PASCAL: IL CALCOLO NUMERICO

TRATTO DA CAMAGNI-NIKOLASSY, CORSO DI INFORMATICA, VOL. 3, HOEPLI

Informatica

Il Calcolo Numerico

Metodo di Calcolo Numerico

- Per risolvere problemi del mondo reale con metodi matematici è necessario rappresentare il problema mediante un modello matematico
- Successivamente la soluzione si implementa a calcolatore mediante un adeguato linguaggio di programmazione
- A volte però la soluzione non esiste in forma esatta oppure se esiste è molto complesso il suo calcolo
- Quindi alle volte si riesce ad avere solamente una soluzione approssimata del problema (*algoritmi approssimati*)
- Tale approccio prende il nome di metodo di calcolo numerico

Metodo di Calcolo Numerico

- Gli algoritmi approssimati si basano su tecniche di :
  - Discretizzazione: passaggio da domini continui a domini discreti
  - Approssimazioni successive: metodi iterativi
- Il più delle volte si eseguono operazioni semplici ma ripetute molte volte

ALGORITMO NUMERICO

In sintesi un algoritmo numerico approssimato viene utilizzato:

- per calcolare l'approssimazione di un valore numerico non altrimenti determinabile con metodi algebrici;
- per calcolare l'approssimazione di un valore numerico non facilmente determinabile in forma esatta con metodi algebrici;
- per calcolare l'approssimazione di un valore numerico con una precisione maggiore rispetto a un algoritmo ottenuto come implementazione di un metodo algebrico esatto.

Calcolo della radice quadrata

- Qualsiasi calcolatrice ha la funzione che calcola la radice quadrata
- In realtà si tratta di un valore approssimato, la cui accuratezza (numero di cifre decimali) dipende dal numero di iterazioni che vengono eseguite.
- Ciò è possibile perché viene implementato un algoritmo di calcolo numerico

Metodo Babilonese

- Dato un valore α la sua radice quadrata $\sqrt{\alpha}$ è:

$$\lim_{n\to\infty} x_n = \sqrt{\alpha}$$

- Dove la successione x_n è così esprimibile:

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{\alpha}{x_n}\right)$$

← Media aritmetica tra due numeri

Implementazione iterativa

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{\alpha}{x_n}\right)$$

- La soluzione si ricava eseguente iterativamente i seguenti passi:
  1. Poni $n = 0$ e scegli $x_0 > 0$ in modo arbitrario
  2. Calcola la media tra x_n e $\frac{\alpha}{x_n}$ e sostituiscilo a x_n
  3. Aumenta n e torna al punto 2.

- Fino a quando si itera?
  - Fino al raggiungimento dell'accuratezza prefissata

Esempio

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{\alpha}{x_n}\right)$$

Formula iterativa che approssima la soluzione

- Supponiamo $\alpha = 4$ e $x_0 = 5$ (⇒ $\sqrt{\alpha} = 2$)

n=0 ⇒ $x_1 = \frac{1}{2}\left(x_0 + \frac{\alpha}{x_0}\right) = \frac{1}{2}\left(5 + \frac{4}{5}\right) = 2{,}9$

n=1 ⇒ $x_2 = \frac{1}{2}\left(x_1 + \frac{\alpha}{x_1}\right) = \frac{1}{2}\left(2{,}9 + \frac{4}{2{,}9}\right) = 2{,}14$

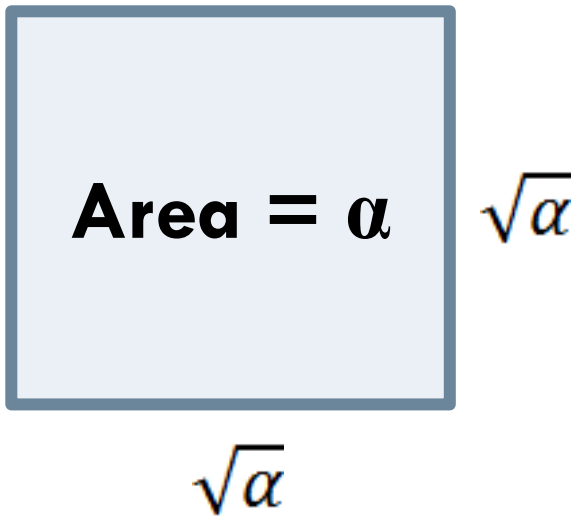
n=2 ⇒ $x_3 = \frac{1}{2}\left(x_2 + \frac{\alpha}{x_2}\right) = \frac{1}{2}\left(2{,}14 + \frac{4}{2{,}14}\right) = 2{,}0046$

n=3 ⇒ $x_4 = \frac{1}{2}\left(x_3 + \frac{\alpha}{x_3}\right) = \frac{1}{2}\left(2{,}0046 + \frac{4}{2{,}0046}\right) = 2{,}000005$

…

Significato geometrico

- Dato un numero α positivo, la sua radice quadrata può essere vista come il lato di un quadrato la cui area è proprio α stesso.



Area = lato x lato = $\sqrt{\alpha}$ x $\sqrt{\alpha}$ = α

Significato geometrico

- L'idea è quella di usare dei rettangoli che possiedano la stessa area del quadrato per arrivare attraverso approssimazioni successive ad ottenere proprio il quadrato che stiamo cercando
- Consideriamo quindi un rettangolo di lati x_0 e $\frac{\alpha}{x_0}$

n = 0

Area = α

$\frac{\alpha}{x_0}$

x_0

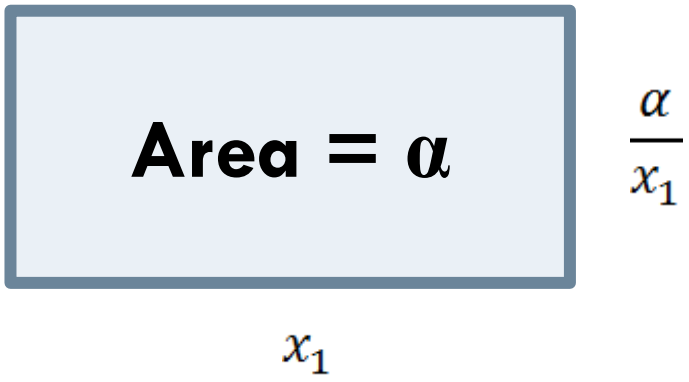
Area = lato x lato = $\frac{\alpha}{x_0}$ x x_0 = α

