Programmazione I

a.a 2008-2009

docente: Carmine Gravino

Sviluppo dei Programmi

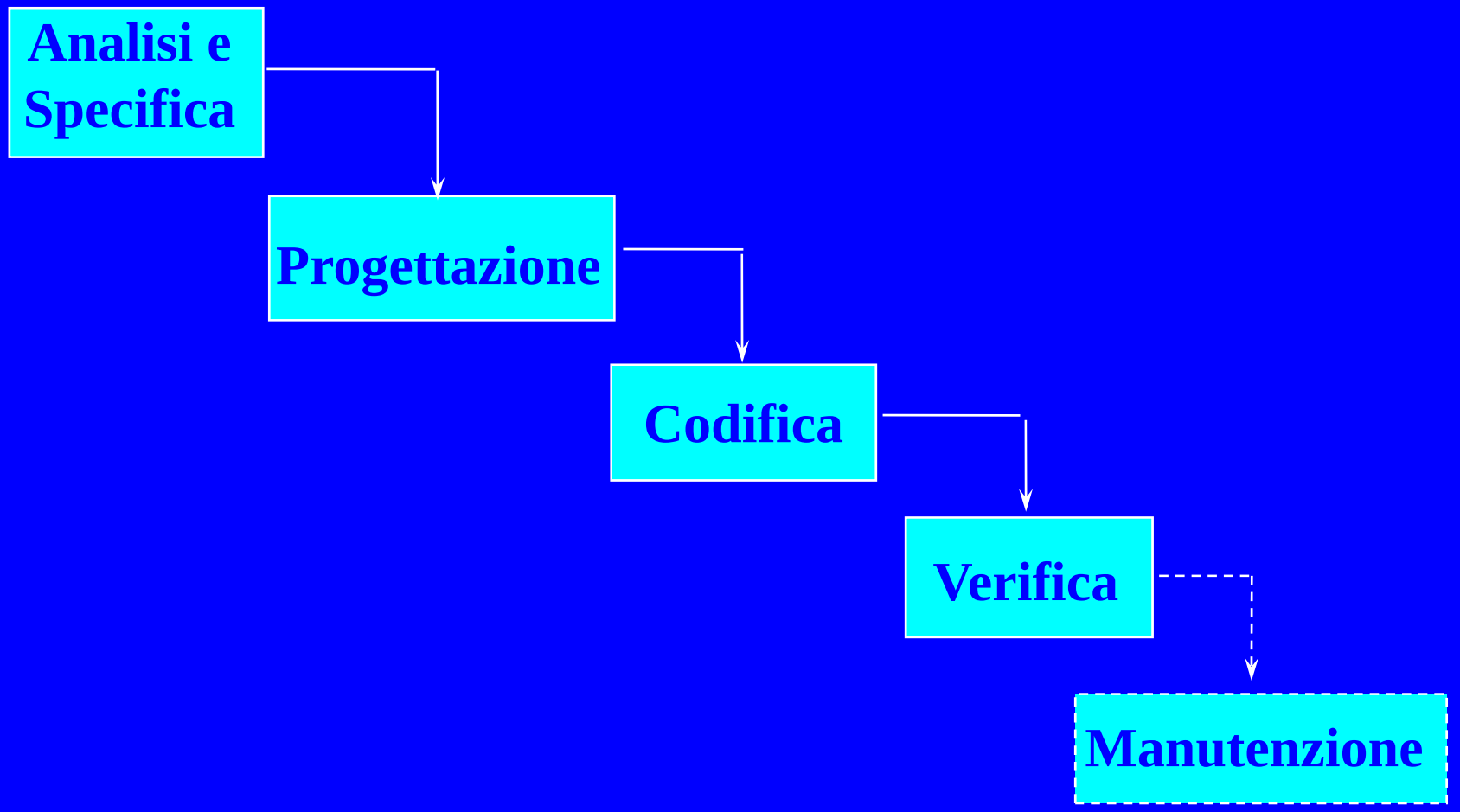
… alcune note su:

decomposizione funzionale

e integrazione di sottoprogrammi

Presentazione realizzata dal Prof. Andrea De Lucia

... il ciclo di vita del software



Decomposizione funzionale

- In fase di progettazione e raffinamento dell'algoritmo (*stepwise refinement*) si può decidere di raffinare azioni complesse utilizzando dei sottoprogrammi;
- *Esempio*:

  ...

  Calcola il fattoriale di n e assegnalo ad fn

  Calcola il fattoriale di m e assegnalo ad fm

  Calcola il massimo di fn e fm e assegnalo a max

  ...

- in questi casi, il primo passo è quello di definire la specifica dei sottoprogrammi ...

Specifica del sottoprogramma

- Ad esempio per il sottoprogramma fattoriale …
  - **Parametri di ingresso: *n***
  - **Parametri di uscita: *fatt***
  - **Precondizione: *n* >= 0**
  - **Postcondizione: se n = 0 allora fatt = 1**
    altrimenti *fatt* = *n* * (*n*-1) * … * 2 * 1

- In questo modo, è possibile definire l'interfaccia (il prototipo) del sottoprogramma:
  - **int fattoriale(int n);**

… nell'esempio si è deciso di realizzare una funzione in cui il parametro di uscita è restituito come valore di ritorno

Come si prosegue ?

