

# Fondamenti di Programmazione (A)

I8 - Allocazione dinamica della memoria

Vincenzo Arceri - Università degli Studi di Parma - [vincenzo.arceri@unipr.it](mailto:vincenzo.arceri@unipr.it)

# Puntate precedenti

- Funzioni
- Passaggio di parametri
  - Passaggio per valore
  - Passaggio per riferimento
  - Differenze fra C e C++
- Call stack
- Funzioni ricorsive

# Allocazione statica della memoria

- Fino ad ora, tutte le variabili dichiarate occupano una quantità di memoria nota a *compile-time*

# Allocazione statica della memoria

- Fino ad ora, tutte le variabili dichiarate occupano una quantità di memoria nota a *compile-time*

```
int x = 10;
```

```
persona p = {"Vincenzo", "Arceri", 31};
```

```
int a[5];
```

# Allocazione statica della memoria

- Fino ad ora, tutte le variabili dichiarate occupano una quantità di memoria nota a *compile-time*

```
int x = 10;
```

```
persona p = {"Vincenzo", "Arceri", 31};
```

```
int a[5];
```

- Non sempre è possibile sapere, prima di eseguire un programma, di *quanta memoria* il programma avrà bisogno

# Layout della memoria

C++



...



...



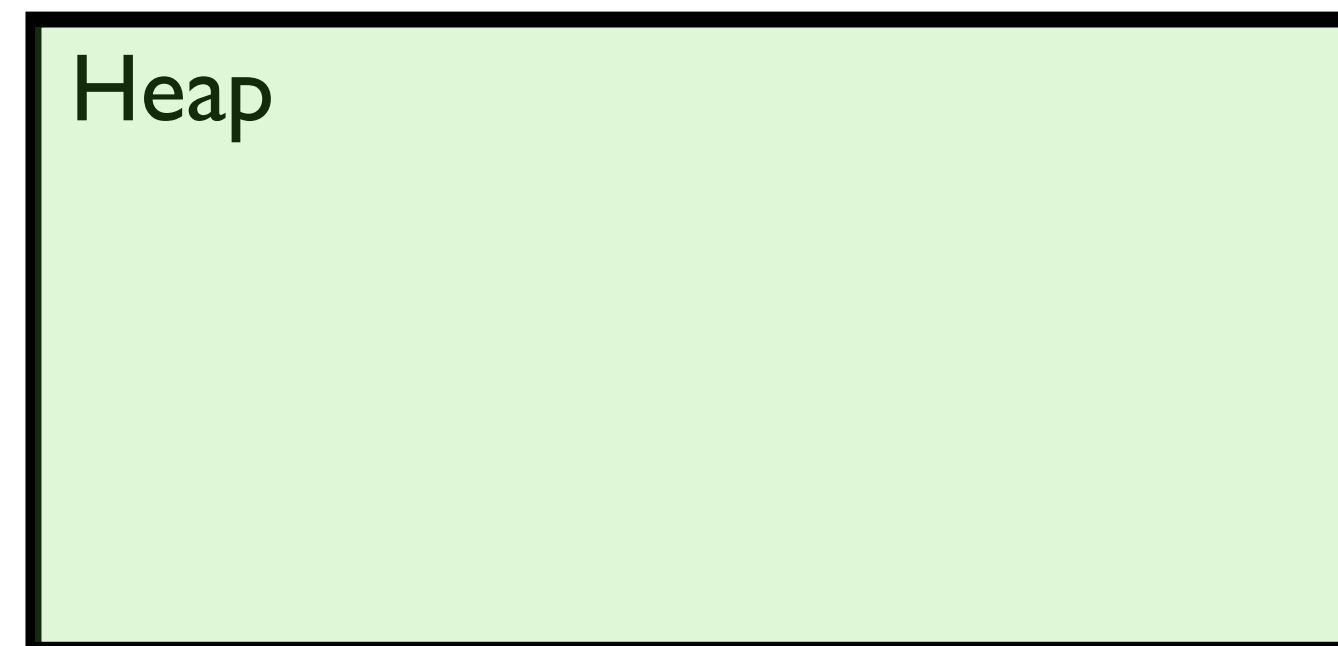
...

# Layout della memoria

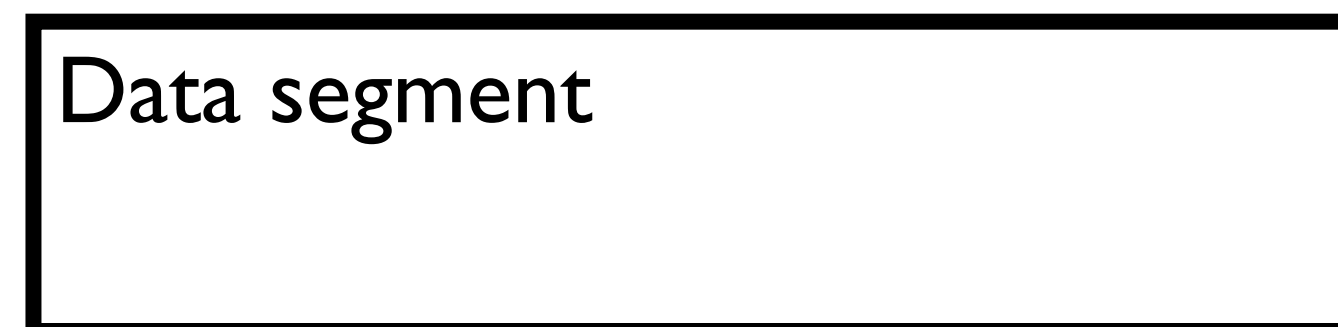
C++



...



...



...

Allocazione dinamica della memoria

# Esempio



# Esempio

- Memorizzare l'anagrafica di uno studente di un corso in un array

# Esempio

- Memorizzare l'anagrafica di uno studente di un corso in un array

```
studente corso[170];
```

# Esempio

- Memorizzare l'anagrafica di uno studente di un corso in un array

```
studente corso[170];
```

- Riservo memoria sufficiente per 170 studenti

# Esempio

- Memorizzare l'anagrafica di uno studente di un corso in un array

```
studente corso[170];
```

- Riservo memoria sufficiente per 170 studenti
- E se aumentano?

# Esempio

- Memorizzare l'anagrafica di uno studente di un corso in un array

```
studente corso[170];
```

- Riservo memoria sufficiente per 170 studenti
- E se aumentano?

```
studente corso[180];
```

# Esempio

- Memorizzare l'anagrafica di uno studente di un corso in un array

```
studente corso[170];
```

- Riservo memoria sufficiente per 170 studenti
- E se aumentano?

```
studente corso[180];
```

- **Problema:** cambio lunghezza dell'array, ricompilare il programma...

# Esempio

# Esempio

- Memorizzare una stringa liberamente digitata dall'utente
- La sua lunghezza non è nota a *compile-time*



# Esempio

- Memorizzare una stringa liberamente digitata dall'utente
- La sua lunghezza non è nota a *compile-time*
- **Problema:** dovrei preallocare in memoria un array di dimensioni enormi con conseguente spreco di memoria (es.: preloco un array di 1 M di elementi, ma l'utente inserisce "ciao")

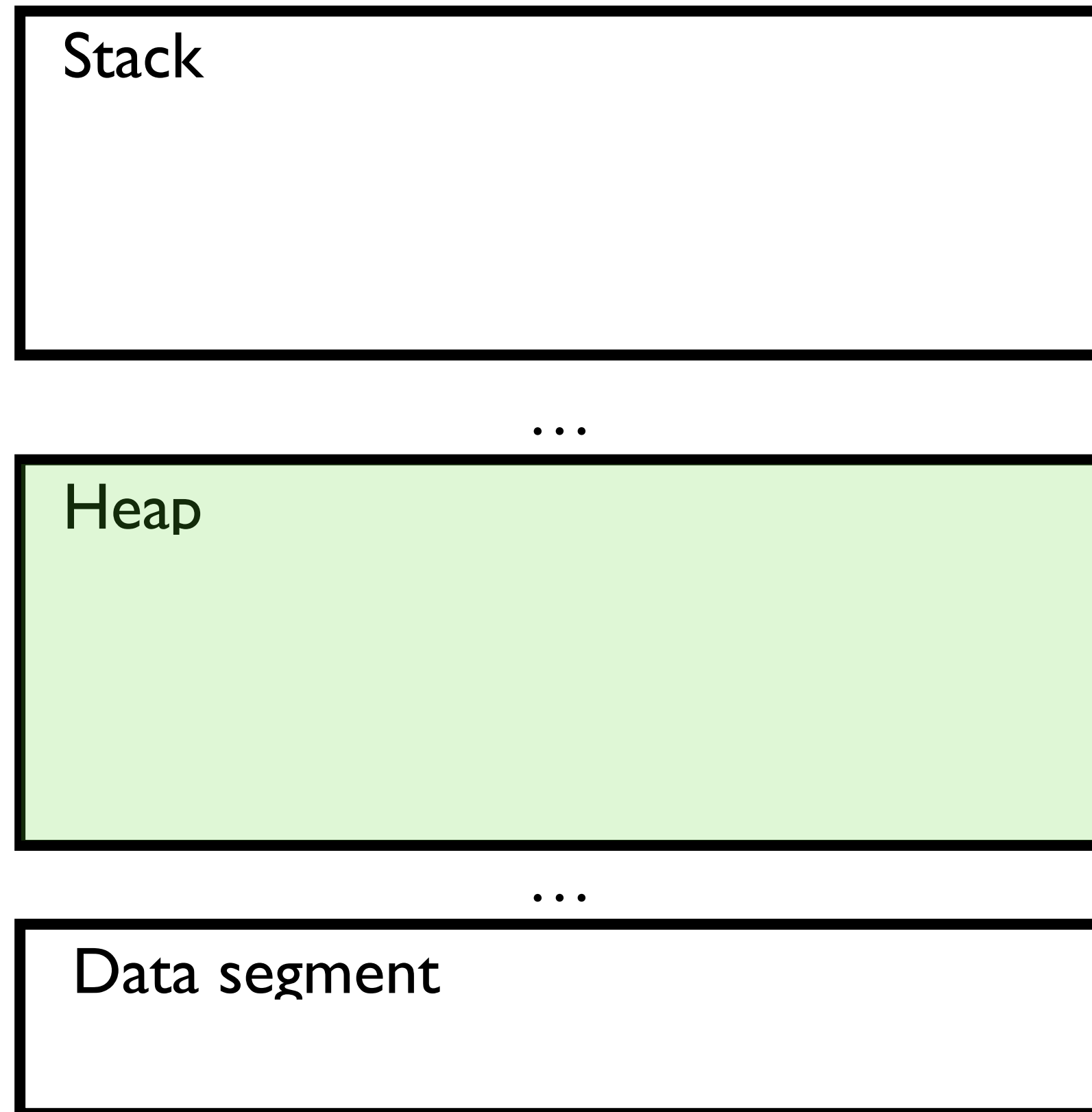
# Allocazione dinamica della memoria

# Allocazione dinamica della memoria

- Heap: parte della memoria che viene allocata dinamicamente (a *run-time*, durante l'esecuzione del programma)

