Sommario

- Valutazione numerica della radice quadrata di un numero positivo (*<u>esempio di soluzione di un'esercitazione</u>*):
  - Descrizione teorica del problema e della sua soluzione.
  - Interpretazione algoritmica della soluzione.
  - Criteri di arresto.
  - Implementazione iterativa dell'approssimazione numerica.
  - Analisi dei risultati.
  - Sviluppo di una funzione di libreria.

Calcolo della radice quadrata

- Si vuole trovare il valore approssimato della radice quadrata $y = \sqrt{x}$ di un numero positivo attraverso approssimazioni successive.

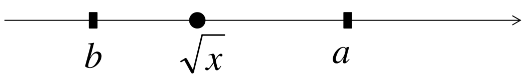
Si sceglie un numero positivo:

$$y_0 = a \quad \text{tale che} \quad a > 0 \quad \text{e} \quad a^2 > x$$

$\Longrightarrow y_0 = a$ rappresenta un'approssimazione per eccesso della radice

Calcolo della radice quadrata

Se $y_0 = a$ rappresenta un valore approssimato per eccesso di $y = \sqrt{x}$ allora $\dfrac{x}{y_0} = b$ rappresenta un valore approssimato per difetto.

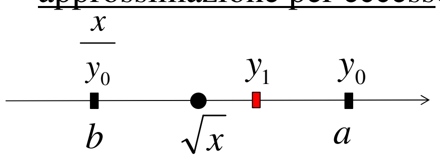


Allora si può approssimare la radice con la media aritmetica di *a* e *b*:

$$y_1 = \frac{a+b}{2}$$

Calcolo della radice quadrata

L'approssimazione y_1 sarà compresa tra $\sqrt{x}$ e y_0 , quindi potrà essere considerata come la <u>nuova approssimazione per eccesso</u> di $\sqrt{x}$:



$$y_1 = \frac{a+b}{2} = \frac{1}{2}\left(y_0 + \frac{x}{y_0}\right)$$

Ciò risulta vero in quanto la *media aritmetica* tra due numeri è maggiore o uguale alla loro *media geometrica*:

$$y_1 = \frac{1}{2}\left(y_0 + \frac{x}{y_0}\right) \geq \sqrt{y_0 \frac{x}{y_0}} = \sqrt{x}$$

Calcolo della radice quadrata

Pertanto si può procedere iterativamente costruendo una successione di approssimazioni $y_0, y_1, \ldots, y_n$ definite dalla seguente formula ricorsiva:

$$\begin{cases} y_0 = a \\ y_{i+1} = \dfrac{1}{2}\left(y_i + \dfrac{x}{y_i} \right) \end{cases}$$

Tale successione è convergente alla radice. Si può interpretare come il restringimento di un intervallo di localizzazione del valore esatto i cui estremi sono l'approssimazione per eccesso y_i e per difetto x/ y_i .

Valutazione iterativa

- Con un metodo iterativo si parte da una prima approssimazione e poi si applica una formula che migliora l'approssimazione a partire dai valori calcolati in precedenza sino al raggiungimento di una precisione richiesta (criterio di arresto):

inizializzare *old_val*
prima approssimazione *new_val*
while (new_val risulta lontano da *old_val)*
{ old_val=new_val
calcolare *new_val*
}

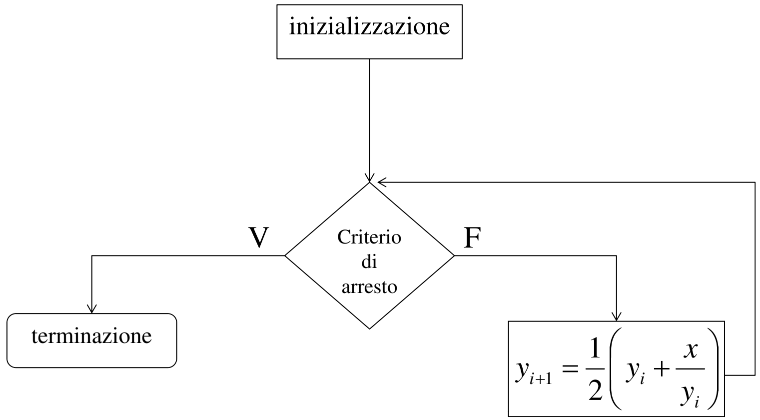
Valutazione iterativa

- Descrizione dell'algoritmo per la risoluzione del problema:

1. Inizializzazione e controlli
2. Calcolare la nuova approssimazione
3. Se il criterio di arresto non viene verificato ripetere passo 2.
4. Tornare l'approssimazione e terminare.

