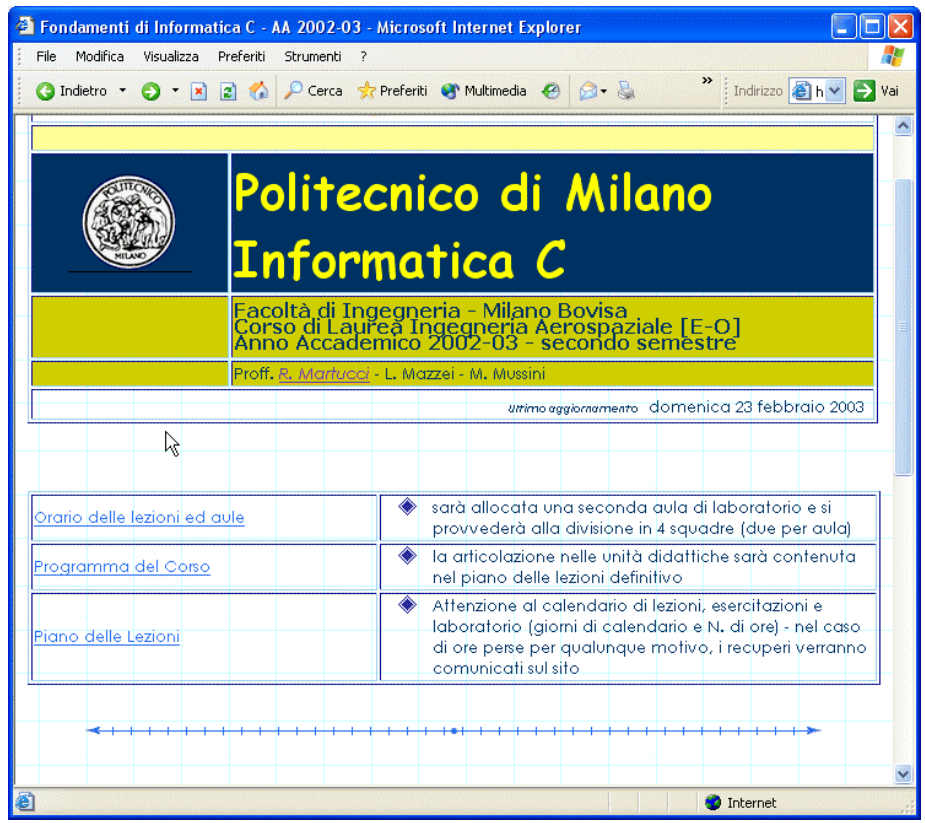
http://www.elet.polimi.it/upload/martucci/index.html

- Lucidi del Corso di Informatica C Aerospaziali
- AA 2002-03
- 1° Modulo
- Es. sugli algoritmi



Esempio 6

Si vuole leggere una sequenza di N valori interi in ingresso (con N qualsiasi) e trovare il valore massimo all'interno della sequenza e la sua posizione. Il valore e la posizione devono essere visualizzati in uscita.

variabili:

`N`	numero di elementi da acquisire
`elemento`	valore dell'elemento letto
`max`	valore del massimo
`posmax`	posizione del massimo all'interno della sequenza
`i`	*`indice corrente dell'elemento letto`*

Algoritmo:

1. acquisizione di N (e verifica dell'accettabilità)
2. acquisizione del primo valore
3. il valore è considerato il massimo attuale
4. da ripetersi finché non si sono acquisiti tutti gli N valori:
   acquisizione del prox valore: se è maggiore del massimo attuale
   aggiornamento del valore del max e della sua posizione
5. visualizzazione del valore del massimo e della sua posizione.

