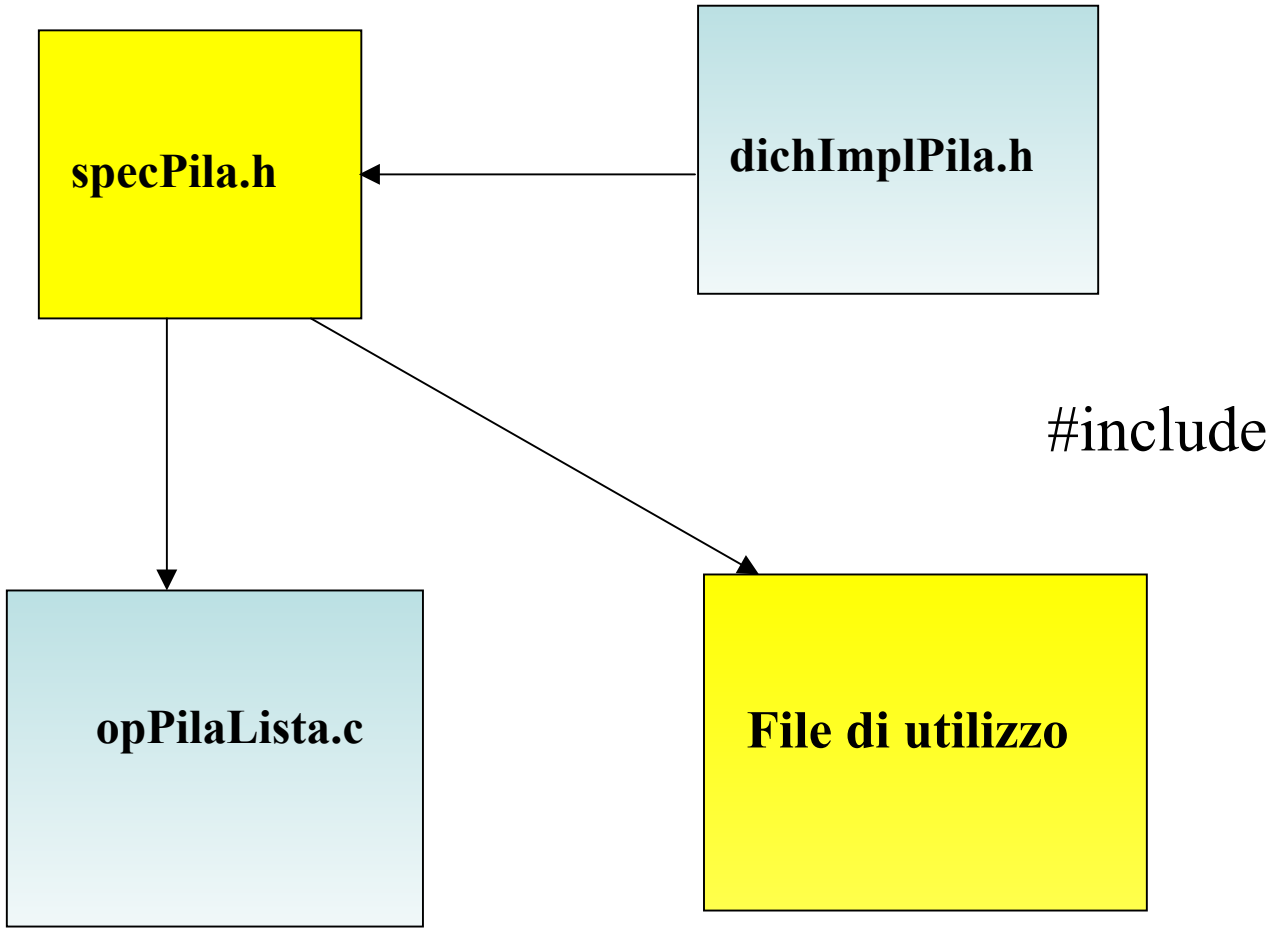

specPila.h
dichImplPila.h
#include
opPilaLista.c
File di utilizzo

NOME FILE :dichImplPila.h

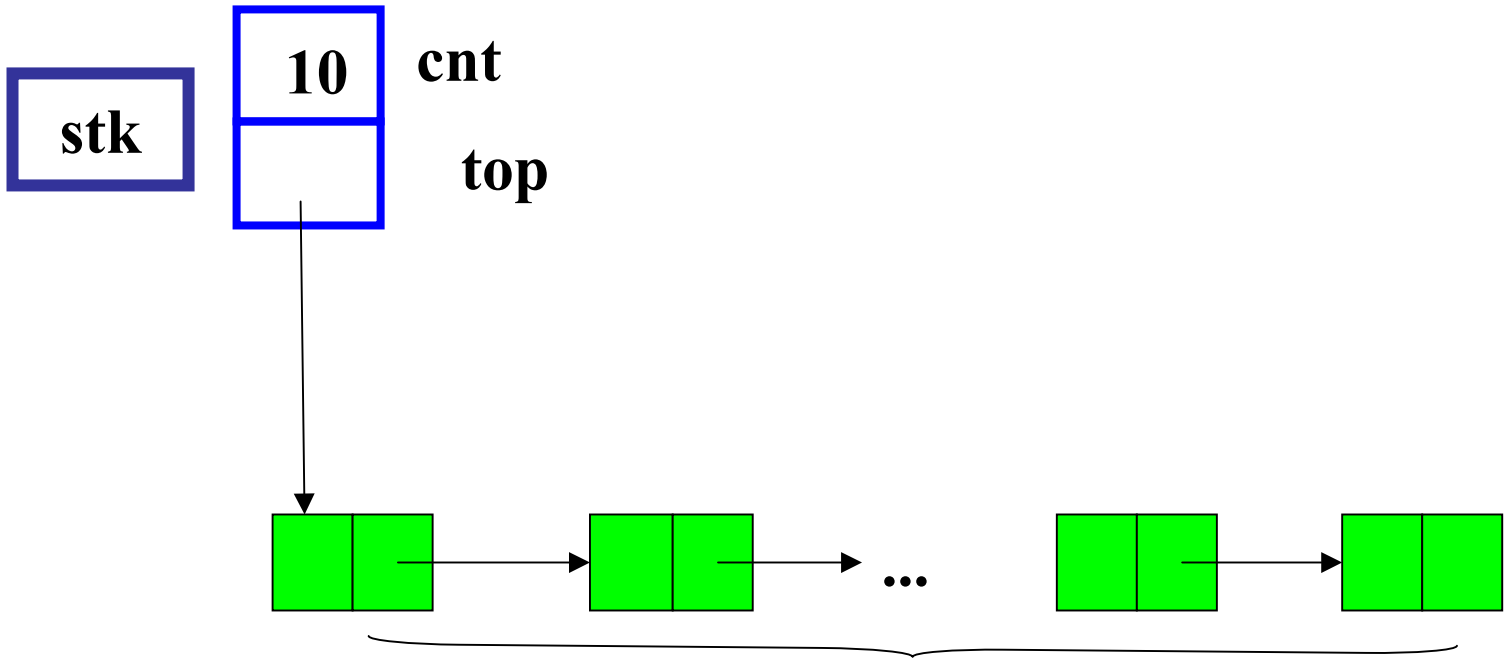
/* presentiamo qui un'implementazione dell'ADT "pila" su lista,
Invariante di implementazione: gli elementi della pila sono memorizzati in una lista, in modo che l'elemento in cima alla pila sia l'ultimo inserito .*/

```
#define EMPY 0
#define FULL  10000

typedef  enum boolean {false,true} boolean;

 typedef  struct elem {
void * d;
struct  elem *next;
} elem;

typedef struct stack {
 int  cnt;
elem * top;
}stack;
```



Elementi della pila

NOME FILE: opPilaList.c

```
#include <assert.h>
#include <stdlib.h>
#include "specPila.h"

void reset(stack *stk)
{assert(stk);
 stk ->cnt = 0;
 stk -> top = NULL
}

boolean empty(const stack *stk)
{assert(stk);
 return((boolean) (stk -> cnt == EMPTY));}

boolean full(const stack *stk)
{assert(stk);
 return((boolean) (stk -> cnt == FULL));}
```

NOME FILE: opPilaList.c

```
void* pop(stack *stk)
{void*   el;
  elem   *p;
  assert(stk);
  assert(!empty(stk));
  el = stk -> top -> d;
  p = stk -> top;
  stk -> top = stk -> top -> next;
  stk -> cnt--;
  free(p);
  return el }
```

NOME FILE: opPilaList.c

```
void push(void * el, stack *stk)
{ elem   *p;
  assert(stk);
  assert(el);
  assert(!full(stk));
   p = malloc(sizeof(elem));
   p -> d = el;
   p -> next = stk -> top;
   stk -> top = p;
   stk -> cnt++;
}

void* top(const stack *stk)
{assert(stk);
 assert(!empty(stk));
 return (stk -> top -> d);
}
```