Università degli Studi di Ferrara

Corso di Laurea in Matematica - A.A. 2021 - 2022

Programmazione
Lezione 4 – il Linguaggio C

Docente: Michele Ferrari - michele.ferrari@unife.it

Nelle lezioni precedenti

- Problemi e Calcolatori: definire i passaggi risolutivi di un problema prima in linguaggio naturale e poi in una forma più vicina al calcolatore
- Algoritmi: definizione della risoluzione di un problema per passi successivi
- Costruire un Algoritmo: scegliere la rappresentazione di un algoritmo
- Programmazione strutturata: rappresentare l'algoritmo per mezzo di sequenze/selezioni/iterazioni
- Dal Problema al Programma: codificare l'algoritmo
- Linguaggi di programmazione: metodi di codifica di un algoritmo in modo che diventi eseguibile in un calcolatore

In questa lezione

- Il Linguaggio C
- Scrivere un programma in C
- Variabili e tipi di dato
- Compilare un programma in C

Il Linguaggio C

- E’ un linguaggio di programmazione di Alto Livello sviluppato da Dennis Ritchie nel 1972 come evoluzione del linguaggio B (e BCPL)

Il Linguaggio C

- Dal linguaggio C derivano direttamente linguaggi sviluppati come evoluzione (C++) o superset (Objective-C) con i quali si sviluppano i software moderni

Linguaggio C: storia

- 1972 Dennis Ritchie implementa il linguaggio C presso i Laboratori Bell per la programmazione del sistema operativo UNIX
- 1978 Dopo qualche anno di evoluzione, il libro "Il linguaggio di programmazione C" di Kernighan e Ritchie definisce il C tradizionale (K&R C).
- 1989 Approvato lo standard ANSI C, indipendente dalla macchina. Esce una versione del testo K&R aggiornata allo standard.
- 1999 Standard C99
- 2011 Standard C11

Il Linguaggio C

Supporta la programmazione strutturata (suddivisione del programma in moduli funzionali)

Introdotto come linguaggio per la programmazione del Sistema Operativo Unix

Utilizzato fino ad oggi per progetti di ogni tipo e complessità: dalla utility più semplice a sistemi operativi come GNU-Linux

Linguaggio C: scheda tecnica

- Statico (tipizzazione)
- Non garbage collected
- Mutabile
- Procedurale
- Strutturato

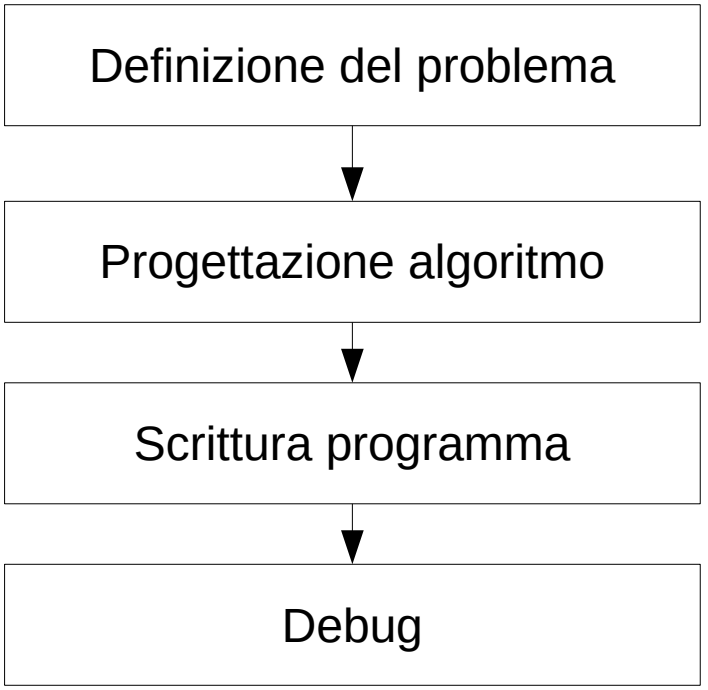
Considerato di basso livello fra i linguaggi di alto livello.

