

Tutorato Lezione 8

Tutor: Maria Angela Pellegrino
mapellegrino@unisa.it

Ricorsione

Esercizi su ricorsione

1. Scrivere una funzione C che calcola, dati due numeri interi M ed N, la potenza M^N . Si progettino le versioni ricorsiva, ricorsiva tail.
2. Scrivere una funzione che calcoli il quoziente della divisione tra interi. Svolgere l'esercizio nelle due versioni ricorsiva e tail ricorsiva.
3. Scrivere una funzione che, data una stringa s, stampi tutte le stringhe ottenute permutando i caratteri di s.
 - a. Ad esempio, l'invocazione permutazioni("abc") deve effettuare la seguente stampa su standard output
 - i. abc/acb/bac/bca/cab/cba
4. La Torre di Hanoi. Sono date tre torri (sinistra, centrale, e destra) e un certo numero N di dischi forati.
 - a. I dischi hanno diametro diverso gli uni dagli altri, e inizialmente sono infilati uno sull'altro (dal basso in alto) dal più grande al più piccolo sulla torre di sinistra.
 - b. Scopo del gioco è portarli tutti sulla torre destra, rispettando due regole:
 - i. si può muovere un solo disco alla volta;
 - ii. un disco grande non può mai stare sopra un disco più piccolo.

Esercizi su ricorsione

1. Scrivere una funzione C che calcola, dati due numeri interi M ed N, la potenza M^N . Si progettino le versioni **ricorsiva** e **tail ricorsiva**.
2. Scrivere una funzione che calcoli il quoziente della divisione tra interi. Svolgere l'esercizio nelle due versioni ricorsiva e tail ricorsiva.
3. Scrivere una funzione che, data una stringa s, stampi tutte le stringhe ottenute permutando i caratteri di s.
 - a. Ad esempio, l'invocazione permutazioni("abc") deve effettuare la seguente stampa su standard output
 - i. abc/acb/bac/bca/cab/cba
4. La Torre di Hanoi. Sono date tre torri (sinistra, centrale, e destra) e un certo numero N di dischi forati.
 - a. I dischi hanno diametro diverso gli uni dagli altri, e inizialmente sono infilati uno sull'altro (dal basso in alto) dal più grande al più piccolo sulla torre di sinistra.
 - b. Scopo del gioco è portarli tutti sulla torre destra, rispettando due regole:
 - i. si può muovere un solo disco alla volta;
 - ii. un disco grande non può mai stare sopra un disco più piccolo.

Sol potenza proposta

pot (m,n) :

if (n==0)

return 1

else

return m*pot(m, n-1)

Potenza
recursive -
draft
soluzione
proposta dai
ragazzi

```
int pot(int m, int n){  
    if (n==0)  
        return 1  
    else  
        return m*pot(m, n-1)  
}
```

Potenza *recursive*

```
int pot(int m, int n){  
    if (n>0) return m*pot(m,n-1);  
    else return 1;  
}
```

Potenza
tail recursive -
draft
soluzione
proposta dai
ragazzi

```
int pot_tail (m,n){  
    return pot_tail_inner(m,n,0,1);  
}  
  
int pot_tail_inner(int m, int n, int i, int prod){  
    if (i<=n){  
        prod=prod*m  
        i=i+1  
        return pot_tail_inner(m,n, i, prod)  
    }  
    else  
        return prod  
}
```


Potenza

tail recursive

```
int pot_tail_inner(int m, int n, int v, int k)
{
    if (k<=n){
        v = v * m;
        k++;
        return pot_tail_inner(m,n,v,k);
    }
    else return v;
}

int pot_tail(int m, int n){
    return pot_tail_inner(m,n,1,1);
}
```

Potenza

```
int pot(int m, int n){  
    if (n>0) return m*pot(m,n-1);  
    else return 1;  
}
```

```
int pot_tail_inner(int m, int n, int v, int k)  
{  
    if (k<=n){  
        v = v * m;  
        k++;  
        return pot_tail_inner(m,n,v,k);  
    }  
    else return v;  
}
```

```
int pot_tail(int m, int n){  
    return pot_tail_inner(m,n,1,1);  
}
```

