



Insegnamento di Programmazione (6 CFU)
 Corso di Laurea in Ingegneria Energetica
 a.a. 2005/2006

Sviluppo di una libreria

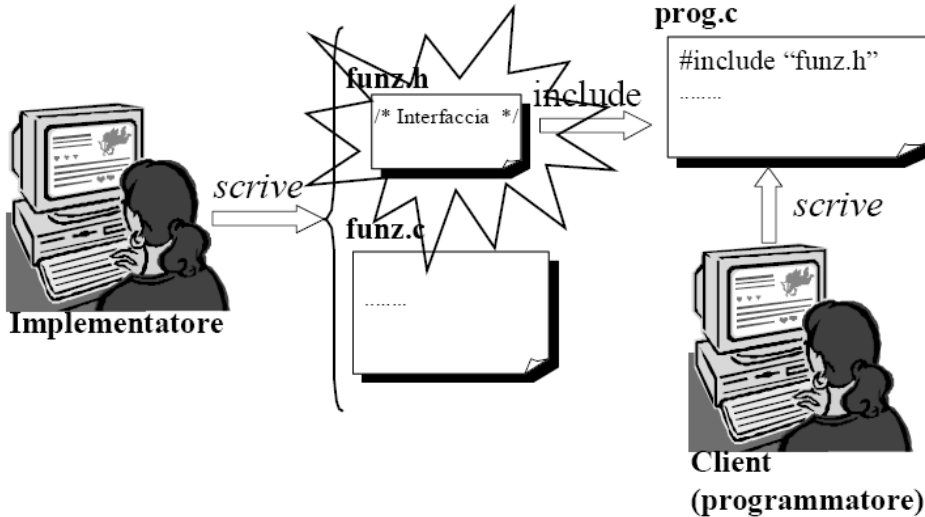
Ing. Nadia Ranaldo
 Dipartimento di Ingegneria
 Università degli Studi del Sannio

Concetto di libreria (1)

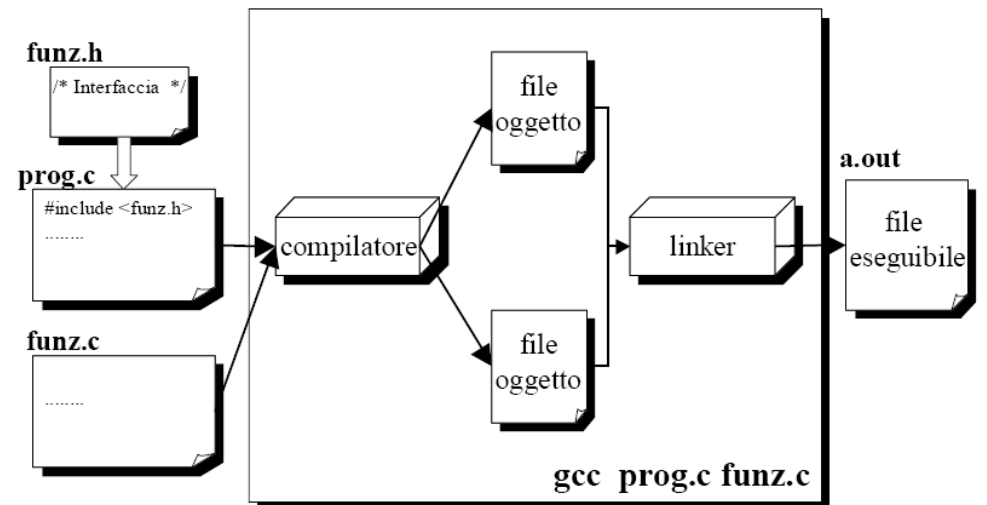


- Una libreria è costituita da un insieme di funzioni e definizioni di tipo che possono utilizzate dai programmi
 - Il programma che usa la libreria non deve conoscere come sono implementate le funzioni (concetto di separazione tra interfaccia e implementazione)
- Una semplice libreria è composta da:
 - Un header file libreria.h, ovvero un file di testo (**interfaccia della libreria**)
 - Un file .c (libreria.c) che include l'header file libreria.h e che contiene il codice delle funzioni (**implementazione della libreria**)
- Un programma (prog.c) (il **client** della libreria) deve
 - includere l'header file della libreria utilizzando i doppi apici ("libreria.h")
 - Utilizzare le funzioni in prog.c
 - Compilare insieme prog.c e libreria.c
- La libreria e il programma potrebbero essere scritti da due persone diverse!

