Programmazione procedurale

Linguaggio di Riferimento: C

Linguaggio macchina

- Il linguaggio macchina costituisce la forma espressiva atta a descrivere programmi e ad essere *direttamente interpretata dall'unità di controllo*
- La programmazione in linguaggio macchina richiederebbe:
  - ☞ la conoscenza dell'architettura del calcolatore
  - ☞ la pianificazione dell'impiego della memoria
  - ☞ la definizione di un numero notevole di istruzioni elementari
- molte cause di errori e difficoltà di controllarli

Linguaggi simbolici

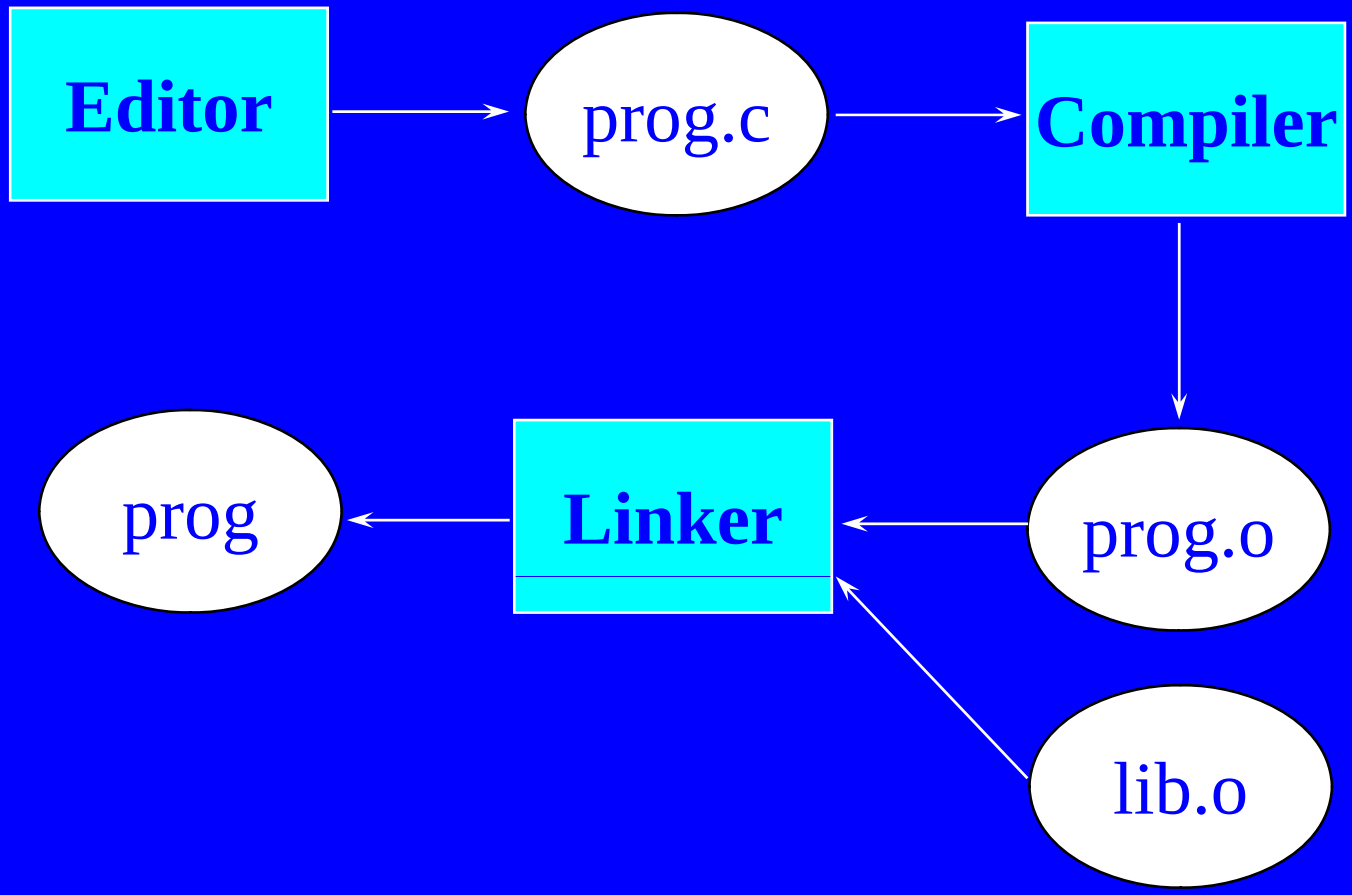
- I linguaggi simbolici consentono di esprimere il programma secondo regole ancora formali, ma con istruzioni più sintetiche e vicine alla logica del pensiero umano
- L'impiego dei linguaggi simbolici è possibile grazie alla disponibilità del *software di base*