Esercizi su ricorsione

1. Scrivere una funzione C che calcola, dati due numeri interi M ed N, la potenza M^N . Si progettino le versioni **ricorsiva** e **tail ricorsiva**.
2. Scrivere una funzione che calcoli il quoziente della divisione tra interi. Svolgere l'esercizio nelle due versioni **ricorsiva** e **tail ricorsiva**.
3. Scrivere una funzione che, data una stringa s, stampi tutte le stringhe ottenute permutando i caratteri di s.
 - a. Ad esempio, l'invocazione permutazioni("abc") deve effettuare la seguente stampa su standard output
 - i. abc/acb/bac/bca/cab/cba
4. La Torre di Hanoi. Sono date tre torri (sinistra, centrale, e destra) e un certo numero N di dischi forati.
 - a. I dischi hanno diametro diverso gli uni dagli altri, e inizialmente sono infilati uno sull'altro (dal basso in alto) dal più grande al più piccolo sulla torre di sinistra.
 - b. Scopo del gioco è portarli tutti sulla torre destra, rispettando due regole:
 - i. si può muovere un solo disco alla volta;
 - ii. un disco grande non può mai stare sopra un disco più piccolo.

Esercizi su ricorsione

1. Scrivere una funzione C che calcola, dati due numeri interi M ed N, la potenza M^N . Si progettino le versioni **ricorsiva** e **tail ricorsiva**.
2. Scrivere una funzione che calcoli il quoziente della divisione tra interi. Svolgere l'esercizio nelle due versioni **ricorsiva** e **tail ricorsiva**.
 - a. Suggerimento: $x/y = (x - y + y)/y = 1 + (x - y)/y$.
3. Scrivere una funzione che, data una stringa s, stampi tutte le stringhe ottenute permutando i caratteri di s.
 - a. Ad esempio, l'invocazione permutazioni("abc") deve effettuare la seguente stampa su standard output
 - i. abc/acb/bac/bca/cab/cba
4. La Torre di Hanoi. Sono date tre torri (sinistra, centrale, e destra) e un certo numero N di dischi forati.
 - a. I dischi hanno diametro diverso gli uni dagli altri, e inizialmente sono infilati uno sull'altro (dal basso in alto) dal più grande al più piccolo sulla torre di sinistra.
 - b. Scopo del gioco è portarli tutti sulla torre destra, rispettando due regole:
 - i. si può muovere un solo disco alla volta;
 - ii. un disco grande non può mai stare sopra un disco più piccolo.

Quoziente
recursive -
draft
soluzione
proposta dai
ragazzi

```
int div(int m, int n){  
    if(m>=n)  
        return 1+quoziente(m-n, n)  
    else  
        return 0  
}
```

Quoziente
recursive

```
int div(int x, int y){  
    if (x>=y) return 1+div(x-y,y);  
    else return 0;  
}
```

Quoziente
tail recursive -
draft
soluzione
proposta dai
ragazzi

```
int quoziente_tail_inner(int m, int n, int q) {  
    if (m < n)  
        return q;  
    else  
        return quoziente_tail_inner(m-n, n, q+1);  
}  
  
int quoziente_tail(int m, int n){  
    return quoziente_tail_inner(m, n, 0);  
}
```

Quoziente *tail recursive*

```
int div_tail_inner(int x, int y, int v, int k) {  
    if (k <= x){  
        v = v + 1;  
        k = k + y;  
        return div_tail_inner(x,y,v,k);  
    }  
    else return v;  
}  
  
int div_tail(int x, int y){  
    return div_tail_inner(x,y,0,y);  
}
```


