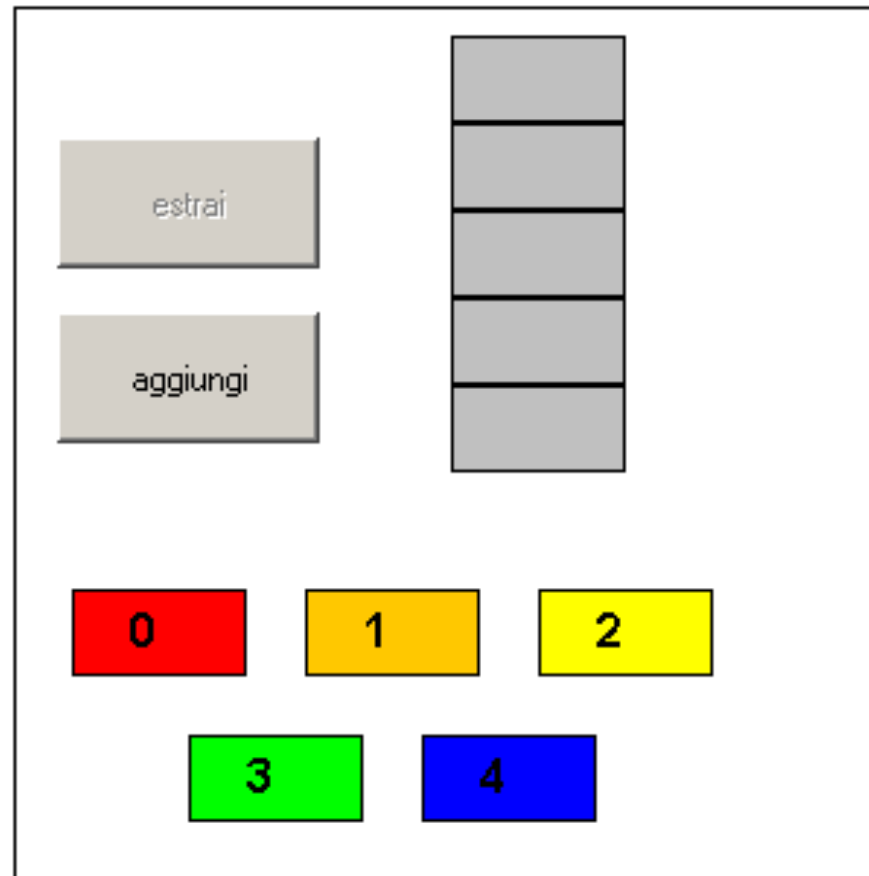


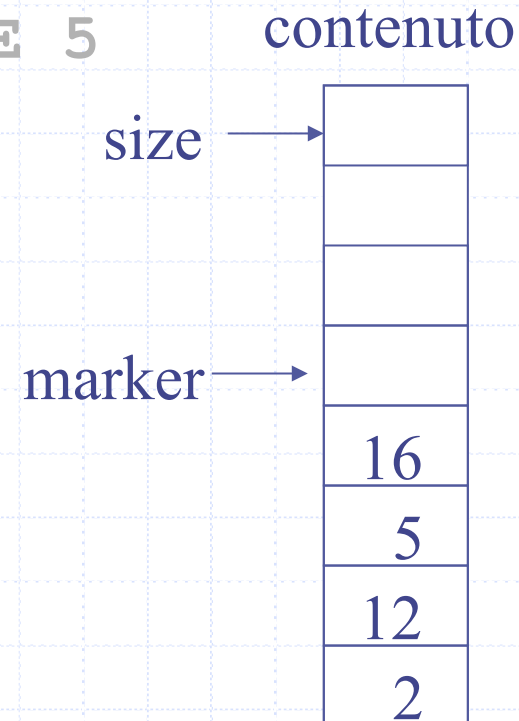
A decorative graphic on the left side of the slide. It features a horizontal line starting from a small circle on the left edge, extending to the right. From the end of this horizontal line, a vertical line extends downwards to the bottom edge of the slide.

