv[i]);
  }

  return 0;
}

//-------------------------------------------------------------
// uso_argc-argv2.c
/*
    uso di argc e argv per passare dei parametri
    al programma da riga di comando

    Variante rispetto a uso_argc-argv1.c
    -> secondo argomento dichiarato come puntatore a puntatori
       (come in arrays.c di F. Safai)

    uso simile, ad esempio
      ./uso_argc-argv2  45 ciao 7 3.14
```

```
*/

#include<stdio.h>
int main(int argc, char** argv) {
  int i;

  printf("argc = %d \n", argc);
  for(i=0; i<argc; i++) {
    printf("i = %d, argv[%d] = %s  \n", i, i, *(argv+i));
  }

  return 0;
}

//-------------------------------------------------------------
// uso_argc-argv3.c
/*
    uso di argc e argv per passare dei parametri
    al programma da riga di comando

    gli elementi di argv sono delle stringhe,
    ovvero arrays ('vettori') di caratteri
    (come si capisce anche da %s nel printf)

    vediamo dei dettagli, stampando a uno a uno (con uno spazio)
    i caratteri della stringa, ad esempio eseguendo il comando

    -> si noti lo ZERO (numero zero e non carattere zero!)
       per terminare la stringa
*/

#include<stdio.h>
int main(int argc, char* argv[]) {
  int i;
  char c;
  char *pc;

  printf("argc = %d \n", argc);
  for(i=0; i<argc; i++) {
    printf("i = %d, argv[%d] = %s  \n", i, i, argv[i]);
    pc = argv[i];
    do {
      c = *pc;
      printf("%c ", c);
      pc++;
    } while(c != 0);
    printf("\n");
    printf("[Ultimo carattere della stringa: %d]\n\n", c);
  }

  return 0;
}

//-------------------------------------------------------------
// ascii_code.c
/*
     Variazione di ascii_code_0.c
     con input da riga di comando
     SOLO se seguito con comando del tipo
```

```
        ./ascii_code_arg a
     (ove a è il carattere di interesse)
                                                              */
#include <stdio.h>
int main(int argc, char *argv[])
{
    int i;
    char c;
    unsigned char b;

    printf("argc = %d\n", argc);

    // argv[0] punta sempre al nome del programma
    printf("Nome del programma %s\n\n", argv[0]);

    // analizziamo gli argomenti della chiamata
    if (argc < 2) {
      printf("Potevi dare l'argomento al volo nel comando!\n");
      printf("Va bene uguale, dallo ora: ");
      scanf("%c", &c);
    } else {
      c = *argv[1];
    }

    printf("ASCII value of %c : %d, 0x%X, ", c, c, c);

    // inizializziamo b con primo bit a sinistra
    b = 1 << 7;

    // scriviamo i vari bit di c
    for (i=0; i<8; i++) {
      if( (b >> i) & c ) {
        printf("1");
      } else {
        printf("0");
      }
    }

   printf("\n");

   return 0;
}

//-------------------------------------------------------------
// uso_argc-argv4.c
/*
    uso di argc e argv per passare dei parametri
    al programma da riga di comando

    Questa versione si aspetta due parametri,
    di cui il primo intero e il secondo float

    Eseguire con il comando
       ./uso_argc-argv4 argomenti

    ad esempio
       ./uso_argc-argv4  2  21.56
*/
```

```
#include<stdio.h>
#include<stdlib.h>

int main(int argc, char* argv[]) {
  int n;
  float f;

  if(argc < 3) {
    printf("Devi dare due parametri: il primo intero, il secondo float\n");
    return 0;
  }

  n = atoi(argv[1]);
  f = atof(argv[2]);

  printf("Ho ricevuto n = %d e f = %f\n", n, f);

  return 0;
}
```

1.21 Introduzione alle string

```
//-----------------------------------------------------
// strings_1.c
#include <stdio.h>
int main() {
  char poeta[] = "Dante Alighieri";
  char *p;
  int i;

  p = poeta;

  printf("Poeta: %s\n", poeta);
  printf("Puntatore a poeta[0]: %p\n", p);

  puts("\nCaratteri a partire da poeta[0]");
  for (i=0; i<20; i++) {
    printf("i = %2d:  %c  [%3d]\n", i, *(p+i), *(p+i));
  }

  puts("\nNotare lo 0 dopo il nome (per i=15)");
  puts("La string finisce lì, ma si può sforare...  Attenzione!!");

  return 0;
}

//-----------------------------------------------------
// strings_2.c
#include <stdio.h>

int main() {
  char poeta[] = {'D', 'a', 'n', 't', 'e', ' ',
                  'A', 'l', 'i', 'g', 'h', 'i', 'e', 'r', 'i'};
  char *p;
  int i;

  p = poeta;
```

```
  printf("Poeta: %s\n", poeta);
  printf("Puntatore a poeta[0]: %p\n", p);

  puts("\nCaratteri a partire da poeta[0]");
  for (i=0; i<20; i++) {
    printf("i = %2d:  %c  [%3d]\n", i, *(p+i), *(p+i));
  }

  puts("\nNotare lo 0 dopo il nome (per i=15)");
  puts("La string finisce lì, ma si può sforare...  Attenzione!!");

  return 0;
}

//-------------------------------------------------
// strings_3.c
#include <stdio.h>

int main() {
  char poeta[] = "Dante Alighieri";
  int i;

  printf("Poeta: %s\n", poeta);

  puts("\nCaratteri a partire da poeta[0]");
  for (i=0; i<20; i++) {
    printf("i = %2d:  %c  [%3d]\n", i, poeta[i], poeta[i]);
  }

  puts("\nNotare lo 0 dopo il nome (per i=15)");
  puts("La string finisce lì, ma si può sforare...  Attenzione!!");

  return 0;
}

//-------------------------------------------------
// strings_4.c
#include <stdio.h>

int main() {
  char poeta[] = "Dante Alighieri";
  int i;

  printf("Poeta: %s\n", poeta);

  puts("\nMettiamo zero in poeta[5]...");
  poeta[5] = 0;
  printf("Poeta: %s\n", poeta);

  puts("\nRimettiamo lo spazio...");
  poeta[5] = ' ';
  printf("Poeta: %s\n", poeta);

  puts("\nMettiamo '\\0' in poeta[5]...");   // notare '\\' per stampare '\'
  poeta[5] = '\0';
  printf("Poeta: %s\n", poeta);

  return 0;
```

```
}

//-------------------------------------------------
// strings_5.c
#include <stdio.h>

int main() {
  char poeta[] = "Dante Alighieri";
  int i;

  printf("Poeta: %s\n", poeta);

  puts("\nSostituiamo lo zero in poeta[15] con uno spazio");
  poeta[15] = ' ';
  printf("Poeta: %s\n", poeta);

  return 0;
}

//-------------------------------------------------
// strings_6.c
#include <stdio.h>

int lunghezza(char s[]) {
  int i, len=0;
  while(1) {
    if (s[len] == '\0') break;
    len++;
  }
  return(len);
}

int main() {
  char poeta[] = "Dante Alighieri";

  printf("Poeta: %s  [%d caratteri]\n", poeta, lunghezza(poeta) );

  return 0;
}
```

1.22 Ancora sulle string

```
//-------------------------------------------------
// concatena.c
#include <stdio.h>

#define MAXLEN 20
// riutilizziamo la function lunghezza  ( ->  strings_6.c )
int lunghezza(char s[]) {
  int i, len=0;
  while(1) {
    if (s[len] == '\0') break;
    len++;
  }
  return(len);
}
```

```
int main() {
  char nome[] = "David";
  char cognome[] = "Hume";
  char NomeCognome[MAXLEN+1]; // non dimentichiamo lo spazio per lo 0 finale!!
  int i, ln, lc, lnc=MAXLEN+1;

  ln = lunghezza(nome);
  lc = lunghezza(cognome);

  printf("Vogliamo concatenare %s [len=%d] e %s [len=%d]\n",
         nome, ln, cognome, lc);

  // controlliamo che MAXLEN sia sufficiente
  // (anche se in questo caso si vede a occhio)
  if( (ln + lc + 1) > MAXLEN) {
    printf("Il vettore NomeCognome è troppo piccolo (%d)\n",  MAXLEN);
    printf("  -> deve essere almeno %d\n", ln + lc + 2);
    printf("  -> ricorda che anche lo spazio è un carattere ");
    printf("(e poi c'è pure lo zero finale)\n");
    return 0;
  }

  for (i=0; i<ln; i++) {
    NomeCognome[i] = nome[i];
  }
  NomeCognome[ln] = ' ';

  for (i=0; i<lc; i++) {
    NomeCognome[ln+1+i] = cognome[i];
  }
  NomeCognome[ln+lc+1] = 0; // RICORDARSI dello zero alla fine!

  printf("Risultato:  %s [len=%d]\n", NomeCognome, lunghezza(NomeCognome));

  return 0;
}

//-----------------------------------------------------
// esempio_sprintf.c
/*
   primo uso di sprintf()
*/

#include <stdio.h>
int main() {

  char saluto[100];  // melius abundare.... sine 'exagerare'...

  printf("Ecco un primo esempio:\n\n");
  // stampiamo una string nei due modi che conosciamo
  puts("ciao a tutti");
  printf("ciao a tutti\n");

  // mettiamo il saluto nel vettore di caratteri saluto[]
  sprintf(saluto, "ciao a tutti");

  // e lo stampiamo in diversi modi
  puts(saluto);
  printf("%s\n", saluto);
```

```
  printf("\nRendiamolo più vivace (anche se non è più un 'saluto'):\n\n");
  int n1=2, n2=3;
  sprintf(saluto, "%s: %d x %d = %d", "semplice moltiplicazione", n1, n2, n1*n2);
  // e lo stampiamo in diversi modi
  puts(saluto);
  printf("%s\n", saluto);

  return 0;
}

//------------------------------------------------------
// test_toUpper.c
#include <stdio.h>

void toUpper(char s[]) {
  int i = 0;
  char diff = 'A' - 'a';

  while(1) {
    if (s[i] == '\0') break;
    if (s[i] >= 'a' && s[i] <= 'z') {
      s[i] = s[i] + diff;
    }
    i++;
  }
}

int main() {
  char poeta[] = "Dante Alighieri";

  printf("String iniziale: %s\n", poeta);

  toUpper(poeta);

  printf("String finale: %s\n", poeta);

  return 0;
}

//------------------------------------------------------
// to_upper.c
#include <stdio.h>

#define LENMAX 50
#define SPAZIO 32

void toUpper(char s[]) {
  int i = 0;
  static char diff = 'A' - 'a';

  while(1) {
    if (s[i] == '\0') break;
    if (s[i] >= 'a' && s[i] <= 'z') {
      s[i] = s[i] + diff;
    }
    i++;
  }
}
```

```
int main(int argc, char* argv[]) {
  int i, len;
  char s[LENMAX+1], *p;

  if (argc == 1) {
    puts("Devi dare la string da convertire (max 50 caratteri)");
    return 0;
  }

  // merge delle varie string di argv[]
  len = 0;
  for (i=1; i<argc; i++) {
    p = argv[i];
    while(1) {
      if (*p == '\0') {
        s[len] = SPAZIO;
        break;
      }
      s[len] = *p;
      p++;
      len++;
      if(len == LENMAX) break;
    }
    len++;
  }
  s[len] = '\0';

  printf("String ricevuta: %s\n", s);

  toUpper(s);

  printf("String convertita: %s\n", s);

  return 0;
}

//-----------------------------------------------------
// to_upper_dbg.c
#include <stdio.h>

#define LENMAX 50
#define SPAZIO 32

void toUpper(char s[]) {
  int i = 0;
  static char diff = 'A' - 'a';

  printf("[to Upper] diff = %d\n", diff);
  while(1) {
    if (s[i] == '\0') break;     // notare i++
    printf("[to Upper] i=%2d  %c (%d)\n", i, s[i], s[i]);
    // se è lettera minuscola viene trasformata in maiuscola
    if (s[i] >= 'a' && s[i] <= 'z') {
      s[i] = s[i] + diff;
    }
    i++;
  }
}
```