Quoziente

```
int div(int x, int y){
    if (x>=y) return 1+div(x-y,y);
    else return 0;
}

int div_tail_inner(int x, int y, int v, int k) {
    if (k<=x){
        v = v + 1;
        k = k + y;
        return div_tail_inner(x,y,v,k);
    }
    else return v;
}

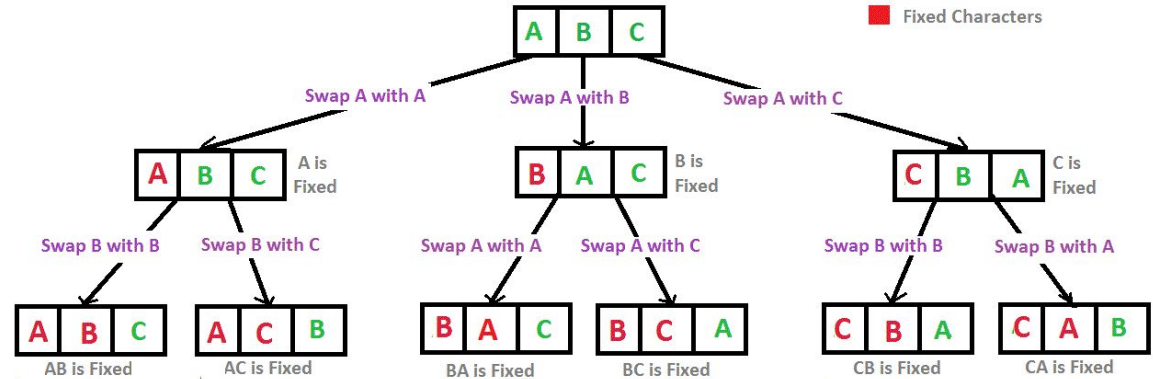
int div_tail(int x, int y){
    return div_tail_inner(x,y,0,y);
}
```

Esercizi su ricorsione

1. Scrivere una funzione C che calcola, dati due numeri interi M ed N, la potenza M^N . Si progettino le versioni **ricorsiva** e **tail ricorsiva**.
2. Scrivere una funzione che calcoli il quoziente della divisione tra interi. Svolgere l'esercizio nelle due versioni **ricorsiva** e **tail ricorsiva**.
 - a. Suggerimento: $x/y = (x - y + y)/y = 1 + (x - y)/y$.
3. Scrivere una funzione che, data una stringa s, stampi tutte le stringhe ottenute permutando i caratteri di s.
 - a. Ad esempio, l'invocazione permutazioni("abc") deve effettuare la seguente stampa su standard output
 - i. abc/acb/bac/bca/cab/cba
4. La Torre di Hanoi. Sono date tre torri (sinistra, centrale, e destra) e un certo numero N di dischi forati.
 - a. I dischi hanno diametro diverso gli uni dagli altri, e inizialmente sono infilati uno sull'altro (dal basso in alto) dal più grande al più piccolo sulla torre di sinistra.
 - b. Scopo del gioco è portarli tutti sulla torre destra, rispettando due regole:
 - i. si può muovere un solo disco alla volta;
 - ii. un disco grande non può mai stare sopra un disco più piccolo.

Permutazione

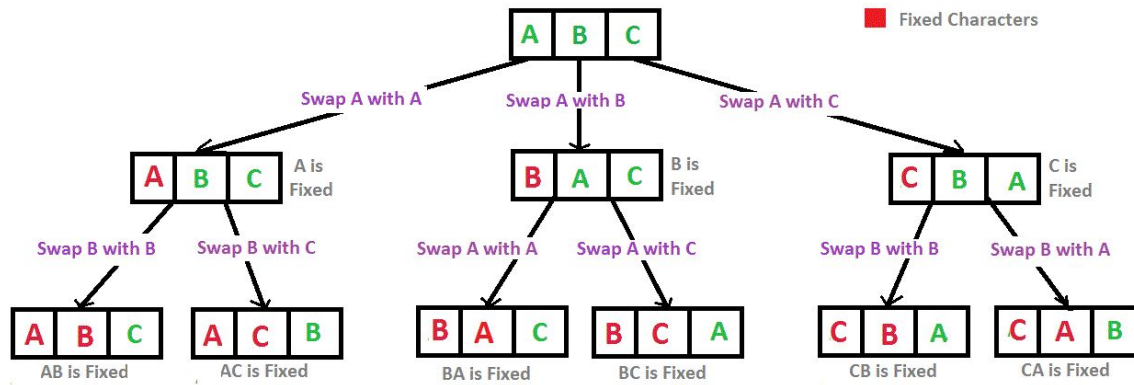
Soluzione - Idea



Recursion Tree for Permutations of String "ABC"

