20. Gestione della memoria

Corso di Algoritmi e Linguaggi di Programmazione Python/C

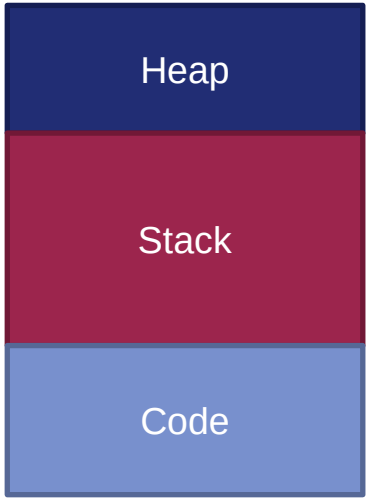
Angelo Cardellicchio
angelo.cardellicchio@stiima.cnr.it

Outline

- Organizzazione della memoria
- Allocazione statica della memoria
- Allocazione dinamica della memoria
  - `malloc`
  - `calloc`
  - `realloc`
  - `free`

Organizzazione della memoria

- **Heap**: sezione di memoria che viene allocata dinamicamente, a seconda delle necessità.
- **Stack**: sezione di memoria dove sono allocate le variabili locali.
- **Code**: sezione di memoria dove sono memorizzate tutte le istruzioni contenute nel codice sorgente.



Allocazione statica della memoria

- Avviene a compile time.
- Non è modificabile a runtime.
- Vantaggi:
  - *Semplice da utilizzare.*
  - *Delega al compilatore dell'allocazione della memoria.*
  - *Efficiente a runtime.*
- Svantaggi:
  - *Memoria non utilizzata al meglio.*
  - *Necessità di conoscere a priori i requisiti in termini di memoria.*
  - *Impossibile riallocare la memoria a runtime.*

```
int main()
{
    int a; // prealloco 4 byte
    long b; // prealloco 8 byte
}
```

Allocazione dinamica della memoria – `malloc`

- Permette di allocare un certo quantitativo di memoria a runtime.
- Specifichiamo il numero di ‘elementi’ da allocare come parametro della funzione.
- Nell’esempio, allochiamo spazio sufficiente a memorizzare dieci interi.
- Restituisce un puntatore alla memoria appena allocata.

```
int num_values = 10;
int *p = (int*) malloc(num_values);
```

Allocazione dinamica della memoria – `calloc`

- Simile alla **`malloc`**, ma con una funzione in più.
- Permette di inizializzare i valori riservati in memoria.
- Accetta due parametri in ingresso: il numero di oggetti per i quali è richiesta l'allocazione, e la dimensione di ciascun oggetto.
- Restituisce un puntatore alla memoria appena allocata.

```
int *p =
    (int*) calloc(num_values, sizeof(int))
```

Allocazione dinamica della memoria – `realloc`

- Ci permette di modificare il blocco di memoria puntato in precedenza da un puntatore.
- Accetta come parametri il puntatore stesso, e la nuova dimensione richiesta.
- Restituisce un puntatore alla memoria appena allocata.

```
int *p = (int*) realloc(*p, new_size)
```

Allocazione dinamica della memoria – `free`

- Permette di liberare la memoria allocata in precedenza.
- Va chiamata per evitare errori di memoria.
- *In realtà, il compilatore libera la memoria automaticamente al termine dell'esecuzione del programma, ma è comunque opportuno utilizzare la funzione* **`free`** *in ogni caso.*

```c
#include <stdlib.h>

int main()
{
    int *p;
    p = (int*) malloc(5 * sizeof(int));
    free(p);
    return 0;
}
```

Allocazione dinamica della memoria

- Vantaggi:
  - *Allocazione a runtime (maggiore flessibilità).*
  - *Possibilità di allocare, deallocare e riallocare ulteriore memoria alla bisogna.*
- Svantaggi:
  - *Maggior tempo di esecuzione (richiesto per allocare dinamicamente le variabili).*
  - *Complicazioni legate alla necessità di gestire l'allocazione in forma esplicita.*

Domande?

42