# Allocazione dinamica della memoria

- Heap: parte della memoria che viene allocata dinamicamente (a *run-time*, durante l'esecuzione del programma)



# Allocazione dinamica della memoria

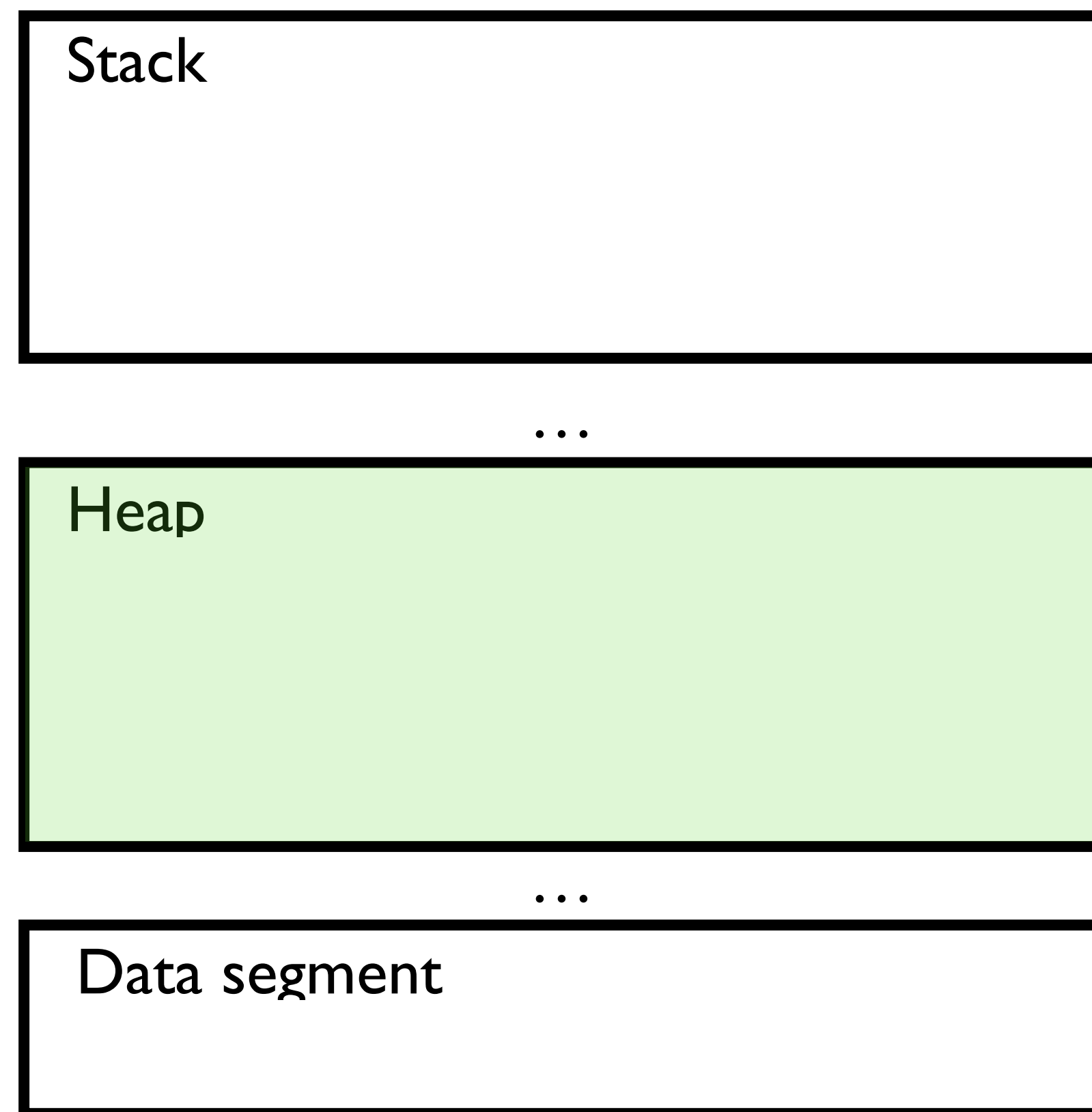
- Heap: parte della memoria che viene allocata dinamicamente (a *run-time*, durante l'esecuzione del programma)



- C++ prevede due operatori per la gestione dinamica della memoria:

# Allocazione dinamica della memoria

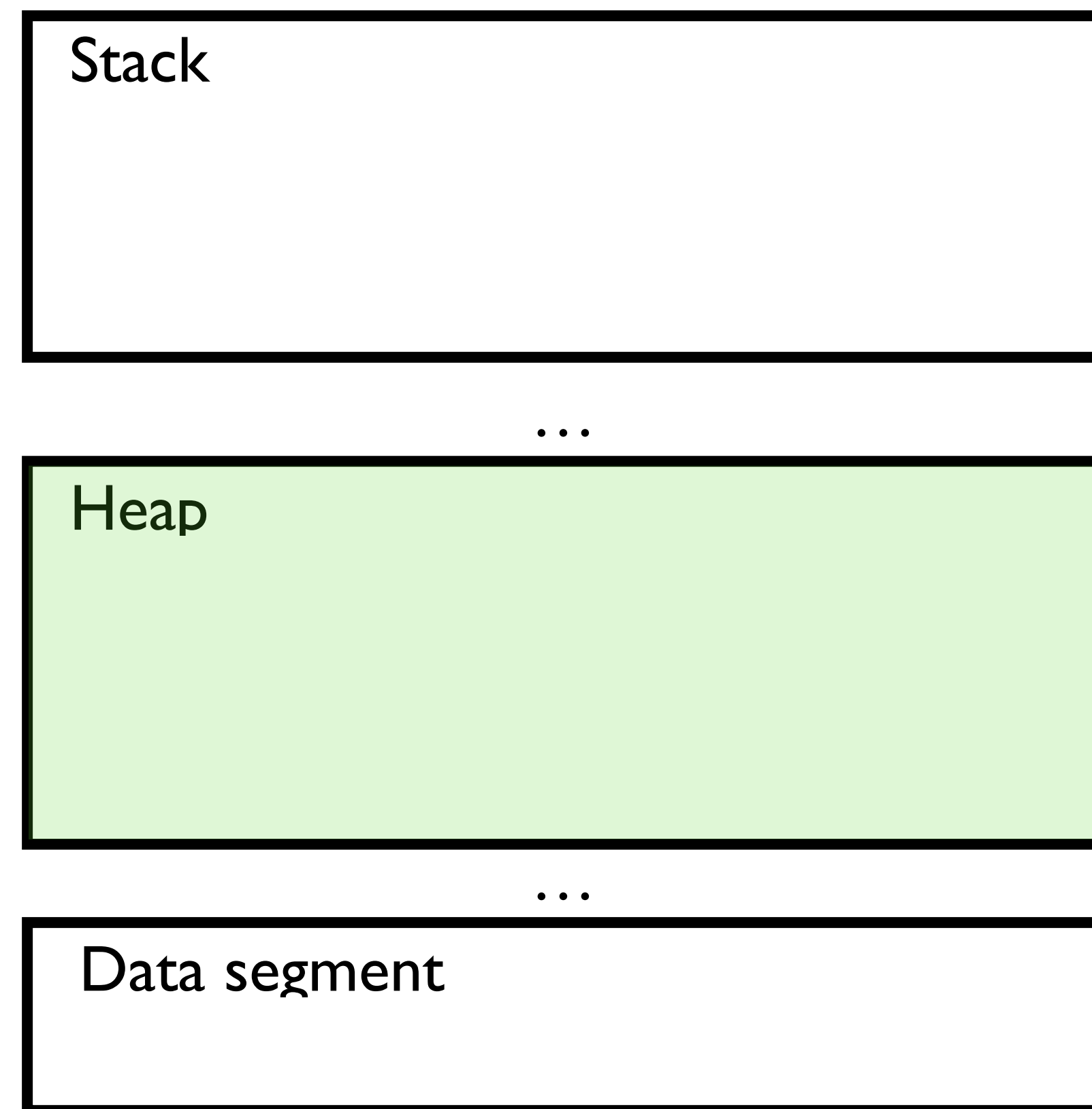
- Heap: parte della memoria che viene allocata dinamicamente (a *run-time*, durante l'esecuzione del programma)



- C++ prevede due operatori per la gestione dinamica della memoria:
  - `new` : “*alloca nello heap spazio sufficiente per contenere valori di tipo T*”

# Allocazione dinamica della memoria

- Heap: parte della memoria che viene allocata dinamicamente (a *run-time*, durante l'esecuzione del programma)



- C++ prevede due operatori per la gestione dinamica della memoria:
  - `new` : “*alloca nello heap spazio sufficiente per contenere valori di tipo T*”
  - `delete`: “*libera dallo heap memoria precedentemente allocata*”

# Allocazione dinamica della memoria

Operatore `new`

`new t`



# Allocazione dinamica della memoria

Operatore `new`

`new t`

- Genera dinamicamente una variabile di tipo *t*

# Allocazione dinamica della memoria

Operatore `new`

`new t`

- Genera dinamicamente una variabile di tipo *t*
- Nello heap viene allocata una zona di memoria della dimensione del tipo *t*

# Allocazione dinamica della memoria

Operatore `new`

`new t`

- Genera dinamicamente una variabile di tipo *t*
- Nello heap viene allocata una zona di memoria della dimensione del tipo *t*
- Ritorna un **puntatore all'indirizzo di memoria allocata**