Costruiamo uno stack



stackapplet.html

```
#include <iostream.h>
#include <cassert>
#define DEBUG
#define DEFAULTGROWTHSIZE 5
struct Pila {
    int size;
    int marker;
    int * contenuto;
};
```



```
Pila * crea(int initialSize) {  
    //crea una Pila  
    #ifdef DEBUG  
        cout<<"entro in crea"<<endl;  
    #endif  
    Pila * s= new Pila ;  
    s->size=initialSize;  
    s->marker=0;  
    s-> contenuto=new int[initialSize];  
    return s;  
}
```

```
void distruggi(Pila * s) {  
    //distruggi la Pila  
    #ifdef DEBUG  
        cout<<"entro in destroy"<<endl;  
    #endif  
    delete [] (s->contenuto);  
    delete s;  
}
```

Il preprocessore

Il preprocessore

Agisce come un Editor prima che la compilazione inizi.

Inclusione di files

`#include <filename>` (cerca in /usr/lib)

`#include "filename"` (specifica la pathname)

Definizione di costanti

```
#define N 100
```

```
    for (k=1; k<N ; k++) cout<< k << endl;
```

SCONSIGLIATO! E' meglio:

```
const int N=100
```

```
void cresci(Pila *s, int increment){
//aumenta la dimensione dello Pila
  cout<<"entro in cresci"<<endl;
  s->size+=increment;
  int * temp=new int[s->size];
  for (int k=0; k<s->marker;k++) {
    temp[k]=s->contenuto[k];
  }
  delete [] (s->contenuto);
  s->contenuto=temp;
}
```



```
void inserisci(Pila *s, int k) {  
    //inserisci un nuovo valore  
    cout<<"entro in inserisci"<<endl;  
    // DEFAULTGROWTHSIZE is 5  
    if (s->size==s->marker)  
        cresci(s,5);  
    s->contenuto[s->marker]=k;  
    s->marker++;  
}
```

```
int estrai(Pila *s) {  
    //estrai l'ultimo valore  
    cout<<"entro in estrai"<<endl;  
    assert(s->marker>0);  
    return s->contenuto[--(s->marker)];  
}
```

```
void stampaStato(Pila *s) {
    //stampa lo stato dello Pila
    cout <<"====="<< endl;
    cout << "size = "<<s->size<<endl;
    cout << "marker = "<<s->marker <<endl;
    for (int k=0;k<s->marker;k++)
        cout << "["<<(s-
>contenuto[k])<<"]";
    cout << endl;
    cout <<"====="<< endl;
}
```

```
Pila * copia(Pila * from) {  
    cout<<"entro in copia"<<endl;  
    Pila * to=crea(from->size);  
    for (int k=0; k<from->marker;k++) {  
        to->contenuto[k]=from->contenuto[k];  
    }  
    to->marker=from->marker;  
    return to;  
}
```

```
int main() {
    Pila * s=crea(5);
    cout<<"s"; stampaStato(s);
    for (int k=1; k<10;k++) inserisci(s,k);
    cout<<"s"; stampaStato(s);
    Pila * w = copia(s);
    cout<<"w"; stampaStato(w);
    for (int k=1; k<8;k++)
        cout<<estrai(s)<<endl;
    cout<<"s"; stampaStato(s);
    distruggi(s);
    cout<<"s"; stampaStato(s);
    for (int k=1; k<15;k++)
        cout<<estrai(w)<<endl;
    cout<<"w"; stampaStato(w);
}
```



```
bash-2.02$ gcc Pila.cc -o Pila.exe
```

```
bash-2.02$ Pila.exe
```

```
entro in crea
```

```
s=====
```

```
size = 5
```

```
marker = 0
```

```
=====
```

```
entro in inserisci
```

```
entro in inserisci
```

```
entro in inserisci
```

```
entro in inserisci
```

```
entro in inserisci
```

```
entro in inserisci
```

```
entro in cresci
```

```
entro in inserisci
```

```
entro in inserisci
```

```
entro in inserisci
```

```
s=====
```

```
size = 10
```

```
marker = 9
```

```
[1][2][3][4][5][6][7][8][9]
```

```
=====
```

```
entro in copia
```

```
w=====
```

```
size = 10
```

```
marker = 9
```

```
[1][2][3][4][5][6][7][8][9]
```

```
=====
```



```

┌
└ entro in estrai
  9
  entro in estrai
  8
  ...
  entro in estrai
  4
  entro in estrai
  3
  s=====
  size = 10
  marker = 2
  [1][2]
  =====