Software di base

- ***editor*** (produzione del *programma origine*);
- ***compilatore*** (produzione del *programma oggetto*, mediante la traduzione del programma origine da linguaggio simbolico a linguaggio macchina)

  ☞... in alternativa si può usare un interprete ...
- ***collegatore*** (produzione del *programma eseguibile* mediante il collegamento del programma oggetto con componenti di librerie)
- ***caricatore*** (caricamento del programma eseguibile in memoria al fine della sua esecuzione)

Dal programma sorgente al programma eseguibile



Struttura dei Programmi

Apertura e chiusura di un commento

```
/*  File: hello.c
    ------------
    Questo programma visualizza il messaggio
    "Hello, world." sullo schermo.  */

#include <stdio.h>
main()
{
    printf("Hello, world.\n");
}
```

inclusione di libreria

programma principale

Librerie

- Una libreria contiene componenti scritte da altri programmatori che possono essere riusate nel nostro programma
  - *le informazioni sulle componenti disponibili nella libreria sono fornite in un* header *file (con estensione '.h') da includere nel programma* (#include <stdio.h>)
  - *l'estensione* .h *indica un header file, così come l'estensione* .c *indica un file contenente un programma C*
  - *la libreria* descritta dall'header file stdio.h *è predefinita*

Il main program

- Un ***programma*** è formato da uno o più blocchi chiamati ***funzioni***
- Una ***funzione*** è formata da una intestazione e da un corpo contenente una o più istruzioni
  - ***in un programma esiste una sola funzione* main**

```
main()
{
   printf("Hello, world.\n");
}
```