# Allocazione dinamica della memoria

Operatore `new`

`new t`

- Genera dinamicamente una variabile di tipo *t*
- Nello heap viene allocata una zona di memoria della dimensione del tipo *t*
- Ritorna un **puntatore all'indirizzo di memoria allocata**
- Sintassi

# Allocazione dinamica della memoria

Operatore `new`

`new t`

- Genera dinamicamente una variabile di tipo *t*
- Nello heap viene allocata una zona di memoria della dimensione del tipo *t*
- Ritorna un **puntatore all'indirizzo di memoria allocata**
- Sintassi
  - `new t` per tipi non array

# Allocazione dinamica della memoria

Operatore `new`


`new t`

- Genera dinamicamente una variabile di tipo *t*
- Nello heap viene allocata una zona di memoria della dimensione del tipo *t*
- Ritorna un **puntatore all'indirizzo di memoria allocata**
- Sintassi
  - `new t` per tipi non array
  - `new t[dim]` per tipi array

# Allocazione dinamica della memoria

## Operatore `new`

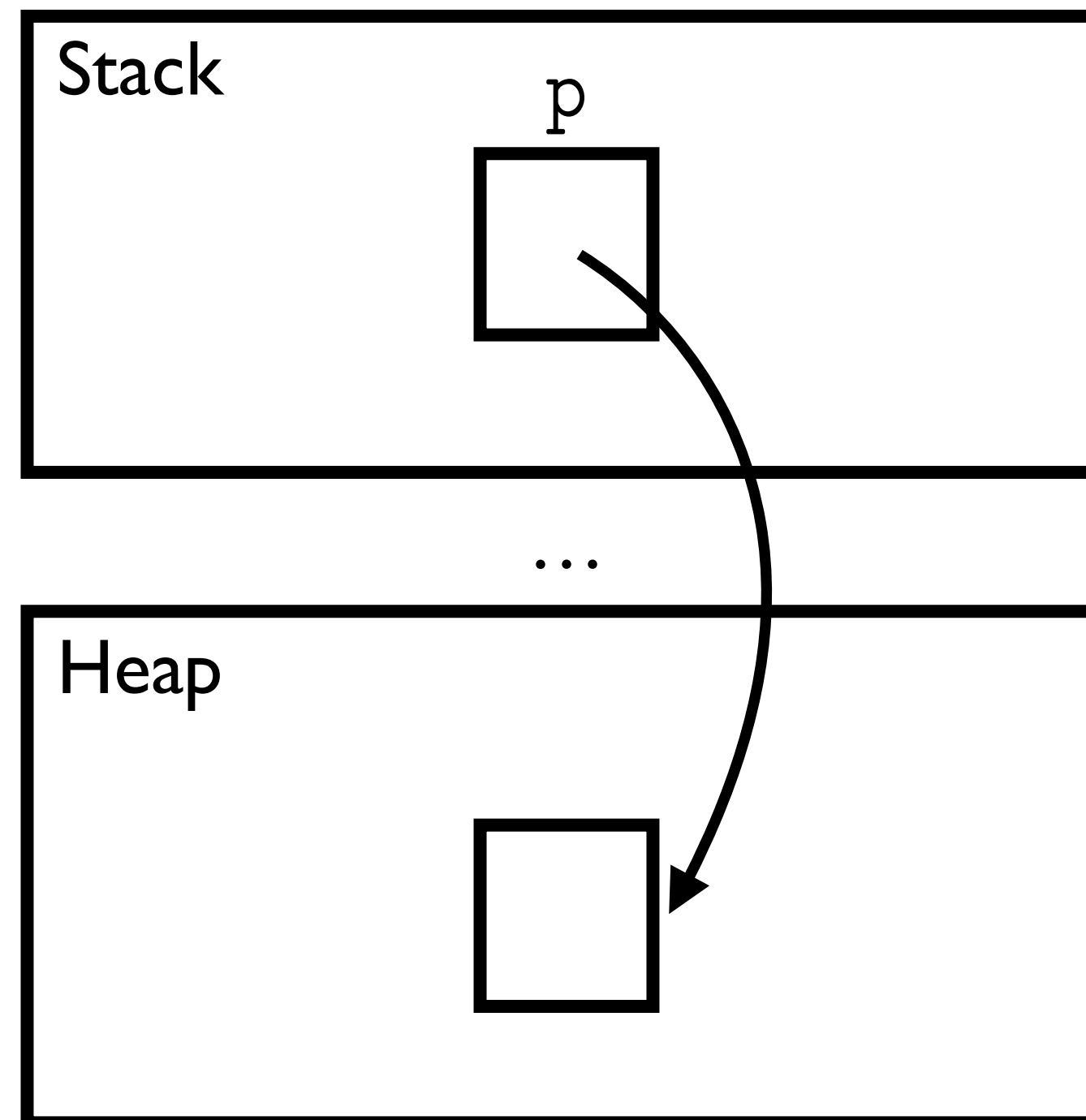
`new t`

- Genera dinamicamente una variabile di tipo *t*
  - Nello heap viene allocata una zona di memoria della dimensione del tipo *t*
  - Ritorna un **puntatore all'indirizzo di memoria allocata**
  - Sintassi
    - `new t` per tipi non array
    - `new t[dim]` per tipi array
- dim* è un'espressione intera!
- 