Concetto di libreria (2)



Concetto di libreria (3)



Compilazione separata



- E' possibile anche compilare separatamente la libreria e poi il programma
- In questo caso la libreria deve essere compilata per produrre il file oggetto ma non deve essere eseguita la fase di linking

- Usare l'opzione `-s` del comando `gcc`, ad esempio:

```
gcc funz.c -s -o funz.o
```

- Successivamente si può compilare il programma
- Specificare sulla linea di comando anche il file oggetto della libreria che si vuole usare, ad esempio

```
gcc client.c funz.o -o client
```

5

Esempio semplice



- Scrivere una libreria che fornisce una funzione `converti_maiuscolo` e un programma che utilizza tale funzione
- Chiamiamo l'header file della libreria `converti.h` e scriviamo (usando un normale editor di testi) il prototipo della funzione

```
void converti_maiuscolo(char s[], int riemp);
```

Chiamiamo `converti.c` il file che contiene l'implementazione della funzione

```
#include "converti.h"
void converti_maiuscolo(char s[], int riemp){
    int i;
    for(i=0;i<riemp-1;i++)
        if(s[i]<='z' && s[i]>='a')
            s[i]= toupper(s[i]);
}
```

- Chiamiamo `client.c` il programma che utilizza la funzione

```
#include <stdio.h>
#include "converti.h"
main() {
    char str[]="Stringa di prova";
    converti_maiuscolo(str,17);
    printf("Conversione in maiuscolo\n");
    printf("%s",str);
}
```

6

Esempio semplice



- Per compilare utilizzando la compilazione congiunta

```
gcc converti.c client.c -o prog
```

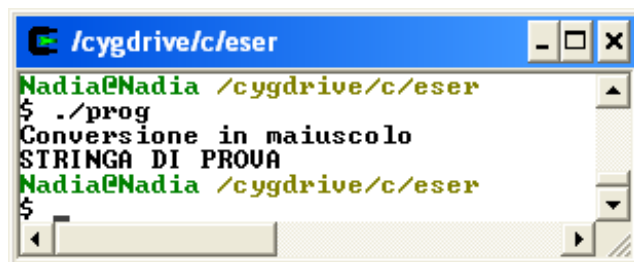
- Utilizzando invece la compilazione separata

```
gcc converti.c -c -o converti.o
```

```
gcc client.c converti.o -o prog
```

- Per eseguire il programma

```
./prog
```



```

/cygdrive/c/eser
Nadia@Nadia /cygdrive/c/eser
$ ./prog
Conversione in maiuscolo
STRINGA DI PROVA
Nadia@Nadia /cygdrive/c/eser
$

```

7

Libreria gestione punti



- Scrivere una libreria che fornisca la definizione del tipo `punto` che rappresenta un punto sul piano cartesiano e un insieme di funzioni che lavorano sui punti
- Recuperiamo l'esempio visto!
- Scriviamo un header file `punto.h` che include la definizione della struttura `puntoT` e i prototipi delle funzioni (per la lettura, la visualizzazione, il calcolo della distanza tra due punti e del punto medio)

8

Header file punto.h



```
typedef struct {
    float x;
    float y;
} puntoT;

void inputPunto(puntoT *p);
void stampa(puntoT p);
float calcola_distanza(puntoT p1, puntoT p2);
void punto_medio (puntoT p1, puntoT p2, puntoT
    *p_medio);
```