```

```

entro in distruggi

```

```

s=====
size = 1627775824
marker = 2
[1627775848][1627775848]
=====

```

```

entro in estrai

```

```

9
entro in estrai

```

```

8

```

```

...
entro in estrai

```

```

2

```

```

entro in estrai

```

```

1

```

```

entro in estrai

```

```

assertion "s->marker>0" failed: file "Pila.cc", line 72

```

```

bash-2.02$

```

```
#include <Pila.h>
```

```
int main() {
```

```
    Pila * s=crea(5);
```

```
    cout<<"s"; stampaStato(s);
```

```
    for (int k=1; k<10;k++) inserisci(s,k);
```

```
    cout<<"s"; stampaStato(s);
```

```
    Pila * w=s;
```

```
    cout<<"w"; stampaStato(w);
```

```
    for (int k=1; k<8;k++)
```

```
        cout<< estrai(s)<<endl;
```

```
    cout<<"s"; stampaStato(s);
```

```
    cout<<"w"; stampaStato(w);
```

```
}
```

Perchè abbiamo scritto
il metodo copia?

Una copia
(troppo)
sbrigativa...


```

s=====
size = 10
marker = 9
[1][2] [3][4] [5] [6] [7] [8] [9]
=====

```

```

w=====
size = 10
marker = 9
[1][2] [3][4] [5] [6] [7] [8] [9]
=====

```

```

entro in estrai
9
entro in estrai
8
...

```

```

...
entro in estrai
4
entro in estrai
3

```

```

s=====
size = 10
marker = 2
[1][2]
=====

```

```

w=====
size = 10
marker = 2
[1][2]
=====

```

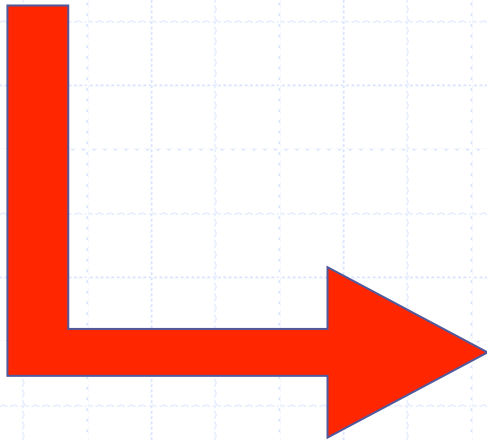
Pila.h

```
struct Pila {  
    int size;  
    int marker;  
    int * contenuto;  
};  
  
Pila * crea(int initialSize) ;  
void distruggi(Pila * s) ;  
Pila * copia(Pila * from) ;  
void cresci(Pila *s, int increment);  
void inserisci(Pila *s, int k) ;  
int estrai(Pila *s) ;  
void stampaStato(Pila *s) ;
```

```
struct Pila {  
    int size;  
    int marker;  
    int * contenuto;  
    int estrai() ;  
} ;  
Pila * crea(int initialSize) ;  
void distruggi(Pila * s) ;  
Pila * copia(Pila * from) ;  
void cresci(Pila *s, int increment);  
void inserisci(Pila *s, int k) ;  
// int estrai(Pila *s) ; vecchia versione  
void stampaStato(Pila *s) ;
```

```
int estrai(Pila *s) {  
    //estrai l' ultimo valore  
    cout<<"entro in estrai"<<endl;  
    assert(s->marker>0);  
    return s->contenuto[--(s->marker)];  
}
```

