

Titolo unità didattica: *Approccio incrementale*

[06]

Titolo modulo : *Algoritmi per somme di potenze e per il massimo comun divisore*

[05-AT]

Sviluppo di algoritmi per il calcolo di particolari sommatorie e per il massimo comun divisore

Argomenti trattati:

- ✓ algoritmi per il calcolo di somme a segni alterni
- ✓ algoritmo per il calcolo di somme di potenze
- ✓ algoritmo di Euclide per il mcd

Prerequisiti richiesti: *AP-06-1-T*

problema:
calcolo della sommatoria

$$z = \sum_{i=1}^n \frac{(-1)^i}{2i+1}$$

$$z = -\frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots + \frac{(-1)^n}{2n+1}$$

dato di input: il numero n (variabile n)

dato di output: il numero z (variabile **somma**)

costrutto ripetitivo: **for**

operazione ripetuta (al generico passo i):

sommare l'addendo i -simo alla somma degli $(i-1)$ addendi precedenti

problema:
calcolo della sommatoria

$$z = \sum_{i=1}^n \frac{(-1)^i}{2i+1}$$

```
float somma_alterni (int n) {  
    int i;  
    float somma, addendo;  
    somma = 0.0 ;  
    for (i=1; i <= n; i++) {  
        addendo = (float)(-1)^i / (2.0*(float)i+1.0);  
        somma = somma + addendo ;  
    }  
    return somma ;  
}
```

2n somme
2n prodotti
n potenze

il calcolo di una potenza **i** richiede **i-1** prodotti

problema:
calcolo della sommatoria

$$z = \sum_{i=1}^n \frac{(-1)^i}{2i+1}$$

```
float somma_alterni (int n) {
```

```
int i;
```

```
float somma, addendo;
```

```
somma = 0.0 ;
```

```
for (i=1; i <= n; i++) {
```

```
    addendo = (float)(-1)^i / (2.0*(float)(i)+1.0);
```

```
    somma = somma + addendo ;
```

```
}
```

```
return somma ;
```

```
}
```

2n somme

$(n^2 - n) / 2 + 2n$ prodotti

il numero complessivo di prodotti per le potenze è
 $0+1+2+3 + \dots + n-1 = n*(n-1)/2$

problema:
calcolo della sommatoria

$$z = \sum_{i=1}^n \frac{(-1)^i}{2i + 1}$$

idea

valore assoluto dell' addendo

```
addendo = 1.0/(2.0*float(i)+1.0)  
somma = somma+segno*addendo
```

segno è una variabile il cui valore è **+1** oppure **-1**
e consente di rappresentare il cambio di segno in
modo immediato

```
segno = -segno
```

problema:
calcolo della sommatoria

$$z = \sum_{i=1}^n \frac{(-1)^i}{2i+1}$$

```
float somma_alterni (int n) {  
    int i;  
    float somma, addendo, segno;  
    somma = 0.0 ;  
    segno = -1.0;  
    for (i=1; i <= n; i++) {  
        addendo = 1.0/(2.0*(float)(i)+1.0);  
        somma = somma + segno*addendo ;  
        segno = - segno;  
    }  
    return somma ;  
}
```

2n somme
3n prodotti

problema: calcolo della sommatoria
di potenze (detta **geometrica**)

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

$$g = 1 + x + x^2 + x^3 + \dots + x^n$$

dati di input: il numero n (variabile **n**), il numero
 x (variabile **x**)

dato di output: il numero g (variabile **somma**)

costrutto ripetitivo: **for**

operazione ripetuta (al generico passo **i**):

sommare l'addendo **i**-simo (x^i) alla somma
degli (**i-1**) addendi precedenti

problema: calcolo della sommatoria di potenze (detta **geometrica**)

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

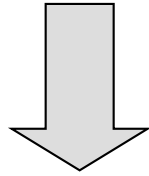
```
float somma_potenze(float x, int n)
int i;
float somma;
somma = 1.0 ;
for (i=1; i <= n; i++) {
    somma = somma + x^i ;
}
return somma ;
end
```

n somme
n potenze

n somme
 $(n^2 - n) / 2$ prodotti

il numero complessivo di prodotti è
 $0 + 1 + 2 + 3 + \dots + n - 1 = n * (n - 1) / 2$

calcolare x^i quando è stato già calcolato,
al passo precedente, il numero x^{i-1}



$$x^i = x \cdot x^{i-1}$$

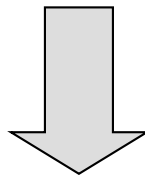
conservare l'ultimo
valore calcolato
della potenza di x

è un numero
già calcolato

calcolare x^i quando è stato già calcolato,
al passo precedente, il numero x^{i-1}

associare alla variabile **potenza_x** l'ultimo
valore calcolato della potenza di x

$$x^i = x \cdot x^{i-1}$$



potenza_x = potenza_x * x

calcolare x^i quando è stato già calcolato,
al passo precedente, il numero x^{i-1}

```
float somma_potenze(float x, int n) {  
    int i ;  
    float somma, potenza_x;  
    somma = 1.0 ;  
    potenza_x = 1.0 ;  
    for (i=1; i <= n; i++) {  
        potenza_x = potenza_x * x ;  
        somma = somma + potenza_x ;  
    }  
    return somma ;  
}
```

n
somme
n
prodotti

problema:

calcolo del **Massimo Comun Divisore** di due numeri (interi positivi)

è il più grande numero intero che divide (resto = 0) ognuno dei due numeri

$$\text{MCD}(30,42) = 6$$

$$\text{MCD}(50,40)=10$$

$$\text{MCD}(32,240)=16$$

$$\text{MCD}(16,23) = 1$$

$\text{MCD}(a,b)$ è il più grande dei fattori comuni della scomposizione di a e b in fattori primi

problema:

calcolo del **Massimo Comun Divisore** di due numeri (interi positivi)

$MCD(a,b)$ è il più grande dei fattori comuni della scomposizione di a e b in fattori primi

$$MCD(32,240)=16$$

$$32 = 2^5$$

$$240 = 2^4 \cdot 3 \cdot 5$$

più grande dei
fattori comuni

$$2^4 = 16$$

proprietà di $\text{MCD}(a,b)$

$$\text{MCD}(a,a)=a$$

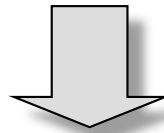
$$\text{MCD}(a,b) = \text{MCD}(a,a-b) = \text{MCD}(b,a-b)$$

$$a > b$$

$$\text{MCD}(a,b) = \text{MCD}(a,r) = \text{MCD}(b,r)$$

$$a = qb + r$$

$$r = a \bmod b$$



se $r=0$ allora b è $\text{MCD}(a,b)$

Algoritmo di Euclide

$$a > b$$

calcolare $r = a \bmod b$

se $r = 0$ allora *MCD* è b

altrimenti calcolare l'*MCD* di b e r

continuare

è considerato il primo algoritmo complesso della storia dell'uomo



325 AC-265 AC

Algoritmo di Euclide

$a > b$

```
int a,b,r,mcd;  
...  
r = mod(a,b) ;  
while (r != 0) {  
    a = b ;  
    b = r ;  
    r = mod(a,b) ;  
}  
mcd = b ;
```



Algoritmo di Euclide

$a > b$

```
int max_com_div(int a, int b) {  
    int r;  
    r = mod(a,b) ;  
    while (r != 0) {  
        a = b ;  
        b = r ;  
        r = mod(a,b) ;  
    }  
    return b ;  
}
```

