## Allocazione della memoria

- Tempo di vita di una variabile: intervallo di tempo fra la sua creazione e la sua distruzione

### Allocazione statica:

Memoria allocata durante la compilazione, non a RUN-TIME (come avviene invece per la memoria allocata dinamicamente o automaticamente)

Lifetime: tempo di esecuzione dell'intero programma

### Allocazione automatica:

Riguarda l'allocazione dei dati locali (un blocco, una funzione...)

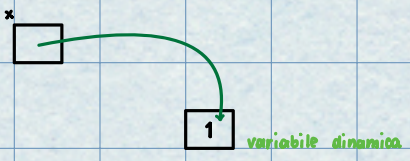
Allocata automaticamente all'entrata del blocco, deallocata automaticamente all'uscita del blocco

Lifetime: tempo di esecuzione del blocco → coincide con lo scope

### Allocazione dinamica:

memoria allocata nello heap tramite operatore **new**

```
int main() {
    int * x = new int[1];
    ...
    delete x;
}
```




Allocata tramite l'operatore **new** e deallocata tramite l'operatore **delete**

Lifetime: il tempo che intercorre tra l'esecuzione della new e la delete

## Allocazione di oggetti

```
void f (int x) {
    Razionale a;
    ...
    return;
}
```

al momento della creazione di un oggetto di una classe C viene invocato automaticamente un costruttore di C

a è una variabile automatica e viene deallocata automaticamente al termine della funzione

```
int main (){
        f(5);
        }
```

Avviene tramite l'invocazione di un distruttore che rilascia la memoria allocata per a

## Distruttore

il distruttore di una classe è una funzione che si occupa di distruggere un oggetto della classe

- una funzione
- public
- Stesso nome della classe, ma con prefisso il carattere ~
- È senza parametri (il distruttore è unico per ciascuna classe)
- non ha tipo di ritorno

### Distruttore di default

- Ogni classe ha un distruttore di default

```
~Razionale () {}
```

- Il fatto che il corpo del distruttore di default sia vuoto non significa che l'oggetto su cui viene invocato non venga distrutto
- La distruzione dell'oggetto avviene comunque in modo automatico dopo l'invocazione del distruttore
- È sempre possibile ridefinire il distruttore di una classe c
- Il distruttore di default rilascia la memoria occupata dall'oggetto (dai suoi attributi) ma non si occupa di rilasciare eventuale memoria allocata dinamicamente

```
class C {
private:
        int a;
        int b;
        int* memory;
public:
```

```
C (int a, int b, int c1){

        a = a1;

        b= b1;

        memory =  new memory [c1];

        }

~C {

        delete  [] memory;

        }
```

nella classe Stack

```
~Stack () {

        if (A != NULL){

                delete [] A;}

        }
```

nella StringSet

```
~StringSet () {

        delete  [] s;

        }
```