```
int main(int argc, char* argv[]) {
  int i, len;
  char s[LENMAX+1], *p;

  printf("argc = %d \n", argc);
  for(i=0; i<argc; i++) {
    printf("i = %d, argv[%d] = %s  \n", i, i, argv[i]);
  }

  if (argc == 1) {
    puts("Devi dare la string da convertire (max 50 caratteri)");
    return 0;
  }

  // merge delle varie string di argv[]
  len = 0;
  for (i=1; i<argc; i++) {
    p = argv[i];
    printf("p : %p  [inizio di %s]\n", p, argv[i]);
    while(1) {
      printf(" %c", *p);
      if (*p == '\0' && i < (argc-1)) {    // spazio fino alla penultima parola
        s[len] = SPAZIO;
        break;
      }
      s[len] = *p;
      p++;
      len++;
      if(len == LENMAX) break;
    }
    len++;
    printf("\n  (len = %d)\n", len);
  }
  s[len] = '\0';
  printf("String ricevuta: %s\n", s);

  toUpper(s);

  printf("String convertita: %s\n", s);

  return 0;
}
```

1.23 Introduzione alle matrici

```
//-------------------------------------------------------
// matrici_1.c

#include <stdio.h>
int main() {
  int i, j, k, x;
  int A[2][3] = {{1, 3, 0}, {-1, 5, 9}};
  int B[3][2] = {{1, 3}, {-1, 0}, {4, 7}};

  puts("A:");
  for(i=0; i<2; i++) {
    for(j=0; j<3; j++) {
```

```
      printf("%3d ", A[i][j]);
    }
    printf("\n");
  }

  puts("\nB:");
  for(i=0; i<3; i++) {
    for(j=0; j<2; j++) {
      printf("%3d ", B[i][j]);
    }
    printf("\n");
  }

  puts("\nA x B:");
  for (i=0; i<2; i++) {
    for (k=0; k<2; k++) {
      x=0;
      for(j=0; j<3; j++) {
        x += A[i][j] * B[j][k];
      }
      printf("%3d ", x);
    }
    printf("\n");
  }
}

//-------------------------------------------------------------
// matrici_2.c

#include <stdio.h>

int main() {
  int i, j, k, x;
  int A[2][3] = {{1, 3, 0}, {-1, 5, 9}} ;
  int B[3][2] = {{1, 3}, {-1, 0}, {4, 7}};
  int C[2][2] = {0};
  int *p;

  puts("A:");
  for(i=0; i<2; i++) {
    for(j=0; j<3; j++) {
      printf("%3d ", A[i][j]);
    }
    printf("\n");
  }

  puts("\nB:");
  for(i=0; i<3; i++) {
    for(j=0; j<2; j++) {
      printf("%3d ", B[i][j]);
    }
    printf("\n");
  }

  puts("\nC:");
  for(i=0; i<2; i++) {
    for(j=0; j<2; j++) {
      printf("%3d ", C[i][j]);
    }
```

```
    printf("\n");
  }

  // cambiamo tutti gli elementi di C
  /*
  p = &C[0][0];
  for (i=0; i<4; i++) {
    *(p+i) = -7;
  }
  */

  // prodotto C = AxB

  for (i=0; i<2; i++) {
    for (k=0; k<2; k++) {
      for(j=0; j<3; j++) {
        C[i][k] += + A[i][j] * B[j][k];
      }
    }
  }

  puts("\nC:");
  for(i=0; i<2; i++) {
    for(j=0; j<2; j++) {
      printf("%3d ", C[i][j]);
    }
    printf("\n");
  }

  return 0;
}

//-------------------------------------------------------------
// matrici_3.c
#include <stdio.h>

void stampaMatriceI(int nr, int nc, int A[nr][nc]) {
  int i, j;

  for(i=0; i<nr; i++) {
    for(j=0; j<nc; j++) {
      printf("%3d ", A[i][j]);
    }
    printf("\n");
  }
}

int main() {
  int i, j, k, x;
  int A[2][3] = {{1, 3, 0}, {-1, 5, 9}} ;
  int B[3][2] = {{1, 3}, {-1, 0}, {4, 7}};
  int C[7][7] = {0};

  puts("A:");
  stampaMatriceI(2, 3, A);
  puts("\nB:");
  stampaMatriceI(3, 2, B);
  puts("\nC:");
  stampaMatriceI(7, 7, C);
```

```
  return 0;
}


//-----------------------------------------------------------
// matrici_4.c
#include <stdio.h>

void stampaMatriceI(int nr, int nc, int A[nr][nc]) {
  int i, j;

  for(i=0; i<nr; i++) {
    for(j=0; j<nc; j++) {
      printf("%3d ", A[i][j]);
    }
    printf("\n");
  }
}

int main() {
  int i, j, k, x;
  int A[2][3] = {{1, 2, 3}, {4, 5, 6}};
  int B[4][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16};

  puts("A:");
  stampaMatriceI(2, 3, A);
  puts("\nA (nr e nc scambiate!):");
  stampaMatriceI(3, 2, A);

  puts("B:");
  stampaMatriceI(4, 4, B);
  puts("B (dimensioni modificate!):");
  stampaMatriceI(2, 8, B);

  puts("\n\nNota: contrariamente a R, in C i valori sono ordinati per riga");
  puts("-> segue in matrici_5.c");

  return 0;
}

//-----------------------------------------------------------
// matrici_5.c
#include <stdio.h>

void stampaMatriceI(int nr, int nc, int A[nr][nc]) {
  int i, j;

  for(i=0; i<nr; i++) {
    for(j=0; j<nc; j++) {
      printf("%3d ", A[i][j]);
    }
    printf("\n");2
  }
}

void stampaMatriceIp(int nr, int nc, int *p) {
  int i, j;
```

```
  for(i=0; i<nr; i++) {
    for(j=0; j<nc; j++) {
      printf("%3d ", *(p + nc*i + j));
    }
    printf("\n");
  }

}

int main() {
  int i, j, k, x;
  int A[2][3] = {1, 2, 3, 4, 5, 6};
  int B[4][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16};

  int *pA, *pB;

  puts("A:");
  stampaMatriceI(2, 3, A);
  puts("B:");
  stampaMatriceI(4, 4, B);

  pA = &A[0][0];
  pB = &B[0][0];

  puts("\nA (mediante puntatore):");
  for(i=0; i < 2*3; i++) {
    printf("%d\n", *(pA+i));
  }
  puts("\nB (mediante puntatore):");
  for(i=0; i < 4*4; i++) {
    printf("%d\n", *(pB+i));
  }

  puts("\nStampa matrice da stampaMatriceIp (passaggio per puntatore)");
  puts("A:");
  stampaMatriceIp(2, 3, pA);
  puts("B:");
  stampaMatriceIp(4, 4, pB);

  return 0;
}

//-----------------------------------------------------------
// matrici_6.c
#include <stdio.h>

void stampaMatriceI(int nr, int nc, int A[nr][nc]) {
  int i, j;

  for(i=0; i<nr; i++) {
    for(j=0; j<nc; j++) {
      printf("%3d ", A[i][j]);
    }
    printf("\n");
  }
}

void setMatrix(int valore, int n, int *p) {
  int i;
```

```
  for(i=0; i<n; i++) *(p+i) = valore;
}


int main() {
  int i, j, k, x;
  int A[2][3] = {1, 2, 3, 4, 5, 6};
  int B[4][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16};

  int *pA, *pB;

  puts("A:");
  stampaMatriceI(2, 3, A);
  puts("B:");
  stampaMatriceI(4, 4, B);

  pA = &A[0][0];
  pB = &B[0][0];

  setMatrix(-1, 2*3, pA);
  setMatrix(0, 4*4, pB);

  puts("\nValori settati a -1 e a 0, risettivamente:");

  puts("A:");
  stampaMatriceI(2, 3, A);
  puts("B:");
  stampaMatriceI(4, 4, B);


  return 0;
}

//-----------------------------------------------------------
// matrici_7.c
#include <stdio.h>

void stampaMatriceI(int nr, int nc, int A[nr][nc]) {
  int i, j;

  for(i=0; i<nr; i++) {
    for(j=0; j<nc; j++) {
      printf("%3d ", A[i][j]);
    }
    printf("\n");
  }
}

// proviamo a stampare una matrice come vettore
void stampaMatriceIasVector(int n, int A[n]) {
  int i;

  for(i=0; i<n; i++) {
      printf("%3d\n", A[i]);
  }

  /* Nota: gcc dà un warning (vedi in fondo), ma la function funziona */

}
```

```
int main() {
  int i, j, k, x;
  int A[2][3] = {1, 2, 3, 4, 5, 6};
  int B[4][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16};

  int *pA, *pB;

  puts("A:");
  stampaMatriceI(2, 3, A);
  puts("B:");
  stampaMatriceI(4, 4, B);

  puts("\nStampa matrice da stampaMatriceIasVector");
  puts("A:");
  stampaMatriceIasVector(2*3, A);
  puts("B:");
  stampaMatriceIasVector(4*4, B);

  return 0;
}

/* Nota: nella compilazione gcc dà il seguente messaggio
         di 'allerta' (warning), ma il programma funziona!

matrici_6.c: In function ‘main’:
matrici_6.c:37:31: warning: passing argument 2 of ‘stampaMatriceIasVector’ from incompatible pointer type [
   stampaMatriceIasVector(2*3, A);
                               ^
matrici_6.c:14:6: note: expected ‘int *’ but argument is of type ‘int (*)[3]’
 void stampaMatriceIasVector(int n, int A[n]) {
      ^~~~~~~~~~~~~~~~~~~~~~
matrici_6.c:39:31: warning: passing argument 2 of ‘stampaMatriceIasVector’ from incompatible pointer type [
   stampaMatriceIasVector(4*4, B);
                               ^
matrici_6.c:14:6: note: expected ‘int *’ but argument is of type ‘int (*)[4]’
 void stampaMatriceIasVector(int n, int A[n]) {
      ^~~~~~~~~~~~~~~~~~~~~~
*/
//-------------------------------------------------------
// matrici_prod.c

include <stdio.h>

void stampaMatriceI(int nr, int nc, int A[nr][nc]) {
  int i, j;

  for(i=0; i<nr; i++) {
    for(j=0; j<nc; j++) {
      printf("%3d ", A[i][j]);
    }
    printf("\n");
  }
}

void matProd(int nrA, int ncA, int A[nrA][ncA],
             int nrB, int ncB, int B[nrB][ncB],
             int nrC, int ncC, int C[nrC][ncC] ) {
  int i, j, k;
```

```
  for (i=0; i<nrA; i++) {
    for (k=0; k<nrC; k++) {
      for(j=0; j<nrB; j++) {
        C[i][k] += + A[i][j] * B[j][k];
      }
    }
  }
}

int main() {
  int i, j, k, x;
  int A[2][3] = {{1, 3, 0}, {-1, 5, 9}} ;
  int B[3][2] = {{1, 3}, {-1, 0}, {4, 7}};
  int C[2][2] = {0};

  puts("A:");
  stampaMatriceI(2, 3, A);

  puts("\nB:");
  stampaMatriceI(3, 2, B);

  puts("\nC (inizializzata):");
  stampaMatriceI(2, 2, C);

  matProd(2, 3, A,
          3, 2, B,
          2, 2, C);

  puts("\nC:");
  stampaMatriceI(2, 2, C);

  return 0;
}
```

1.24 Primi esempi di ordinamento (alquanto naive)

```
//-----------------------------------------------------
//  inserisci_ordinato_ossatura.c
*  genera numeri casuali da 0 a 999
    e li mette, in ordine crecente, in un vetttore di N elementi

    ossatura da completare
*/

#include <stdio.h>
#include <stdlib.h>

#define N 10

int main() {
  int i, j, n=0, new, v[N];
  int inserito;
  int dbg = 1;

  // il primo valore va sicuramente nella prima posizione
  v[0] = rand() % 1000;
```