Significato geometrico

- Calcoliamo la media: $x_1 = \frac{1}{2}\left(x_0 + \frac{\alpha}{x_0}\right)$
  - Se $x_1 = x_0$ abbiamo trovato la radice quadrata $\sqrt{\alpha} = x_0 = x_1$
  - Altrimenti la media è diversa da x_0 e procediamo con il calcolo di una nuova media con n=1.

n = 1

$$x_2 = \frac{1}{2}\left(x_1 + \frac{\alpha}{x_1}\right)$$

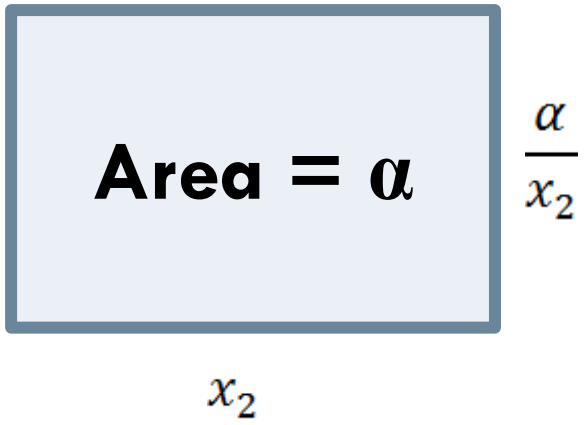


Area = lato x lato = x_1 x $\frac{\alpha}{x_1}$ = α

Significato geometrico

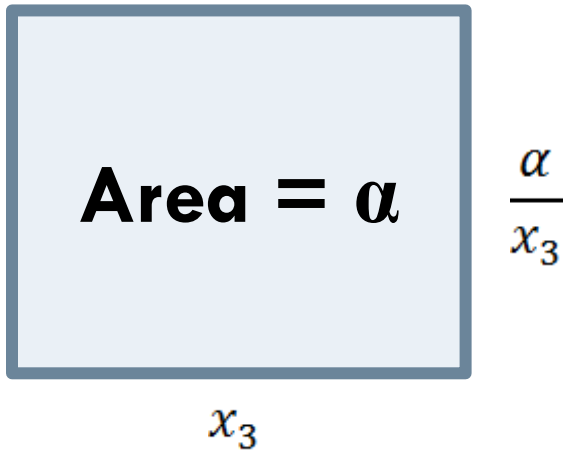
n = 2

$$x_3 = \frac{1}{2}\left(x_2 + \frac{\alpha}{x_2}\right)$$



n = 3

$$x_4 = \frac{1}{2}\left(x_3 + \frac{\alpha}{x_3}\right)$$



n = n+1

…

Esempio

- Supponiamo che α=4, vogliamo calcolare la $\sqrt{\alpha}$
- Scegliamo come valore di partenza $x_0 = 5$

n = 0

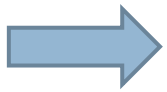
Area = 4

$$\frac{\alpha}{x_0} = \frac{4}{5} = 0.8$$

$$x_0 = 5$$

Area = lato x lato = 5 x 0.8 = 4

$$x_1 = \frac{1}{2}\left(x_0 + \frac{\alpha}{x_0}\right) = \frac{1}{2}\left(5 + \frac{4}{5}\right) = 2{,}9$$



$$\begin{cases} x_0 = 5 \\ x_1 = 2.9 \end{cases}$$

Esempio

n = 1

Area ≈ 4

$$\frac{\alpha}{x_1} = \frac{4}{2.9} = 1.38$$

$$x_1 = 2.9$$

Area = lato x lato = 2.9 x 1.38 ≈ 4

$$x_2 = \frac{1}{2}\left(x_1 + \frac{\alpha}{x_1}\right) = \frac{1}{2}\left(2{,}9 + \frac{4}{2{,}9}\right) = 2{,}14$$

$$\begin{cases} x_1 = 2.9 \\ x_2 = 2.14 \end{cases}$$

Esempio

n = 2

Area ≈ 4

$$\frac{\alpha}{x_2} = \frac{4}{2.14} = 1.87$$

$$x_2 = 2.14$$

Area = lato x lato = 2.14 x 1.87 ≈ 4

$$x_3 = \frac{1}{2}\left(x_2 + \frac{\alpha}{x_2}\right) = \frac{1}{2}\left(2{,}14 + \frac{4}{2{,}14}\right) = 2{,}0046$$

$$\begin{cases} x_2 = 2.14 \\ x_3 = 2.004 \end{cases}$$

Esempio

n = 3

Area ≈ 4

$$\frac{\alpha}{x_3} = \frac{4}{2.004} = 1.996$$

$$x_3 = 2.004$$

Area = lato x lato = 2.004 x 1.996 ≈ 4

$$x_4 = \frac{1}{2}\left(x_3 + \frac{\alpha}{x_3}\right) = \frac{1}{2}\left(2{,}0046 + \frac{4}{2{,}0046}\right) = 2{,}000005$$

$$x_3 = 2.004$$

$$x_4 = 2.000005$$

$$x_3 \cong x_4 \quad \Rightarrow \quad \sqrt{4} = 2$$

Implementazione in Pascal

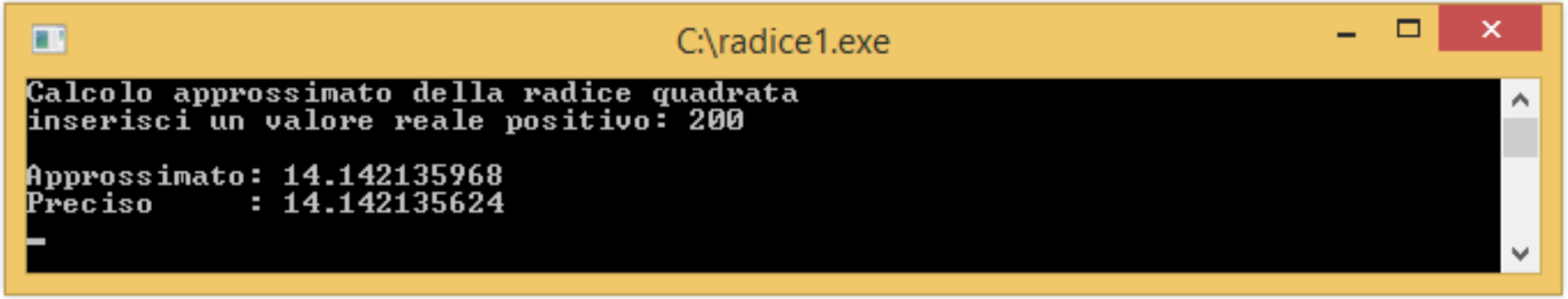
```
PROGRAM radice1;
const
  MAX = 5;
var
  numero: real;

 function radice_q(num:real): real;
 var
   Q:real;
   QPrec:real;
   n: integer;
 begin
   n:=0;
   QPrec := num / 2;                    // primo termine
   for n:=0 to MAX do
   begin
     Q := (QPrec + num / QPrec) / 2; // calcolo nuovo termine
     QPrec := Q;
   end;
  radice_q := q;
 end;
```

