Puntatori a funzioni

Puntatori a funzioni

- Sono essenzialmente un nuovo tipo di dato
- Sebbene una funzione non e' una variabile, essa è identificata dall'indirizzo del proprio codice, indirizzo che può essere assegnato ad un puntatore
- Tale indirizzo consiste nel punto di ingresso di una funzione
- Il puntatore a funzione consente di passare come argomento di una funzione una funzione

<tipodiritorno> (*nome)();

Esempio

```
void main (){
    char s1[80],s2[80];
    int (*p)();

    p = strcmp;
    gets(s1);
    gets(s2);
    controlla(s1,s2,p);
}
```

```
void controlla(char *a, char* b, int (*cmp)()){
    prinf("verifica dell'ugualglianza fra stringhe");
    if (!(*cmp)(a,b)) printf("uguali");
    else printf("diverse");
}
```

Esempio

```
void main (){
    char s1[80],s2[80];
    int i;
    gets(s1);
    gets(s2);
    scanf("%d", &i)
    if (i)
        controlla(s1,s2,strcmp);
    else controlla(s1,s2,numcmp);
}
```

```
int numcmp(char *a, char *b){
        return atoi(a) ==atoi(b);
}
```

```
void controlla(char *a, char* b,int (*cmp)()){
    prinf("verifica dell'ugualglianza fra stringhe");
    if (!(*cmp)(a,b)) printf("uguali");
    else printf("diverse");
}
```

Allocazione dinamica

La gestione della memoria heap

- I puntatori forniscono il supporto per la gestione della memoria dinamica
- Il sistema di gestione della memoria dinamica consente di gestire la memoria heap
- Le funzioni della stdlib.h
  - malloc
  - free

malloc

void *malloc(size_f numero_di_byte)

- La funzione malloc
  - alloca una regione della memoria heap di dimenzione pari a numero_di_byte
  - ritorna un puntatore a void contenente il valore dell'indirizzo del primo byte della memoria heap allocata
  - tale puntatore è assegnabile ad un qualsiasi tipo di puntatore
- Quando nell'heap non esiste più memoria disponibile viene segnalato un errore di allocazione e la funzione restituisce un puntatore uguale a **null**
- size_f è un tipo definito nella stdlib
- Esempio

```
char * p
p = (char*)malloc(1000)
```

```
int *x;
x = (int*)malloc(50 *sizeof(int));
```

free

- È la funzione complementare a malloc

  void free(void *p)

- Rilascia la memoria precedentemente allocata il cui indirizzo è contenuto in p

Attenzione

- È necessario verificare
  - Che l'allocazione abbia avuto successo
- È necessario eseguire
  - il cast al puntatore nel tipo opportuno
- È necessario fare attenzione a
  - Rilasciare la memoria prima che sia inaccessibile