```
  // e poniamo a 1 il numro totale di valori nel vettore
  n = 1;
  printf("v[0] = %d\n", v[0]);

  // facciamo le altre estrazioni e sistemiamo i valori opportunamente
  while (n<N) {
    if (dbg) printf("n = %d\n", n);
    // nuovo valore
    new = .....
    printf("new = %d\n", new);

    inserito = 0; // Settiamo a 0 il 'flag' inserito
    for(i=0; i<n; i++) {
      if (dbg)  printf("Prima dell'if: v[%d] = %d\n", i, v[i]);
      if (new < v[i])  {
        if (dbg)  printf("%d va messo nella posizione %d\n", new, i);
        // prima facciamogli posto traslando i valori da i in poi
        ..........
        // posizioniamo new in v[i]
        ..........
        // ci segniamo che new è già stato inserito
        inserito = 1;
        break;
      }
    }
    if (dbg) printf("Inserito? %d \n", inserito);
    if (! inserito) {
      // se non è stato inserito prima di in vecchio valore va messo alla fine
      if (dbg)  printf("2. vado a mettere %d nella posizione %d\n", new, n);
      .........
    }
    // incrementiamo il valore di n
    ......

    if (dbg) {
      for(i=0; i<n; i++) printf(" %d", v[i]);
      puts("\n");
    }
  }

  puts("\nOrdinamento finale");
  for(i=0; i<n; i++) printf(" %d", v[i]);
  puts("");

  return 0;
}

//-----------------------------------------------------
// sorting_0.c
*
    primo tentativo di sorting, alquanto ingenuo
*/

int indMin(int n, int v[]) {

  int i, ind = 0;
  int vmin=v[0];

  for (i=0; i<n; i++) {
```

```
    if (v[i] < vmin) {
      ind = i;
      vmin = v[i];
    }
  }
  return(ind);
}

#define N 30

#include <stdio.h>
#include <stdlib.h>

int main() {
  int i, j, ind;
  int v[N], nv, v1[N], nv1;

  // Nota: non settiamo il seed della random per controllare meglio
  for(i=0; i<N; i++) v[i] = rand() % 100;     // nr random fra 0 e 99

  for(i=0; i<N; i++) printf("%2d ", v[i]);
  puts("");

  nv = N;
  nv1 = 0;

  while(nv > 0) {
    // troviamo il minimo
    ind = indMin(nv, v);
    // printf("min: v[%d] = %2d \n", ind, v[ind]);

    // lo mettiamo in v1[]
    v1[nv1] =  v[ind];
    nv1++;

    // e lo togliamo da v[]
    for(i=ind; i < nv-1; i++) v[i] = v[i+1];
    nv--;
    // for(i=0; i<nv; i++) printf(" %2d ", v[i]);
    // puts("");
  }

  for(i=0; i<N; i++) printf("%2d ", v1[i]);
  puts("");

  return 0;
}

//-----------------------------------------------------
// sorting_1.c
/*
    Variante di sorting_0.c, basato sulla stessa idea
    di cercare il valore minimo fra quelli rimasti,
    ma inserendo il valore trovato nel vettore
    di partenza
 */

#define N 20
```

```
#include <stdio.h>
#include <stdlib.h>

int indMin(int n, int v[]) {
  int i, ind = 0;
  int vmin=v[0];

  for (i=0; i<n; i++) {
    if (v[i] < vmin) {
      ind = i;
      vmin = v[i];
    }
  }

  return(ind);
}

int main() {
  int i, j, ind;
  int v[N], tmp, nOrd;

  // Nota: non settiamo il seed della random per controllare meglio
  for(i=0; i<N; i++) v[i] = rand() % 100;     // nr random fra 0 e 99

  for(i=0; i<N; i++) printf("%2d ", v[i]);
  puts("");

  nOrd = 0;  // numero di valori messi in ordine

  while(nOrd < N) {
    // troviamo l'indice del minimo ma spostando il punto
    // di partenza della ricerca [riutilizzando, con un piccolo
    // trucco, indMin() -- in pratica le passiamo un vettore
    // che a mano a mano si riduce di lunghezza!]
    ind = indMin(N-nOrd, &v[nOrd]) + nOrd;        // <- si noti "+nOrd" !

    printf("min: v[%d] = %2d \n", ind, v[ind]);

    // lo spostiamo, scambiandolo con quello che c'era
    // nella posizione di arrivo
    tmp = v[nOrd];
    v[nOrd] =  v[ind];
    v[ind] = tmp;
    nOrd++;
  }

  for(i=0; i<N; i++) printf("%2d ", v[i]);
  puts("");

  return 0;
}
```

1.25 coefficienti_andamenti_ossatura.c

```
#include <stdio.h>
#include <math.h>
```

```
void lepLaw(double x1, double x2, double y1, double y2,
    double *a, double *b, char type) {

  /*  lineare        y = a + b*x
      esponenziale   y = a * exp(b*x)
      potenza        y = a * pow(x, b)
  */

  switch(type) {
    case 'l' :
      printf("Andamento lineare\n");
      // .....
      // ..... fare i conti per calcolare a e b
      break;

    case 'e' :
      printf("Andamento esponenziale\n");
      // .....
      // ..... fare i conti per calcolare a e b
      break;

    case 'p' :
      printf("Legge di potenza\n");
      // .....
      // ..... fare i conti per calcolare a e b
      break;

    default:
      printf("Andamento '%c' non riconosciuto\n", type);
  }

}

int main() {
  double x1, x2, y1, y2, a, b;

  x1 = 1;
  x2 = 2;
  y1 = 5;
  y2 = 10;

  lepLaw(x1, x2, y1, y2, &a, &b, 'l');
  // stampare i coefficienti
  lepLaw(x1, x2, y1, y2, &a, &b, 'e');
  // stampare i coefficienti
  lepLaw(x1, x2, y1, y2, &a, &b, 'p');
  // stampare i coefficienti
  lepLaw(x1, x2, y1, y2, &a, &b, 'x');
}
```

1.26 calcolatrice.c

```
/* Calcolatrice a riga di comando
   (quattro operazioni + potenze)

   Nota: per la moltiplicazione sono dati due simboli, x e *,
         ma se si usa * nella riga di comando va 'escaped',
         o mettendolo fra apicetti singoli, oppure scrivedo \*
```

```
   Per il numero di cifre del printf regolarsi come si crede opportuno
   (anche in base ai numeri che si vogliono utilizzare - per sicurezza
   il risultato è dato anche in notazione esponenziale)

 */

#include <stdio.h>
#include <stdlib.h>   // per atoi()

int main(int argc, char *argv[]) {
  float a, b, ris;
  char op, *p;

  if (argc < 4) {
    printf("Immettere l'operazione nel formato, ad es., \"3 + 4\"\n");
    return 0;
  }

  printf("Operazione da effettuare: %s %s %s\n", argv[1], argv[2], argv[3]);

  a = atof(argv[1]);
  b = atof(argv[3]);
  p = argv[2];  //  si noti l'uso del puntatore
  op = *p;      //  -> per prendere il primo carattere della string

  printf("                  ovvero: %f %c %f\n", a, op, b);

  switch(op) {
    case '+':
      ris = a + b;
      break;
    case '-':
      ris = a - b;
      break;
    case 'x':
    case '*':
      ris = a * b;
      break;
    case '/':
      ris = a / b;
      break;
    default:
      printf("operatore %c non riconosciuto\n", op);
      return 0;
  }
  printf("Risultato: %f %c %f = %f [%.2e]\n", a, op, b, ris, ris);

  return 0;
}
```

1.27 uso_for.c

```
// uso non banale del for  (e altre peculiarità del linguaggio))

#include <stdio.h>

int main() {
```

```
  int i, j, n;

  // normale sommatoria dei quadrati di i da 1 a 9
  n=0;
  for (i=1; i<=10; i++) n += i*i;
  printf("  Primo metodo: n = %d\n", n);

  for (i=1, n=0; i<=10; n+=i*i, i++) ;    // notare il solo punto e virgola!
  printf("Secondo metodo: n = %d\n", n);
                                          // provare a toglierlo...
  printf("(Ma l'ordine è importante: provare con \"i++, n+=i*i\"");
  for (i=1, n=0; i<=10; i++, n+=i*i) ;
  printf(" --> n = %d\n", n);
  puts("");

  // a proposito, "i++;" è diverso da "++i"
  i = 1;
  printf("i=1;  \"10 + i++\" --> %d\n", 10 + i++);
  i = 1;
  printf("i=1;  \"10 + ++i\" --> %d\n", 10 + ++i);
  puts("");

  // somma di i, finché è inferiore a 100
  // A) metodo 'normale'
  i = n = 0;            // <- altra peculiarità
  while (n < 100)
    n += i++;
  i -=2;  // era stato aumentato una volta di troppo!
  printf("i affinché Sum_i i < 100:  %d\n", i);

  // A) usando il for
  for(i=n=0; n < 100; n +=i, i++) ;
  i -= 2;
  printf("i affinché Sum_i i < 100:  %d\n", i);

  puts("\n Ma anche la leggibilità del codice ha un suo valore!");

  puts("\nAddendum (dopo la lezione): for() usato come fosse un while");
  for(i=n=0; 1; n +=i, i++) if(n>100) break;
  printf("i = %d; n = %d\n", i, n);
  puts(" -> Questo esempio serve anche a mostrare come la variabile");
  puts("    del loop esiste anche finito il for(), con l'ultimo incremento");

  return 0;
}
```

1.28 sin_to_file.c

```
/* primo esempio di scrittura su file */

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

# define N 1000

int main() {
  const float T = 1.0; // secondi
```

```
  const float pi = acos(-1.);
  const float omega = 2*pi/T;
  const float X0 = 1;  // cm
  const float tmax = 3*T;
  const float dt = tmax / N;
  FILE * fp;                    // puntatore al file
  int i;
  float t, x, v;

  printf("T = %.2f s; omega = %.2f s^-1; X0 = %.2f cm\n", T, omega, X0);

  fp = fopen("t_x_v.txt", "w");

  fprintf(fp, "t x v\n");
  for (i=0; i<N; i++) {
    t = i*dt;
    x = X0*cos(omega*t);
    v = omega*X0*cos(omega*t+pi/2);
    fprintf(fp, "%.3f %.3f %.3f\n", t, x, v);
  }

  fclose(fp);

  return 0;
}
```

1.29 lancio_2D.c

```
/* Soluzione numerica del problema del
   punto materiale lanciato con un certo angolo   */

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
  int step=0;
  float v0, theta, vx, vy, vxm, vym, ax, ay, dt;
  float vx0, vy0; // ci servono per la soluzione analitica
  float t=0, x = 0., y = 0.;
  const float grad2rad = acos(-1) / 180.;
  const float g = 9.8;   // m/s^2

  /* legge i parametri del lancio (v0 e theta e dt) da riga di comando

     ./lancio_2D  10  45 0.01

  */

  if (argc < 4) {
    puts("Devi dare anche velocità iniziale (m/s), angolo (gradi) e dt (s)");
    return 0;
  }

  v0 = atof(argv[1]);
  theta = atof(argv[2]);
  dt = atof(argv[3]);
  printf("Parametri del lancio: v0 = %.2f m/s, theta = %.0f gradi\n", v0, theta);
```