Implementazione in Pascal

```
begin
  writeln ('Calcolo approssimato della radice quadrata ');
  write   ('inserisci un valore reale positivo: ');
  readln  (numero);
  writeln;
  writeln ('Approssimato: ', radice_q(numero):12:9);
  writeln ('Preciso     : ', sqrt(numero):12:9);
  readln;
end.
```

- Il codice fornisce il seguente output



Esercizio

- Modifica il programma precedente facendo in modo che l'utente inserisca il numero di iterazioni che devono essere eseguite.

Esercizi 1-5, pag. 7

1 Modifica l'algoritmo in modo che l'utente scelga il numero massimo di iterazioni e la precisione desiderata; quindi visualizzi a ogni iterazione l'errore che viene commesso.
Introduci anche un controllo sui numeri inseriti che devono essere positivi.

2 Scrivi un nuovo programma dove viene inserito dall'utente anche il primo termine x_0: analizza il comportamento dell'algoritmo al variare di tale valore.

3 Scrivi un programma che acquisisca dall'utente il valore della precisione EPS desiderata compresa tra 0,1 e 0,000001 (verificando la correttezza dell'inserimento) ed il valore del numero del quale si vuole calcolare la radice, verificando che sia positivo.
Esegui il calcolo della radice mediante l'algoritmo babilonese terminando l'elaborazione quando si raggiunge il valore dell'errore e visualizzando sullo schermo il numero delle iterazioni che sono state fatte.

4 Scrivi un programma che acquisisca dall'utente il valore della precisione EPS desiderata compresa tra 0,1 e 0,000001 (verificando la correttezza dell'inserimento) ed il valore del numero $a > 10^6$ del quale si vuole calcolare la radice, verificando che sia positivo e rispetti la condizione desiderata.
Esegui il calcolo della radice mediante entrambi gli algoritmi confrontando i valori per ogni ordine di grandezza dell'errore.

5 Modifica l'esercizio precedente in modo che esegua automaticamente la ricerca della radice di un numero $a > 10^6$ con entrambi i metodi descritti al variare del valore della precisione EPS desiderata a partire da 0,1 fin ad arrivare a 0,0000001, ogni volta diminuendo EPS di 10 volte.
Costruisci una tabella visualizzando le sette coppie di valori ottenuti.
Ripeti l'elaborazione per almeno 10 volte: quali osservazioni si possono fare?

La generazione di numeri pseudocasuali

Processi deterministici e pseudocasuali

- In molte applicazioni è fondamentale la generazione di numeri casuali (cioè statisticamente non predicibili), come per esempio:
  - Simulazione di sistemi stocastici
  - Analisi numerica basata sul metodo Monte Carlo
  - Crittografia
  - Protocolli di comunicazione sicura

Un esempio di casualità lo si ottiene dal lancio di un dado: l'imprevedibilità del numero ottenuto come punteggio, compreso tra 1 e 6, conferisce allo stesso una forma di casualità.

- Essendo il computer regolato da eventi deterministici è difficile generare con esso numeri perfettamente casuali
- Infatti il calcolatore può generare solo numeri pseudocasuali

Processi pseudocasuali

PROCESSO PSEUDOCASUALE

Un processo pseudocasuale è un processo che sembra essere casuale ma non lo è. Una sequenza pseudocasuale sembra possedere casualità statistica anche se viene generata da un processo interamente deterministico.

- Una sequenza pseudocasuale, prima o poi si ripete ed è caratterizzata da un periodo di ripetizione chiamato periodo della sequenza.

Maggiore è la lunghezza di tale periodo e maggiore è la bontà del generatore di numeri pseudocasuali.

- Esempio: Sequenza binaria pseudocasuale

1101011011000101110101101100010111010110110001 01.....

periodo della sequenza = 16

Processi pseudocasuali

- Una sequenza pseudocasuale non è casuale ma è completamente determinata conoscendo:
  - Alcuni valori iniziali
  - Il seme (seed) della sequenza che a volte è il primo elemento della sequenza

Usando due volte lo stesso seme si ottiene la stessa sequenza di numeri!

Numeri pseudocasuali in Pascal

- Il linguaggio Pascal mette a disposizione due funzioni di libreria per generare sequenze di numeri pseudocasuali:
  - randomize() per inizializzare la sequenza
  - random() per generare un numero della sequenza

Esempio: numero casuale tra 0 e 1

- Il seguente codice genera un numero casuale compreso tra 0 e 1

```
program casuale;
var
 numero : real;
begin
  randomize;                // inizializzo il seme
  numero:=random();   // genero un numero casuale tra 0 e 1
  writeln ('il numero casuale generato e" : ', numero:2:6);
  readln;
end.
```

Esempio: Generazione di numero casuale in un range

- È possibile generare un numero casuale compreso in uno specifico intervallo (o range)
- È sufficiente passare come parametro alla funzione random() il numero desiderato
- Per esempio la funzione `random(num)` genera un numero casuale tra 0 e num-1
- random(70) genera un numero compreso nell'intervallo [0,69]

Esempio 2: la tombola

- Generiamo i numeri per la tombola, che devono avere un range [1, 90].
- Quindi si ha:
  - random(90) genera un numero nel range [0,89]
  - random(90)+1 genera un numero nel range [1,90]

```
program casuale2;
var
 numero : integer;
begin
  randomize;                  // inizializzo il seme
  numero:=random(90)+1;   // genero un numero casuale tra 0 e 1
  writeln ('il numero casuale generato e" : ', numero);
  readln;
end.
```