Ottimo per applicazioni che richiedono controllo del processo (software di sistema, calcolo numerico, multimedia fra le tante).

Tra i linguaggi più usati (insieme a Java).

Disponibile per praticamente qualsiasi piattaforma hardware e portabile **se si segue lo standard ANSI C**.

Realizzare un programma: le fasi



Codifica di un programma C

- Editing
- Compilazione
  - Precompilazione
  - Compilazione
  - Assemblaggio
  - Link
- Esecuzione
  - Caricamento in memoria
  - Esecuzione

Codifica di un programma C

La codifica di un programma in C comprende 4 fasi (inclusa l'esecuzione)

- Editing
- Compilazione
- Link
- Esecuzione

Editing

È la fase in cui il programmatore scrive il codice necessario a codificare l'algoritmo.

La scrittura del codice, in genere avviene attraverso l'uso di editor specifici che aiutano il programmatore nella stesura del codice evidenziando con colori diversi le parole chiave del linguaggio.

Il file sul quale viene salvato il codice sorgente deve avere estensione .c e il nome del file NON deve avere spazi

Editing - Convenzioni

Ogni linguaggio di programmazione presenta delle **Convenzioni Stilistiche,** che è importante seguire:

- Posizione delle parentesi { } (per quelli che li prevedono)
- Stile dei nomi delle funzioni (o metodi)

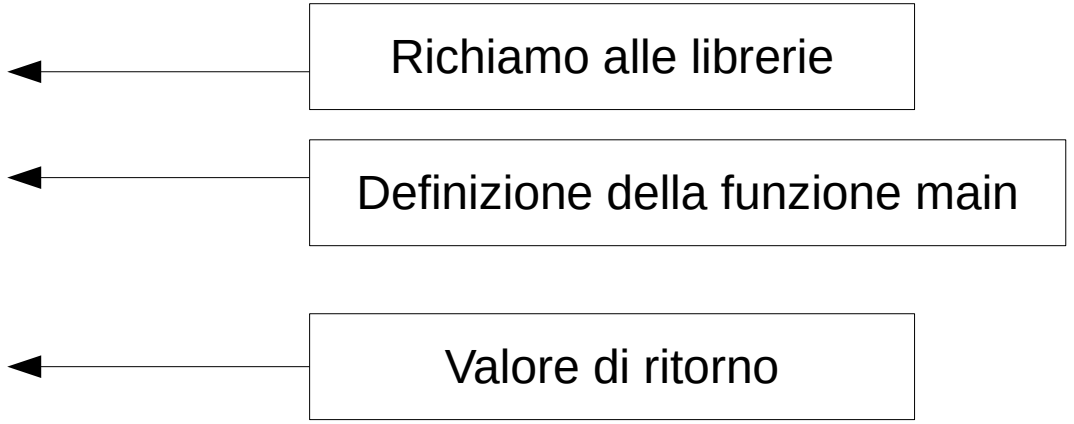
TUTTI i linguaggi di programmazione prevedono una stessa “buona pratica”:

- Indentazione
- Commenti al codice

Struttura di un programma in C

Tutti i programmi scritti in C presentano questa struttura:

```
#include <stdio.h>
int main (void) {
// codice
return 0;
}
```



Compilazione

Una volta terminata la scrittura del codice sorgente, il programmatore esegue la compilazione del codice sorgente per produrre il file eseguibile dal calcolatore.

La compilazione comprende diverse fasi:

- Precompilazione
- Compilazione
- Assemblaggio

Precompilazione

Viene eseguita dal preprocessore C che ha il compito di espandere alcune forme abbreviate.

Vengono copiati all’interno del sorgente i file header indicati attraverso la parola chiave

#include

Vengono Sostituiti alle costanti i loro valori indicati con la macro istruzione #define

Compilazione

Il codice espanso, viene tradotto in codice assembly.