```
  printf("Discretizzazione: dt = %f s\n", dt);

  vx = vx0 = v0 * cos(theta * grad2rad);   // notare la doppia assegnazione
  vy = vy0 = v0 * sin(theta * grad2rad);
  printf("\nVelocità iniziale: (%.2f, %.2f) m/s\n", vx, vy);

  ax = 0.;    // anche se è nulla (può servire per altri casi)
  ay = -g;

  printf(" t=%.3f s; x=%.2f m, y=%.2f m;  vx=%.2f m/s, vy=%.2f m/s\n",
   t, x, y, vx, vy);
  printf("            x=%.2f m, y=%.2f m;  vx=%.2f m/s, vy=%.2f m/s\n",
      vx0 * t, vy0*t + ay*t*t / 2, vx0, vy0 + ay*t);

  while (y >= 0) {
    vxm = vx + ax * dt/2;  // velocita' media: (v + (v+a*dt))/2
    vym = vy + ay * dt/2;

    vx += ax * dt;         // velocita' dopo intervallo dt
    vy += ay * dt;

    x  += vxm * dt;        /* posizione dopo intervallo dt in cui
      il corpo si e' mosso con  v media vm  */
    y  += vym * dt;

    t  += dt;              // tempo dopo intervallo dt

    printf(" t=%.3f s; x=%.2f m, y=%.2f m;  vx=%.2f m/s, vy=%.2f m/s\n",
   t, x, y, vx, vy);
    // aggiungiamo anche la soluzione analitica
    printf("            x=%.2f m, y=%.2f m;  vx=%.2f m/s, vy=%.2f m/s\n",
        vx0 * t, vy0*t + ay*t*t / 2, vx0, vy0 + ay*t);
  }

  return 0;
}
```

1.30 lancio_2D_file.c

```
/* Soluzione numerica del problema del
   punto materiale lanciato con un certo angolo

   Variante (rispetto a lancio_2D.c) che scrive su file
*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define HEADER  "t x y vx vy"     // mette i nomi all'inizio delle variabili

int main(int argc, char *argv[]) {
  int step=0;
  float v0, theta, vx, vy, vxm, vym, ax, ay, dt;
  float t=0, x = 0., y = 0.;
  const float grad2rad = acos(-1) / 180.;
  const float g = 9.8;   // m/s^2
  FILE * fp;             // puntatore al file
```

```
  char nomeFile[50];     // vettore di caratteri per il nome del file

  /* legge i parametri del lancio (v0 e theta e dt) da riga di comando
        ./lancio_2D_file  10  45 0.01
  */

  if (argc < 4) {
    puts("Devi dare anche velocità iniziale (m/s), angolo (gradi) e dt (s)");
    return 0;
  }

  v0 = atof(argv[1]);
  theta = atof(argv[2]);
  dt = atof(argv[3]);
  printf("Parametri del lancio: v0 = %.2f m/s, theta = %.0f gradi\n", v0, theta);
  printf("Discretizzazione: dt = %f s\n", dt);

  vx = v0 * cos(theta * grad2rad);
  vy = v0 * sin(theta * grad2rad);
  printf("\nVelocità iniziale: (%.2f, %.2f) m/s\n", vx, vy);

  ax = 0.;    // anche se è nulla (può servire per altri casi)
  ay = -g;

  // apertura file
  sprintf(nomeFile, "lancio_%s_%s_%s.txt", argv[1], argv[2], argv[3]);
  printf("I dati andranno sul file %s\n", nomeFile);
  fp = fopen(nomeFile, "w");

  #ifdef HEADER
  printf("Header definito: %s\n", HEADER);
  fprintf(fp, "%s\n", HEADER);
  #endif

  fprintf(fp, "%.5f %.5f %.5f %.5f %.5f\n", t, x, y, vx, vy);

  while (y >= 0) {
    vxm = vx + ax * dt/2;  // velocita' media: (v + (v+a*dt))/2
    vym = vy + ay * dt/2;

    vx += ax * dt;         // velocita' dopo intervallo dt
    vy += ay * dt;

    x  += vxm * dt;        /* posizione dopo intervallo dt in cui
      il corpo si e' mosso con  v media vm  */
    y  += vym * dt;

    t  += dt;              // tempo dopo intervallo dt

    fprintf(fp, "%.5f %.5f %.5f %.5f %.5f\n", t, x, y, vx, vy);
  }

  close(fp);
  return 0;
}
```

1.31 lancio_2D_csv.c

```
/* Soluzione numerica del problema del
   punto materiale lanciato con un certo angolo

   Variante di lancio_2D_file.c per scrivere csv
*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define HEADER  "t, x, y, vx, vy"   // mette i nomi all'inizio delle variabili

int main(int argc, char *argv[]) {
  int step=0;
  float v0, theta, vx, vy, vxm, vym, ax, ay, dt;
  float t=0, x = 0., y = 0.;
  const float grad2rad = acos(-1) / 180.;
  const float g = 9.8;   // m/s^2
  FILE * fp;             // puntatore al file
  char nomeFile[50];     // vettore di caratteri per il nome del file

  /* legge i parametri del lancio (v0 e theta e dt) da riga di comando

       ./lancio_2D_csv  10  45 0.01
  */

  if (argc < 4) {
    puts("Devi dare anche velocità iniziale (m/s), angolo (gradi) e dt (s)");
    return 0;
  }

  v0 = atof(argv[1]);
  theta = atof(argv[2]);
  dt = atof(argv[3]);
  printf("Parametri del lancio: v0 = %.2f m/s, theta = %.0f gradi\n", v0, theta);
  printf("Discretizzazione: dt = %f s\n", dt);

  vx = v0 * cos(theta * grad2rad);
  vy = v0 * sin(theta * grad2rad);
  printf("\nVelocità iniziale: (%.2f, %.2f) m/s\n", vx, vy);

  ax = 0.;    // anche se è nulla (può servire per altri casi)
  ay = -g;

  // apertura file
  sprintf(nomeFile, "lancio_%s_%s_%s.csv", argv[1], argv[2], argv[3]);
  printf("I dati andranno sul file %s\n", nomeFile);
  fp = fopen(nomeFile, "w");

  #ifdef HEADER
  printf("Header definito: %s\n", HEADER);
  fprintf(fp, "%s\n", HEADER);
  #endif

  fprintf(fp, "%.5f, %.5f, %.5f, %.5f, %.5f\n", t, x, y, vx, vy);

  while (y >= 0) {
```

```
    vxm = vx + ax * dt/2;  // velocita' media: (v + (v+a*dt))/2
    vym = vy + ay * dt/2;

    vx += ax * dt;         // velocita' dopo intervallo dt
    vy += ay * dt;

    x  += vxm * dt;        /* posizione dopo intervallo dt in cui
      il corpo si e' mosso con  v media vm  */
    y  += vym * dt;

    t  += dt;              // tempo dopo intervallo dt

    fprintf(fp, "%.5f, %.5f, %.5f, %.5f, %.5f\n", t, x, y, vx, vy);
  }

  fclose(fp);
  return 0;
}
```

1.32 'Prova' del 10 dicembre

```
//============================================================
// somma_log2.c
#include <stdio.h>
#include <math.h>

#define N 10

int main() {
  int i;
  float s;

  for (i=1; i<=N; i++) {
    s = s + log(i)/log(2);
  }

  printf("s = %.2f\n", s);
}

//============================================================
// set_arrays.c
#include <stdio.h>


void setArray(int n, int *pA, int val) {
  int i;
  for (i=0; i<n; i++) {
    *(pA+i) = val;
  }

}

int main() {
  int i, j;
  int A[3][4] = {0};
  int *pA;
  pA = &A[0][0];
```

```
  for (i=0; i<3; i++) {
    for(j=0; j<4; j++) {
      printf("%3d ", A[i][j]);
    }
    printf("\n");
  }
   printf("\n");

  setArray(3*4, pA, 7);

  for (i=0; i<3; i++) {
    for(j=0; j<4; j++) {
      printf("%3d ", A[i][j]);
    }
    printf("\n");
  }
  printf("\n");

  /* la function setArray può essere utilizzata per settare
     allo stesso valore solo gli elementi di una stessa
     riga, in quanto in C gli elementi sono ordinati per riga

     Seguono esempi
  */

  setArray(4, pA, 1);
  setArray(4, &A[1][0], 2);
  setArray(4, &A[2][0], 3);

  for (i=0; i<3; i++) {
    for(j=0; j<4; j++) {
      printf("%3d ", A[i][j]);
    }
    printf("\n");
  }

  return 0;
}

//============================================================
// prova_string.c
#include <stdio.h>

int main(int argc, char *argv[]) {
  int n=0;
  char *p;
  const int diff = 'A' - 'a';

  if (argc < 2) {
    printf("Devi dare una sequenza di caratteri nella riga di comando\n");
    return 0;
  } else {
    printf("Sequenza ricevuta: %s\n", argv[1]);
  }

  p = argv[1];

  while(1) {
    if ( *(p+n) == 0)
```

```
      break;

    if ('a' <= *(p+n) && *(p+n) <= 'z') {
      *(p+n) += diff;
    } else if('A' <= *(p+n) && *(p+n) <= 'Z') {
      *(p+n) -= diff;
    }

    printf("%c", *(p+n));
    n++;
  }
  printf("\n");
  printf("(%d caratteri)\n", n);

  return 0;
}

//============================================================
// prova_bubble.c
#include <stdio.h>
#include <stdlib.h>

void bubbleSorting(int n, float v[]){
  int i, ind, swapped;
  float tmp;
  while(1) {
    swapped=0;
    for (i=0; i<(n-1); i++) {
      if (v[i] > v[i+1]){
        tmp = v[i];
        v[i] = v[i+1];
        v[i+1] = tmp;
        swapped++;
      }
    }
    if(!swapped) break;
  }
}

int main() {
  float v[10];
  int i;

  srand(123456789);     // dovrebbe essere il nr di matricola

  for (i=0; i<10; i++) {
    v[i] = rand() / (float) RAND_MAX;
    printf("%.3f ", v[i]);
  }
  printf("\n");

  bubbleSorting(10, v);
  for (i=0; i<10; i++) {
    printf("%.3f ", v[i]);
  }
  printf("\n");

  return 0;
}
```

1.33 Format variabile in printf (etc.)

```
//-------------------------------------
// cifre_pi.c
#include <stdio.h>
#include <math.h>

#define N 13

int main() {
  int i;
  char fmt[30];
  double pi = acos(-1);

  //  pi = acos(-1);
  for(i=0; i<N; i++) {
    sprintf(fmt, "%%.%df\n", i);
    printf(fmt, pi);
  }

  return 0;
}

//-------------------------------------------------------------------------
//  stampaMatriceDfmt.c
/*  stampa matrice di double, con formattazione
    di printf passata come argomento
    (vedi prova_sprintf.c)
*/

#include <stdio.h>
#include <stdlib.h>

#define NR 3
#define NC 4

//---------------------------------------------
float drunif(double min, double max) {
  float rzerouno;
  if(min >= max) return(0);
  rzerouno = (double) rand() / (double) RAND_MAX;
  return( min + rzerouno * (max-min) );
}

//-------------------------------------------------------------------------
void stampaMatriceDfmt(int nr, int nc, double A[nr][nc], char ftm[]) {
  int i, j;
  char formato[20];

  sprintf(formato, "%%%s ", ftm);
  printf("Formato: %s\n", formato);
  for(i=0; i<nr; i++) {
    for(j=0; j<nc; j++) {
      printf(formato, A[i][j]);
    }
    printf("\n");
  }
```

```
}

int main() {
  double A[3][4] = {0.};
  double *p;
  int i;

  p = &A[0][0];
  for (i=0; i < NR*NC; i++)  *(p+i) = drunif(0., 1.);

  stampaMatriceDfmt(NR, NC, A, ".2f");
  puts("");
  stampaMatriceDfmt(NR, NC, A, ".4f");
  puts("");
  stampaMatriceDfmt(NR, NC, A, ".3e");

  // moltiplichiamo tutti gli elementi per 100 e ristampiamo
  for (i=0; i < NR*NC; i++)  *(p+i) = *(p+i) * 100.;

  printf("\nValori moltiplicati per 100 (e format cambiati)\n");
  stampaMatriceDfmt(NR, NC, A, "8.2f");
  puts("");
  stampaMatriceDfmt(NR, NC, A, "8.4f");
  puts("");
  stampaMatriceDfmt(NR, NC, A, ".3e");

  return 0;
}
```

1.34 lancio_2D_aria.c

```
/* Soluzione numerica del problema del
   punto materiale lanciato con un certo angolo

   Variante (rispetto a lancio_2D_file.c):
     -> resistenza dell'aria modellizzata realisticamente
        dipendente dal quadrato della velocità
        (e non semplicemente dalla velocità)
           https://en.wikipedia.org/wiki/Drag_(physics)