Esempio 3

- Generazione di numeri compresi tra 50 e 149

```
program casuali3;
var
  tanti, conta : integer;
begin
  randomize;                    // inizializzo il seme

  write  ('Quanti numeri compresi tra  50 e 149 genero ? ');
  readln (tanti);

  writeln('La sequenza generata e" la seguente: ');
  for conta:=1 to tanti do
    write(random(3)+ 50:5);   // genero un numero casuale tra 50 e 149

  readln;
end.
```

Algoritmi che generano le sequenze

- Non tutti gli algoritmi sono buoni: come distinguere un buon algoritmo ?
- Un buon algoritmo deve generare numeri che:
  - Siano **uniformemente distribuiti**
  - Non siano **correlati**
- E che l'algoritmo:
  - Sia riproducibile
  - Sia portabile
  - Abbia un lungo periodo
  - Sia computazionalmente veloce

Tra tutti i parametri i più significativi utilizzati per valutare la bontà sono la "casualità" della sequenza e il periodo del generatore.

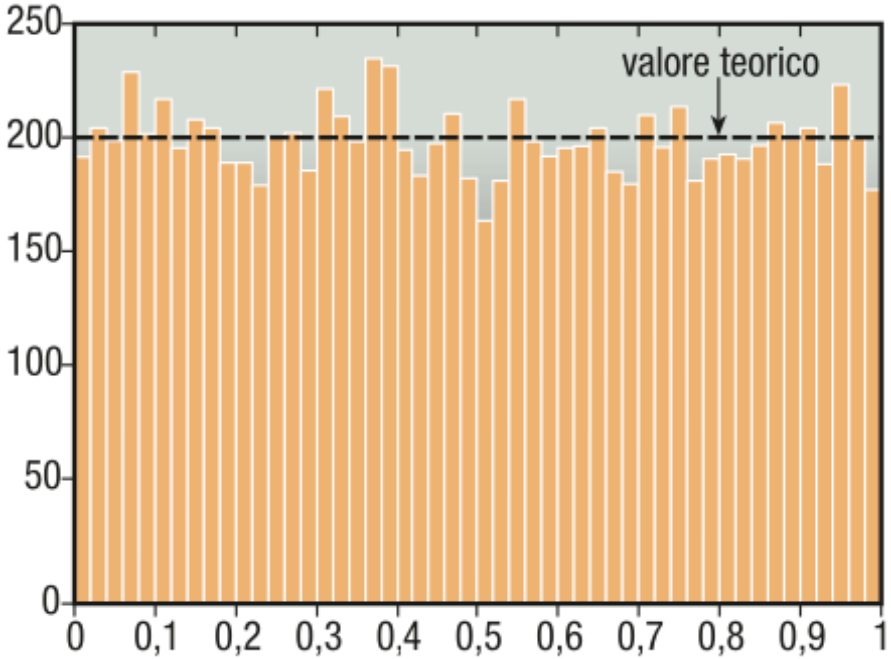
Distribuzione uniforme: Istogramma di frequenza

- Utilizziamo un generatore di numeri pseudocasuali con valore compreso tra 0 e 1 e suddividiamo l'intervallo di variabilità ad esempio in M = 50 sotto intervalli di pari ampiezza (range/nr. intervalli = 1/50 = 0,2).
- Calcoliamo ora il numero di valori "che cadono" in ciascun intervallo con la generazione di un numero elevato di numeri casuali, ad esempio N=100
- Teoricamente, in caso di equidistribuzione, si dovrebbe avere una frequenza costante pari al rapporto tra il numero di lanci e il numero di intervalli, cioè

$$valore\,teorico = \frac{Nr.lanci}{Nr.intervalli} = \frac{N}{M} = \frac{10000}{50} = 200$$

Istogramma di frequenza

- Riportando graficamente i risultati otteniamo ad esempio la situazione della figura



- Dal confronto dei grafici è possibile individuare l'algoritmo che più si avvicina al valore teorico e che quindi risulta avere una maggior equiprobabilità nella generazione dei numeri.