# Allocazione dinamica della memoria

## Operatore new

```
int * p = new int;
```

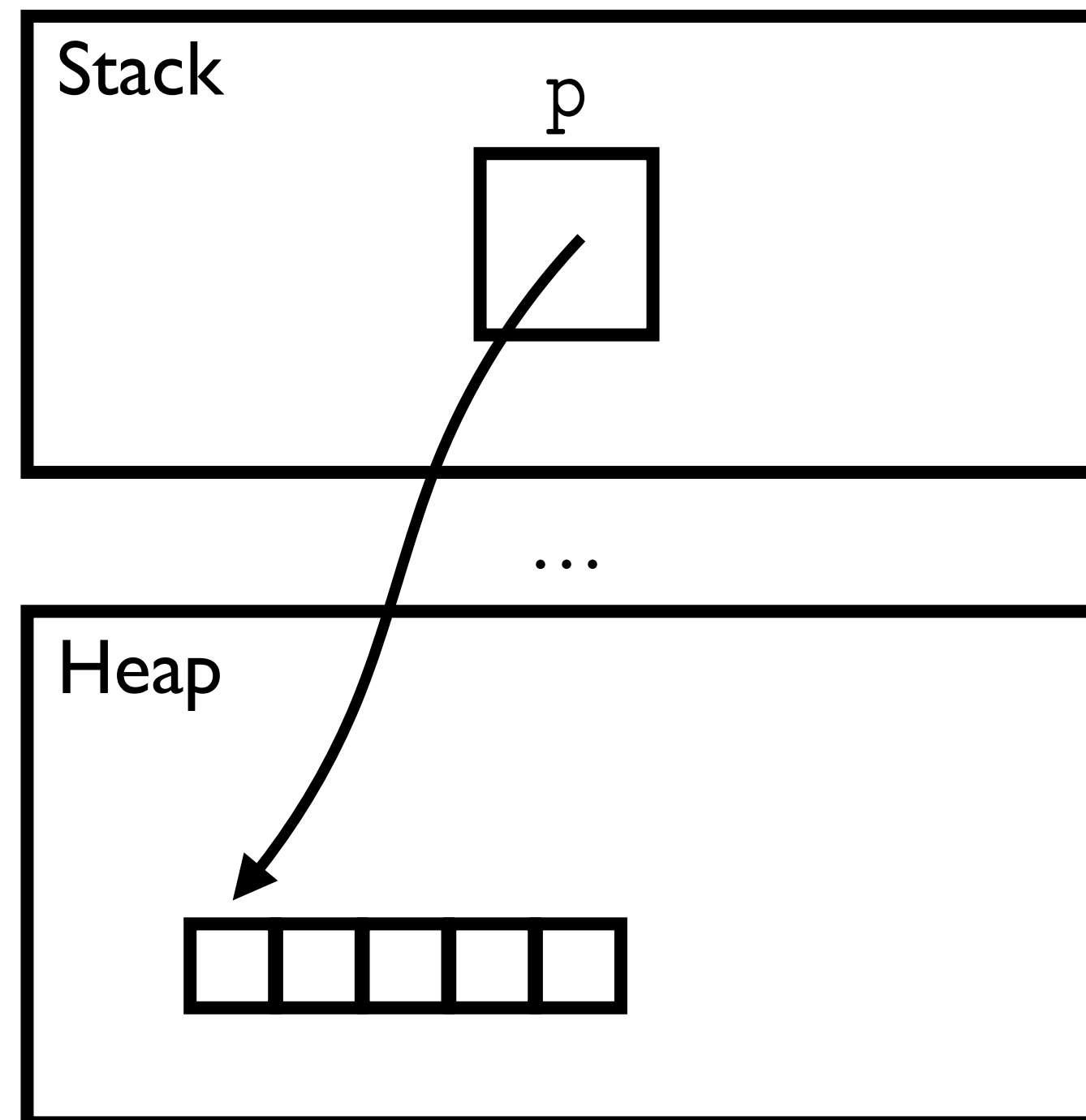




# Allocazione dinamica della memoria

## Operatore new

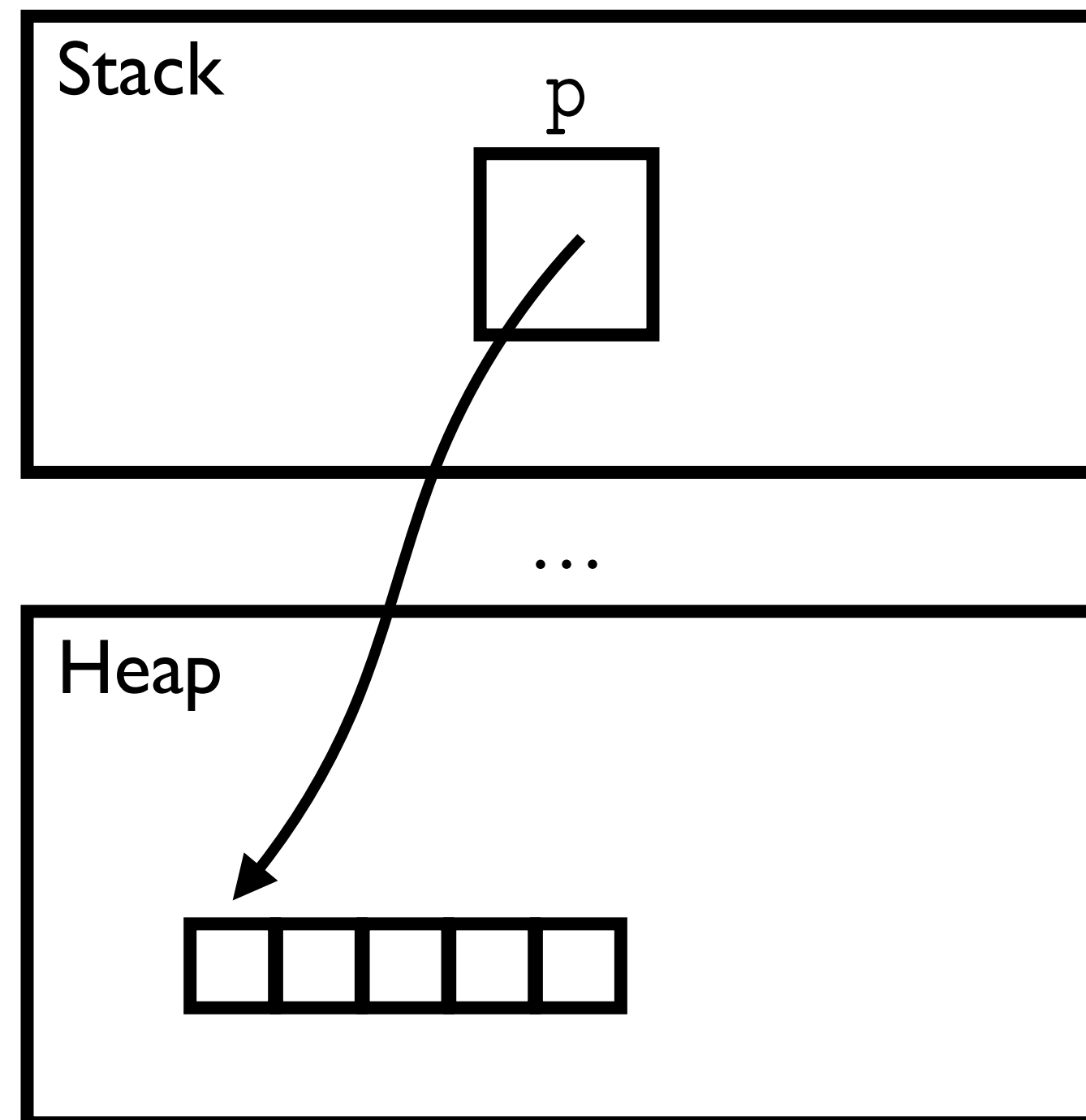
```
int * p = new int[5];
```



# Allocazione dinamica della memoria

Operatore new

```
int * p;  
p = new int[5];
```

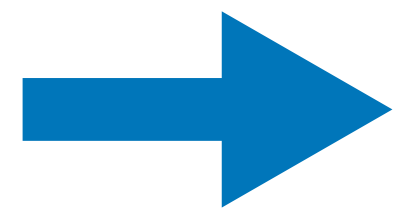


# Allocazione dinamica della memoria

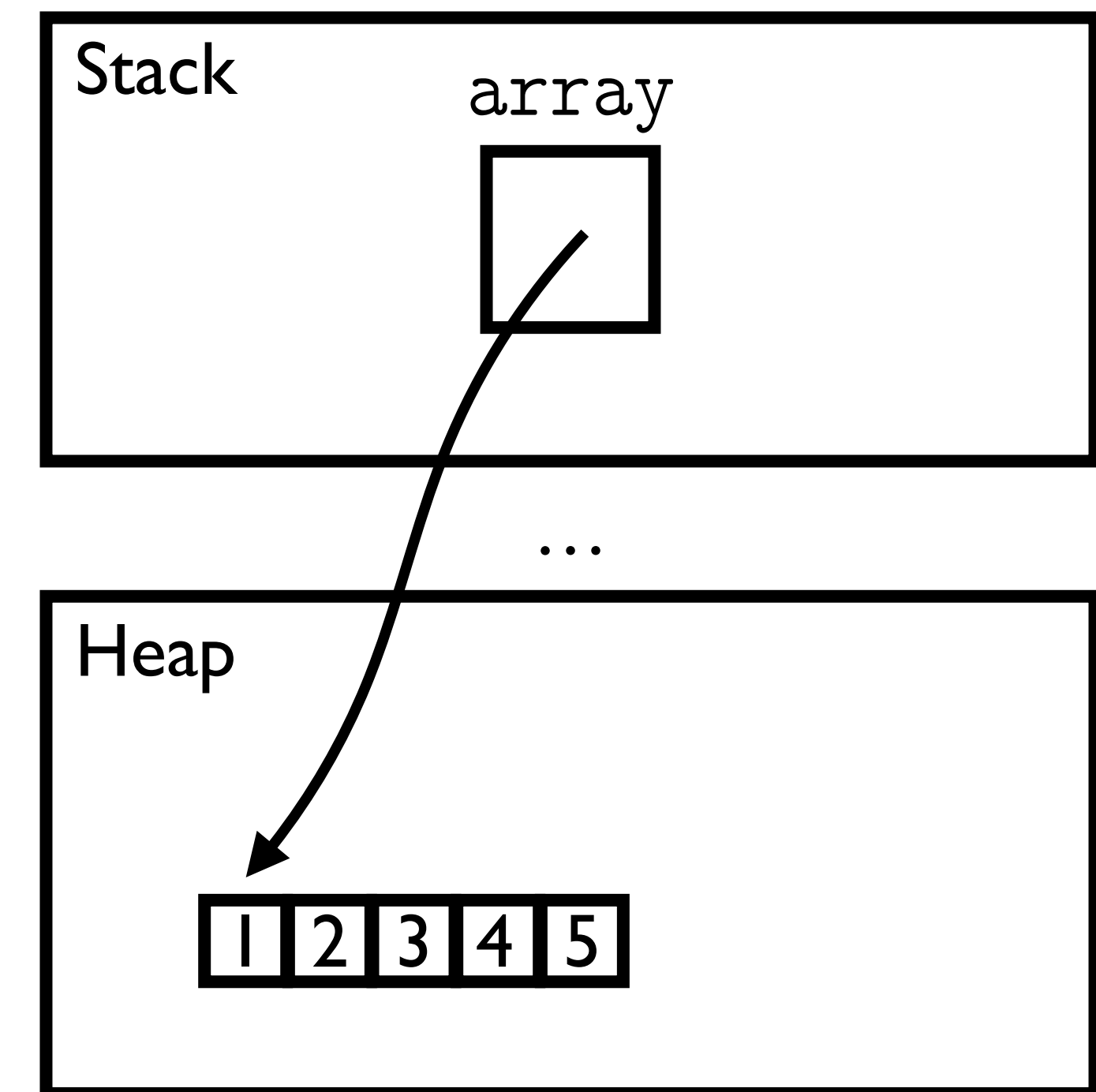
## Esercizio

- **Problema:** memorizzare in un array dinamico `array` una serie indefinita di interi positivi inseriti dall'utente (inserimento termina quando l'utente digita -1). L'array inizialmente contiene 5 elementi.
- Si scriva una funzione `raddoppia` che, preso in input l'array dinamico `array` e la sua dimensione, raddoppi la sua dimensione per poter consentire nuove memorizzazioni al suo interno.

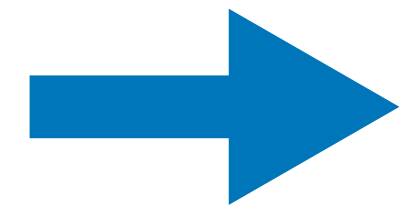
# Allocazione dinamica della memoria



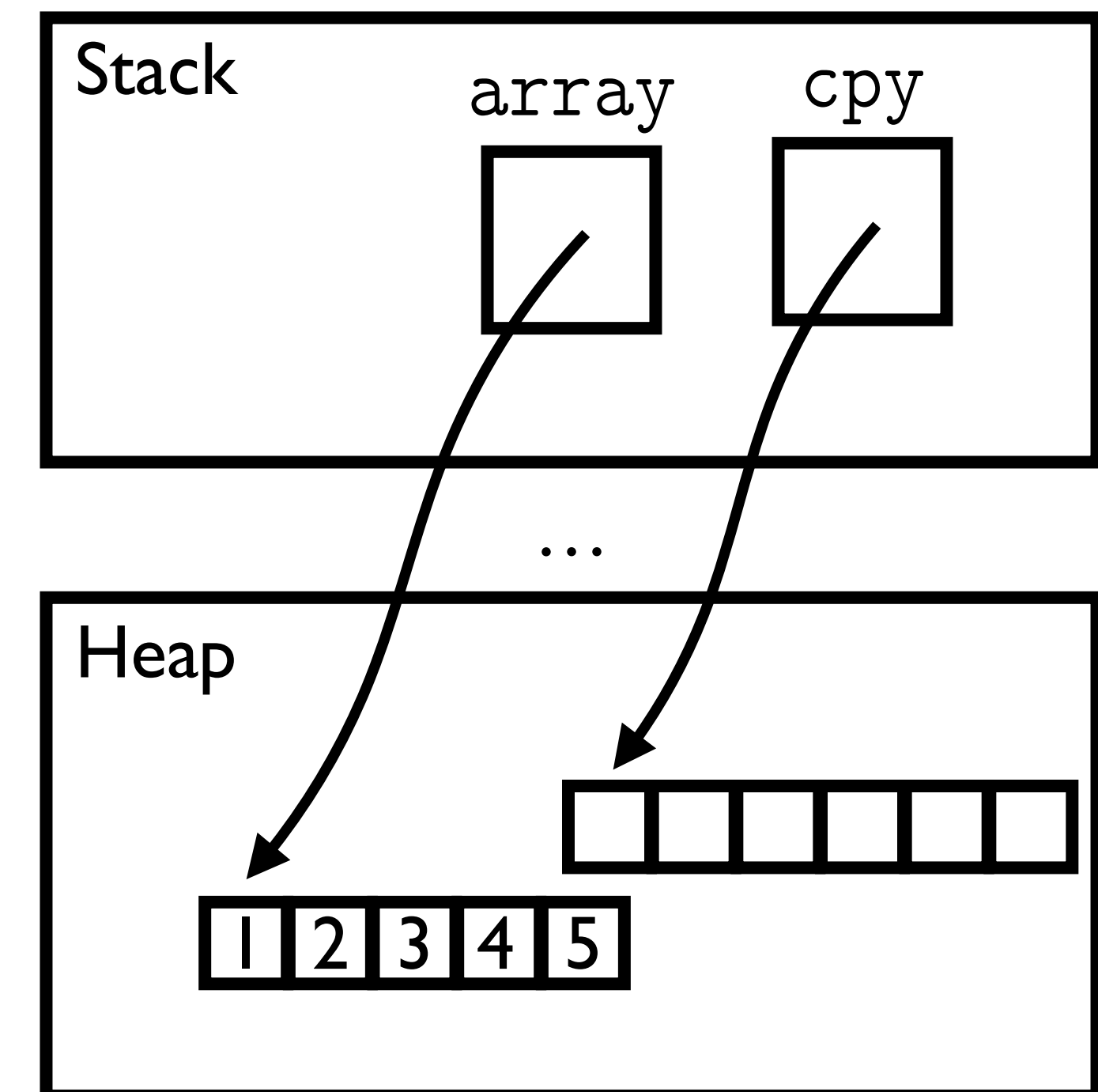
```
int* cpy = new int[dim + 1];  
  
for (int j = 0; j < dim; j++)  
    cpy[j] = array[j];  
  
cpy[i] = num;  
i++;  
array = cpy;  
dim++;
```