Re-implementazione di estrai



```
int estrai() {  
    //estrai l' ultimo valore  
    cout<<"entro in estrai"<<endl;  
    assert(this->marker>0);  
    return this->contenuto[--(this->marker)];  
}
```

Re-implementazione del main



```
int main() {
    Pila * s=crea(5);
    cout<<"s"; stampaStato(s);
    for (int k=1; k<10;k++) inserisci(s,k);
    cout<<"s"; stampaStato(s);
    Pila * w = copia(s);
    cout<<"w"; stampaStato(w);
    for (int k=1; k<8;k++)
        //cout<<estrai(s)<<endl;
        cout<<s->estrai()<<endl;
    ...
}
```

Re-implementazione di estrai: dove scrivo il codice?

```
struct Pila {  
    int size;  
    int marker;  
    int * contenuto;  
    int estrai() {  
        //estrai l'ultimo valore  
        cout<<"entro in estrai"<<endl;  
        assert(this->marker>0);  
        return this->contenuto[--(this->marker)];  
    }  
};
```

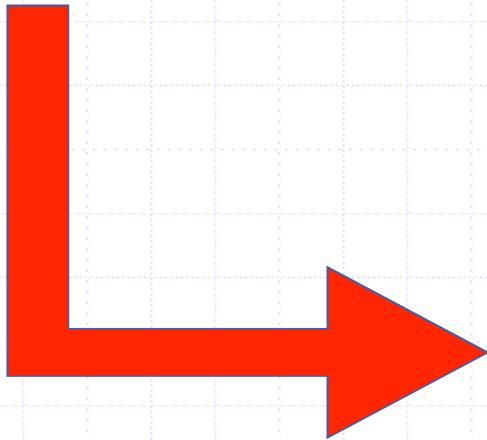
Re-implementazione di estrai: dove scrivo il codice?



```
struct Pila {  
    int size;  
    int marker;  
    int * contenuto;  
    int estrai();  
};  
int Pila::estrai() {  
    //estrai l'ultimo valore  
    cout<<"entro in estrai"<<endl;  
    assert(this->marker>0);  
    return this->contenuto[--(this->marker)];  
}
```

```
int estrai(Pila *s) {  
    //estrai l' ultimo valore  
    cout<<"entro in estrai"<<endl;  
    assert(s->marker>0);  
    return s->contenuto[--(s->marker)];  
}
```

Re-implementazione
di estrai con this
implicito

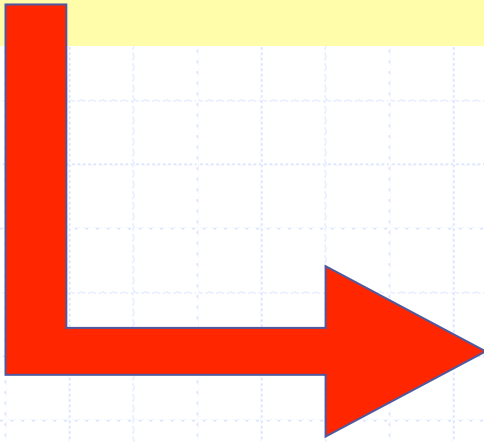


```
int estrai() {  
    //estrai l' ultimo valore  
    cout<<"entro in estrai"<<endl;  
    assert(marker>0);  
    return contenuto[--(marker)];  
}
```



```
Pila * crea(int initialSize) {  
    Pila * s= new Pila ;  
    s->size=initialSize;  
    s->marker=0;  
    s-> contenuto=new int[initialSize];  
    return s;  
}
```