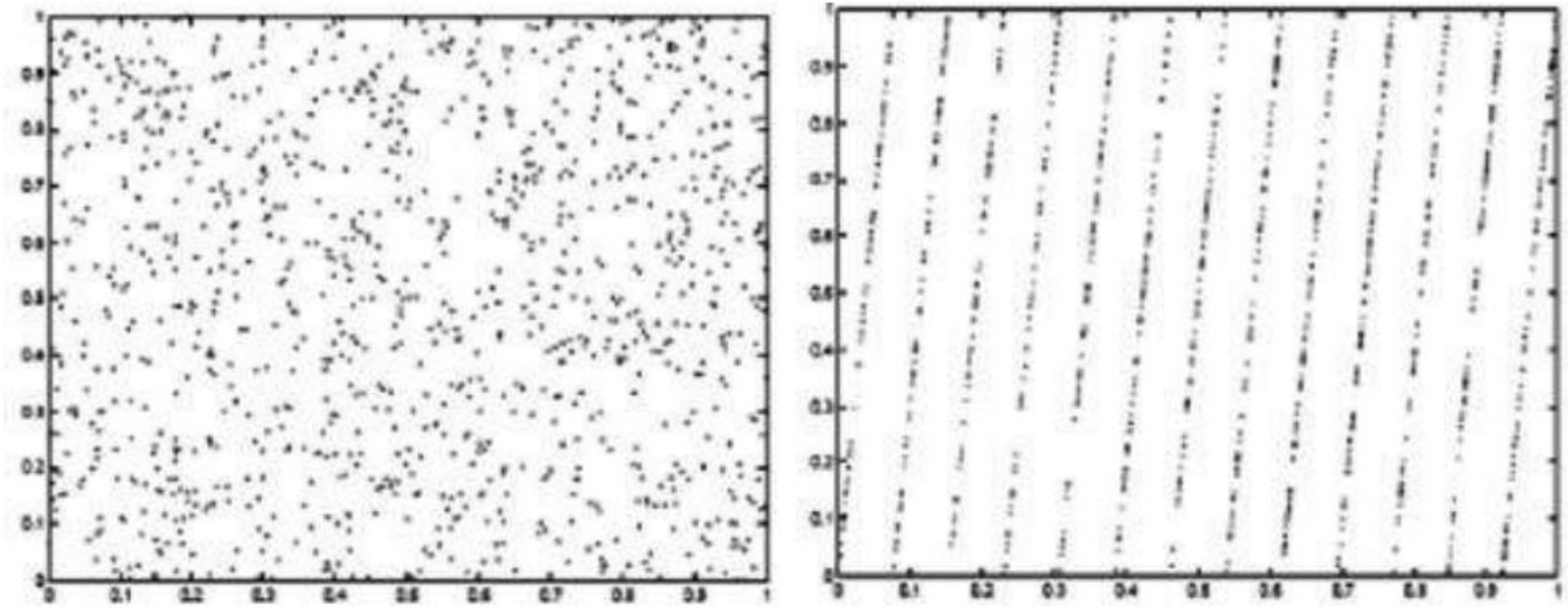
Correlazione

- Un buon algoritmo deve avere una bassa correlazione tra parti di sequenze generate
- Ciò implica che due numeri consecutivi non devono avere nessun legame tra loro
- **Esempio:**
  - Data una sequenza di numeri generati nel range [0,1]
  - Consideriamo i numeri a coppie e utilizziamoli come ascissa e ordinata di un punto $P_i(x_n, x_{n+1})$

In questo grafico non dovranno comparire linee, forme o altre strutture regolari ma punti distribuiti "omogeneamente a caso" nel piano.

Correlazione

- Il primo grafico rappresenta un generatore con distribuzione ottimale mentre il secondo, dato che i punti sono disposti su rette, sicuramente produce sequenze correlate e prevedibili.



Questi due grafici sono ottenuti dal medesimo algoritmo generatore, l'LCG che sarà di seguito descritto: la differenza tra le due situazioni è dovuta ai parametri di "settaggio" della formula generatrice.

Linear Congruential Generator (LCG)

- Il metodo LCG ha bisogno di un seme per inizializzare la sequenza di numeri secondo la seguente regola:

$$x_n = (ax_{n-1} + c) \bmod m$$

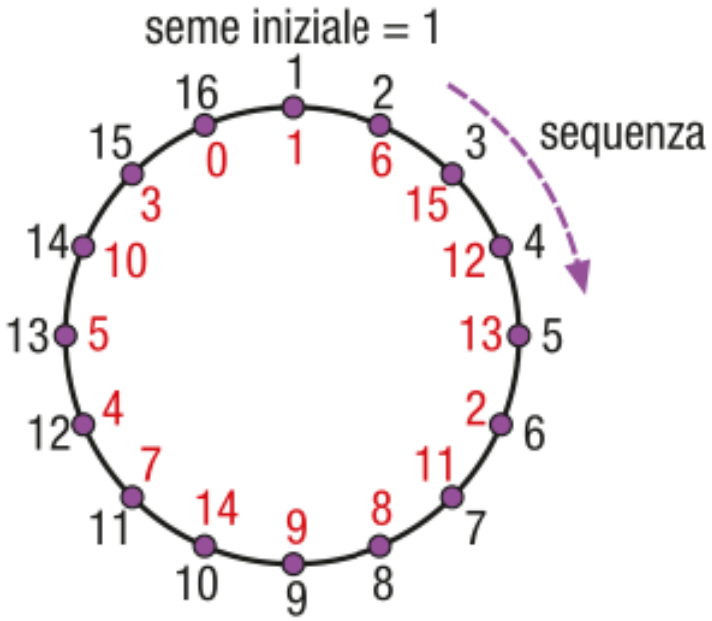
- Dove:
  - X_n è l'n-esimo numero casuale generato;
  - X_{n-1} è l'n-esimo numero casuale generato;
  - a, c, ed m sono costanti ($m>0$; $0<a<m$; $0<c<m$; $x_0<m$). Tipicamente m è 2^{32} oppure 2^{64}.
    Esistono delle relazioni tra di essi:
    - c ed m sono primi fra loro;
    - a-1 è divisibile per tutti i fattori primi di m;
    - a-1 è un multiplo di 4 se m è un multiplo di 4.

Esempio: Calcolo di 16 numeri con il metodo LCG

- Utilizziamo i seguenti valori:
  - LCG (a, c, m, x_0) = LCG(5,1,16,1)
- Il primo termine ($x_0 = 1$) è il seme della sequenza, calcoliamo i successivi termini.

$$x_2 = (ax_1 + c) \,\%\, m = (5x_1 + 1) \,\%\, 16 = 6$$
$$x_3 = (5x_2 + 1) \,\%\, 16 = 15$$
$$x_4 = (5x_3 + 1) \,\%\, 16 = 12$$

- E via di seguito fino a ottenere i seguenti 16 numeri:



Codice Pascal LCG

```
program casuali4;
var
    numero, seme, a, c, m, tanti, conta : integer;
begin
    seme:=1;       // parametri del metodo di Lemer
    a:=5;
    c:=1;
    m:=16;         // modulo della sequenza
    write  ('Quanti numeri della sequenza genero ? ');
    readln (tanti);

    writeln('La sequenza generata e" la seguente: ');
    for conta:=1 to tanti do
    begin
        numero := (a * seme + c) mod m;
        write (numero:4);
        seme := numero;
     end;
     readln;
end.
```

Output

```
Quanti numeri della sequenza genero ? 15
La sequenza generata e' la seguente:
   6  15  12  13   2  11   8   9  14   7   4   5  10   3   0
```

Esercizi 1-3, pag. 17

1 Un algoritmo proposto da Knuth nel 1981 può essere visto come una variante del metodo lineare congruenziale: i termini pseudocasuali si ottengono dalla seguente espressione:

$$x_{n+1} = (ax_n^2 + bx_n + c) \% m$$

La presenza di termini con esponente 2 dà al metodo il nome di congruenza quadratica.

Scrivi un programma con i seguenti parametri iniziali: $x_0 = 2$, $a = 2$, $b = 3$, $c = 1$, $m = 10$ e confronta i primi 10 termini con quelli generati dal metodo lineare.

2 Scrivi un programma che, utilizzando il metodo lineare congruenziale, distribuisca le carte per un torneo di tresette, assegnando ad ogni smazzata un numero identificativo in modo che questa possa successivamente essere rigiocata da un altro tavolo.
(Nel gioco del tresette ogni giocatore riceve 10 carte e le suddivide per seme e quindi le dispone in ordine decrescente secondo la seguente regola: 3,2,1,K,Q,J,7,6,5,4).
Visualizza a schermo le carte in mano ai quattro giocatori.