# Allocazione dinamica della memoria



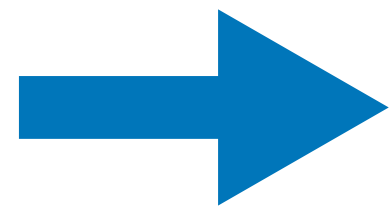
```
int* cpy = new int[dim + 1];  
  
for (int j = 0; j < dim; j++)  
    cpy[j] = array[j];  
  
cpy[i] = num;  
i++;  
array = cpy;  
dim++;
```



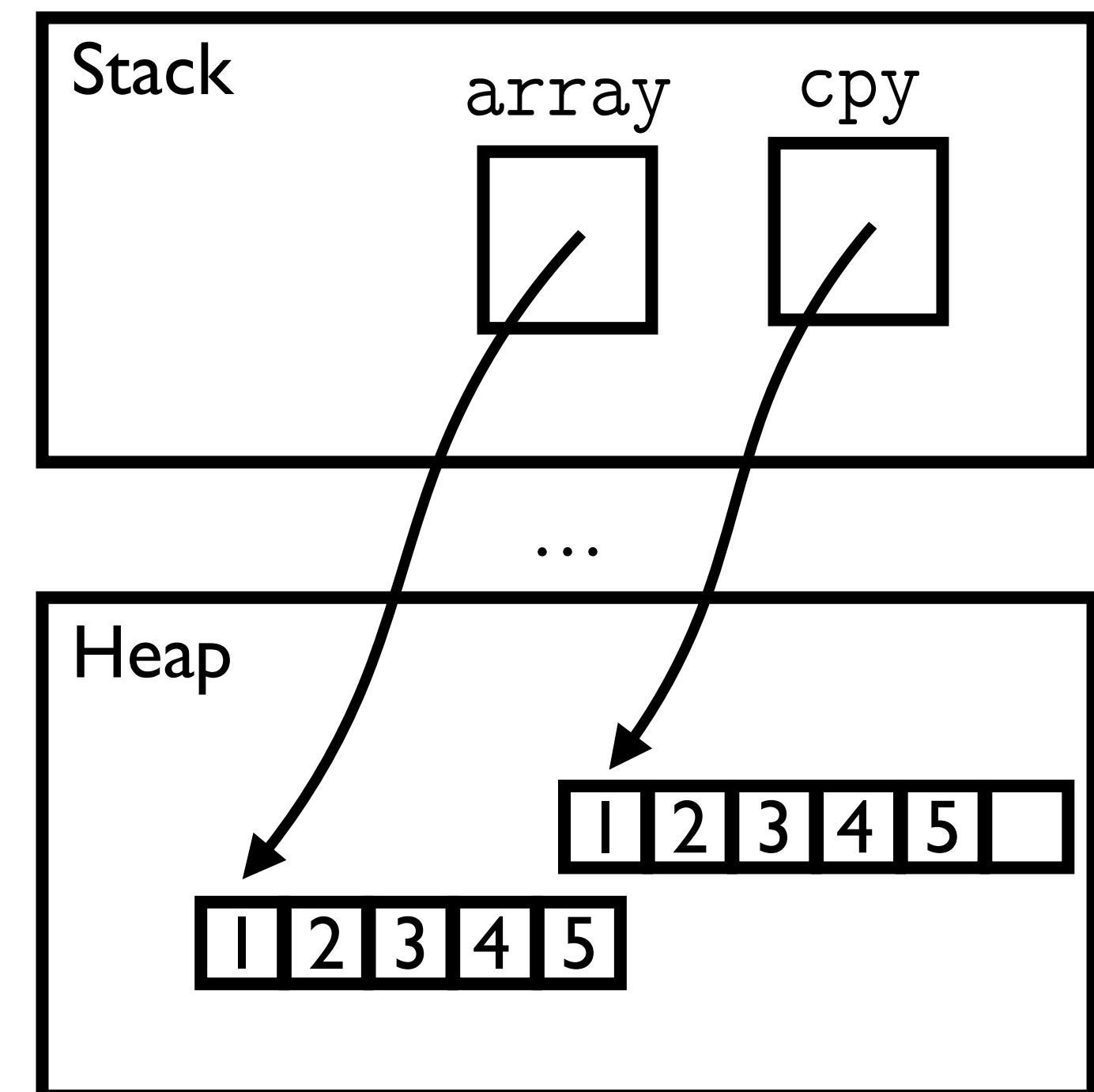
# Allocazione dinamica della memoria

```
int* cpy = new int[dim + 1];
```

```
for (int j = 0; j < dim; j++)  
    cpy[j] = array[j];
```

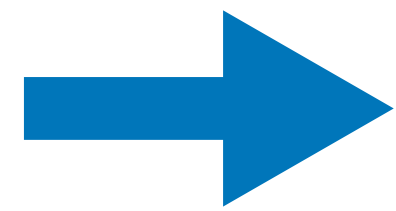


```
cpy[i] = num;  
i++;  
array = cpy;  
dim++;
```

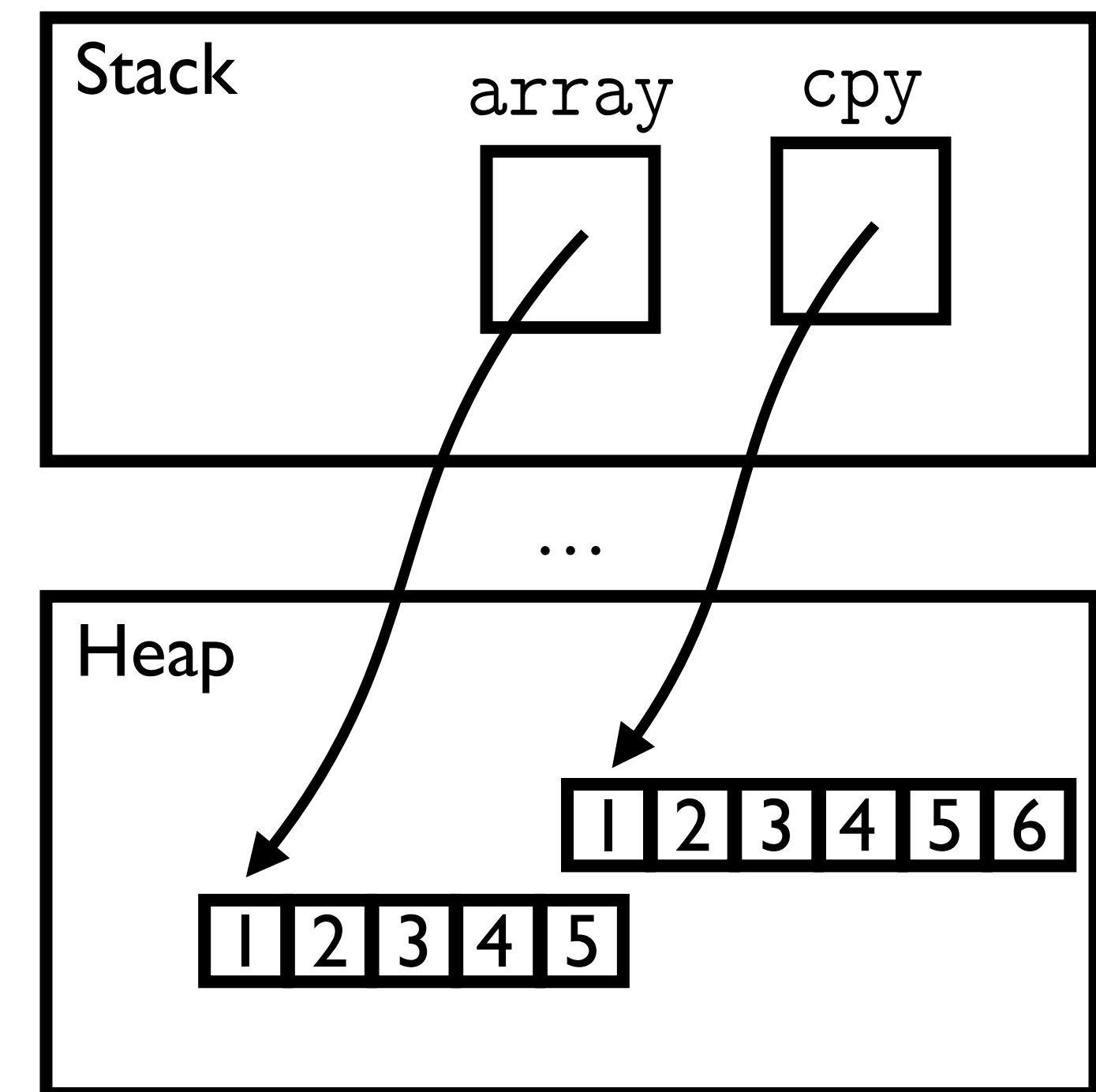


# Allocazione dinamica della memoria

```
int* cpy = new int[dim + 1];  
  
for (int j = 0; j < dim; j++)  
    cpy[j] = array[j];
```

