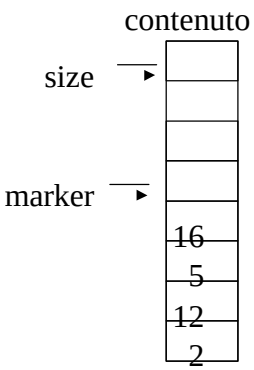
```
#include <iostream.h>
#include <cassert>
#define DEBUG

struct Pila {
  int size;
  int defaultGrowthSize;
  int marker;
  int * contenuto;
} ;
```




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

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

```
void cresci(Pila *s, int increment){
//aumenta la dimensione dello Pila
#ifdef DEBUG
  cout<<"entro in cresci"<<endl;
#endif
  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
#ifdef DEBUG
  cout<<"entro in inserisci"<<endl;
#endif
  if (s->size==s->marker)
     cresci(s,s->defaultGrowthSize);
  s->contenuto[s->marker]=k;
  s->marker++;
}
```

```
int estrai(Pila *s) {
//estrai l'ultimo valore
#ifdef DEBUG
  cout<<"entro in estrai"<<endl;
#endif
  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 << "defaultGrowthSize = “
          <<s->defaultGrowthSize<<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) {
#ifdef DEBUG
  cout<<"entro in copia"<<endl;
#endif
  Pila * to=crea(from->size);
  to->defaultGrowthSize=from->defaultGrowthSize;
  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  -lstdc++  -o Pila.exe
bash-2.02$ Pila.exe
entro in crea
s==================
size = 5
defaultGrowthSize = 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
defaultGrowthSize = 5
marker = 9
[1][2][3][4][5][6][7][8][9]
==================
entro in copia
w==================
size = 10
defaultGrowthSize = 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
defaultGrowthSize = 5
marker = 2
[1][2]
==================
```

```
entro in distruggi
s==================
size = 1627775824
defaultGrowthSize = 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$
```

Perchè abbiamo scritto il metodo copia?

```
#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);
}
```

Una copia (troppo) sbrigativa…

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

Pila.h

```
struct Pila {
  int size;
  int defaultGrowthSize;
  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) ;
```

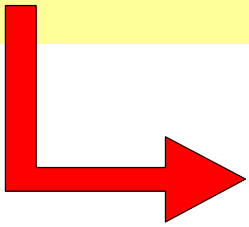
Pila.h

versione 2

```
struct Pila {
  int size;
  int defaultGrowthSize;
  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) ;
```

Re-implementazione di estrai

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



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

Re-implementazione
di estrai: dove scrivo il codice?

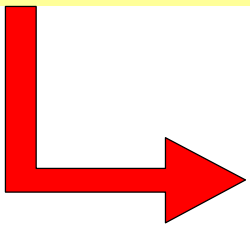
```
struct Pila {
  int size;
  int defaultGrowthSize;
  int marker;
  int * contenuto;
  int estrai() {
      //estrai l'ultimo valore
#ifdef DEBUG
      cout<<"entro in estrai"<<endl;
#endif
      assert(this->marker>0);
      return this->contenuto[--(this->marker)];
   }
};
```

Re-implementazione
di estrai: dove scrivo il codice?

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

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

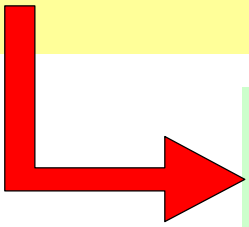
Re-implementazione di estrai con this implicito



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

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

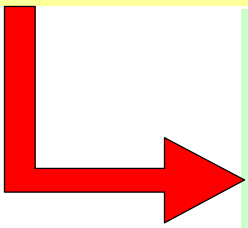
Re-implementazione di crea



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

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

Re-implementazione di distruggi



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

Re-implementazione del main

```
int main() {
  Pila  * s=new Pila(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;
   cout<<"s"; s->stampaStato();
   for (int k=1; k<15;k++)
      cout<< w->estrai()<<endl;
   cout<<"w"; w->stampaStato();
}
```

Pila.h
versione 3

```
struct Pila {
  int size;
  int defaultGrowthSize;
  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,
Funzoni membro

Polimorfismo

Possono esistere piu' metodi con uguale nome!

La FIRMA (signature) di un metodo e' data da

NOME + tipo degli argomenti passati

(Il tipo del valore di ritorno non conta!)

Vale anche per i costruttori

Overloading del costruttore

```
Pila::Pila() { //crea una Pila
  cout<<"eseguo il costruttore Pila()"<<endl;
  int initialSize=10;
  this->size=initialSize;
  this->defaultGrowthSize=initialSize;
  this->marker=0;
  this->contenuto=new int[initialSize];}
//-----------------------------------------
Pila::Pila(int initialSize) { //crea una Pila
   cout<<"eseguo il costruttore Pila(int)"<<endl;
   this->size=initialSize;
   this->defaultGrowthSize=initialSize;
   this->marker=0;
   this->contenuto=new int[initialSize];
}
```

Overloading del costruttore

```
void Pila::inizializza(int initialSize) {
  this->size=initialSize;
  this->defaultGrowthSize=initialSize;
  this->marker=0;
  this->contenuto=new int[initialSize];
}
//-----------------------------------------
Pila::Pila() { //crea una Pila
  cout<<"eseguo il costruttore Pila()"<<endl;
  inizializza(15); //  valore di default
}
//-----------------------------------------
Pila::Pila(int initialSize) { //crea una Pila
   cout<<"eseguo il costruttore Pila(int)"<<endl;
   inizializza(initialSize);
}
```

Overloading del costruttore

```
inline void Pila::inizializza(int initialSize) {
  this->size=initialSize;
  this->defaultGrowthSize=initialSize;
  this->marker=0;
  this->contenuto=new int[initialSize];
}
//----------------------------------------
Pila::Pila() { //crea una Pila
  cout<<"eseguo il costruttore Pila()"<<endl;
  inizializza(15); //  valore di default
}
//----------------------------------------
Pila::Pila(int initialSize) { //crea una Pila
   cout<<"eseguo il costruttore Pila(int)"<<endl;
   inizializza(initialSize);
}
```

Overloading del costruttore

Parametri con valori di default

```
Pila::Pila(int initialSize=10) { //crea una Pila
   cout<<"eseguo il costruttore Pila(int)"<<endl;
   this->size=initialSize;
   this->defaultGrowthSize=initialSize;
   this->marker=0;
   this->contenuto=new int[initialSize];
}
```

Pila.h
versione 4

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

Pila.h
versione 5

```
class Pila {
        int size;
        int defaultGrowthSize;
        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 6

```
struct Pila {
  private:
        int size;
        int defaultGrowthSize;
        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 defaultGrowthSize;
        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() ;
} ;
```

Oggetti in Stack

```
int main() {
  //Una Pila in Stack…
  Pila s(5);
  cout<<"s";s.stampaStato();

  for (int k=1; k<10;k++) s.inserisci(k);
  cout<<"s";s.stampaStato();

  Pila * w=new Pila(10);
  cout<<"w"; w->stampaStato(w);
  s.copy(w);
  cout<<"w";w->stampaStato();

 for (int k=1; k<8;k++)
        cout<<s.estrai()<<endl;
 cout<<"s"; s.stampaStato();
```

Re-implementazione del main

```
//NO! delete &s;

cout<<"w";w->stampaStato();

for (int k=1; k<15;k++)
        cout<<w->estrai()<<endl;
cout<<"w";w->stampaStato();
}
```