3 Scrivi un programma che, utilizzando il metodo lineare congruenziale, effettui la simulazione del lancio di un dado per $N = 10000$ e visualizzi l'istogramma delle frequenze in almeno 3 terne diverse di valori dei coefficienti. Osservando i risultati, quali osservazioni possono essere fatte sui semi e sul coefficiente a?
Ripeti ora l'esecuzione del programma ponendo $c = 0$ in ogni situazione: quali osservazioni puoi fare?

Esercizi 4-5, pag. 17

4 Crea un vettore di 20 elementi di valori random usando il metodo di congruenza lineare con $m = 2^{31} - 1$ $a = 75$ $c = 0$. Considerando che ogni numero può essere permutato con qualsiasi altro numero con probabilità uniforme, effettua $n = 15$ permutazioni e confronta a video i risultati.

5 Scrivere un programma che effettui la simulazione del lancio contemporaneo di due dadi e riportare in una tabella come la seguente i risultati ottenuti al variare del numero di lanci.

	Probabilità con $N = 500$	Probabilità con $N = 3000$	Probabilità con $N = 10.000$	Probabilità con $N = 50.000$	Probabilità con $N = 1.00000$
2					
3					
4					
5					
6					
…					
12					

Metodo Monte Carlo e calcolo di π

Metodo Monte Carlo

- Il metodo Monte Carlo fa parte della famiglia dei metodi probabilistici: questi hanno una lunga storia ma solo dopo il 1944 si sono studiati e applicati in modo sistematico

È stato valutato che quasi la metà delle applicazioni del calcolo scientifico ad alte prestazioni utilizza l'approccio con metodi tipo Monte Carlo.

- Alla base dei metodi probabilistici c'è il problema della generazione di numeri casuali

Il nome di metodo Monte Carlo che viene dato alle simulazioni probabilistiche è stato dato in onore del famoso casinò di Monaco per sottolineare il forte legame che esiste tra il gioco e le simulazioni probabilistiche.

Calcolo di π

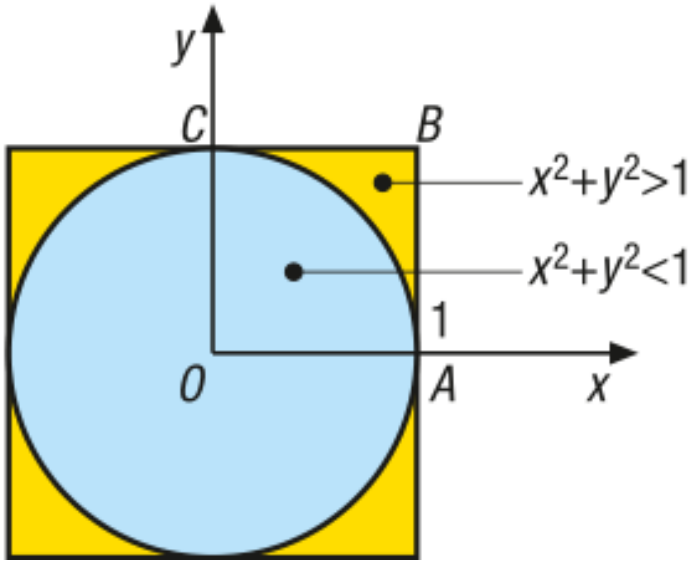
- Consideriamo una circonferenza di raggio unitario

$$r = 1 \quad \Rightarrow \quad Area_{Cerchio} = \pi \cdot r^2 = \pi \cdot 1^2 = \pi$$

- Consideriamo il quadrato che circoscrive la circonferenza

$$l = 2 \cdot r = 2 \quad \Rightarrow \quad Area_{Quadrato} = l \cdot l = 2 \cdot 2 = 4$$

- I punti all'interno del quadrato hanno coordinate x e $y < 1$
- Di questi, quelli che distano meno di 1 dal centro $(x^2 + y^2 < 1)$ sono nel cerchio
- Quelli per cui $(x^2 + y^2 > 1)$ sono fuori da cerchio



Calcolo di π

- Se disegniamo punti a caso nel quadrato alcuni capiteranno nel cerchio e altri fuori.
- Il metodo Monte Carlo ci assicura che il rapporto tra il numero di punti che cade nel cerchio e il numero totali di punti disegnati è pari al rapporto delle corrispondenti aree:

$$\frac{N.Punti\ nel\ cerchio}{N.Punti\ Totali} = \frac{Area\ del\ cerchio}{Area\ del\ quadrato} = \frac{\pi \cdot r^2}{(2 \cdot r)^2} = \frac{\pi}{4}$$

$$\pi = 4 \cdot \frac{N.Punti\ nel\ cerchio}{N.Punti\ Totali}$$

È necessario che i punti siano **uniformemente distribuiti**

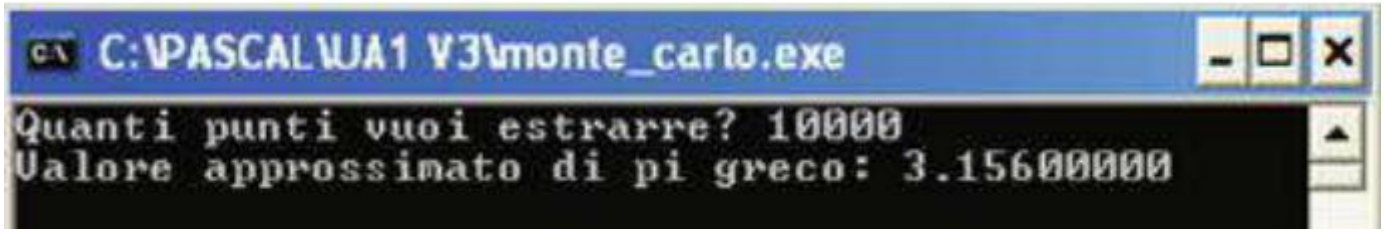
La **precisione** del calcolo migliora all'aumentare del numero di punti generati

Calcolo di π in Pascal

```
program pigreco;  // calcola pigreco col metoto Montecarlo
uses crt;
var i, n, dentro : integer;
     x, y : real;
begin
    randomize;
    clrscr;
    write('Quanti punti vuoi estrarre? ');
    readln(n);
    dentro:=0;
    for i := 1 to n do
    begin
        x:=random();    // estrae un punto nel quadrato di raggio unitario
        y:=random();
        if (x*x + y*y) < 1
        then dentro:=dentro + 1;
    end;
    writeln('Valore approssimato di pi greco: ', (4 * dentro / n):10:8);
    readln;
end.
```