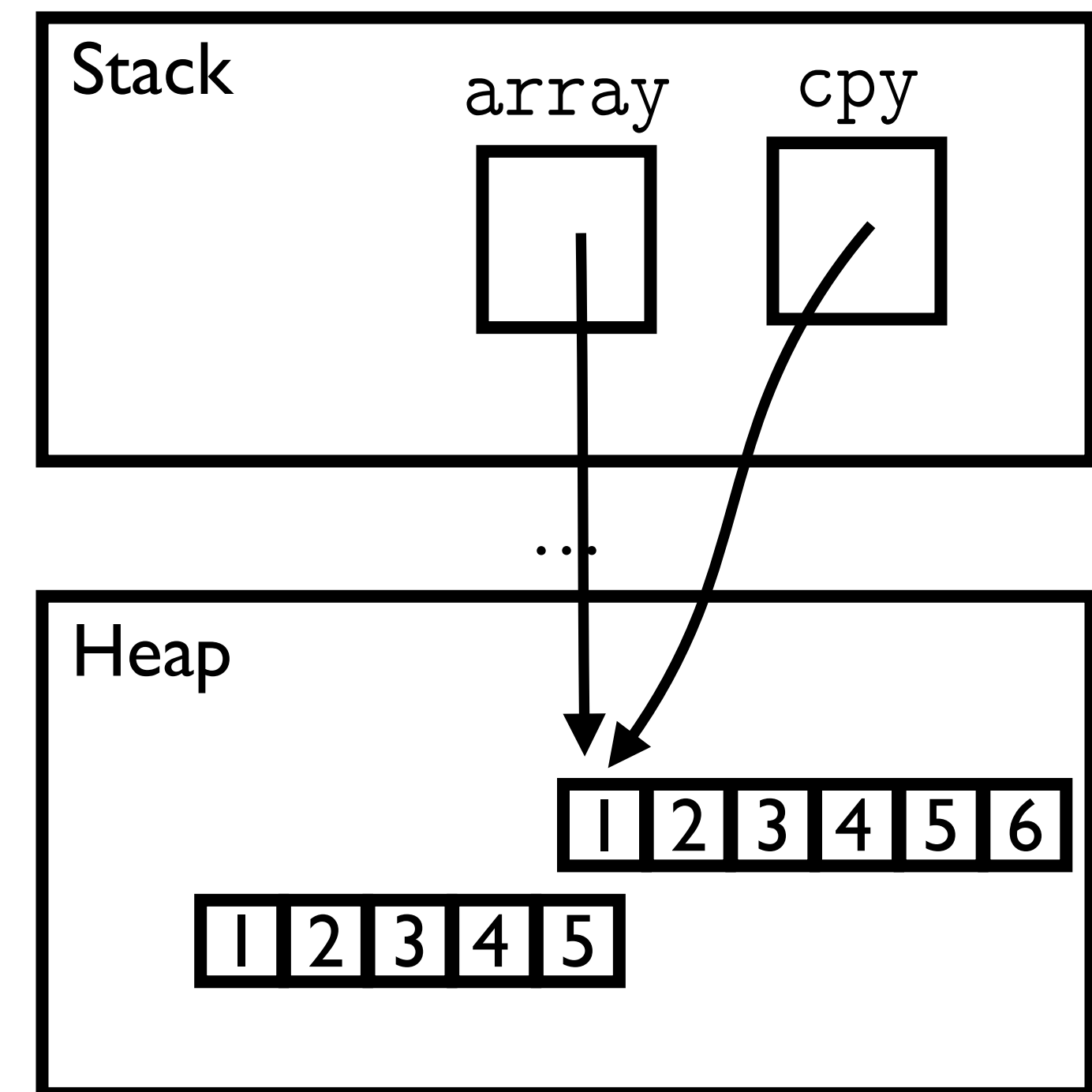
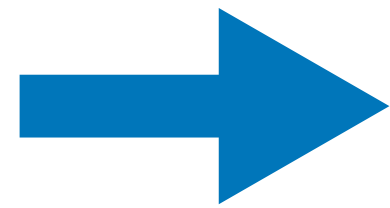


```
cpy[i] = num;  
i++;  
array = cpy;  
dim++;
```



# Allocazione dinamica della memoria

```
int* cpy = new int[dim + 1];  
  
for (int j = 0; j < dim; j++)  
    cpy[j] = array[j];  
  
cpy[i] = num;  
i++;  
array = cpy;  
dim++;
```

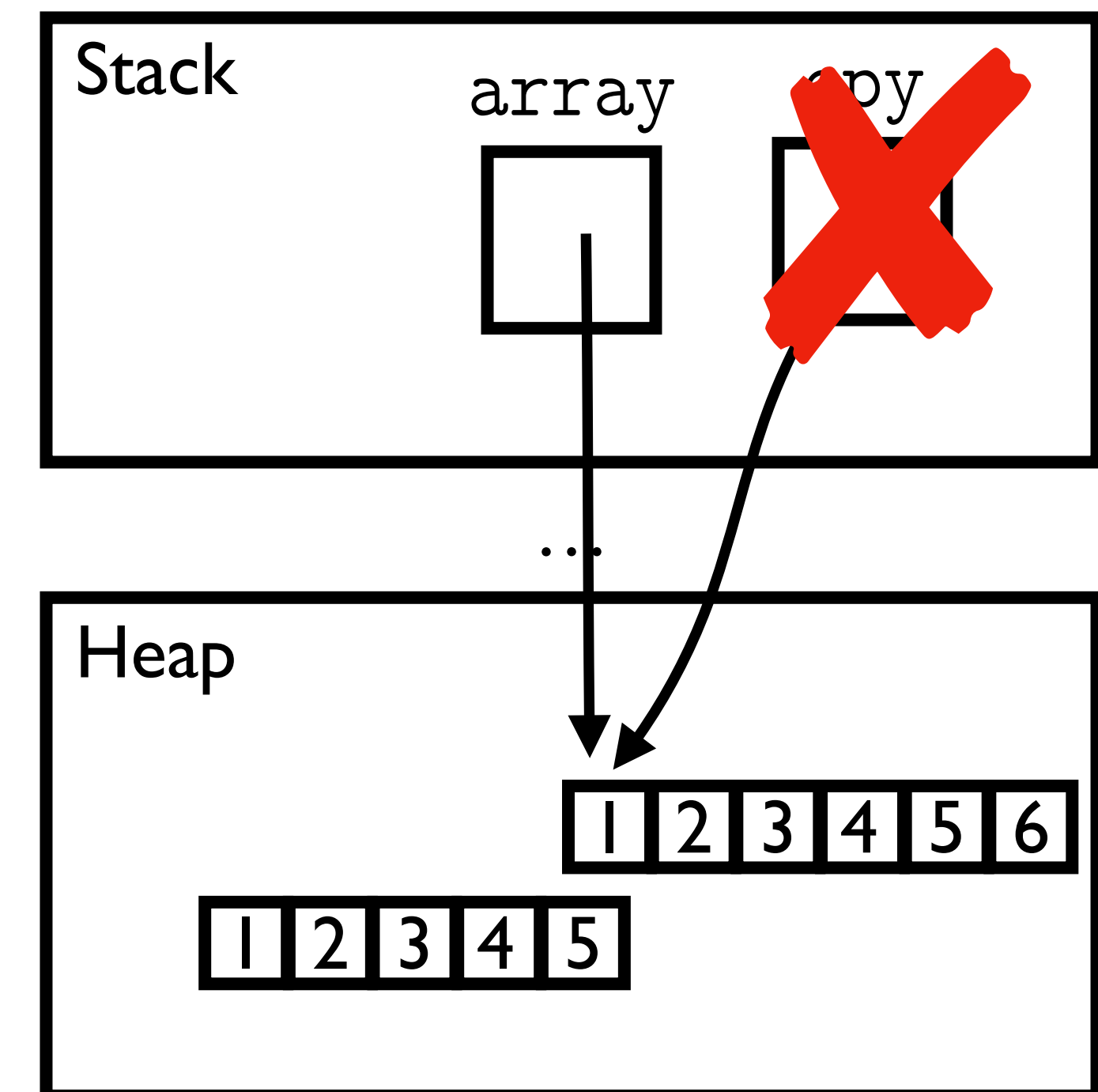
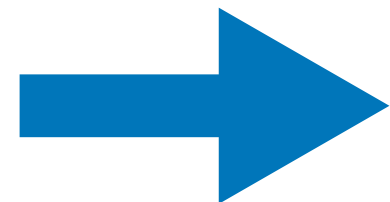