      -> |F_A| = eta * v^2

   Per la logica vedere la soluzione numerica della molla:
     -> molla_xva_smorz.R
        (http://www.roma1.infn.it/~dagos/LabC/molla_1.html)

   Legge da riga di comando
      - i parametri fisici (m dell'oggetto e eta di resistenza)
      - i parametri del lancio (v0, theta)
      - il parametro di discretizzazione del tempo (dt)

        ./lancio_2D_aria_file m eta v0 theta ylim dt

   Es:
        ./lancio_2D  1.0 0.1 10 45 -1. 0.05

         (Att: ymin deve essere <= 0, altrimenti viene posto a 0)
```

```
*/


#include <stdio.h>
#include <math.h>
#include <stdlib.h>

// #define TO_FILE 1
#define HEADER  "t x y vx vy"     // mette i nomi all'inizio delle variabili

int main(int argc, char *argv[]) {
  int step=0;
  float v0, theta, vx, vy, vx0, vy0, vxm0, vym0, ax, ay, amx, amy, dt;
  float ymin;
  float Fx, Fy, m, v2, Fmx, Fmy, vm2;
  float t=0, x = 0., y = 0.;
  const float grad2rad = acos(-1) / 180.;
  const float g = 9.8;     // m/s^2
  float eta;               // N / (m/s)^2     -> da riga di comando
  int   toFile=0;          // 'flag' per scrittura su file
  FILE * fp;               // puntatore al file
  char nomeFile[50];       // vettore di caratteri per il nome del file
  char *p;                 // puntatore a carattere (-> argv[6])


  if (argc < 7) {
    puts("Devi dare anche: m eta v0 theta ymin dt");
    return 0;
  }

  m     = atof(argv[1]);
  eta   = atof(argv[2]);
  v0    = atof(argv[3]);
  theta = atof(argv[4]);
  ymin  = atof(argv[5]);
  if(ymin > 0) ymin = 0. ;
  dt    = atof(argv[6]);
  printf("Parametri fisici: m = %.2f kg,  eta = %.3f N / (m/s)^2\n", m, eta);
  printf("Parametri lancio: v0=%.2f m/s, theta=%.0f gradi\n", v0, theta);
  printf("Discretizzazione: dt = %.3f s\n", dt);

  vx = v0 * cos(theta * grad2rad);
  vy = v0 * sin(theta * grad2rad);
  printf("\nVelocità iniziale: (%.2f, %.2f) m/s\n", vx, vy);

  if (argc > 7) {
    p = argv[7];
    if (*p == 'f' || *p == 'F') {
      toFile=1;
    }
  }

  // apertura file
  if (toFile) {
    sprintf(nomeFile, "lancio_%s_%s_%s_%s_%s_%s.txt",
    argv[1], argv[2], argv[3], argv[4], argv[5], argv[6]);
    printf("I dati andranno sul file %s\n", nomeFile);
    fp = fopen(nomeFile, "w");
    #ifdef HEADER
```

```
  printf("Header definito: %s\n", HEADER);
  fprintf(fp, "%s\n", HEADER);
  #endif
  fprintf(fp, "%.5f %.5f %.5f %.5f %.5f\n", t, x, y, vx, vy);
} else {
  printf("%.5f %.5f %.5f %.5f %.5f\n", t, x, y, vx, vy);
}

while (y >= ymin) {
  // salviamo la velocità in t (all'inizio dell'intervallino dt)
  vx0 = vx;
  vy0 = vy;

  // quadrato della velocità e forza calcolata in t
  v2 = vx*vx + vy*vy;
  Fx = 0    - eta * v2 * vx/sqrt(v2);  // vx/sqrt(v2) : versore x della velocità
  Fy = -m*g - eta * v2 * vy/sqrt(v2);  // vy/sqrt(v2) : versore y della velocità

  // accelerazione calcolata in t
  ax = Fx / m;
  ay = Fy / m;

  // velocità media (di prima approssimazione) nell'intervallino dt
  // (calcolata dall'accelerazione in t)
  vxm0 = vx + ax * dt/2;  // velocita' media: (v + (v+a*dt))/2
  vym0 = vy + ay * dt/2;

  // riaggiorniamo la velocità quadra con quella media nell'intervallino
  // e ricalcoliamo forze e accelerazioni
  vm2  = vxm0*vxm0 + vym0*vym0;
  Fmx  = 0    - eta * vm2 * vxm0/sqrt(vm2);
  Fmy  = -m*g - eta * vm2 * vym0/sqrt(vm2);
  amx  = Fmx / m;                          // ax in t + dt/2
  amy  = Fmy / m;                          // ay in t + dt/2

  // velocità dopo l'intervallo dt
  vx += amx * dt;
  vy += amy * dt;

  // posizione dopo l'intervallino dt
  x  += (vx0+vx)/2. * dt;            /* coordinata x dopo l'intervallino dt,
                                        durante il quale il corpo si e' mosso
                                        con v media (vx0+vx)/2               */

  y  += (vy0+vy)/2 * dt;             // idem per la coordinata y

  t  += dt;                          // tempo dopo intervallo dt

  if (toFile) {
    fprintf(fp, "%.5f %.5f %.5f %.5f %.5f\n", t, x, y, vx, vy);
  } else {
    printf("%.5f %.5f %.5f %.5f %.5f\n", t, x, y, vx, vy);
  }
}

if (toFile) {
  fclose(fp);
}
```

```
  return 0;
}
```

1.35 Introduzione alle strutture

```
//=========================================================
// struttura_studenti.c
#include <stdio.h>
#include <string.h>

int main() {
  int i;

  struct Studente{
    char nome[30];
    char cognome[30];
    int annoNascita;
    int voto1;
    int voto2;
    int voto3;
  };

  struct Studente studente1;
  struct Studente studente2;

  //  studente1.nome = "Antonio";    <- NO !

  strcpy(studente1.nome, "Antonio");
  strcpy(studente1.cognome, "Persichetti");
  studente1.annoNascita = 1999;
  studente1.voto1 = 30;
  studente1.voto2 = 31;
  studente1.voto3 = 28;

  strcpy(studente2.nome, "Daniele");
  strcpy(studente2.cognome, "Zurlo");
  studente2.annoNascita = 1997;
  studente2.voto1 = 23;
  studente2.voto2 = 18;
  studente2.voto3 = 14;

  printf("%s %s (%d): %d %d %d \n",
         studente1.nome, studente1.cognome, studente1.annoNascita,
         studente1.voto1, studente1.voto2, studente1.voto3);

  printf("%s %s (%d): %d %d %d \n",
         studente2.nome, studente2.cognome, studente2.annoNascita,
         studente2.voto1, studente2.voto2, studente2.voto3);

  struct Studente altri[10];

  strcpy(altri[0].nome, "Giuseppe");
  strcpy(altri[0].cognome, "Garibaldi");
  altri[0].annoNascita = 1999;
  altri[0].voto1 = 23;
  altri[0].voto2 = 18;
  altri[0].voto3 = 14;
```

```
  strcpy(altri[1].nome, "Giuseppe");
  strcpy(altri[1].cognome, "Mazzini");
  altri[1].annoNascita = 1999;
  altri[1].voto1 = 18;
  altri[1].voto2 = 19;
  altri[1].voto3 = 20;

  strcpy(altri[2].nome, "Alessandro");
  strcpy(altri[2].cognome, "Manzoni");
  altri[2].annoNascita = 2000;
  altri[2].voto1 = 30;
  altri[2].voto2 = 30;
  altri[2].voto3 = 30;
puts("\nAltri:");
  for(i=0; i<3; i++) {
    printf("%s %s (%d): %d %d %d \n",
         altri[i].nome, altri[i].cognome, altri[i].annoNascita,
         altri[i].voto1, altri[i].voto2, altri[i].voto3);
  }

  puts("\nAltra stampa:");
  for(i=0; i<10; i++) {
    printf("%s %s (%d): %d %d %d \n",
         altri[i].nome, altri[i].cognome, altri[i].annoNascita,
         altri[i].voto1, altri[i].voto2, altri[i].voto3);
  }
}

//=====================================================================
// struttura_studenti_1.c
/* variante di struttura_studenti.c
   con vettori di voti
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main() {
  int i, s;

  struct Studente{
    char nome[30];
    char cognome[30];
    int annoNascita;
    int voti[30];
    int NrEsami;
  };

  struct Studente studenti[10];
  int NrStudenti = 0;

  strcpy(studenti[0].nome, "Antonio");
  strcpy(studenti[0].cognome, "Persichetti");
  studenti[0].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[1].nome, "Daniele");
  strcpy(studenti[1].cognome, "Zurlo");
  studenti[1].annoNascita = 1995;
```

```
  NrStudenti++;
  strcpy(studenti[2].nome, "Giuseppe");
  strcpy(studenti[2].cognome, "Garibaldi");
  studenti[2].annoNascita = 1997;
  NrStudenti++;
  strcpy(studenti[3].nome, "Giuseppe");
  strcpy(studenti[3].cognome, "Mazzini");
  studenti[3].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[4].nome, "Alessandro");
  strcpy(studenti[4].cognome, "Manzoni");
  studenti[4].annoNascita = 1997;
  NrStudenti++;

  for (s=0; s<NrStudenti; s++) {
    studenti[s].NrEsami = rand() % 25 + 5;
    for (i=0; i<studenti[s].NrEsami; i++) {
      studenti[s].voti[i] = rand() % 13 + 18;
    }
  }

  for (s=0; s<NrStudenti; s++) {
       printf("%s %s (%d):   %d esami superati\n",
              studenti[s].nome, studenti[s].cognome,
              studenti[s].annoNascita, studenti[s].NrEsami);
       printf("  Voti: ");
         for (i=0; i<studenti[s].NrEsami; i++) {
         printf("%d ", studenti[s].voti[i]);
       }
       printf("\n");
  }

  return 0;
}
//=================================================================
// struttura_studenti_2.c
/* variante di struttura_studenti_1.c
   passaggio a function della struttura
*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

//-----------------------------------------------------------
/* struttura definita all'esterno del main, in modo che sia
   riconosciuta anche altrove (nella function)  */
struct Studente {
  char nome[30];
  char cognome[30];
  int annoNascita;
  int voti[30];
  int NrEsami;
};

//-----------------------------------------------------------
float calcolaMedia(struct Studente stud) {
  int i, somma;
  for(i=somma=0; i<stud.NrEsami; somma+=stud.voti[i++]) ;
  return( (float) somma / (float) stud.NrEsami );
```

```
}

//-------------------------------------------------------------
int main() {
  int i, s;

  struct Studente studenti[10];
  int NrStudenti = 0;

  strcpy(studenti[0].nome, "Antonio");
  strcpy(studenti[0].cognome, "Persichetti");
  studenti[0].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[1].nome, "Daniele");
  strcpy(studenti[1].cognome, "Zurlo");
  studenti[1].annoNascita = 1995;
  NrStudenti++;
  strcpy(studenti[2].nome, "Giuseppe");
  strcpy(studenti[2].cognome, "Garibaldi");
  studenti[2].annoNascita = 1997;
  NrStudenti++;
  strcpy(studenti[3].nome, "Giuseppe");
  strcpy(studenti[3].cognome, "Mazzini");
  studenti[3].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[4].nome, "Alessandro");
  strcpy(studenti[4].cognome, "Manzoni");
  studenti[4].annoNascita = 1997;
  NrStudenti++;

  // mettiamo 'a caso' i voti
  for (s=0; s<NrStudenti; s++) {
    studenti[s].NrEsami = rand() % 25 + 5;
    for (i=0; i<studenti[s].NrEsami; i++) {
      studenti[s].voti[i] = rand() % 13 + 18;
    }
  }

  for (s=0; s<NrStudenti; s++) {
       printf("%s %s (%d):   %d esami superati\n",
              studenti[s].nome, studenti[s].cognome,
              studenti[s].annoNascita, studenti[s].NrEsami);
       printf("  Voti: ");
       for (i=0; i<studenti[s].NrEsami; i++) {
         printf("%d ", studenti[s].voti[i]);
       }
       printf(" [media %.1f]\n", calcolaMedia(studenti[s]) );
  }

  return 0;
}

//=============================================================
// struttura_studenti_3.c
/* variante di struttura_studenti_1.c
   passaaggio a function della struttura
*/