9

Implementazione file punto.c



```
#include<stdio.h>
#include<math.h>
#include"punto.h"
void inputPunto(puntoT *p){
    printf("Inserire la coordinata x\n");
    scanf("%f", &p->x);
    printf("Inserire la coordinata y\n");
    scanf("%f", &p->y);
}
void stampa(puntoT p){
    printf("x: %.2f\n",p.x);
    printf("y: %.2f\n", p.y);
}
/* etc. */
```

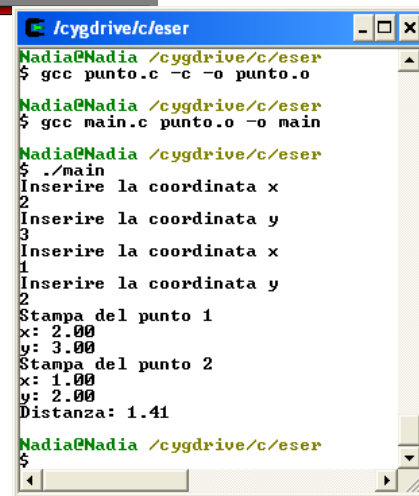
Compilazione:
gcc punto.c -c -o punto.o

10

Programma main.c



```
#include<stdio.h>
#include"punto.h"
main() {
    puntoT p1,p2;
    float dist;
    inputPunto(&p1);
    inputPunto(&p2);
    printf("Stampa del punto 1\n");
    stampa(p1);
    printf("Stampa del punto 2\n");
    stampa(p2);
    dist = calcola_distanza(p1, p2);
    printf("Distanza: %.2f\n",dist);
    /*etc. */
}
```



```
~/cygdrive/c/eser
Nadia@Nadia /cygdrive/c/eser
$ gcc punto.c -c -o punto.o
Nadia@Nadia /cygdrive/c/eser
$ gcc main.c punto.o -o main
Nadia@Nadia /cygdrive/c/eser
$ ./main
Inserire la coordinata x
2
Inserire la coordinata y
3
Inserire la coordinata x
1
Inserire la coordinata y
2
Stampa del punto 1
x: 2.00
y: 3.00
Stampa del punto 2
x: 1.00
y: 2.00
Distanza: 1.41
Nadia@Nadia /cygdrive/c/eser
$
```

Compilazione:
gcc main.c punto.o -o main

12

Esempio di libreria



- Scrivere una libreria che offre un insieme di funzioni sulle matrici
- Si considera una matrice di numeri reali di una certa dimensione $N \times M$ per la quale si forniscono le operazioni di:
 - Lettura da tastiera o da file
 - Si considera la lettura delle seguenti informazioni:
 - il numero di righe, il numero di colonne
 - gli elementi della matrice per riga, fornita mediante una serie di linea. Ogni linea contiene gli elementi separati da uno spazio
 - Scrittura a video o su file
 - Calcolo della trasposta
 - La trasposta si ottiene scambiando ordinatamente le righe con le colonne
 - Confronto tra due matrici
 - Due matrici sono uguali se hanno lo stesso numero di righe e di colonne e tutti gli elementi corrispondenti sono uguali
- Etc.



- Per rappresentare una matrice si considera una struttura costituita da:

- Un puntatore ad un array bidimensionale allocato dinamicamente
- Un intero che rappresenta il numero di righe
- Un intero che rappresenta il numero di colonne

```
typedef struct {  
    float **arr;  
    int righe;  
    int colonne;  
} matriceT;
```

- Per ciascuna operazione di definisce un prototipo di funzione:

- void leggiMatrice(matriceT *mat);
- void leggiMatriceDaFile(matriceT *mat, char * nomeFile);
- void scriviMatrice(matriceT *mat);
- void scriviMatriceSuFile(matriceT *mat, char *nomeFile);
- void trasposta(matriceT *mat, matriceT *trasposta);
- int confronta(matriceT *mat1, matriceT *mat2);
 - Restituisce un valore ==0 se le matrici sono uguali, != se sono diverse