Valutazione iterativa

- Descrizione dell'algoritmo per la risoluzione del problema:



Valutazione iterativa

- Si possono definire diversi <u>criteri di arresto </u>con differenti prestazioni:
  - – Ricerca del valore esatto $y_{i+1}^2 != x$
  - – Errore assoluto $\left|y_{i+1}^2 - x\right| \le \varepsilon_{toll}$
  - – Errore relativo $\left|y_{i+1}^2 - x\right| \le y_{i+1}^2 \varepsilon_{toll}$
  - – Errore relativo approssimato $\left|y_{i+1} - y_i\right| \le \left|y_{i+1}\right| \varepsilon_{toll}$

Esempio di esercitazione

1. Sviluppare un programma che implementi il calcolo della radice quadrata di un numero positivo. Verificare con esempi le prestazioni dei diversi criteri di arresto. Commentare i risultati.

2. Sviluppare una corrispondente funzione di libreria e verificarne il funzionamento. La funzione ritorna un valore intero che indica quale condizione si è verificata.

Esempio di soluzione: analisi

```
#include <iostream>
#include <cmath>
using namespace std;
int main(){
        float x=9;
        float e=5e-3;
        float newy,oldy;

        newy=x;
        do{
                oldy=newy;
                newy=0.5*(oldy+x/oldy);
                cout<<"rif="<<sqrt(x)<<" y="<<newy<<endl;
        }while(newy*newy!=x);
        //}while((newy*newy-x)>e);
        //}while((newy*newy-x)>e*newy*newy);
        //}while(fabs(newy-oldy)>e*newy);

        cout<<"rif-y="<<sqrt(x)-newy<<endl;
        return 0;
}
```

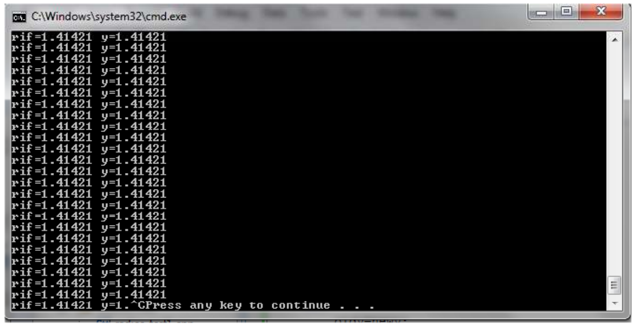

Esempio di soluzione: analisi

```
}while(newy*newy!=x);
```

Una possibile uscita con x=9:

rif=3 y=5
rif=3 y=3.4
rif=3 y=3.02353
rif−3 y−3.00009
rif=3 y=3
rif-y=0

Una possibile uscita con x=2:



Esempio di soluzione: analisi

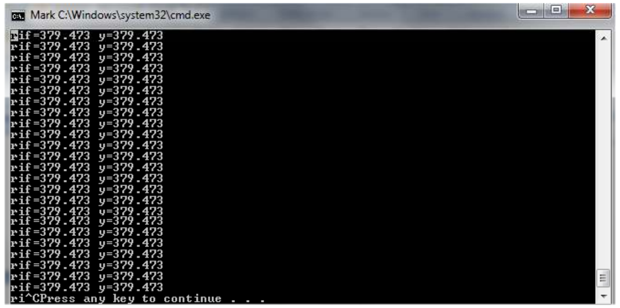
Una possibile uscita con x=2:

```
rif=1.41421 y=1.5
rif=1.41421 y=1.41667
rif=1.41421 y=1.41422
rif-y−-2.14577e-006
```

$$\varepsilon_{toll} = 5e-3$$

```
}while((newy*newy-x)>e);
```

Una possibile uscita con x= 144000 :