#include <stdio.h>
```

```
#include <string.h>
#include <stdlib.h>

//-------------------------------------------------------------
/* struttura definita all'esterno del main, in modo che sia
   riconosciuta anche altrove (nella function)  */
struct Studente {
  char nome[30];
  char cognome[30];
  int annoNascita;
  int voti[30];
  int NrEsami;
};

//-------------------------------------------------------------
float calcolaMedia(struct Studente *stud) {
  int i, somma;
  for(i=somma=0; i<stud->NrEsami; somma+=stud->voti[i++]) ;
  return( (float) somma / (float) stud->NrEsami );
}

//-------------------------------------------------------------
int main() {
  int i, s;

  struct Studente studenti[10];
  int NrStudenti = 0;

  strcpy(studenti[0].nome, "Antonio");
  strcpy(studenti[0].cognome, "Persichetti");
  studenti[0].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[1].nome, "Daniele");
  strcpy(studenti[1].cognome, "Zurlo");
  studenti[1].annoNascita = 1995;
  NrStudenti++;
  strcpy(studenti[2].nome, "Giuseppe");
  strcpy(studenti[2].cognome, "Garibaldi");
  studenti[2].annoNascita = 1997;
  NrStudenti++;
  strcpy(studenti[3].nome, "Giuseppe");
  strcpy(studenti[3].cognome, "Mazzini");
  studenti[3].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[4].nome, "Alessandro");
  strcpy(studenti[4].cognome, "Manzoni");
  studenti[4].annoNascita = 1997;
  NrStudenti++;

  // mettiamo 'a caso' i voti
  for (s=0; s<NrStudenti; s++) {
    studenti[s].NrEsami = rand() % 25 + 5;
    for (i=0; i<studenti[s].NrEsami; i++) {
      studenti[s].voti[i] = rand() % 13 + 18;
    }
  }

  for (s=0; s<NrStudenti; s++) {
       printf("%s %s (%d):   %d esami superati\n",
```

```
              studenti[s].nome, studenti[s].cognome,
              studenti[s].annoNascita, studenti[s].NrEsami);
       printf("  Voti: ");
       for (i=0; i<studenti[s].NrEsami; i++) {
         printf("%d ", studenti[s].voti[i]);
       }
       printf(" [media %.1f]\n", calcolaMedia(&studenti[s]) );
  }

  return 0;
}

//=========================================================
// struttura_studenti_4.c
/* Passare a una function una struttura in un argomento
   è più o meno equivalente a passare una normale variabile:
   se ne passa una copia e quindi quello che si fa su di essa
   all'interno della function non ha influenza sulla struttura
   della function chiamante
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

//-----------------------------------------------------------
/* struttura definita all'esterno del main, in modo che sia
   riconosciuta anche altrove (nella function)  */
struct Studente {
  char nome[30];
  char cognome[30];
  int annoNascita;
  int voti[30];
  int NrEsami;
};

//-----------------------------------------------------------
float AssegnaVoti(struct Studente stud) {
  int i;
  // mettiamo 'a caso' i voti
    stud.NrEsami = rand() % 25 + 5;
    for (i=0; i<stud.NrEsami; i++) {
      stud.voti[i] = rand() % 13 + 18;
    }
}

//-----------------------------------------------------------
float calcolaMedia(struct Studente stud) {
  int i, somma;
  for(i=somma=0; i<stud.NrEsami; somma+=stud.voti[i++]) ;
  return( (float) somma / (float) stud.NrEsami );
}

//-----------------------------------------------------------
int main() {
  int i, s;

  struct Studente studenti[10];
  int NrStudenti = 0;
```

```
  strcpy(studenti[0].nome, "Antonio");
  strcpy(studenti[0].cognome, "Persichetti");
  studenti[0].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[1].nome, "Daniele");
  strcpy(studenti[1].cognome, "Zurlo");
  studenti[1].annoNascita = 1995;
  NrStudenti++;
  strcpy(studenti[2].nome, "Giuseppe");
  strcpy(studenti[2].cognome, "Garibaldi");
  studenti[2].annoNascita = 1997;
  NrStudenti++;
  strcpy(studenti[3].nome, "Giuseppe");
  strcpy(studenti[3].cognome, "Mazzini");
  studenti[3].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[4].nome, "Alessandro");
  strcpy(studenti[4].cognome, "Manzoni");
  studenti[4].annoNascita = 1997;
  NrStudenti++;

  for(s=0; s<NrStudenti; s++) {
    AssegnaVoti(studenti[s]);
  }

  for (s=0; s<NrStudenti; s++) {
       printf("%s %s (%d):   %d esami superati\n",
              studenti[s].nome, studenti[s].cognome,
              studenti[s].annoNascita, studenti[s].NrEsami);
       printf("  Voti: ");
       for (i=0; i<studenti[s].NrEsami; i++) {
         printf("%d ", studenti[s].voti[i]);
       }
       printf(" [media %.1f]\n", calcolaMedia(studenti[s]) );
  }

  return 0;
}

//============================================================
// struttura_studenti_5.c
/* ... diverso è invece il caso se si passa il puntatore alla struttura
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

//------------------------------------------------------------
/* struttura definita all'esterno del main, in modo che sia
   riconosciuta anche altrove (nella function)  */
struct Studente {
  char nome[30];
  char cognome[30];
  int annoNascita;
  int voti[30];
  int NrEsami;
};
```

```
//-------------------------------------------------------------
float AssegnaVotiStruct(struct Studente *stud) {
  int i;
  // mettiamo 'a caso' i voti
  stud->NrEsami = rand() % 25 + 5;
  for(i=0; i<stud->NrEsami; i++)
    stud->voti[i] = rand() % 13 + 18;
}

//-------------------------------------------------------------
float calcolaMedia(struct Studente *stud) {
  int i, somma;
  for(i=somma=0; i<stud->NrEsami; somma+=stud->voti[i++]) ;
  return( (float) somma / (float) stud->NrEsami );
}

//-------------------------------------------------------------
int main() {
  int i, s;

  struct Studente studenti[10];
  int NrStudenti = 0;

  strcpy(studenti[0].nome, "Antonio");
  strcpy(studenti[0].cognome, "Persichetti");
  studenti[0].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[1].nome, "Daniele");
  strcpy(studenti[1].cognome, "Zurlo");
  studenti[1].annoNascita = 1995;
  NrStudenti++;
  strcpy(studenti[2].nome, "Giuseppe");
  strcpy(studenti[2].cognome, "Garibaldi");
  studenti[2].annoNascita = 1997;
  NrStudenti++;
  strcpy(studenti[3].nome, "Giuseppe");
  strcpy(studenti[3].cognome, "Mazzini");
  studenti[3].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[4].nome, "Alessandro");
  strcpy(studenti[4].cognome, "Manzoni");
  studenti[4].annoNascita = 1997;
  NrStudenti++;

  // mettiamo 'a caso' i voti
  for (s=0; s<NrStudenti; s++) {
     AssegnaVotiStruct(&studenti[s]);
  }

  for (s=0; s<NrStudenti; s++) {
       printf("%s %s (%d):   %d esami superati\n",
              studenti[s].nome, studenti[s].cognome,
              studenti[s].annoNascita, studenti[s].NrEsami);
       printf("  Voti: ");
       for (i=0; i<studenti[s].NrEsami; i++) {
         printf("%d ", studenti[s].voti[i]);
       }
```

```
      printf(" [media %.1f]\n", calcolaMedia(&studenti[s]) );
  }

  return 0;
}

//=========================================================
// struttura_studenti_6.c
/* Ma ovviamente si può anche passare l'intero vettore
   di strutture (equivalente a passare un vettore di
   interi o di float o altro) e la cosa funziona:
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

//---------------------------------------------------------
/* struttura definita all'esterno del main, in modo che sia
   riconosciuta anche altrove (nella function)  */
struct Studente {
  char nome[30];
  char cognome[30];
  int annoNascita;
  int voti[30];
  int NrEsami;
};

//---------------------------------------------------------
float AssegnaVoti(int NrStudenti, int n, struct Studente stud[]) {
  int i, s;
  // mettiamo 'a caso' i voti
  for(s=0; s<NrStudenti; s++) {
    stud[s].NrEsami = rand() % 25 + 5;
    for (i=0; i<stud[s].NrEsami; i++) {
      stud[s].voti[i] = rand() % 13 + 18;
    }
  }
}

//---------------------------------------------------------
float calcolaMedia(struct Studente stud) {
  int i, somma;
  for(i=somma=0; i<stud.NrEsami; somma+=stud.voti[i++]) ;
  return( (float) somma / (float) stud.NrEsami );
}

//---------------------------------------------------------
int main() {
  int i, s;
  struct Studente studenti[10];
  int NrStudenti = 0;

  strcpy(studenti[0].nome, "Antonio");
  strcpy(studenti[0].cognome, "Persichetti");
  studenti[0].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[1].nome, "Daniele");
  strcpy(studenti[1].cognome, "Zurlo");
```

```
  studenti[1].annoNascita = 1995;
  NrStudenti++;
  strcpy(studenti[2].nome, "Giuseppe");
  strcpy(studenti[2].cognome, "Garibaldi");
  studenti[2].annoNascita = 1997;
  NrStudenti++;
  strcpy(studenti[3].nome, "Giuseppe");
  strcpy(studenti[3].cognome, "Mazzini");
  studenti[3].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[4].nome, "Alessandro");
  strcpy(studenti[4].cognome, "Manzoni");
  studenti[4].annoNascita = 1997;
  NrStudenti++;

  AssegnaVoti(NrStudenti, 10, studenti);

  for (s=0; s<NrStudenti; s++) {
       printf("%s %s (%d):   %d esami superati\n",
              studenti[s].nome, studenti[s].cognome,
              studenti[s].annoNascita, studenti[s].NrEsami);
       printf("  Voti: ");
       for (i=0; i<studenti[s].NrEsami; i++) {
         printf("%d ", studenti[s].voti[i]);
       }
       printf(" [media %.1f]\n", calcolaMedia(studenti[s]) );
  }

  return 0;
}
```

1.36 pozzo.c

```
/*  soluzione iterativa della profondità del pozzo
    dal tempo di discesa più di ritorno del suono

    Nota: in un problema fisico è inutile esagerare
          con il numero di cifre!!
*/

#include <stdio.h>
#include <math.h>

int main() {
  const double  g = 9.8;         // m/s^2
  const double vs = 340;         // m/s
  const double tm = 3;           // s    tempo misurato
  const double epsilon = 1e-11;  /* s    (precisione assurda dal punto di vista fisico,
                                          ma solo per testare la velocità dell'algoritmo)
  double tc;                     //      tempo 'vero' di caduta
  double b, c, s, s0, s1, s2, sa;
  int i;

  tc = tm;
  i=0;
  printf("i = %2d,  s = %.12f\n", i, g*tc*tc/2.);
```

```
  do {
    i++;
    s0 = g*tc*tc/2.;
    tc = tm - s0 / vs;
    s = g*tc*tc/2.;
    printf("i = %2d,  s = %.12f\n", i, s);
  } while (fabs(s-s0) > epsilon);

  printf("\nSoluzione equazioni (s): ");
  b = - 2. * ( tm *vs + vs*vs/g);
  c = pow(tm*vs, 2);
  s1 = -b/2 - sqrt(  pow(b/2, 2) - c );
  s2 = -b/2 + sqrt(  pow(b/2, 2) - c);
  printf("%.12f m;  %.f m (soluzione spuria)\n", s1, s2);

  return 0;
}
```

1.37 n_unif.c

```
// Nota: usa random_lib.c

#include <stdio.h>
#include <stdlib.h>
#include "random_lib.h"

void Nirunif(int N, int v[N], int min, int max) {
  int i;
  for(i=0; i<N; i++) v[i] = irunif(min, max);
}

int main() {
  int i=0, n=20, v[20] = {};

  setRandomSeed(0);