Esempio 6: pseudo-codice

```
main ()

{
int N;              numero di elementi da acquisire
int elemento;       valore dell'elemento letto
int max;            valore del massimo
int posmax;         posizione del massimo all'interno della sequenza
int i;              indice corrente relativo al numero di elementi già letti

leggi (N);
if (N è un valore accettabile)
  {
     leggi (elemento);
      i = 1; /* aggiorna l'indice corrente */
      considera questo elemento come massimo attuale

    /* ciclo acquisizione e ricerca massimo per i  rimanenti
valori */

    while (non si sono letti N valori)
    {
     leggi (elemento);
      i = i +1; /* aggiorna l'indice corrente */
      if (l'elemento letto è maggiore del massimo attuale)
          {
           considera questo elemento come massimo attuale
          }
    } /* fine del costrutto while */

    scrivi (max);
    scrivi (posmax);
  } /*fine ramo if eseguito se N è  valore accettabile*/

  else /* N non è un valore accettabile */
  {
    scrivi («il valore di N non è accettabile»)
  }/*fine ramo if eseguito se N non è  accettabile */
                                    }
```

Esempio 6: pseudo-codice

```
main ()
{

int N;            /* numero di elementi da acquisire */
int elemento;     /* valore dell'elemento letto */
int max;          /* valore del massimo */
int posmax;       /* posizione massimo nella sequenza */
int i;            /*  indice  corrente  relativo  al  numero  di
                  elementi già letti */

leggi (N);
if (N > 0 ) /*l'utente vuole inserire almeno un valore */
  {
     leggi (elemento);
      i = 1; /* aggiorna l'indice corrente */
      max = elemento;
      posmax = i;

    /* ciclo acquisizione e ricerca massimo per i  rimanenti
valori */

    while (i < N)
    {
     leggi (elemento);
      i = i +1; /* aggiorna l'indice corrente */
      if (elemento > max)
          {
           max = elemento;
           posmax = i;
          }
    } /* fine del costrutto while */

    scrivi (max);
    scrivi (posmax);
  } /* fine ramo if eseguito se N è valore accettabile */

  else /*N non è un valore accettabile */
  {
    scrivi («il valore di N non è accettabile»)
  }/* fine ramo if eseguito se N non è accettabile */
}
```

Esempio 7

Si vuole costruire la retta passante per due punti del piano X-Y.

Algoritmo:
1. acquisizione delle coordinate X e Y dei due punti.
2. calcolo dei parametri caratteristici dell'equazione della retta:
   - $x=k_x$
   - $y= k_y$
   - $y=mx+q$
3. visualizzazione dei parametri.

variabili:

`x1,x2,y1,y2`	coord. reali dei punti
`kx,ky, m, q`	param. della retta (reali)

Esempio 7: pseudo-codice

```
main ()
{
  float x1, x2, y1,y2;    /*coordinate dei punti */
  float kx;               /*parm retta parallela asse y*/
  float ky;               /*parm retta parallela asse x*/
  float m, q;             /*parm retta generica*/

  leggi (x1);
  leggi (x2);
  leggi (y1);
  leggi (y2);

  if (i due punti sono coincidenti)
    {
     scrivi («problema mal posto: punti coincidenti»);
    }

    else  /* esiste la retta */
    {
     if (ascisse coincidenti)
         {
           kx=x1;
         scrivi(«la retta e' parall. asse y e ......»);
         }
         else  /* ascisse non coincidenti */
         {
             if (ordinate coincidenti)
               {
               ky=y1;
               scrivi(«retta parallela  asse x e ......»);
               }
             else /*ascisse e ordinate non coincidenti */
               {
               calcolare m e q;
               scrivi(«i parametri della retta sono....»);
               } /*fine ramo retta generica */
         } /*fine ramo ascisse non coincidenti */
    } /*fine ramo esiste retta */
} /*fine del programma */
```