Esempio di soluzione: analisi

Una possibile uscita con x= 144000 :

```
rif=379.473 y=72000.5
rif=379.473 y=36001.3
rif=379.473 y=18002.6
rif−379.473 y−9005.31
rif=379.473 y=4510.65
rif=379.473 y=2271.29
rif=379.473 y=1167.34
rif=379.473 y=645.35
rif=379.473 y=434.242
rif=379.473 y=382.927
rif=379.473 y=379.489
rif-y=-0.015564
```

$$\varepsilon_{toll} = 5e-3$$

```
}while((newy*newy-x)>e*newy*newy);
```

Una possibile uscita con x= 1440000 :

```
rif=1200 y=720001
rif=1200 y=360001
rif=1200 y=180003
rif=1200 y=90005.3
rif=1200 y=45010.7
rif=1200 y=22521.3
rif=1200 y=11292.6
rif=1200 y=5710.07
rif=1200 y=2981.13
rif=1200 y=1732.08
rif=1200 y=1281.73
rif=1200 y=1202.61
rif-y=-2.60559
```


Esempio di soluzione: analisi

Una possibile uscita con x= 1440000 :

```
rif=1200 y=720001
rif=1200 y=360001
rif=1200 y=180003
rif−1200 y−90005.3
rif=1200 y=45010.7
rif=1200 y=22521.3
rif=1200 y=11292.6
rif=1200 y=5710.07
rif=1200 y=2981.13
rif=1200 y=1732.08
rif=1200 y=1281.73
rif=1200 y=1202.61
rif=1200 y=1200
rif-y=-0.00280762
```

$$\varepsilon_{toll} = 5e-3$$

```
}while(fabs(newy-oldy)>e*newy);
```

Una possibile uscita con x= 3e-6 :

```
rif=0.00173205 y=0.500001
rif=0.00173205 y=0.250004
rif=0.00173205 y=0.125008
rif=0.00173205 y=0.0625159
rif=0.00173205 y=0.031282
rif=0.00173205 y=0.0156889
rif=0.00173205 y=0.00794008
rif=0.00173205 y=0.00415895
rif=0.00173205 y=0.00244014
rif=0.00173205 y=0.00183479
rif=0.00173205 y=0.00173493
rif=0.00173205 y=0.00173205
rif-y=-2.44472e-009
```


Esempio di soluzione: funzione di libreria

radice.h

```
#ifndef    RADICE_H
#define    RADICE_H
int Radice(float x,float e,float
    *y);
#endif
```

radice.cpp

```
#include <cmath>
#include "radice.h"
int Radice(float x, float e, float *y){
        float newy,oldy;
        if (x==0 || x==1){
                *y=x;
                return 0;
        }
        if (x<0){
                *y=0;
                return 1;
        }
        newy=x;
        do{
                oldy=newy;
                newy=0.5*(oldy+x/oldy);
        }while(fabs(newy-oldy)>e*newy);
        *y=newy;
        return 0;
}
```

Esempio di soluzione: funzione di libreria

radice_test.cpp

```
#include <iostream>
#include <cmath>
#include "radice.h"
using namespace std;
int main(){
        float x, e, y;
        int err;
        x=0; e=5e-3;
        err=Radice(x,e,&y);
        cout<<"err="<<err<<";   rif-y="<<sqrt(x)-y<<";   y="<<y<<endl;
        x=-1; e=5e-3;
        err=Radice(x,e,&y);
        cout<<"err="<<err<<";   rif-y="<<sqrt(x)-y<<";   y="<<y<<endl;
        x=1440000; e=5e-3;
        err=Radice(x,e,&y);
        cout<<"err="<<err<<";   rif-y="<<sqrt(x)-y<<";   y="<<y<<endl;
        x=3e-6; e=5e-3;
        err=Radice(x,e,&y);
        cout<<"err="<<err<<";   rif-y="<<sqrt(x)-y<<";   y="<<y<<endl;
        return 0;
}
```


Esempio di soluzione: funzione di libreria

Una possibile uscita :