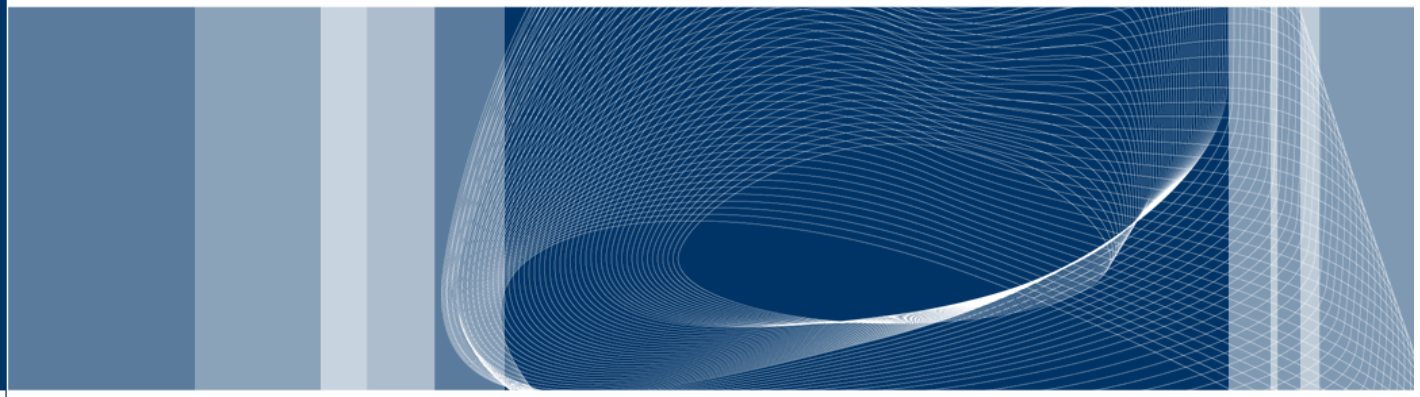
POLITECNICO DI MILANO



Allocazione dinamica della memoria

Fondamenti di Informatica

Daniele Loiacono

Gestione *dinamica* della memoria?

- In C dobbiamo dichiarare una variabile ogni volta che abbiamo bisogno di riservare uno spazio nella memoria di un calcolatore per ospitare dei dati…
- … tuttavia a volte sarebbe comodo avere uno strumento più *flessibile* della dichiarazione di una variabile per *creare* questo spazio di memoria
  - non sempre siamo in grado di prevedere *a priori* quanti dati dobbiamo memorizzare;
  - la quantità di informazioni da memorizzare può cambiare *durante* l'esecuzione del programma;
  - in una funzione, vogliamo poter creare uno spazio di memoria (ed assegnarli un valore) che *sopravviva* alla conclusione della funzione stessa.

Allocare la memoria

- In C è possibile *allocare* uno spazio di memoria tramite la funzione `malloc` contenuta nella libreria `stdlib`:

```
tipo *p = malloc (size);
```

  - `size` è la quantità (in byte) di memoria da allocare ed è in genere espressa come `n*sizeof(tipo)`
  - la funzione riserva uno spazio di memoria delle dimensioni richieste e ne restituisce l'indirizzo
  - nel caso non sia possibile riservare uno spazio di memoria con le caratteristiche richieste, la funzione ritorna `NULL`
- Esempio

```
float *v = malloc(5*sizeof(float)); //alloca 5 float
```

Esempio: malloc e funzioni

```
int *array_vuoto(int n)
{
      int *a = malloc(n*sizeof(int));
      for (int i = 0; i < n; i++)
        a[i]=0;

      return a;
}

int main()
{
  …
  int *v = array_vuoto(10);
  …
}
```

Allocare ed inizializzare la memoria

- Un altra funzione in C che consente di allocare la memoria è la `calloc` contenuta nella libreria `stdlib`:

```
tipo *p = calloc (n, size);
```

  - `n` è il numero di elementi da allocare
  - `size` è la quantità (in byte) di memoria di ***ciascun elemento*** da allocare, in genere espressa come `sizeof(tipo)`
  - la funzione riserva uno spazio di memoria delle dimensioni richieste, inizializza tutti i bit della memoria allocata a 0 e ne restituisce l'indirizzo
  - nel caso non sia possibile riservare uno spazio di memoria con le caratteristiche richieste, la funzione ritorna `NULL`
- Esempio

```
float *v = calloc(5, sizeof(float)); //alloca 5 float
```

Esempio: calloc

```
int *array_vuoto2(int n)
{
      return calloc(n,sizeof(int));
}

int main()
{
  …
  int *v = array_vuoto2(10);
  …
}
```

Deallocare la memoria

- In C è possibile *deallocare* (liberare) uno spazio di memoria con la funzione `free` contenuta nella libreria `stdlib`:

```
free(indirizzo);
```

  - la funzione riceve come parametro in ingresso l'indirizzo di uno spazio di memoria **precedentemente allocato** e lo libera (rendendolo riutilizzabile in seguito)
- **Attenzione a non liberare uno spazio di memoria ancora in uso o usare uno spazio di memoria dopo averlo liberato!**
- Esempio

```
…
float *v = calloc(500, sizeof(float));
…
free(v);
```