Esempio 7: pseudo-codice

```
main ()

{
  float x1, x2, y1,y2;    /*coordinate dei punti */
  float kx;               /*parm retta parallela asse y*/
  float ky;               /*parm retta parallela asse x*/
  float m, q;             /*parm retta generica*/

  leggi (x1);
  leggi (x2);
  leggi (y1);
  leggi (y2);

  if (i due punti sono coincidenti)
    {
     scrivi («problema mal posto: punti  coincidenti»);
    }

    else  /* esiste la retta */
    {
     if (x1==x2)
         {
           kx=x1;
         scrivi(«retta parallela asse y e ......»);
         }
         else  /* ascisse non coincidenti */
         {
             if (y1==y2)
               {
               ky=y1;
               scrivi(«retta parallela asse x e ......»);
               }
             else  /*ascisse e ordinate non coincidenti */
               {
               m=(y2-y1)/(x2-x1);
               calcolare q;
               scrivi(«i parametri della retta sono.....»);
               } /*fine ramo retta generica */
         } /*fine ramo ascisse non coincidenti */
    } /*fine ramo esiste retta */
} /*fine del programma */
```

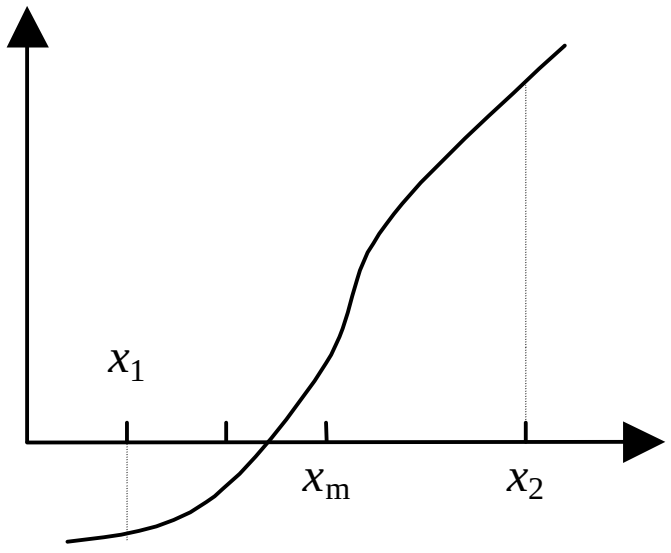
Esempio 8

Si vuole scrivere un programma per la ricerca delle radici di una funzione polinomiale di grado *n* del tipo

$$y = \sum_{i=0}^{n} a_i x^n$$

in un intervallo compreso tra x_1 e x_2.
La ricerca della radice (supposta unica e esistente) avviene con il **metodo di bisezione**:

- si calcola il punto medio $x_m = (x_1+x_2)/2$ tra i punti x_1 e x_2;
- si valuta il segno del polinomio *y* in corrispondenza dei punti x_1, x_2 e x_m;
- si scarta l'intervallo in cui il segno del polinomio valutato agli estremi è uguale (nell'esempio in figura, si scarta l'intervallo tra x_m e x_2);
- si riapplica il procedimento all'intervallo i cui estremi hanno segno del polinomio diverso (nell'esempio in figura, si riapplica il procedimento all'intervallo tra x_1 e x_m);
- il procedimento continua fino a che l'ampiezza dell'intervallo considerato diventa minore di un valore di tolleranza *t*;
- la soluzione cercata è il punto medio dell'ultimo intervallo trovato.



Traccia dell'algoritmo (1)

Il programma

1. riceve in ingresso dall'utente il grado *n* del polinomio;
2. riceve in ingresso dall'utente gli *n*+1 coefficienti a_i del polinomio;
3. riceve in ingresso dall'utente gli estremi x_1 e x_2;
4. riceve in ingresso dall'utente la tolleranza *t*;
5. **calcola la radice del polinomio con il metodo di bisezione;**
6. visualizza la coordinata *x* della radice trovata e il numero di iterazioni richieste per trovare la soluzione.