- Definita l'interfaccia tra i sottoprogrammi, sono possibili due strade alternative:
  - ☞Continuare a raffinare l'algoritmo da cui è scaturita la decomposizione funzionale, ad esempio**:**

    …

    fn = fattoriale(n);

    fm = fattoriale(m);

    Calcola il massimo di fn e fm e assegnalo a max

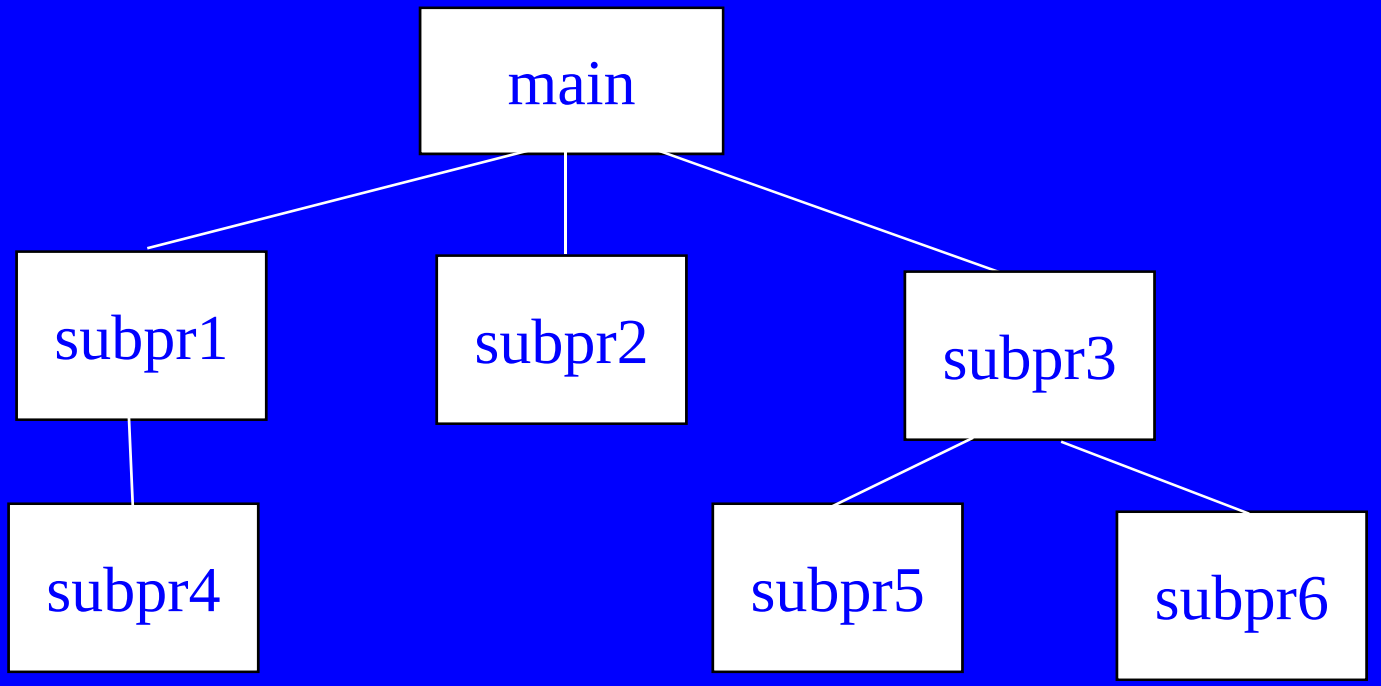
    …
  - ☞Progettare l'algoritmo del sottoprogramma …
    - ***… nel nostro caso il fattoriale ...***

Verifica

- Finora abbiamo visto come selezionare i casi di prova e verificare programmi costituiti dal solo main ...
- Come integrare e verificare un programma costituito da più sottoprogrammi ?
- Prima ipotesi: integra il programma con tutti i sottoprogrammi e verificalo nel suo insieme (strategia big-bang)
  - ☞poco efficace …
  - ☞difficile il debugging ...

… strategie più efficaci se si considera la struttura delle chiamate tra sottoprogrammi (architettura del programma) ...

Strategia bottom-up



- verificare prima i sottoprogrammi terminali (più in basso) e poi via via quelli di livello superiore ...
  - ☞**un sottoprogramma può essere verificato se tutti i sottoprogrammi che usa (chiama) sono stati verificati**

Strategia bottom-up e driver

- per ogni sottoprogramma da verificare è necessario costruire un programma (detto ***driver***) che:
  - ☞acquisisce i dati di ingresso necessari al sottoprogramma;
  - ☞invoca il sottoprogramma passandogli i dati di ingresso e ottenendo i dati di uscita;
  - ☞visualizza i dati di uscita del sottoprogramma

… occorre individuare i casi di prova dalla specifica del sottoprogramma ...

Esempio driver per sottoprogramma fattoriale

```
#include <stdio.h>

main()
{
   int n;
   printf(“Inserisci un intero positivo: ”);
   scanf(“%d”, &n);
   printf( “Fattoriale di %d: %d \n”, n, fattoriale(n));
}
```

… possibili casi di prova con valori di n pari a -2, 0, 4 ...

Vantaggi di strategie di integrazione

- ***Vantaggi***: migliore localizzazione di errori
  - ☞ **nella strategia bottom-up se un sottoprogramma è stato verificato (bene), è improbabile che un malfunzionamento che si verifica in seguito alla sua integrazione con un sottoprogramma chiamante (di livello superiore) sia causato da un errore nel codice del sottoprogramma già verificato**
- Costi maggiori di progettazione, ma costi minori di debugging rispetto alla strategia big-bang
- Altre strategie di integrazione ...
  - ☞ ***top-down***
  - ☞ ***sandwich***