

# Parte 8

# Glib



[W. Kandinsky – Composition VII, 1913]

# Glib

## *Libreria C di utilità generale*

- Offre molteplici facilitazioni e sostituti ai costrutti C standard
- Ha un unico file header "glib.h"
- Fornisce e implementa molte ***strutture dati comunemente usate***
  - Es. Liste semplici (single-linked) e doppie (double-linked)
- Per installare su Ubuntu:
  - `sudo apt-get install libglib2.0-dev`

# Elemento della lista

- Strutture **GSList** e **GList**
- Definite in *glib.h*

```
typedef void* gpointer;  
Un puntatore non  
tipizzato ( void* )
```

**Lista semplice**

```
typedef struct {  
    gpointer data;  
    GSList *next;  
} GSList;
```

**Lista doppia**

```
typedef struct {  
    gpointer data;  
    GList *next;  
    GList *prev;  
} GList;
```

# Creazione di una lista

- Non c'e' una funzione specifica per creare una lista
- E' sufficiente:
  - creare una variabile di tipo `GSList*` (o `GList*`)
  - settare il suo valore a `NULL`
  - `GSList* list = NULL;`
- Il valore `NULL` del puntatore è considerato come una **lista vuota**

# Inserimento di elementi

- Per **aggiungere elementi** alla lista si usano le routine (lista semplice):
  - `g_slist_append()`
  - `g_slist_prepend()`
  - `g_slist_insert()`
  - `g_slist_insert_sorted()`
- Tutte accettano un puntatore all'inizio della lista e ritornano il nuovo puntatore all'inizio della lista
- Vediamo la sintassi...

# Funzioni per l'inserimento

- `GSList *g_slist_append( GSList *list, gpointer data );`  
`//Aggiunge in coda`
- `GSList *g_slist_prepend( GSList *list, gpointer data );`  
`//Aggiunge in testa`
- `GSList *g_slist_insert( GSList *list, gpointer data,`  
`gint position );`  
`//Aggiunge nella posizione specificata da position`
- `GSList* g_slist_insert_sorted( GSList *list, gpointer`  
`data, GCompareFunc func);`  
`//Aggiunge un elemento usando la funzione func per`  
`//mantenere l'ordinamento e determinare la posizione`

# Estrazione ed eliminazione

- `GSList *g_slist_remove( GSList *list, gpointer data );`  
`// Rimuove l'elemento corrispondente al valore`  
`//"data".`
- `void g_slist_free( GSList *list );`  
`// Libera tutta la memoria usata da GSList`  
`// Ma non azzera il puntatore alla lista`
- *Per un elenco completo cercare su Google*  
*“GSList gnome library”*

# Accesso agli elementi

- Per accedere ai dati del primo elemento:

```
char * my_data = (char *) list->data;
```

- Per muoversi all'interno della lista:


```
GSLIST* tmp = list;
```

```
while (tmp != NULL) {
```

```
    cout << "List data: " << tmp->data << endl;
```

```
    tmp = g_slist_next(tmp);
```

```
}
```



E' una funzione sicura: ritorna il prossimo elemento, o NULL se non ci sono più elementi



# Altre funzioni

- `GSList* g_slist_find (GSList *list, gpointer data);`  
// Trova l'elemento nella lista che contiene i  
// data specificati
- `guint g_slist_length (GSList *list);`  
// Ritorna il numero di elementi contenuti nella  
// lista

# Programma

- *Programma list\_glib.cc*
- Compilare con  
`g++ -Wall -o list_glib list_glib.cc `pkg-config --cflags --libs glib-2.0``
- La dicitura ``comando`` viene sostituita dall'output del comando
- In pratica, stiamo indicando al compilatore le opzioni necessarie per trovare le librerie glib