Calcolo di π in Pascal

- Si ha il seguente output:



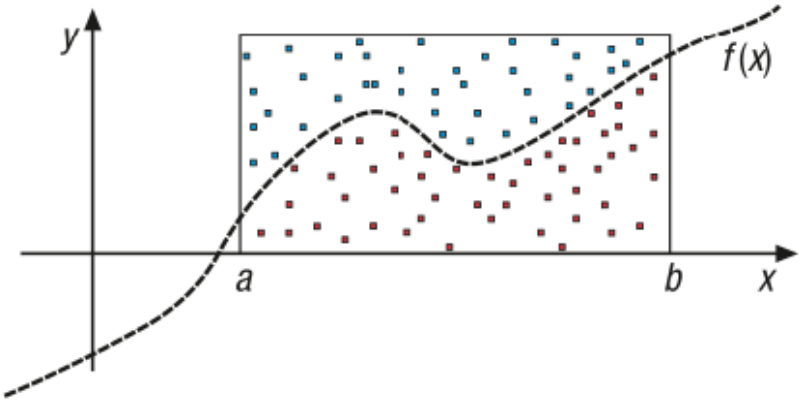
Naturalmente a ogni esecuzione avremo un risultato diverso dato che i punti sono generati casualmente.

- All'aumentare dei punti migliora la precisione:

Numero di punti	Valore
100	3,24
1000	3,128
10000	3,1256
100000	3,14028
1000000	3,142296
10000000	3,1418284
100000000	3,14158556

Integrazione numerica con il metodo Monte Carlo

- Il metodo Monte Carlo può anche essere utilizzato per effettuare il calcolo approssimato dell'integrale definito.



- Si individua un rettangolo di area nota
- Si generano punti uniformemente distribuiti nel rettangolo

$$\frac{N.\,Punti\ sotto\ la\ curva}{N.\,Punti\ nel\ rettangolo} = \frac{Valore\ dell'integrale}{Area\ del\ rettangolo}$$

$$Valore\ dell'integrale = Area\ del\ rettangolo \cdot \frac{N.\,Punti\ sotto\ la\ curva}{N.\,Punti\ nel\ rettangolo}$$

Esempio: Calcolo dell'area sottesa da una curva

- Si consideri la seguente funzione:

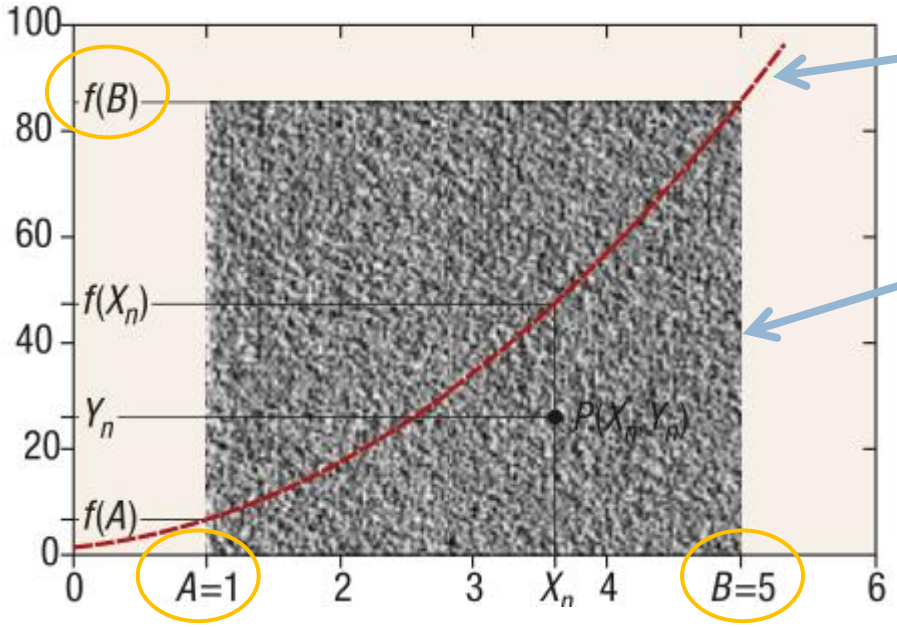
$$f(x) = 3x^2 + 2x + 1$$

- e il relativo integrale definito:

$$\int_A^B f(x)\,dx = \int_1^5 3x^2 + 2x + 1 = 152$$

- Bisogna generare una coppia di numeri $\mathbf{x_n}$ e $\mathbf{y_n}$ pseudocasuali uniformemente distribuiti, tali che:
  - $\mathbf{x_n}$ sia compreso tra A e B
  - $\mathbf{y_n}$ sia compreso tra o e f(B)

Esempio: Calcolo dell'area sottesa da una curva



$$f(x) = 3x^2 + 2x + 1$$

$$f(5) = 3 \cdot 5^2 + 2 \cdot 5 + 1 = 86$$

$$Area_{Rett} = f(B) \cdot (B - A) = 86 \cdot (5 - 1) = 344$$

$$\frac{Numero\ punti\ nell'area\ sottesa\ alla\ curva}{Numero\ di\ punti\ interni\ al\ rettangolo} =$$

$$= \frac{Area\ sottesa\ alla\ curva}{Area\ del\ rettangolo} = \frac{\int_A^B f(x)\,dx}{(B-A)\cdot f(B)}$$

$$\int_A^B f(x)dx = (B - A) \cdot f(B) \frac{N.\,punti\ sotto\ la\ curva}{N.\,totale\ dei\ punti}$$

Implementazione in Pascal

- Per calcolare il numero di punti appartenenti all'area sottesa dalla curva è sufficiente confrontare il valore della funzione nel punto X_n con il valore di Y_n e
  - se $f(X_n) \leq Y_n$ allora il punto appartiene all'area sottesa dalla curva;
  - Altrimenti il punto si trova al di sopra della curva.
- **Esercizio:**
  - Scrivi il codice Pascal che calcola l'integrale definito della funzione $f(x) = 3x^2 + 2x + 1$

Esercizi 1-4, pag. 27

1 Scrivi un programma che riceve in input il numero delle prove che deve eseguire e quante volte deve ripeterle. Quindi visualizza sullo schermo una tabella che riassume tutti i risultati ottenuti.

