



# Pile - Lezione 17

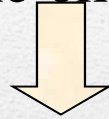
---

# Pile - stack

## **Applicazione:**

chiamate a funzioni

A chiama B che chiama C ...



last-in-first-out

## **Applicazione:**

reverse di una riga di input

---

# Pile

```
/* prima di includere questo file si deve dichiarare il tipo degli  
elementi da inserire nella pila, mediante una dichiarazione
```

```
typedef ..... TipoElemPila;
```

```
*/
```

```
typedef TipoElemLista TipoElemPila;
```

```
/* inclusione del file contenente la definizione del tipo TipoLista */
```

```
#include "tipolis.c"
```

```
typedef TipoLista TipoPila;
```

```
/* inclusione del file contenente l'implementazione delle  
operazioni primitive sulle liste */
```

```
#include "liste.c"
```

# Pila

```
/* implementazione delle operazioni primitive sulle pile */

void InitPila(TipoPila *p)
/* inizializza la pila p ponendo a NULL il puntatore all'elemento affiorante della pila */
{
    InitLista(p);
} /* InitPila */

bool TestPilaVuota(TipoPila p)
/* restituisce TRUE se la pila p e' vuota, FALSE altrimenti */
{
    return (TestListaVuota(p));
} /* TestPilaVuota */

void TopPila(TipoPila p, TipoElemPila *v)
/* restituisce in v l'elemento affiorante della pila p, senza modificare la pila */
{
    TestaLista(p, v);
} /* TopPila */
```

# Pile

```
/* implementazione delle operazioni primitive sulle pile */  
  
void Push(TipoPila *p, TipoElemPila v)  
/* inserisce l'elemento v in cima alla pila p */  
{  
    InserisciTestaLista(p, v);  
} /* Push */  
  
void Pop(TipoPila *p, TipoElemPila *v)  
/* elimina l'elemento affiorante della pila p, restituendone il valore in v */  
{  
    TopPila(*p, v);  
    CancellaPrimoLista(p);  
} /* Pop */
```

# Pile sequenziali

```
/* prima di includere questo file si deve dichiarare:  
  (i) la lunghezza massima della pila;  
  (ii) il tipo degli elementi da inserire nella pila.  
Questo viene realizzato mediante le seguenti dichiarazioni:
```

```
#define MaxPila ...
```

```
typedef ..... TipoElemPila;
```

```
*/
```

```
typedef int TipoPosPila;
```

```
struct StructPila {  
    TipoElemPila pila[MaxPila];  
    TipoPosPila pos;  
};
```

```
typedef struct StructPila TipoPila;
```

# Pile sequenziali

```
/* implementazione delle operazioni primitive sulle pile */

void InitPila(TipoPila *p)
/* inizializza la pila p ponendo a -1 il puntatore all'elemento
   affiorante della pila */
{
    p->pos = -1;
} /* InitPila */

bool TestPilaVuota(TipoPila p)
/* restituisce TRUE se la pila p e' vuota, FALSE altrimenti */
{
    return (p.pos == -1);
} /* TestPilaVuota */
```

# Pile sequenziali

```
void TopPila(TipoPila p, TipoElemPila *v)
/* restituisce in v l'elemento affiorante della pila p,
  senza modificare la pila */
{
  if (TestPilaVuota(p))
    printf("ERRORE: PILA VUOTA\n");
  else
    *v = p.pila[p.pos];
} /* TopPila */

bool TestPilaPiena(TipoPila p)
/* restituisce TRUE se la pila p e' piena, FALSE altrimenti */
{
  return (p.pos == (MaxPila - 1));
} /* TestPilaPiena */
```

# Pile sequenziali

```
void Push(TipoPila *p, TipoElemPila v)
/* inserisce l'elemento v in cima alla pila p */
{
    if (TestPilaPiena(*p))
        printf("ERRORE: PILA PIENA\n");
    else {
        p->pos++;
        p->pila[p->pos] = v;
    }
} /* Push */
```

```
void Pop(TipoPila *p, TipoElemPila *v)
/* elimina l'elemento affiorante della pila p, restituendone il valore in v */
{
    if (TestPilaVuota(*p))
        printf("ERRORE: PILA VUOTA\n");
    else {
        *v = p->pila[p->pos];
        p->pos--;
    }
} /* Pop */
```

# Pile dinamiche

```
/* prima di includere questo file si deve dichiarare:  
  (i) la lunghezza massima iniziale della pila;  
  (ii) il tipo degli elementi da inserire nella pila.  
mediante le seguenti dichiarazioni:
```

```
#define DimInizialePila ...
```

```
typedef ..... TipoElemPila;
```

```
*/
```

```
typedef int TipoPosPila;
```

```
struct StructPila {  
  TipoElemPila *pila;  
  TipoPosPila pos;  
  TipoPosPila dimCorrente;  
};
```

```
typedef struct StructPila TipoPila;
```

# Pile dinamiche

```
/* implementazione delle operazioni primitive sulle pile */

void InitPila(TipoPila *p)
/* inizializza la pila p, allocando la memoria per il vettore p->pila e
   ponendo a -1 il puntatore all'elemento affiorante della pila */
{
    p->pila = malloc(DimInizialePila * sizeof(TipoElemPila));
    p->pos = -1;
    p->dimCorrente = DimInizialePila;
} /* InitPila */

bool TestPilaVuota(TipoPila p)
/* restituisce TRUE se la pila p e' vuota, FALSE altrimenti */
{
    return (p.pos == -1);
} /* TestPilaVuota */
```

# Pile dinamiche

```
void TopPila(TipoPila p, TipoElemPila *v)
/* restituisce in v l'elemento affiorante della pila p, senza modificare la pila */
{
    if (TestPilaVuota(p))
        printf("ERRORE: PILA VUOTA\n");
    else
        *v = p.pila[p.pos];
} /* TopPila */

void Push(TipoPila *p, TipoElemPila v)
/* inserisce v in cima alla pila p e, se e' piena, aumenta la dimensione della pila */
{
    if (p->pos == (p->dimCorrente - 1))
    { /* pila piena: raddoppia la dimensione del vettore p->pila */
        p->dimCorrente = p->dimCorrente * 2;
        p->pila = realloc(p->pila, p->dimCorrente * sizeof(TipoElemPila));
    }
    /* inserisce l'elemento v in cima alla pila p */
    p->pos++;
    p->pila[p->pos] = v;
} /* Push */
```

# Pile dinamiche

```
void Pop(TipoPila *p, TipoElemPila *v)
/* elimina l'elemento affiorante della pila p, restituendone
   il valore in v, e se necessario riduce la dimensione della pila */
{
  if (TestPilaVuota(*p))
    printf("ERRORE: PILA VUOTA\n");
  else {
    *v = p->pila[p->pos];
    p->pos--;
    /* se la pila contiene un numero di elementi maggiore di
       DimInizialePila e inferiore ad un terzo della sua dimensione
       corrente, viene dimezzata la sua dimensione */
    if (p->dimCorrente > DimInizialePila && p->pos < p->dimCorrente/3)
      {
        p->dimCorrente=p->dimCorrente/2;
        p->pila = realloc(p->pila, p->dimCorrente * sizeof(TipoElemPila));
      }
  }
} /* Pop */
```

# Pile dinamiche

Esercizio:

Data una sequenza di parentesi tonde, quadre e graffe, verificarne il corretto annidamento

---