# Allocazione dinamica della memoria

cpy verrà distrutta all'uscita dal blocco

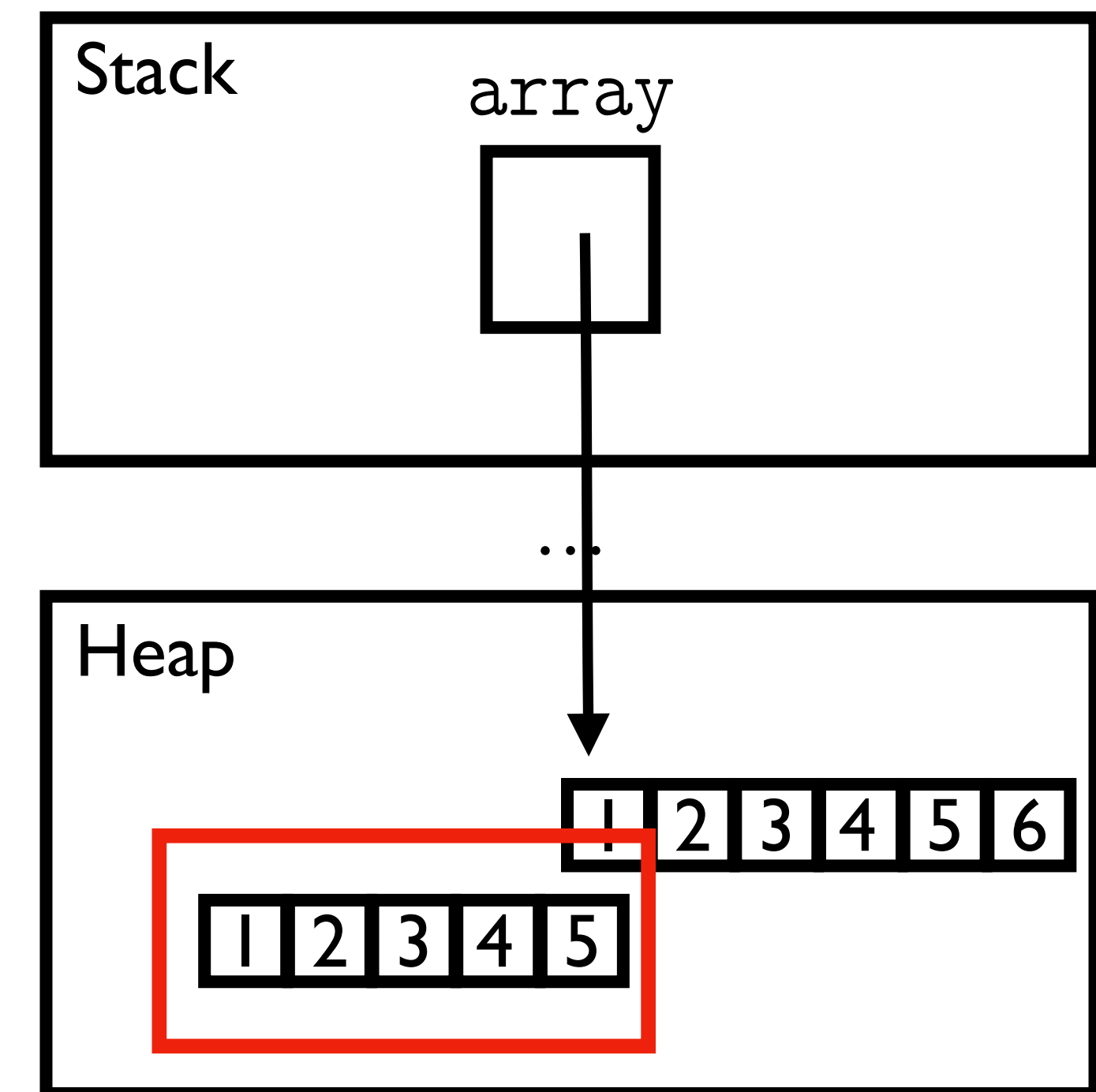
```
int* cpy = new int[dim + 1];  
  
for (int j = 0; j < dim; j++)  
    cpy[j] = array[j];  
  
cpy[i] = num;  
i++;  
array = cpy;  
dim++;
```



# Allocazione dinamica della memoria

cpy verrà distrutta all'uscita dal blocco

```
int* cpy = new int[dim + 1];  
  
for (int j = 0; j < dim; j++)  
    cpy[j] = array[j];  
  
cpy[i] = num;  
i++;  
array = cpy;  
dim++;
```



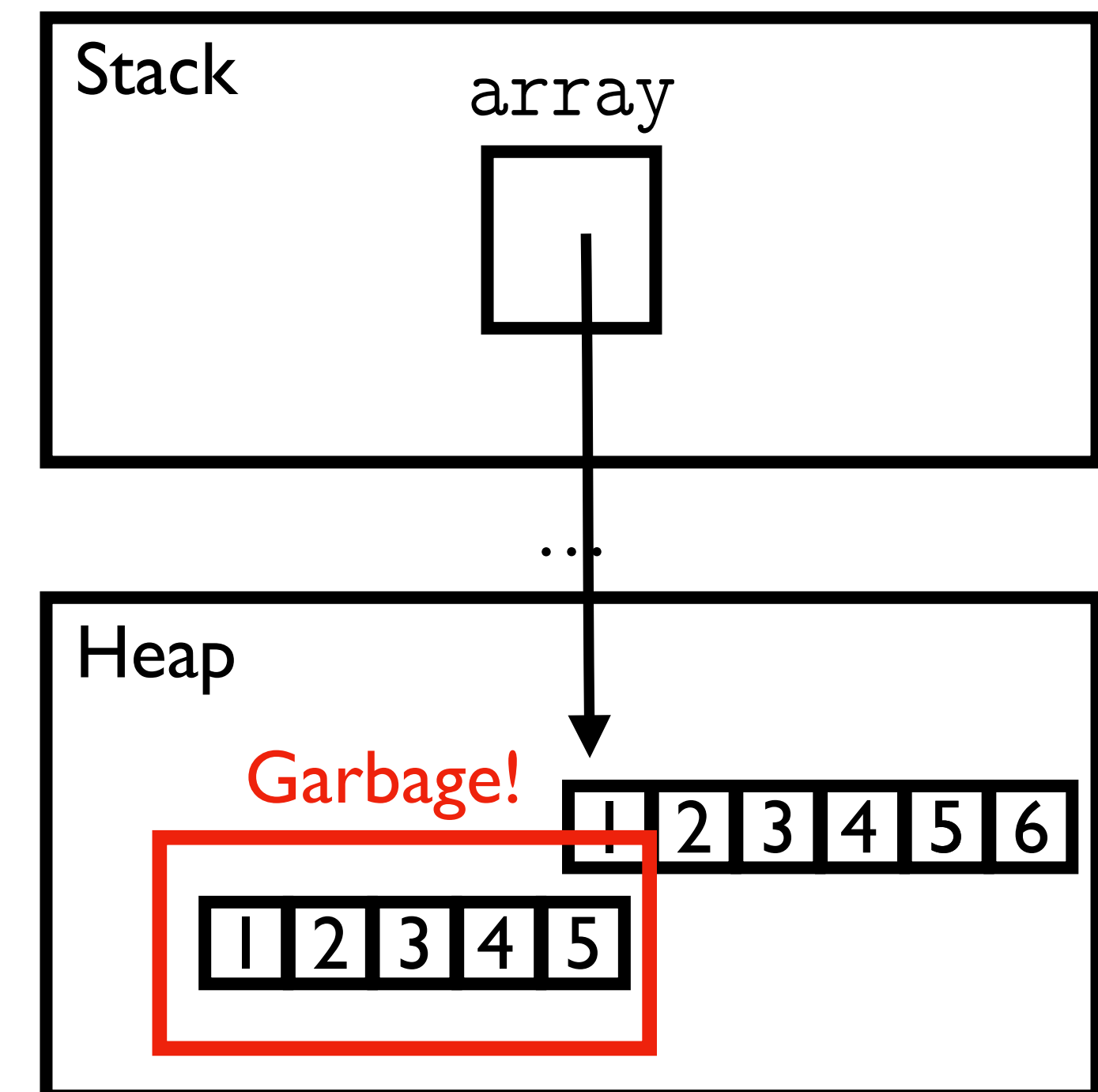
Un blocco di memoria dello heap non viene puntato da nessuno (è irraggiungibile) ma è comunque occupata!

# Allocazione dinamica della memoria

cpy verrà distrutta all'uscita dal blocco

```
int* cpy = new int[dim + 1];  
  
for (int j = 0; j < dim; j++)  
    cpy[j] = array[j];  
  
cpy[i] = num;  
i++;  
array = cpy;  
dim++;
```

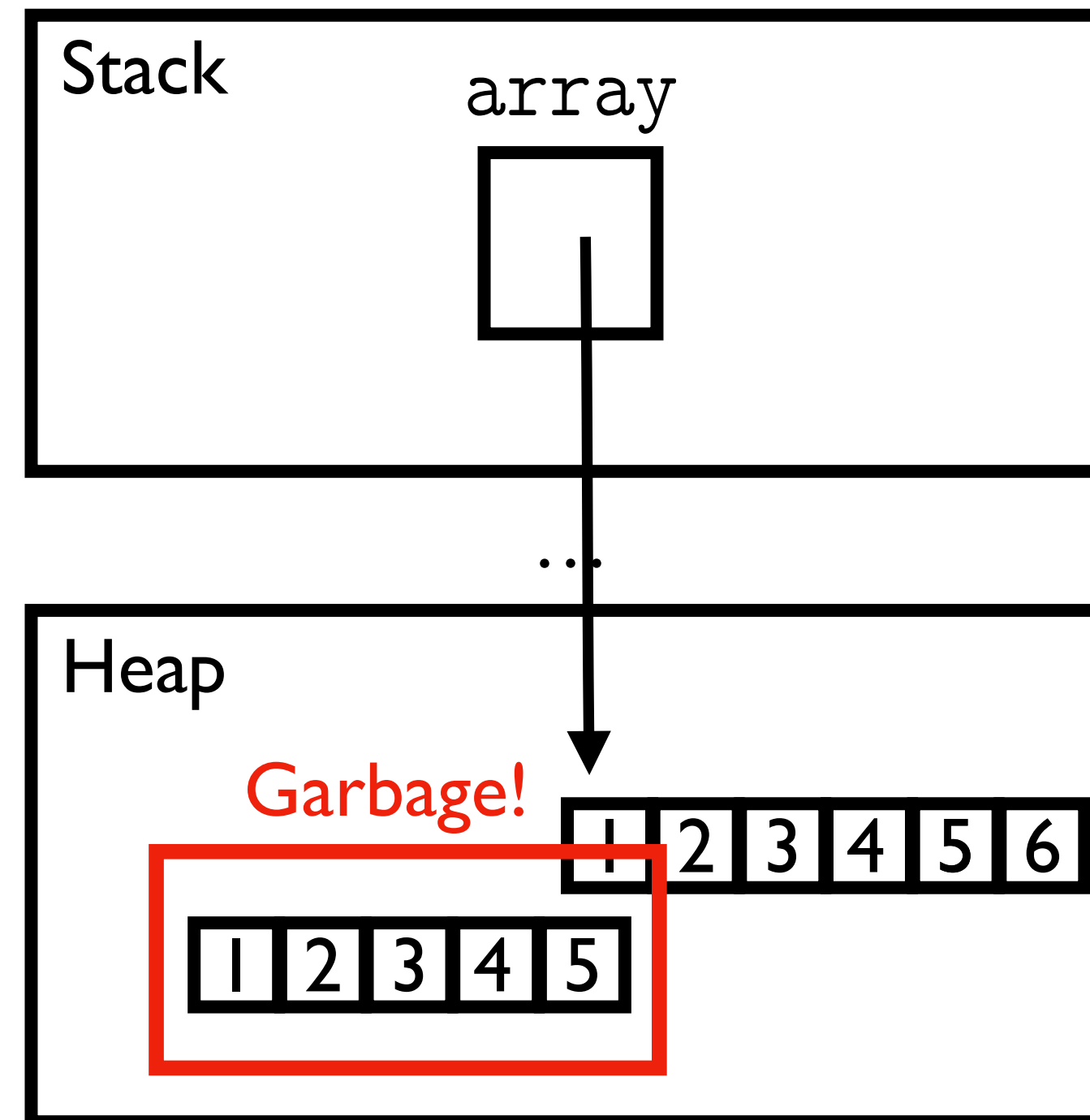
Non può essere riallocata!