intestazione

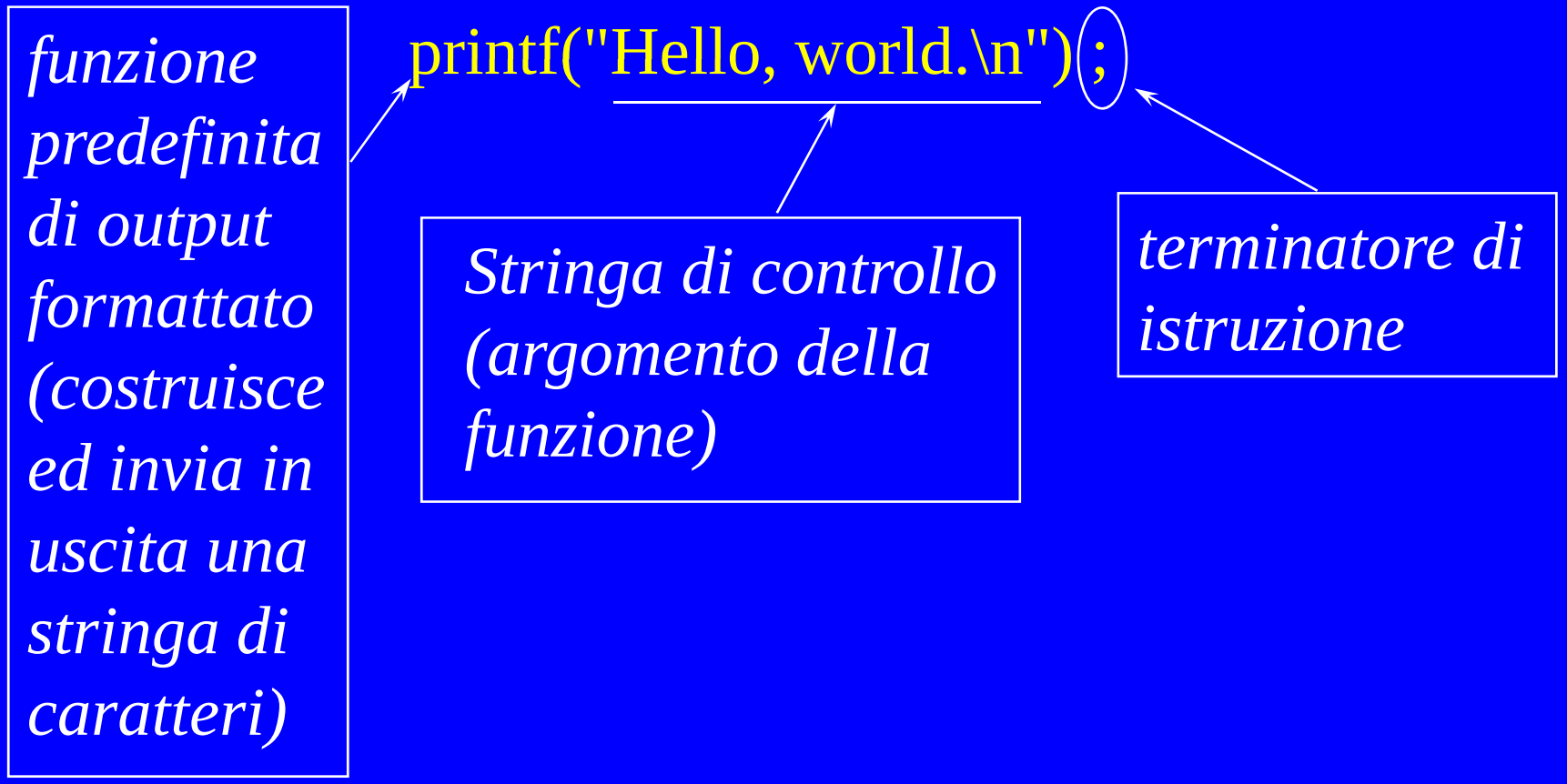
inizio del corpo

istruzione

fine del corpo

Istruzione di uscita

- Nella libreria **stdio.h** l'uscita standard viene normalmente inviata sullo schermo



Dati di uscita

- Nell'esempio precedente la stringa di controllo è una sequenza di caratteri (stringa) costante:

Hello, world.\n ← *carattere di ritorno a capo (**newline**)*

- In C una stringa di caratteri viene racchiusa tra doppi apici
- Sullo schermo comparirà il messaggio:

```
Hello, world.
```

Istruzioni di I/O

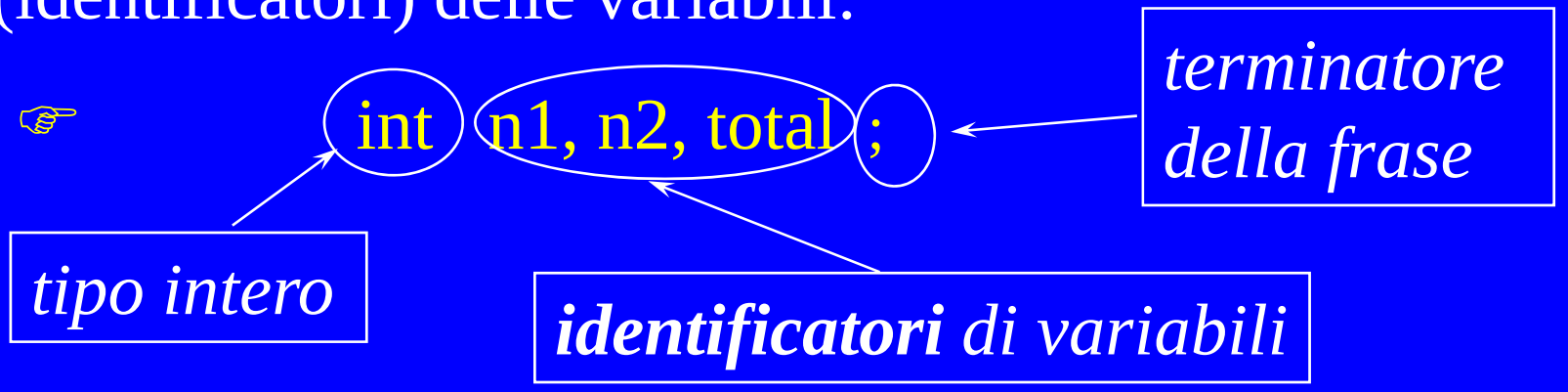
- In C non sono definite istruzioni o operatori specifici volti a realizzare le classiche operazioni di ingresso/uscita
- queste funzioni sono delegate ad apposite librerie esterne predefinite
  - ***... di cui bisogna includere l'header file nel programma***
  - **stdio.h** ***per il C***

Variabili e tipi

- Una ***variabile*** è una entità che può assumere un valore qualunque all'interno di un insieme di valori (***tipo***)
- I tipi possono essere ***predefiniti*** dal linguaggio di programmazione (***tipi primitivi***) o definibili attraverso appositi costrutti messi a disposizione dal linguaggio (***tipi di utente***)
- I tipi possono essere ***semplici*** (l'informazione è definita come un oggetto non decomponibile in altri oggetti) o ***strutturati*** (l'informazione è definita come un oggetto decomponibile in altri oggetti)

Dichiarazioni di Variabili

- Le informazioni usate in un programma (e quindi anche le variabili) vanno definite nel tipo e nell'attributo (nome), prima di essere usate
- I linguaggi mettono a disposizione apposite frasi per dichiarare le ***variabili*** usate nel programma
- Queste frasi sono in generale costituite da una parola chiave (per il tipo) e dalla lista dei nomi (identificatori) delle variabili:

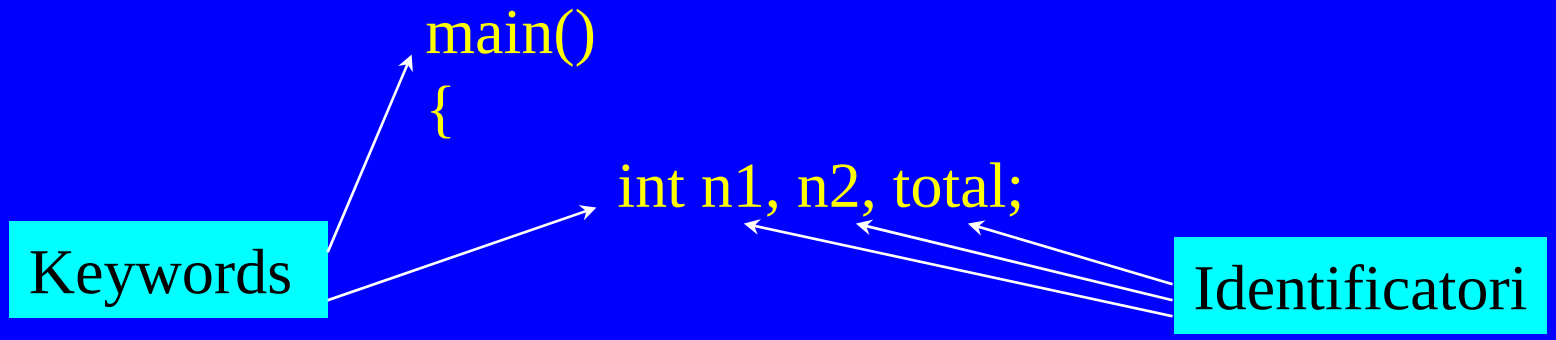


Identificatori e Keywords

Un programma C è scritto utilizzando:

- Keywords: parole proprie del linguaggio di C (ad es. Istruzioni o tipi predefiniti)
- Identificatori: parole definite dall'utente (es. variabili, costanti, tipi definiti dall'utente)

  ☞*non possono coincidere con alcuna keyword ...*



Oltre a keywords e identificatori si usano anche simboli speciali (ad esempio parentesi, simboli di interpunzione)

Identificatori in C

- Gli identificatori di qualsiasi oggetto in un programma possono consistere di un numero qualsiasi di caratteri alfanumerici minuscoli o maiuscoli incluso il carattere "_" (*underscore*).
- Il primo carattere deve necessariamente essere una lettera oppure il carattere underscore.
- Il compilatore fa differenza tra lettere minuscole e maiuscole

  Esempi validi:

  sp_addr sp2_addr F_lock_user _found

  Esempi non validi:

  20_secolo -pippo

Preprocessore C

- Una estensione al linguaggio che fornisce le seguenti possibilità:
  - ☞inclusione di file
  - ☞definizione di costanti
- Inoltre:
  - ☞definizione di macro sostituzioni
  - ☞compilazione condizionale
- Noi useremo solo le prime due …

Preprocessore C

- I comandi del preprocessore iniziano con # nella prima colonna del file sorgente e non richiedono il “**;**” alla fine della linea.
- Un compilatore C esegue la compilazione di un programma in due passi successivi.
  - ☞Nel primo passo usa il preprocessore per sostituire ogni occorrenza testuale definita attraverso la direttiva # con il corrispondente testo da inserire (file, costanti, macro)
  - ☞La compilazione vera e propria avviene nel secondo passo

Preprocessore C: Inclusione di file

- Il comando di inclusione **#include** permette di inserire nel sorgente che contiene la direttiva di inclusione, il file specificato, a partire dal punto in cui è presente la direttiva.

  #include “const.h”

  #include <stdio.h>

- Nel primo caso il file da includere verrà ricercato nella directory corrente, nel secondo verrà cercato in quella di default.
- E’ convenzione che i file da includere abbiano estensione **.h** (header file).

Preprocessore C: Macro

- L'uso della direttiva **#define** consente anche di definire delle macro.
  - **Una macro è una porzione di codice molto breve che è possibile rappresentare attraverso un nome; il preprocessore provvederà ad espandere il corrispondente codice in linea.**
- Una macro può accettare degli argomenti, nel senso che il testo da sostituire dipenderà dai parametri utilizzati all'atto del suo utilizzo. Il preprocessore espanderà il corrispondente codice in linea avendo cura di rimpiazzare ogni occorrenza del parametro formale con il corrispondente argomento reale.