Permutazione

```
void permute(char *a, int l, int r) {
    int i;
    if (l == r)
        printf("%s\n", a);
    else
    {
        for (i = l; i <= r; i++)
        {
            swap((a+l), (a+i));
            permute(a, l+1, r);
            swap((a+l), (a+i)); // backtrack
        }
    }
}
```



Recursion Tree for Permutations of String "ABC"

```
void permute(char *a, int l, int r) {
    int i;
    if (l == r)
        printf("%s\n", a);
    else
    {
        for (i = l; i <= r; i++)
        {
            swap((a+l), (a+i));
            permute(a, l+1, r);
            swap((a+l), (a+i)); // backtrack
        }
    }
}
```

Esercizi su ricorsione


1. Scrivere una funzione che calcoli il quoziente della divisione tra interi. Svolgere l'esercizio nelle due versioni **ricorsiva** e **tail ricorsiva**.
 - a. Suggerimento: $x/y = (x - y + y)/y = 1 + (x - y)/y$.
2. Scrivere una funzione C che calcola, dati due numeri interi M ed N, la potenza M^N . Si progettino le versioni ricorsiva, ricorsiva tail e iterativa.
3. Scrivere una funzione che, data una **stringa** s, stampi tutte le stringhe ottenute **permutando** i caratteri di s.
 - a. Ad esempio, l'invocazione `permutazioni("abc")` deve effettuare la seguente stampa su standard output
 - i. `abc/acb/bac/bca/cab/cba`
4. La **Torre di Hanoi**. Sono date tre torri (sinistra, centrale, e destra) e un certo numero N di dischi forati.
 - a. I dischi hanno diametro diverso gli uni dagli altri, e inizialmente sono infilati uno sull'altro (dal basso in alto) dal più grande al più piccolo sulla torre di sinistra.
 - b. Scopo del gioco è portarli tutti sulla torre destra, rispettando due regole:
 - i. si può muovere un solo disco alla volta;
 - ii. un disco grande non può mai stare sopra un disco più piccolo.

Hanoi


```
void hanoi(int n, int origine, int destinazione, int appoggio){  
    if (n==1)  
        printf("Muovo da %d a %d\n", origine,destinazione);  
    else{  
        hanoi(n-1, origine, appoggio, destinazione);  
        printf("Muovo da %d a %d\n", origine, destinazione);  
        hanoi(n-1, appoggio, destinazione, origine);  
    }  
}
```

Esercizi Esame


Esercizi d'esame

- 
1. Implementare l'ADT Coda mediante l'ausilio di un Array.
 2. Verificare che due alberi dati in input siano uguali
 3. Stampa degli elementi di una lista in modo ricorsivo.

Esercizi d'esame

- 
1. Implementare l'ADT Coda mediante l'ausilio di un Array.
 2. Verificare che due alberi dati in input siano uguali
 3. Stampa degli elementi di una lista in modo ricorsivo.

Coda con
array



La coda è disponibile sul Teams del corso.

Esercizi d'esame

1. Implementare l'ADT Coda mediante l'ausilio di un Array.
2. Verificare che due alberi dati in input siano uguali
3. Stampa degli elementi di una lista in modo ricorsivo.

Confronta alberi

```
int same_tree(Btree first, Btree second){  
    if (first == NULL && second == NULL) {  
        return 1;  
    }  
  
    if ((first == NULL && second != NULL) ||  
        (first != NULL && second == NULL) ||  
        (first -> value != second -> value))  
        return 0;  
  
    return same_tree(first -> left, second -> left)  
        && same_tree(first -> right, second -> right);  
}
```

Esercizi d'esame

1. Implementare l'ADT Coda mediante l'ausilio di un Array.
2. Verificare che due alberi dati in input siano uguali
3. Stampa degli elementi di una lista in modo ricorsivo.

Stampa ricorsiva

```
void recursivePrintList(List list){
    struct node *p = getHead(list);
    recursivePrintListInner(p);
}

void recursivePrintListInner(Node* n){
    if(n){
        outputItem(n->item);
        recursivePrintListInner(n->next);
    }
}
```