Un blocco di memoria dello heap non viene puntato da nessuno (è irraggiungibile) ma è comunque occupata!

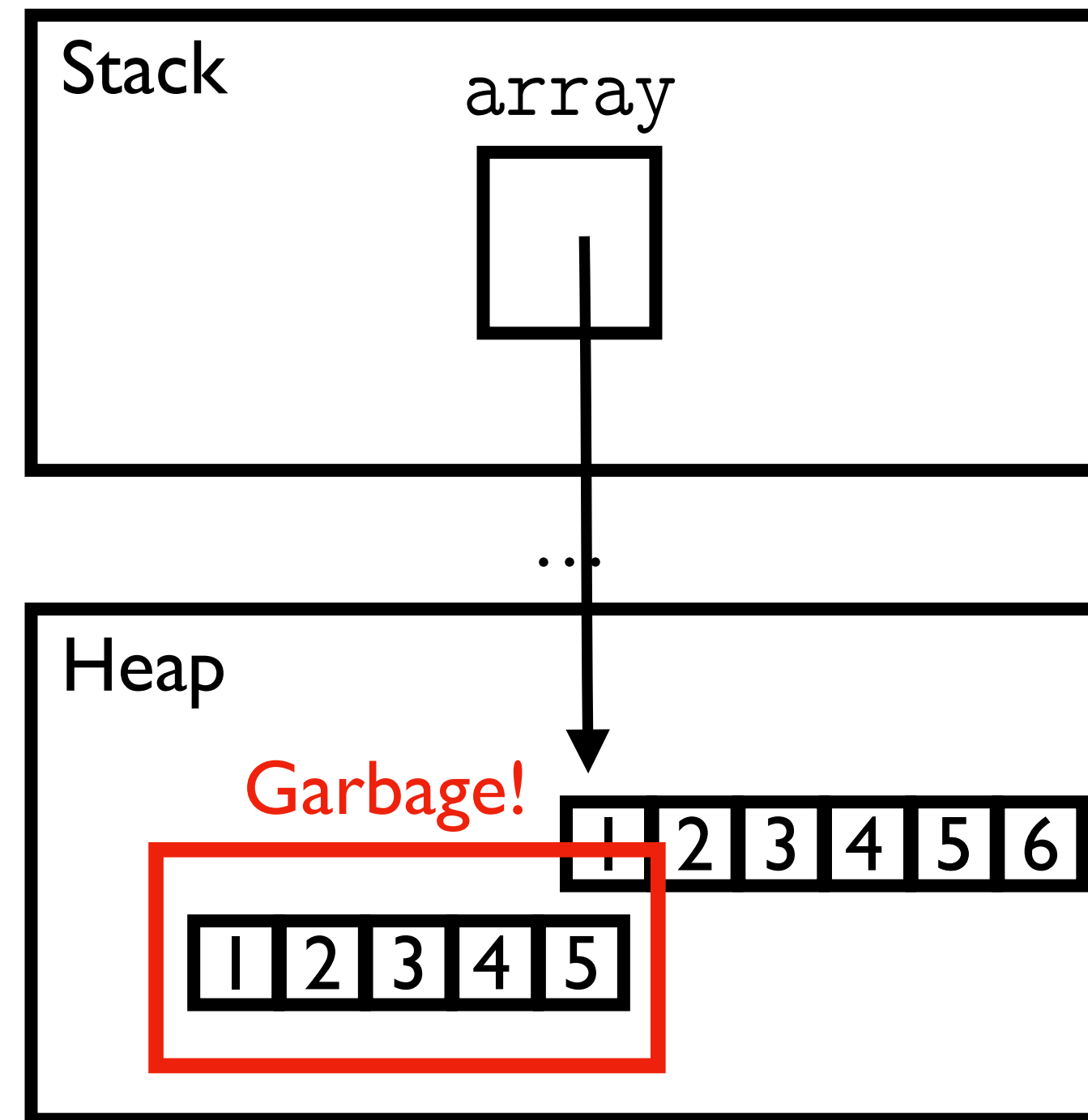
# Allocazione dinamica della memoria

## Garbage collection



# Allocazione dinamica della memoria

## Garbage collection



- C++: garbage collection deve essere effettuata dal programmatore
- Java: gestita automaticamente dal *garbage collector*

# Allocazione dinamica della memoria

Operatore delete

delete *pointer*

# Allocazione dinamica della memoria

## Operatore delete

`delete pointer`

- Dealloca la zona di memoria puntata da *pointer*
- La zona di memoria deallocata potrà essere eventualmente riallocata

# Allocazione dinamica della memoria

## Operatore delete

`delete pointer`

- Dealloca la zona di memoria puntata da *pointer*
- La zona di memoria deallocata potrà essere eventualmente riallocata
- Sintassi



# Allocazione dinamica della memoria

## Operatore delete

`delete pointer`

- Dealloca la zona di memoria puntata da *pointer*
- La zona di memoria deallocata potrà essere eventualmente riallocata
- Sintassi
  - `delete p` per tipi non array

# Allocazione dinamica della memoria

## Operatore delete

`delete pointer`

- Dealloca la zona di memoria puntata da *pointer*
- La zona di memoria deallocata potrà essere eventualmente riallocata
- Sintassi
  - `delete p` per tipi non array
  - `delete [] p` per tipi array

# Allocazione dinamica della memoria

## Esercizio

- **Problema:** memorizzare in un array dinamico una serie indefinita di interi positivi inseriti dall'utente (inserimento termina quando l'utente digita -1)

# Allocazione dinamica della memoria

Corrispondenza `new` – `delete`

```
int* p = new int[dim]; // alloco sullo heap
...
... // uso la memoria puntata da p
...
delete[]p; // dealloco
```

# Allocazione dinamica della memoria

## Out-of-memory

```
int* p = new int[10000000]; // spazio insufficiente nello heap
```

Heap



# Allocazione dinamica della memoria

## Out-of-memory

```
int* p = new int[10000000]; // spazio insufficiente nello heap
```

Heap



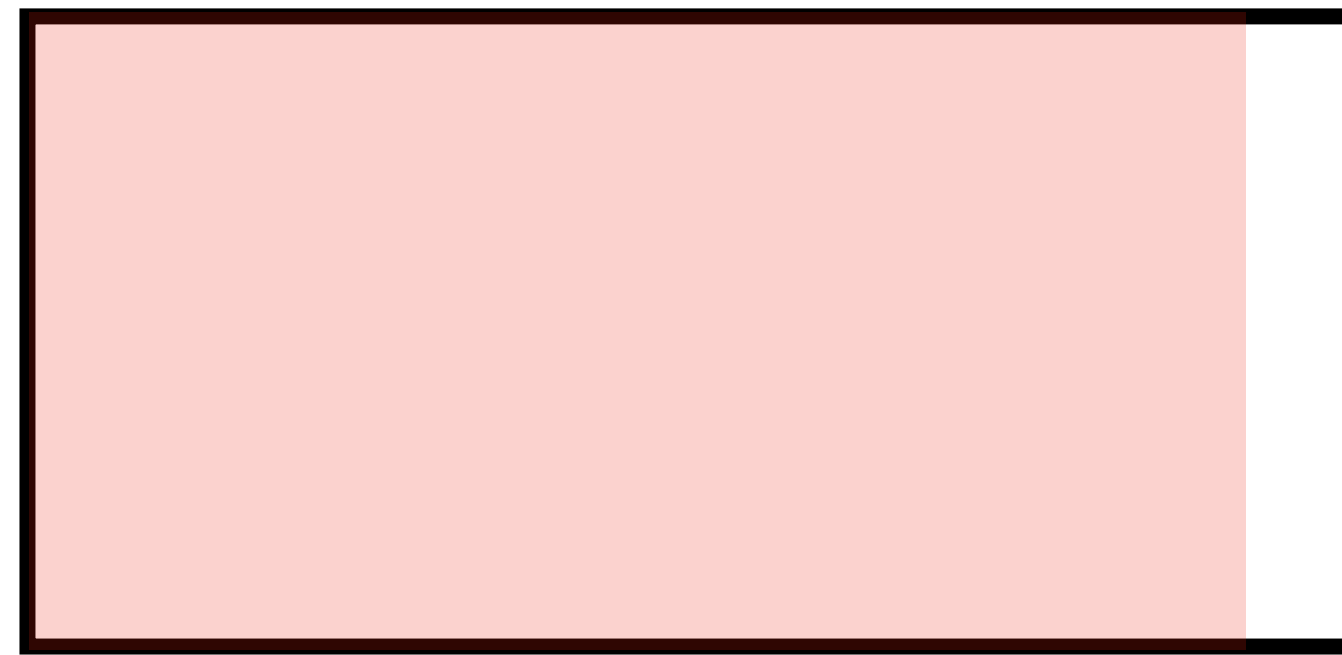
- Lo heap non è infinito

# Allocazione dinamica della memoria

## Out-of-memory

```
int* p = new int[10000000]; // spazio insufficiente nello heap
```

Heap



- Lo heap non è infinito
- Se lo heap non contiene spazio sufficiente per allocare la memoria specificata dall'operatore `new`, viene lanciata un'eccezione (`bad_alloc`)