ALLOCAZIONE DINAMICA DELLA MEMORIA

-

MALLOC()

Dott. Manzali Matteo

Allocazione dinamica della memoria

Un puntatore deve puntare ad una zona di memoria
• a cui il sistema operativo permette di accedere
• che non viene modificata inaspettatamente

Finora abbiamo visto un modo per soddisfare questi requisiti: assegnare ad un puntatore l'indirizzo di una delle variabili del programma.

Metodo alternativo: allocazione dinamica della memoria, attraverso una chiamata di funzione che crea una nuova zona di memoria e ne restituisce l'indirizzo iniziale.

• la zona di memoria è accessibile al programma

• la zona di memoria non viene usata per altri scopi (ad esempio variabili in altre funzioni)

• ad ogni chiamata della funzione viene allocata una nuova zona di memoria

Funzione `malloc()`

La funzione `malloc()` è dichiarata in `<stdlib.h>` con prototipo:

```
void * malloc(size_t);
```

da allocare (`size\_t` è il tipo restituito da `sizeof` e usato per le dimensioni in byte delle variabili, ad esempio potrebbe essere `unsigned long`)

- alloca (riserva) la zona di memoria
- restituisce il puntatore iniziale alla zona allocata (è una funzione che restituisce un puntatore)

N.B. La funzione `malloc()` restituisce un puntatore di tipo `void*`, compatibile con tutti i tipi puntatore

Esempio:

```
float *p;
p = malloc(4);
```

Funzione `malloc()`

Uso tipico di `malloc()` è con `sizeof(tipo)` come parametro.

Esempio:

```
#include <stdlib.h>
...
int *p; p = malloc(sizeof(int)); *p = 12;
(*p)++;
printf("*p vale %d\n", *p);
```

• attivando `malloc(sizeof(int))` viene allocata una zona di memoria adatta a contenere un intero; ovvero viene creata una nuova variabile intera

• il puntatore restituito da `malloc()` viene assegnato a `p` !!

• `*p` si riferisce alla nuova variabile appena creata...

Lo heap (o memoria dinamica)

La zona di memoria allocata attraverso `malloc()` si trova in un'area di memoria speciale, detta heap (o memoria dinamica).

Abbiamo 4 aree di memoria:
- zona programma: contiene il codice macchina
- stack: contiene la pila dei RDA
- statica: contiene le variabili statiche
- heap: contiene dati allocati dinamicamente

Funzionamento dello heap:
- gestito dal sistema operativo
- le zone di memoria sono marcate libere o occupate
    a) marcata libera: può venire utilizzata per la prossima malloc
    b) marcata occupata: non si tocca

Lo heap (o memoria dinamica)

Potrebbe mancare la memoria per allocare la zona richiesta. In questo caso `malloc()` restituisce il puntatore `NULL`: bisogna sempre verificare cosa restituisce `malloc()`.

Esempio:

```
p = malloc(sizeof(int));
if (p == NULL) {
   printf("Non ho abbastanza memoria per
   l'allocazione\n");
   exit(1);
} ...
```

La costante `NULL`
- è una costante di tipo `void*` (quindi compatibile con tutti i tipi puntatore)
- indica un puntatore che non punta a nulla: non può essere dereferenziato
- ha tipicamente valore 0
- definita in `<stdlib.h>` (ed in altri file header)

Deallocazione della memoria dinamica

Le celle di memoria allocate dinamicamente devono essere deallocate (o rilasciate) quando non servono più.

Si utilizza la funzione `free`, che è dichiarata in `<stdlib.h>`:
void * free(void *);

Esempio:

```
int *p;
...
p = malloc(sizeof(int));
...
free(p);
```

- il parametro `p` deve essere un puntatore ad una zona di memoria allocata precedentemente con `malloc` (altrimenti il risultato non è determinato)
- la zona di memoria viene resa disponibile (viene marcata libera)
- `p` non punta più ad una locazione significativa (valore arbitrario)