2 Calcola il valore di π per diversi valori di n (anche fino a $n \approx 50000$) e costruisci una tabella in modo da poter comparare i risultati per analizzare la velocità di conversione di questo metodo.

3 Invece del numero di punti in $\{x^2 + y^2 \leq 1\}$, conta il numero di punti in $\{x^2 + y^2 \leq 1/2\}$. A quale numero converge? Converge più o meno velocemente del metodo precedente?

4 In riferimento al calcolo dell'integrale della funzione

$$\int_A^B f(x)dx = \int_1^5 3x^2 + 2x + 1 = 152$$

Scrivi un algoritmo che esegue 5 diversi calcoli variando volta per volta il numero di punti e completa la tabella di seguito riportata.

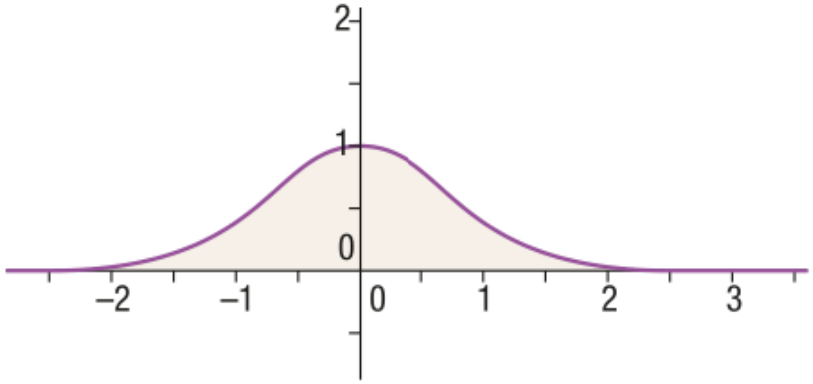
Numero di punti	Valore calcolato	Errore % sul calcolo dell'integrale
500		
10.000		
100.000		
1.000.000		

Esercizi 5-7, pag. 27

5 Possiamo verificare un integrale difficilmente calcolabile: l'integrale di Gauss (integrale che si presenta nello studio della densità di probabilità per variabili aleatorie continue):

$$\int_{-\infty}^{+\infty} e^{-x^2}\, dx = \sqrt{\pi}$$

La curva è la seguente:



Osservandola deduciamo che possiamo studiare solo mezza figura, quella nel primo quadrante, e considerare il rettangolo di altezza unitaria e di base ad esempio 3 (e non infinita!).

6 Prova a calcolare col metodo Monte Carlo il volume di una sfera unitaria.

7 Generalizza il problema di Buffon utilizzando piastrelle di forma rettangolare qualsiasi avente lati 2a e 2b: dopo aver ricavato l'espressione analitica, verifica il risultato applicando il metodo Monte Carlo.

Il numero e

Il numero e

- Il numero **e** non è periodico, irrazionale e la sua approssimazione a 20 cifre decimali è la seguente:

  e = 2,71828 18284 59045 23536

- Ha generalmente due nomi:
  - numero di **Eulero**, in ambito internazionale, collegandolo alla funzione esponenziale;
  - numero di **Nepero**, usato prevalentemente in Italia, come riconoscimento di paternità.

- Nepero assunse questo numero come base dei logaritmi in numero e = 2,71828…. perché vi conducevano naturalmente dei problemi finanziari, relativi alla ricerca dell'interesse

Calcolo del numero e

$$e = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n$$

- Eulero si dedicò al calcolo del suo valore e, inoltre dimostrò che:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \ldots = \sum_{k=0}^{\infty} \frac{1}{k!}$$

- ed è la formula che utilizzeremo nel nostro algoritmo per calcolare una approssimazione di e.

Funzione *fat()*

- Scriviamo una funzione che calcola il fattoriale.

```
function fat(n : integer): integer;
begin
   if(n>1)
   then
      fat:=(n*fat(n-1))
   else
      fat:=1;
end;
```

Funzione *calcolo_e()*

- La funzione che calcola il numero e è:

```
function calcolo_e(n:integer): real;
var
   x:integer;
   numeroe: real;
begin
   numeroe := 0;
   for x := 0 to n do
      numeroe := numeroe + (1.0 / fat(x));
   calcolo_e:=numeroe;
end;
```

Il programma completo

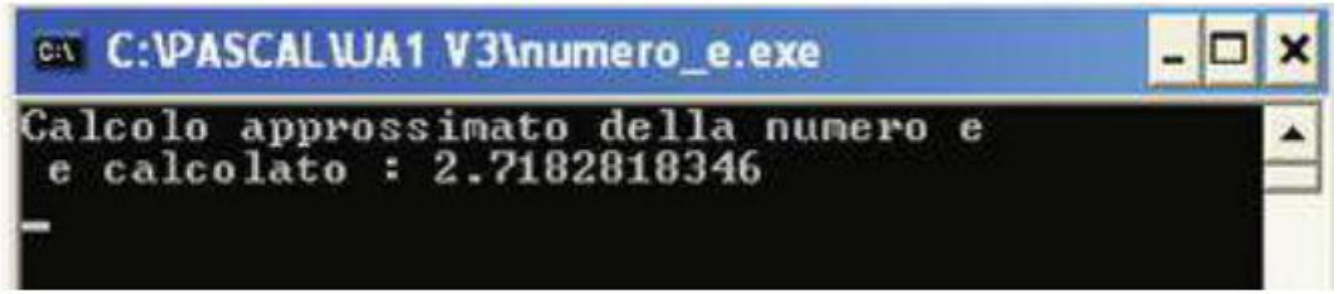
```
program numero_e;
var
   tanti : integer;
   numero_e: real ;

function fat(n : integer): integer;
function calcolo_e(n:integer): real;

begin
   tanti := 20;
   writeln('Calcolo approssimato della numero e ');
   numero_e := calcolo_e(tanti);
   writeln(' e calcolato : ',  numero_e:12:10 );
   readln;
end.
```

Output

- Il risultato con 20 iterazioni è il seguente:



- che ben si avvicina al valore di e = 2,71828182845904

Esercizio

Prova adesso!

Modifica il programma facendo calcolare il numero *e* con un numero di iterazioni variabile, leggendolo da input: attenzione ai problemi di overflow dei numeri.