.

Libreria standard

Stefano Ferrari

Università degli Studi di Milano
stefano.ferrari@unimi.it

Programmazione

anno accademico 2017–2018

Libreria standard

La modularità consente di costruire librerie cioè raccolte di funzioni che arricchiscono il linguaggio con nuove operazioni di uso comune

Il C offre una libreria standard presente in ogni compilatore per

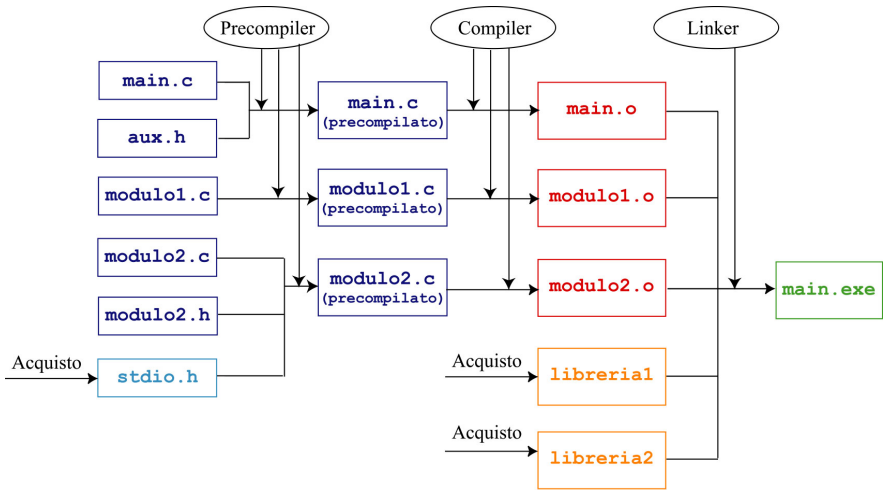
- gestire l'ingresso di dati (da tastiera, file, ecc. . .) e l'uscita di risultati (a video, file, ecc. . .)
- gestire le stringhe (vettori di caratteri)
- gestire l'allocazione e deallocazione di memoria dinamica
- eseguire operazioni matematiche superiori (potenze, logaritmi, funzioni trigonometriche)
- gestire data e ora, misurare tempi di calcolo

.

Inclusione

Per usare una libreria basta includerla nel codice

```
#include <libreria.h>
```



Componenti standard (1)

La libreria standard ha 15 componenti

1. `assert.h` fornisce un'istruzione per la diagnosi di errori

   `assert(`*espressione*`);`

   Se l'*espressione* ha valore falso, interrompe il programma
2. `ctype.h` fornisce funzioni per classificare i caratteri e convertirli da maiuscolo a minuscolo e viceversa
3. `errno.h` definisce
   - una variabile globale `errno`
   - alcune costanti simboliche associate a errori

   Se una funzione di libreria riscontra un errore, la variabile (inizialmente nulla) assume il valore della costante associata

.

Componenti standard (2)

4. `float.h` definisce le costanti che indicano intervallo di definizione e accuratezza dei tipi reali
5. `limits.h` definisce le costanti che indicano intervallo di definizione dei tipi interi e naturali
6. `locale.h` fornisce una struttura che conserva convenzioni locali sulla stampa di numeri, valori monetari, date e ore e consente di modificarle
7. `math.h` fornisce le funzioni matematiche superiori di uso comune: trigonometriche, logaritmiche, esponenziali, di valore assoluto, elevamento a potenza, arrotondamento, ecc. . .

Componenti standard (3)

8. `setjmp.h` fornisce funzioni per consentire salti da una funzione a un'altra per gestire problemi seri durante l'esecuzione
9. `signal.h` fornisce funzioni per gestire situazioni eccezionali (segnali) che indicano errori o eventi esterni al programma (*interrupt*)
10. `stdarg.h` consente di scrivere funzioni con un numero di argomenti non specificato a priori, fornendo funzioni per
    - cominciare la lettura degli argomenti
    - accedere all'argomento successivo
    - terminare la lettura degli argomenti

.

Componenti standard (4)

11. `stddef.h` definisce alcuni tipi e costanti di uso frequente (`size_t`, `NULL`)
12. `stdio.h` fornisce funzioni per leggere dati (da tastiera, file, ecc...) e scrivere risultati (a video, su file, ecc...)
13. `stdlib.h` raccoglie funzioni varie di uso comune
    - allocazione e deallocazione della memoria dinamica

```
void* malloc (size_t sizeObject);
void* calloc (size_t sizeObjCnt, size_t
sizeObject);
void* realloc (void* pObject, size_t sizeNew);
void free (void* pObject);
```

    - interruzione dell'esecuzione

```
void exit (int nStatus);
```

Componenti standard (5)

13. `stdlib.h` raccoglie funzioni varie di uso comune
    - esecuzioni di comandi esterni come da terminale

```
int system (const char* szCommand);
```

    - fare ricerche e ordinamenti di vettori (`bsearch`, `qsort`)
14. `string.h` fornisce funzioni per operare su
    - stringhe (`strcpy`, `strcmp`, `strcat`, ...)
    - blocchi di memoria (`memcpy`, `memcmp`, `memset`)
15. `time.h` fornisce tipi e funzioni per manipolare ore e date e determinare quelle correnti

```
clock_t clock(void)
time_t time(time_t *pt)
double difftime(time_t t1, time_t t2)
```