  Nirunif(n, v, 1, 6);

  for(i=0; i<n; i++) printf("%2d", v[i]);
  puts("");

  return 0;
}
```

1.38 sampleI.c

```
#include <stdio.h>
#include <stdlib.h>
#include "random_lib.h"

void sampleI(int n, int v[n])  {
  int i, ind, tmp;

  for (i=0; i<n; i++) {
    ind = irunif(i, n-1);
    tmp = v[ind];
```

```
    v[ind] = v[i];
    v[i] = tmp;
  }

}

int main() {
  int i, n=10, v[] = {1,2,3,4,5,6,7,8,9,10};

  setRandomSeed(0);

  for(i=0; i<n; i++) printf("%3d", v[i]);
  puts("");

  sampleI(n, v);

  for(i=0; i<n; i++) printf("%3d", v[i]);
  puts("");

  return 0;
}
```

1.39 sqrt_babilonesi.c

```
/*  Calcolo iterativo di radice quadrata di numeri positivi

    1)  se x > sqrt(N) ->        x^2 > N
                           x^2 + x^2 > N + x^2
                              2*x^2  > N + x^2
                                 2*x > N/x + x
                                   x > (x + N/x) / 2

      Ovvero, se x > sqrt(N), x è maggiore della media aritmetica
                              fra x stesso e N/x

    2) Ma una media aritmetica di due numeri positivi è sempre
       maggiore della media geometrica, ovvero
                   (x + N/x) / 2  > sqrt( x * N/x ) = sqrt(N)


    1)+2):   sqrt(N) < (x + N/x) / 2 < x

    3) Ma, se chiamiamo x1 = (x + N/x) / 2, essendo > sqrt(N),
       segue

             sqrt(N) < (x1 + N/x1) / 2 < x1

    Etc. etc.

      => si può cominciare con x=N

 */


#include <stdio.h>
#include <math.h>
#include <stdlib.h>
```

```
int main(int argc, char *argv[]) {
  double x, N;
  int i;

  if (argc < 2) {
    printf("Sintassi:  ./sqrt_babilonesi N\n");
    return 0;
  }
  N = atof(argv[1]);

  x = N;                              // ma si può anche
                                      // partire da un numero >> N  (!!)
  printf("i=%2d: %.17f\n", 0, x);
  for (i=1; i<=10; i++) {
    x = (x+N/x) / 2.;
    printf("i=%2d: %.17f\n", i, x);
  }
  printf("\nCon sqrt(): %.17f\n", sqrt(N));

  printf("\nCon bc:     1.41421356237309504880\n");
  printf("\[ echo 'sqrt(2)' |  bc -l ]\n");

  return 0;
}
```

1.40 Panoramica di funzioni I/O

```
//=====================================================0
// prova_fgetc_putchar.c
/*
    prove di fgetc() per leggere da file
    un carattere alla volta, successivamente
    mostrato sul terminale ('stampato') con putchar()
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
  FILE *fp;
  char c, pi[1010];
  int i, nc;

  fp = fopen("pi1000.txt", "r");

  nc=0;
  while( ( c = fgetc(fp) ) != EOF) {
    pi[nc++] = c;
    putchar(c);
  }

  fclose(fp);

  printf("letti %d caratteri\n", nc);
```

```
  return 0;
}

//=====================================================0
// fgets_fputs.c
/*
    lettura del solito file di pi greco con fgets()
    e scrittura su altro file con fputs()
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]) {
  FILE *fps, *fpd;
  char pi[1010];

  if (argc<3) {
    printf("devi dare, in ordine, i nomi file sorgente e destinazione\n");
    return 0;
  }

  fps = fopen(argv[1], "r");
  fpd = fopen(argv[2], "w");

  fgets(pi, 1010, fps);

  printf("\nLetti %lu caratteri\n", strlen(pi));

  fputs(pi, fpd);

  fclose(fps);
  fclose(fpd);

  return 0;
}

//=====================================================0
// copia_file.c
/*
    prove di fgetc() e fputc() per copiare un file su un altro
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]) {
  FILE *fps, *fpd;
  char c;
  int nc;

  if (argc<3) {
    printf("devi dare, in ordine, i nomi file sorgente e destinazione\n");
    return 0;
  }

  fps = fopen(argv[1], "r");
```

```
  fpd = fopen(argv[2], "w");

  nc=0;
  while( ( c = fgetc(fps) ) != EOF) {
    fputc(c, fpd);
    nc++;
  }
  printf("\nCopiati %d caratteri\n", nc);

  fclose(fps);
  fclose(fpd);
  return 0;
}

//=================================================0
// prova_getchar2file1.c
/*
    Prova di uso di getchar() e putc()

    Legge un testo da tastiera (anche su più righe)
    e lo scrive su un file

    Il testo è immagazzinato su un file

    Attenzione: MANCA il controllo sul numero di caratteri che
                si stanno leggendo e immagazzinando in v[]
                (ma per piccole prove va bene)
 */


#include <stdio.h>
#include <stdlib.h>

int main() {
  int i, nc=0;
   char c, v[1000];
   char nomeFile[50];
   FILE *fp;

   printf("Dai il nome del file su cui andare a inserire quanto scrivi: ");
   scanf("%s", nomeFile);
   printf("Scriverò su %s ", nomeFile);
   printf("cancellando il contenuto precedente se il file esisteva (!)\n");

   printf("scrivi dei caratteri e termina con Ctrl-d ('EOF')\n");
   while ( ( c = getchar() ) != EOF) {
     v[nc] = c;
     nc++;
   }
   v[nc] = 0;
   printf("\n %d caratteri letti\n", nc);

   fp = fopen(nomeFile, "w");
   for (i=0; i<nc; i++) {
     putc(v[i], fp);
   }
   printf("\n %d caratteri scritti\n", nc);
   fclose(fp);
```

```
   return 0;
}

//=====================================================0
// prova_getchar2file2.c
/* Variante di prova_getchar2file1.c
   nella quale la scrittura sul file viene fatta
   in un solo colpo con fputs()

   (Attenzione: vale l'avvertenza messa nell'altro file
    sulla mancanza del controllo del numero di caratteri
    immagazzinati in v[])
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
  int i, nc=0;
   char c, v[1000];
   char nomeFile[50];
   FILE *fp;

   printf("Dai il nome del file su cui andare a inserire quanto scrivi: ");
   scanf("%s", nomeFile);
   printf("Scriverò su %s ", nomeFile);
   printf("cancellando il contenuto precedente se il file esisteva (!)\n");

   printf("scrivi dei caratteri e termina con Ctrl-d ('EOF')\n");
   while ( ( c = getchar() ) != EOF) {
     v[nc] = c;
     nc++;
   }
   v[nc] = 0;
   printf("\n %d caratteri letti\n", nc);

   fp = fopen(nomeFile, "w");
   fputs(v, fp);
   printf("\n %lu caratteri scritti\n", strlen(v));
   fclose(fp);

   return 0;
}
```

1.41 Scrittura/rilettura di file binari

```
//============================================================
// scrivi_struttura.c
/*
    come scrivere il vettore di struttura su file binario
    mediante fwrite()

    (basato su struttura_studenti_6.c)

    -> Andare direttamente in fondo ('Parte Nuova')
```

```
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

//-------------------------------------------------------------
/* struttura definita all'esterno del main, in modo che sia
   riconosciuta anche altrove (nella function)  */

struct Studente {
  char nome[30];
  char cognome[30];
  int annoNascita;
  int voti[30];
  int NrEsami;
};

//-------------------------------------------------------------
float AssegnaVoti(int NrStudenti, int n, struct Studente stud[]) {
  int i, s;
  // mettiamo 'a caso' i voti
  for(s=0; s<NrStudenti; s++) {
    stud[s].NrEsami = rand() % 25 + 5;
    for (i=0; i<stud[s].NrEsami; i++) {
      stud[s].voti[i] = rand() % 13 + 18;
    }
  }
}

//-------------------------------------------------------------
float calcolaMedia(struct Studente stud) {
  int i, somma;
  for(i=somma=0; i<stud.NrEsami; somma+=stud.voti[i++]) ;
  return( (float) somma / (float) stud.NrEsami );
}

//-------------------------------------------------------------
int main() {
  int i, s;
  struct Studente studenti[10];
  int NrStudenti = 0;

  strcpy(studenti[0].nome, "Antonio");
  strcpy(studenti[0].cognome, "Persichetti");
  studenti[0].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[1].nome, "Daniele");
  strcpy(studenti[1].cognome, "Zurlo");
  studenti[1].annoNascita = 1995;
  NrStudenti++;
  strcpy(studenti[2].nome, "Giuseppe");
  strcpy(studenti[2].cognome, "Garibaldi");
  studenti[2].annoNascita = 1997;
  NrStudenti++;
  strcpy(studenti[3].nome, "Giuseppe");
  strcpy(studenti[3].cognome, "Mazzini");
  studenti[3].annoNascita = 1996;
```

```
  NrStudenti++;
  strcpy(studenti[4].nome, "Alessandro");
  strcpy(studenti[4].cognome, "Manzoni");
  studenti[4].annoNascita = 1997;
  NrStudenti++;

  AssegnaVoti(NrStudenti, 10, studenti);

  for (s=0; s<NrStudenti; s++) {
       printf("%s %s (%d):   %d esami superati\n",
              studenti[s].nome, studenti[s].cognome,
              studenti[s].annoNascita, studenti[s].NrEsami);
       printf("  Voti: ");
       for (i=0; i<studenti[s].NrEsami; i++) {
         printf("%d ", studenti[s].voti[i]);
       }
       printf(" [media %.1f]\n", calcolaMedia(studenti[s]) );
  }

  //-------------------------------------------------------
  //  Parte nuova con la scrittura su file del vettore di strutture

  FILE *fp;
  char NomeFile[] = "DataBaseStudenti.bin";

    printf("\nVado a scrivere i dati su %s\n", NomeFile);
  if ( (fp = fopen(NomeFile, "wb") ) == NULL) {
    // Questa volta mettiamo anche un controllo
    // per intercettare eventuali errori ad aprire il file
       printf("Errore nell'apertura del file!\n");
       exit(1);
   }

  for (s=0; s<NrStudenti; s++) {
    fwrite(&studenti[s], sizeof(struct Studente), 1, fp);
  }
  //                                          '1' : un elemento alla volta

  fclose(fp);
  return 0;
}

//=====================================================================
// rileggi_struttura.c
*
    rilettura mediante fread() del file binario
    scritto con scrivi_struttura.c

*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

//-------------------------------------------------------------
/* struttura definita all'esterno del main, in modo che sia
   riconosciuta anche altrove (nella function)  */

struct Studente {
```

```
  char nome[30];
  char cognome[30];
  int annoNascita;
  int voti[30];
  int NrEsami;
};

//-------------------------------------------------------------
float calcolaMedia(struct Studente stud) {
  int i, somma;
  for(i=somma=0; i<stud.NrEsami; somma+=stud.voti[i++]) ;
  return( (float) somma / (float) stud.NrEsami );
}

//-------------------------------------------------------------
int main() {
  int i, s;
  struct Studente studenti[10];
  int NrStudenti = 0;

  //-----------------------------------------------------------------
  // Rilettura dei dati
  //
  FILE *fp;
  char NomeFile[] = "DataBaseStudenti.bin";

  printf("\nVado a leggere i dati da %s\n", NomeFile);
  if ( (fp = fopen(NomeFile, "rb") ) == NULL) {
    // Rimettiamo il controllo
    // per intercettare eventuali errori ad aprire il file
       printf("Errore nell'apertura del file!\n");
       exit(1);
   }

  NrStudenti = 0;
  // prova a leggere fino a un massimo di 10 elementi
  for (s=0; s<10; s++) {
     if( fread(&studenti[s], sizeof(struct Studente), 1, fp) != 1) break;
     NrStudenti++;
  }
  fclose(fp);

  printf("\nHo riletto i dati di %d studenti\n\n", NrStudenti);