Re-implementazione di crea



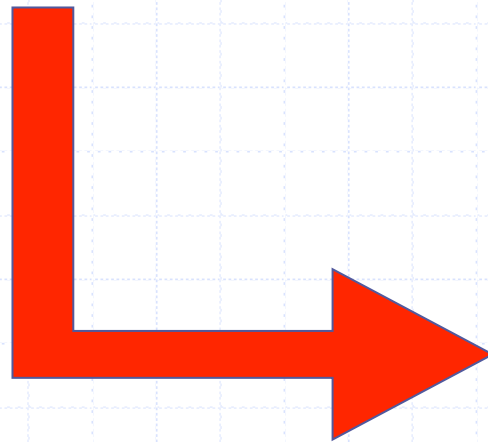
```
Pila::Pila(int initialSize) {  
    size=initialSize;  
    marker=0;  
    contenuto=new int[initialSize];  
}
```

“Il costruttore”

```
void Pila:: distruggi () {  
//distruggi lo Pila  
    cout<<"entro in distruggi"<<endl;  
    delete []contenuto;  
    delete this;  
}
```

Re-implementazione di distruggi

```
Pila::~~Pila() {  
//distruggi lo Pila  
    cout<<"entro nel distruttore"<<endl;  
    delete []contenuto;  
    // NO! delete this;  
}
```



"Il distruttore"

del main

```
int main() {
    Pila * s=new Pila(5); // OLD: =crea(5)
    cout<<"s"; s->stampaStato();
    for (int k=1; k<10;k++) s->inserisci(k);
    cout<<"s"; s->stampaStato();
    Pila * w = s->copia();
    cout<<"w"; w->stampaStato();
    for (int k=1; k<8;k++)
        cout<< s->estrai()<<endl;
    cout<<"s"; s->stampaStato();
    delete s; // OLD: s->distruuggi();
    cout<<"s"; s->stampaStato();
    for (int k=1; k<15;k++)
        cout<< w->estrai()<<endl;
    cout<<"w"; w->stampaStato();
}
```

versione 3

```
struct Pila {  
    int size;  
    int marker;  
    int * contenuto;  
    Pila(int initialSize) ;  
    ~Pila() ;  
    Pila * copia() ;  
    void cresci(int increment) ;  
    void inserisci(int k) ;  
    int estrai() ;  
    void stampaStato() ;  
} ;
```

Variabili di istanza,
Dati membro

Metodi,
Funzioni membro

```
struct Pila {  
    Pila(int initialSize) ;  
    Pila();  
    ~Pila() ;  
    void copia(Pila * to) ;  
    void inserisci(int k) ;  
    int estrai() ;  
    void stampaStato() ;  
  
    private:  
    int size;  
    int marker;  
    int * contenuto;  
    void cresci(int increment);  
};
```

Pila.h

versione 4

```
class Pila {  
    int size;  
    int marker;  
    int * contenuto;  
    void cresci(int increment);  
    public:  
        Pila(int initialSize) ;  
        Pila();  
        ~Pila() ;  
        void copy(Pila * to) ;  
        void inserisci(int k) ;  
        int estrai() ;  
        void stampaStato() ;  
};
```

Pila.h

versione 5

```
struct Pila {  
    private:  
        int size;  
        int marker;  
        int * contenuto;  
        void cresci(int increment);  
    public:  
        Pila(int initialSize) ;  
        Pila();  
        ~Pila() ;  
        void copy(Pila * to) ;  
        void inserisci(int k) ;  
        int estrai() ;  
        void stampaStato() ;  
};
```

```
class Pila {  
    private:  
        int size;  
        int marker;  
        int * contenuto;  
        void cresci(int increment);  
    public:  
        Pila(int initialSize) ;  
        Pila();  
        ~Pila() ;  
        void copy(Pila * to) ;  
        void inserisci(int k) ;  
        int estrai() ;  
        void stampaStato() ;  
};
```