Preprocessore C: Costanti

- Attraverso la direttiva #define del preprocessore è possibile definire delle costanti:

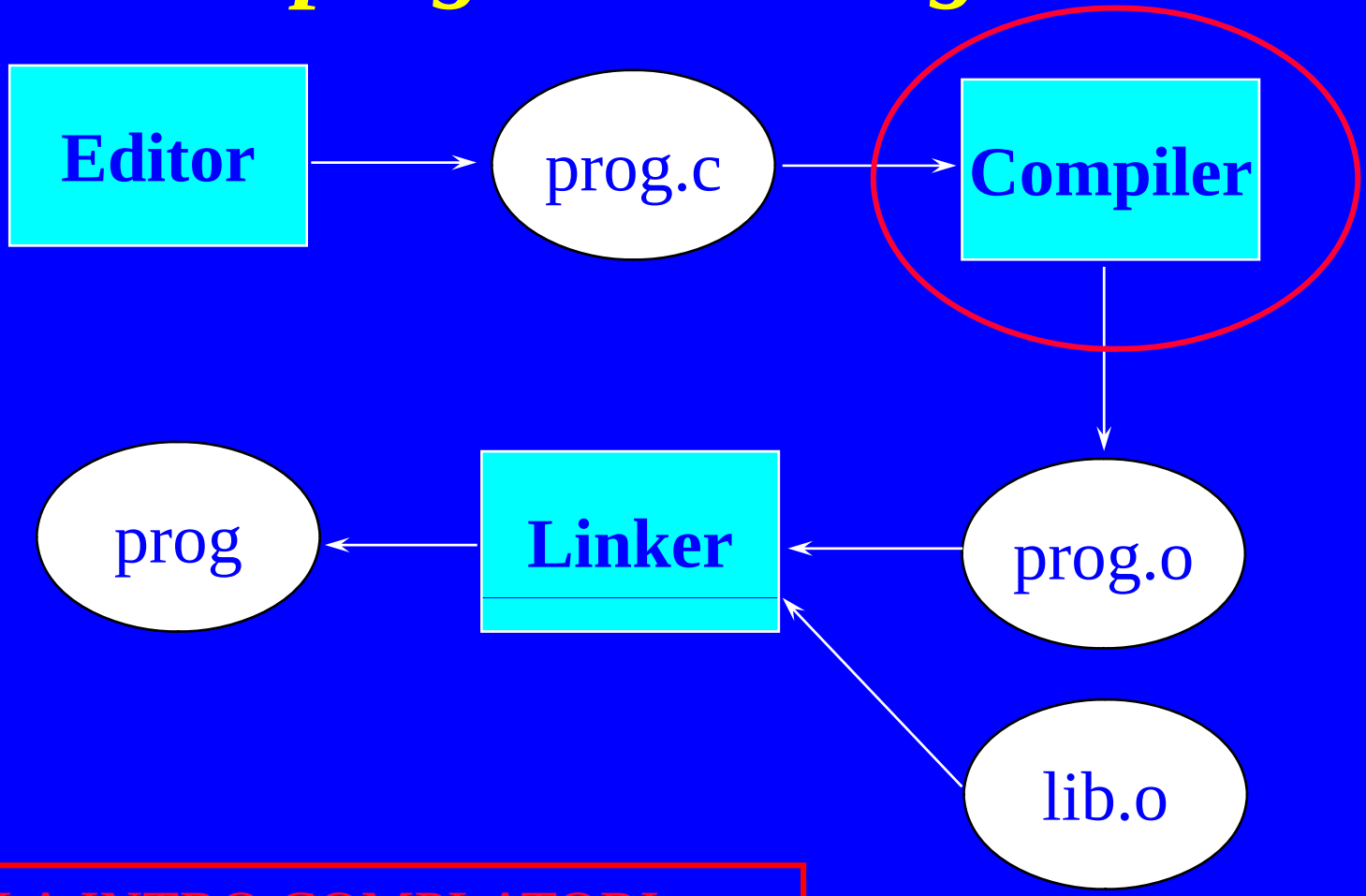
  #define ***nome*** ***testo da sostituire***

- Esempi

```
#define     MAXLEN   100
#define     YES      1
#define     NO               0
#define  ERROR    “File non trovato\n”
```

- E’ uso comune indicare per le costanti (e per le macro in genere) le lettere maiuscole.
- L’uso di costanti e macro favoriscono la leggibilità del programma e consentono un facile riuso del codice

Dal programma sorgente al programma eseguibile



VAI A INTRO COMPLATORI ->