  //---------------------------------------------------------------
  // Controllo del contenuto
  //
  for (s=0; s<NrStudenti; s++) {
       printf("%s %s (%d):   %d esami superati\n",
              studenti[s].nome, studenti[s].cognome,
              studenti[s].annoNascita, studenti[s].NrEsami);
       printf("  Voti: ");
       for (i=0; i<studenti[s].NrEsami; i++) {
         printf("%d ", studenti[s].voti[i]);
       }
       printf(" [media %.1f]\n", calcolaMedia(studenti[s]) );
  }

  return 0;
```

```
}

//=====================================================================
// rileggi_struttura_tutto.c
/*
    rilettura mediante fread() del file binario
    scritto con scrivi_struttura.c

*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

//-------------------------------------------------------------
/* struttura definita all'esterno del main, in modo che sia
   riconosciuta anche altrove (nella function)  */

struct Studente {
  char nome[30];
  char cognome[30];
  int annoNascita;
  int voti[30];
  int NrEsami;
};

//-------------------------------------------------------------
float calcolaMedia(struct Studente stud) {
  int i, somma;
  for(i=somma=0; i<stud.NrEsami; somma+=stud.voti[i++]) ;
  return( (float) somma / (float) stud.NrEsami );
}

//-------------------------------------------------------------
int main() {
  int i, s;
  struct Studente studenti[10];
  int NrStudenti = 0;

  //-----------------------------------------------------------------
  // Rilettura dei dati
  //
  FILE *fp;
  char NomeFile[] = "DataBaseStudenti.bin";

  printf("\nVado a leggere i dati da %s\n", NomeFile);
  if ( (fp = fopen(NomeFile, "rb") ) == NULL) {
    // Rimettiamo il controllo
    // per intercettare eventuali errori ad aprire il file
       printf("Errore nell'apertura del file!\n");
       exit(1);
   }

  NrStudenti = 0;

  // proviamo a leggere in un solo fino a 10 studenti
  NrStudenti = fread(studenti, sizeof(struct Studente), 10, fp);
  fclose(fp);
```

```
  printf("\nHo riletto i dati di %d studenti\n\n", NrStudenti);

  if(NrStudenti == 0) {
    printf("\nUhm, qualcosa non ha funzionato... \n");
    exit(1);
  }

  //-------------------------------------------------------------
  // Controllo del contenuto
  //
  for (s=0; s<NrStudenti; s++) {
       printf("%s %s (%d):   %d esami superati\n",
              studenti[s].nome, studenti[s].cognome,
              studenti[s].annoNascita, studenti[s].NrEsami);
       printf("  Voti: ");
       for (i=0; i<studenti[s].NrEsami; i++) {
         printf("%d ", studenti[s].voti[i]);
       }
       printf(" [media %.1f]\n", calcolaMedia(studenti[s]) );
  }

  return 0;
}

//=============================================================
// scrivi_struttura_tutto.c
/*
    come scrivere il vettore di struttura su file binario
    mediante fwrite()

    variante di scrivi_struttura.c, in cui scriviamo
    i NrStudenti in un solo colpo

     file: DataBaseStudenti_1.bin

    -> Andare direttamente in fondo ('Parte Nuova')

*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

//-------------------------------------------------------------
/* struttura definita all'esterno del main, in modo che sia
   riconosciuta anche altrove (nella function)  */

struct Studente {
  char nome[30];
  char cognome[30];
  int annoNascita;
  int voti[30];
  int NrEsami;
};

//-------------------------------------------------------------
float AssegnaVoti(int NrStudenti, int n, struct Studente stud[]) {
  int i, s;
  // mettiamo 'a caso' i voti
```

```
  for(s=0; s<NrStudenti; s++) {
    stud[s].NrEsami = rand() % 25 + 5;
    for (i=0; i<stud[s].NrEsami; i++) {
      stud[s].voti[i] = rand() % 13 + 18;
    }
  }
}

//-------------------------------------------------------------
float calcolaMedia(struct Studente stud) {
  int i, somma;
  for(i=somma=0; i<stud.NrEsami; somma+=stud.voti[i++]) ;
  return( (float) somma / (float) stud.NrEsami );
}

//-------------------------------------------------------------
int main() {
  int i, s;
  struct Studente studenti[10];
  int NrStudenti = 0;

  strcpy(studenti[0].nome, "Antonio");
  strcpy(studenti[0].cognome, "Persichetti");
  studenti[0].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[1].nome, "Daniele");
  strcpy(studenti[1].cognome, "Zurlo");
  studenti[1].annoNascita = 1995;
  NrStudenti++;
  strcpy(studenti[2].nome, "Giuseppe");
  strcpy(studenti[2].cognome, "Garibaldi");
  studenti[2].annoNascita = 1997;
  NrStudenti++;
  strcpy(studenti[3].nome, "Giuseppe");
  strcpy(studenti[3].cognome, "Mazzini");
  studenti[3].annoNascita = 1996;
  NrStudenti++;
  strcpy(studenti[4].nome, "Alessandro");
  strcpy(studenti[4].cognome, "Manzoni");
  studenti[4].annoNascita = 1997;
  NrStudenti++;

  AssegnaVoti(NrStudenti, 10, studenti);

  for (s=0; s<NrStudenti; s++) {
       printf("%s %s (%d):   %d esami superati\n",
              studenti[s].nome, studenti[s].cognome,
              studenti[s].annoNascita, studenti[s].NrEsami);
       printf("  Voti: ");
       for (i=0; i<studenti[s].NrEsami; i++) {
         printf("%d ", studenti[s].voti[i]);
       }
       printf(" [media %.1f]\n", calcolaMedia(studenti[s]) );
  }

  //-------------------------------------------------------------
  //  Parte nuova con la scrittura su file del vettore di strutture

  FILE *fp;
```

```
  char NomeFile[] = "DataBaseStudenti_1.bin";

  printf("\nVado a scrivere i dati su %s\n", NomeFile);
  if ( (fp = fopen(NomeFile, "wb") ) == NULL) {
    // Questa volta mettiamo anche un controllo
    // per intercettare eventuali errori ad aprire il file
       printf("Errore nell'apertura del file!\n");
       exit(1);
   }

  fwrite(studenti, sizeof(struct Studente), NrStudenti, fp);

  fclose(fp);
  return 0;
}
```

1.42 Crivello di Eratostene, anche con scrittura su file e allocazione dinamica

```
//============================================================
// eratostene.c
/*  implementazione del crivello di Eratostene
    per la ricerca dei numeri primi
*/

#include <stdio.h>
#include <math.h>

#define N 1000

//-------------------------------------------------
void rimuoviZeriI(int *pn, int v[]) {
  int i, nnz = 0;

  for (i=0; i < *pn; i++) {
    if (v[i] != 0) {
      v[nnz++] = v[i];
    }
  }
  *pn = nnz;
}

//-------------------------------------------------
int main() {
  int i, ipr, pr, nPrimi=0, nMax, n=N, v[N];

  for(i=0; i<n; i++) v[i]=i+1;

  ipr  = 1;                 // indice del primo numero primo != 1  (-> 2)
  nMax = floor( sqrt(n));   // massimo nr con cui provare
  while (ipr < nMax) {
    pr = v[ipr];
    if(pr != 0) {           // se non è già stato cancellato è primo
      for(i=(ipr+1); i<n; i++){
        if( !(v[i]%pr) ) {
          v[i] = 0;
        }
```

```
      }
    }
    ipr++;
  }

  rimuoviZeriI(&n, v);

  for(i=0; i<n; i++) {
    if(v[i] !=0) {
      nPrimi++;
      printf("%d ", v[i]);
    }
  }
  puts("\n");
  printf("Totale numeri primi trovati: %d (Max %d)\n", nPrimi, v[n-1]);

  return 0;
}

//=================================================================
// eratostene_file.c
/*  implementazione del crivello di Eratostene
    per la ricerca dei numeri primi, con scrittura su file
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//-------------------------------------------------
void rimuoviZeriI(int *pn, int v[]) {
  int i, nnz = 0;

  for (i=0; i < *pn; i++) {
    if (v[i] != 0) {
      v[nnz++] = v[i];
    }
  }
  *pn = nnz;
}

//-------------------------------------------------
int main(int argc, char *argv[]) {
  int i, ipr, pr, nPrimi=0, nMax, n;
  char nomeFile[50];
  FILE * fp;

  if (argc < 2) {
    printf("\n  devi dare il nr max a riga di comando\n\n");
    return 0;
  }
  n = atoi(argv[1]);

  int v[n];

  sprintf(nomeFile, "numeri_primi_%d.txt", n);
  printf("I numeri primi andrannno scritti in %s\n", nomeFile);
  fp = fopen(nomeFile, "w");
```

```
  for(i=0; i<n; i++) v[i]=i+1;

  ipr  = 1;                 // indice del primo numero primo != 1  (-> 2)
  nMax = floor( sqrt(n));   // massimo nr con cui provare
  while (ipr < nMax) {
    pr = v[ipr];
    if(pr != 0) {           // se non è già stato cancellato è primo
      for(i=(ipr+1); i<n; i++){
        if( !(v[i]%pr) ) {
          v[i] = 0;
        }
      }
    }
    ipr++;
  }

  rimuoviZeriI(&n, v);

  for(i=0; i<n; i++) {
    if(v[i] !=0) {
      nPrimi++;
      fprintf(fp, "%d\n", v[i]);
    }
  }

  printf("Totale numeri primi trovati: %d (max %d)\n", nPrimi, v[nPrimi-1]);

  fclose(fp);
  return 0;
}

//================================================================
// eratostene_file_mall.c
*  implementazione del crivello di Eratostene
    per la ricerca dei numeri primi

    -> scrittura su file
    -> vettore riservato con malloc()

*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//-----------------------------------------------
void rimuoviZeriI(int *pn, int *ptr) {
  int i, nnz = 0;

  for (i=0; i < *pn; i++) {
    if (*(ptr+i) != 0) {
      *(ptr+nnz++) = *(ptr+i);
    }
  }
  *pn = nnz;
}

//-----------------------------------------------
int main(int argc, char *argv[]) {
```

```
  int i, ipr, pr, nPrimi=0, nMax, n;
  char nomeFile[50];
  FILE * fp;
  int *ptr;

  if (argc < 2) {
    printf("\n  devi dare il nr max a riga di comando\n\n");
    return 0;
  }
  n = atoi(argv[1]);

  sprintf(nomeFile, "numeri_primi_%d.txt", n);
  printf("I numeri primi andrannno scritti in %s\n", nomeFile);
  fp = fopen(nomeFile, "w");

  printf(" %d interi saranno allocati dinamicamente \n", n);
  ptr = (int*) malloc(n * sizeof(int));
                                        // v[i] -> *(ptr+i)
  for(i=0; i<n; i++) *(ptr+i) = i+1;

  // for(i=0; i<n; i++) printf("%d ", *(ptr+i) );
  // puts(" ");

  ipr  = 1;                 // indice del primo numero primo != 1  (-> 2)
  nMax = floor( sqrt(n));   // massimo nr con cui provare
  while (ipr < nMax) {
    pr = *(ptr+ipr) ;
    if(pr != 0) {           // se non è già stato cancellato è primo
      for(i=(ipr+1); i<n; i++){
        if( !( *(ptr+i)  % pr) ) {
          *(ptr+i)  = 0;
        }
      }
    }
    ipr++;
  }

  // for(i=0; i<n; i++) printf("%d ", *(ptr+i) );
  // puts(" ");

  rimuoviZeriI(&n, ptr);

  for (i=0; i<n; i++) {
    if (*(ptr+i)  !=0) {
      nPrimi++;
      fprintf(fp, "%d\n", *(ptr+i));
    }
  }

  printf("Totale numeri primi trovati: %d (max %d)\n",
         nPrimi, *(ptr+nPrimi-1));

  free( ptr );   // IMPORTANTE liberare lo spazio allocato con malloc() !
  fclose(fp);    // e anche 'chiudere' il file

  return 0;
}
```