13



```
#include<stdio.h>  
#include"matrice.h"  
void leggiMatrice(matriceT *mat){  
    int i,j;  
    printf("Inserire numero di righe: ");  
    scanf("%d", &mat->righe);  
    printf("Numero di righe: %d\n", mat->righe);  
    printf("Inserire numero di colonne: ");  
    scanf("%d", &mat->colonne);  
    printf("Numero di colonne: %d\n", mat->colonne);  
  
    mat->arr = (float**)malloc(sizeof(float)*mat->righe);  
    for (i=0; i<mat->righe; i++){  
        mat->arr[i] = (float *)malloc(sizeof(float)*mat->colonne);  
    }  
    printf("Inserire gli elementi della matrice per righe\n");  
    for (i=0; i<mat->righe; i++){  
        printf("Inserire gli elementi della riga n. %d\n", i);  
        for (j=0; j<mat->colonne; j++){  
            scanf("%f", &mat->arr[i][j]);  
        }  
    }  
}
```

14



```
void leggiMatriceDaFile(matriceT *mat, char * nomeFile){  
    FILE *f;  
    int i,j;  
    f = fopen(nomeFile, "r");  
    if (f==NULL) {  
        printf("Errore di apertura del file %s\n",nomeFile);  
        exit(0);  
    }  
    fscanf(f, "%d", &mat->righe);  
    printf("Numero di righe: %d\n", mat->righe);  
    fscanf(f, "%d", &mat->colonne);  
    printf("Numero di colonne: %d\n", mat->colonne);  
  
    mat->arr = (float**)malloc(sizeof(float)*mat->righe);  
    for (i=0; i<mat->righe; i++){  
        mat->arr[i] = (float *)malloc(sizeof(float)*mat->colonne);  
    }  
    for (i=0; i<mat->righe; i++){  
        for (j=0; j<mat->colonne; j++){  
            fscanf(f, "%f", &mat->arr[i][j]);  
        }  
    }  
    fclose(f);  
}
```

15



```
void scriviMatrice(matriceT *mat){  
    int i,j;  
    printf("Numero di righe: ");  
    printf("%d\n", mat->righe);  
    printf("Numero di colonne: ");  
    printf("%d\n", mat->colonne);  
    for (i=0; i<mat->righe; i++){  
        for (j=0; j<mat->colonne; j++){  
            printf("%.2f ", mat->arr[i][j]);  
        }  
        printf("\n");  
    }  
}
```

16

Implementazione matrice.c



```
void scriviMatriceSuFile(matriceT *mat, char *nomeFile){
    int i,j;
    FILE *f;
    f = fopen(nomeFile, "w");
    if (f==NULL) {
        printf("Errore di apertura del file %s\n",nomeFile);
        exit(0);
    }
    fprintf(f, "%d\n", mat->righe);
    fprintf(f, "%d\n", mat->colonne);
    for (i=0; i<mat->righe; i++){
        for (j=0; j<mat->colonne; j++){
            fprintf(f, "%.2f ", mat->arr[i][j]);
        }
        fprintf(f, "\n");
    }
    fclose(f);
}
```

17

Implementazione matrice.c



```
void trasposta(matriceT *mat, matriceT *trasposta){
    int i,j;
    trasposta->righe = mat->colonne;
    trasposta->colonne = mat->righe;
    trasposta->arr = (float**)malloc(sizeof(float*)*trasposta->righe);
    for (i=0; i<trasposta->righe; i++){
        trasposta->arr[i] = (float *)malloc(sizeof(float)*trasposta->
            colonne);
    }
    for (i=0; i<trasposta->righe; i++){
        for (j=0; j<trasposta->colonne; j++){
            trasposta->arr[i][j] = mat->arr[j][i];
        }
    }
}
```