Raffinamento del punto 5. (vedi *metodo di bisezione*)

numero di iterazioni = 0
ripete ciclicamente finchè l'ampiezza dell'intervallo x_2-x_1 è maggiore della tolleranza *t*

- calcola il punto medio $x_m = (x_1+x_2)/2$ tra i punti x_1 e x_2;
- valuta il polinomio *y* in corrispondenza dei punti x_1, x_2 e x_m (y_1,y_2,y_m)
- valuta il segno del polinomio e aggiorna gli estremi dell'intervallo
- incrementa di uno il numero di iterazioni
- torna a valutare la condizione sull'ampiezza dell'intervallo

al termine della ripetizione ciclica, calcola la coordinata *x* della radice: $x = (x_1+x_2)/2$

Traccia dell'algoritmo (2)

Specifiche sul punto **1.**

il grado n del polinomio è minore di un grado massimo. Per l'esempio si sceglie il valore arbitrario di **4**

Variabili necessarie al programma

variabili di ingresso

- **N** di tipo intero: grado effettivo del polinomio da analizzare

- variabili di tipo reale per rappresentare il valore dei coefficienti del polinomio da analizzare (sono grado massimo +1)

  per le conoscenze attuali si adotta la soluzione di dichiarare una variabile di tipo reale per ogni coefficiente: **a0**, **a1**, **a2**, **a3**, **a4** (esiste un tipo di dato più indicato per rappresentare queste informazioni)

- **x1** e **x2** di tipo reale per rappresentare gli estremi dell'intervallo di definizione del polinomio

- **toll** di tipo reale per rappresentare la tolleranza voluta

variabili di uscita

- **n_iterazioni** di tipo intero
- **x** di tipo reale per rappresentare la coordinata della radice

variabili per l'elaborazione

- **xm** di tipo reale per rappresentare la coordinata del punto medio
- **y1**, **y2**, **ym** di tipo reale per rappresentare il valore del polinomio nei punti corrispondenti

Calcolo delle radici: pseudo-codice

```
main()
{
  int N, n_iterazioni;
  float x1,x2,xm, x, toll;
  float a0,a1,a2,a3,a4;
  float y1,y2,ym;


  scrivi("Inserire il grado del polinomio <= 4»);
  leggi(N);
  if (N <= 4)
    { /* algoritmo per il calcolo della radice */

    /* Acquisizione dei dati di ingresso */

    scrivi("Inserire il valore del coeff a0»);
    leggi(a0);
    scrivi("Inserire il valore del coeff a1»);
```

```
leggi(a1);

leggi(a2);

leggi(a3);

leggi(a4);

scrivi("Inserire il valore di x1»);
leggi(x1);

leggi(x2);

leggi(toll);
```

```
   /* algoritmo di calcolo */

   n_iterazioni = 0;

   /* ciclo di bisezione */
   while ((x2 – x1) > toll)
     {
      xm = (x1+x2)/2;
      y1 = calcola_ordinata(a0, a1, a2, a3,a4, x1);
      y2 = calcola_ordinata(a0, a1, a2, a3,a4, x2);
      ym = calcola_ordinata(a0, a1, a2, a3,a4, xm);

      if (y1*ym <= 0)
/* y1 e ym discordi: deve essere scartato l'intervallo x2 xm e per la
   prossima iterazione l'intervallo da considerare diventa x1, x2
   uguale al punto medio trovato  */

         x2=xm;
       else  /* y2 e ym discordi */
         x1=xm;

      n_iterazioni = n_iterazioni +1;
```

```
      } /* termine ciclo di bisezione */

     x = xm;

     scrivi("La soluzione e'», x);
     scrivi("Trovata dopo numero di iterazioni pari a",
                                        n_iterazioni);
    } /* fine ramo di calcolo */

  else /* calcolo della radice non possibile */
    {
    scrivi("Il calcolo della radice non e'possibile: e' stato
           inserito un polinomio di grado troppo elevato»);
    }

} /* fine del programma principale */
```

```
/* SOTTOPROGRAMMA per il calcolo del valore del polinomio */

/* Riceve come parametri in ingresso i valori dei coefficienti
   e  il valore della ascissa x. Restituisce come valore
   l'ordinata corrispondente */

float calcola_ordinata(float a0, float a1, float a2, float a3,
                       float a4, float x);
{
  float y;

  y = a4*(x*x*x*x) + a3*(x*x*x) + a2*(x*x) + a1*x + a0;
  return y;
}
```