18

Programma client mainmatrice.c



- Il programma permette di eseguire delle operazioni sulle matrici a seconda della scelta effettuata dall'utente su un menu principale
- Il menu permette in particolare di scegliere tra:
 - Calcolare la trasposta di una matrice letta da tastiera e di scrivere tale matrice a video o su un file
 - Calcolare la trasposta di una matrice letta da file e di scrivere tale matrice a video o su un file
 - Confrontare due matrici lette da tastiera o da file
- Ad ogni operazione è associato un numero intero che l'utente deve inserire per selezionarla
- La struttura principale del programma contiene un ciclo che permette di ripetere una di queste operazioni fino a che l'utente non sceglie di terminare
- Si utilizza una funzione visualizzaMenu() che contiene un insieme di stampe a video relative al menu principale

19

Programma client mainmatrice.c



- Dopo aver visualizzato il menu, si attende la scelta effettuata dall'utente per eseguire una delle possibili operazioni
 - Si utilizza una struttura di controllo switch
- Per ogni scelta si scrive una funzione che esegue l'operazione corrispondente
- Se l'utente sceglie di calcolare la trasposta di una matrice letta da tastiera
 - void calcolaTrasposta();
- Se l'utente sceglie di calcolare la trasposta di una mat. letta da un file
 - void calcolaTraspostaDaFile();
- Se l'utente sceglie di confrontare due matrici lette da tastiera
 - void confronta();
- Se l'utente sceglie di confrontare due matrici lette da due file
 - void confrontaDaFile();
- Dopo aver calcolato la trasposta viene data la possibilità di visualizzarla a video e/o scriverla su file
- Per fare questa operazione si utilizza una funzione che fornisce un ulteriore menu che permette di fare queste operazioni:
 - void scriviTrasposta(matriceT *trasp);

20

Programma client mainmatrice.c



```
#include<stdio.h>
#include"matrice.h"
void visualizzaMenu();
void calcolaTrasposta();
void calcolaTraspostaDaFile();
void confrontaMatrici();
void confrontaMatriciDaFile();
void scriviTrasposta();
main(){
    int scelta;
    do {
        visualizzaMenu();
        scanf("%d",&scelta);
        switch (scelta){
            case 1: calcolaTrasposta(); break;
            case 2: confrontaMatrici (); break;
            case 3: calcolaTraspostaDaFile(); break;
            case 4: confrontaMatriciDaFile(); break;
            case 9: break;
            default: printf("Opzione inserita non valida!");
        }
    } while(scelta!=9);
}
```

21

Programma client mainmatrice.c



```
void visualizzaMenu(){
    printf("----- OPERAZIONI SU MATRICI -----\n");
    printf("-1- Calcola trasposta (lettura da tastiera)\n");
    printf("-2- Confronta due matrici (lettura da tastiera)\n");
    printf("-3- Calcola trasposta (lettura da file)\n");
    printf("-4- Confronta due matrici (lettura da file)\n");
    printf("-9- Esci\n");
}
void calcolaTrasposta(){
    char nomeFile[30];
    matriceT mat, trasp;
    printf("Inserire la matrice da tastiera\n");
    leggiMatrice(&mat);
    trasposta(&mat, &trasp);
    scriviTrasposta(&trasp);
}
```

22

Programma client mainmatrice.c



```
void calcolaTraspostaDaFile(){
    char nomeFileMat[30];
    matriceT mat, trasp;
    printf("Inserire il nome del file\n");
    scanf("%s",nomeFileMat);
    leggiMatriceDaFile(&mat, nomeFileMat);
    trasposta(&mat, &trasp);
    scriviTrasposta(&trasp);
}
```

23

Programma client mainmatrice.c



```
void scriviTrasposta(matriceT *trasp){
    char nomeFile[30];
    int scelta;
    do {
        printf("-1- Scrivi la trasposta a video\n");
        printf("-2- Scrivi la trasposta su file\n");
        printf("-3- Esci da questo menu\n");
        scanf("%d",&scelta);
        switch (scelta){
            case 1: scriviMatrice(trasp); break;
            case 2:
                printf("Inserire il nome del file\n");
                scanf("%s",nomeFile);
                scriviMatriceSuFile(trasp, nomeFile);
                break;
            case 3: break;
            default: printf("Inserire una opzione valida!");
        }
    } while(scelta!=3);
}
```

24