In questa fase vengono evidenziati gli errori di sintassi all'interno del codice prodotto.

Nell'ultima fase della compilazione, il codice scritto in linguaggio assembly, viene tradotto in una forma chiamata codice oggetto rilocabile, viene salvato in un file con estensione".o"

• Codice Oggetto: codice in linguaggio macchina nella maggior parte dei casi non eseguibile direttamente.

• Il codice oggetto è composto normalmente da codice eseguibile, più una serie di informazioni che permettono al linker di unirlo, se richiesto, con altri codici oggetto per generare un programma funzionante.

Assemblaggio

Il codice assembly prodotto durante la fase di compilazione subisce un ulteriore trasformazione.

Il file .o viene assemblato e trasformato in codice macchina eseguibile dal calcolatore.

Link

Durante la fase di link, nel caso il nostro codice sorgente sia diviso in più file, questi vengono uniti e viene incluso anche il codice delle librerie utilizzate.

Se il codice sorgente è incluso in un unico file, solitamente, questa fase è trasparente (cioè non si distingue questa fase dalla compilazione).

Il comando gcc

gcc -o nome_programma file_sorgente.c

- gcc comando del compilatore
- -o opzione di compilazione
- nome_programma nome che avrà il programma eseguibile
- file_sorgente.c nome del file .c che contiene il nostro codice sorgente

Codifica di un Algoritmo

Lo scopo principale di scrivere programmi è quello di “far eseguire” gli algoritmi al calcolatore.

- Come un semplice algoritmo viene codificato con il linguaggio C?

Codifica di un Algoritmo

Riprendiamo il classico esempio di algoritmo per calcolare l'area di un rettangolo.

Per semplicità, per questo primo esempio faremo riferimento al calcolo dell'area di un rettangolo specifico.

- Assegna a Base il valore 3;
- Assegna ad Altezza il valore 5;
- Assegna ad Area il valore Base * Altezza;
- Scrivi Area;

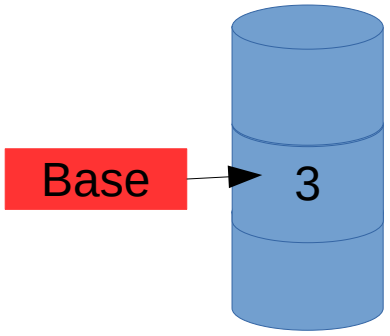
Le Variabili

Le variabili sono delle porzioni di memoria a cui viene assegnato (per comodità) un nome.

Durante la vita del nostro programma alle variabili è possibile assegnare più valori e utilizzarle per svolgere i calcoli definiti dall'algoritmo.

Prendiamo ad esempio la prima riga del nostro algoritmo:

Assegna a Base il valore 3;



Base è il nome di una variabile
che contiene il valore 3

Variabili - Dichiarazione

Il linguaggio C è un linguaggio tipizzato, per questo motivo, prima di utilizzare una variabile è necessario dichiarare che intendiamo utilizzarla attraverso un'operazione detta

dichiarazione della variabile

Con l’operazione di dichiarazione della variabile, definiamo il suo nome ed anche il suo tipo.

La sintassi per dichiarare una variabile è la seguente:

```
<tipo-variabile> <nome-variabile>;
```

Variabili - Tipi di Dato

- Int (e.g. → 5)
- Float (e.g. → 3.4)
- Double (e.g. → 3.5678890)
- Char (e.g. → m)

Nella realtà queste dimensioni sono fortemente dipendenti dalla architettura, e sono definite nel file limits.h in cui sono definiti i valori massimi e minimi per ogni tipo di dato.

Variabili - Nomi

Esistono inoltre delle regole da rispettare nella costruzione degli identificatori:

- Devono iniziare con una lettera o con un underscore “_”
- Possono contenere lettere, cifre e “_”
- Il C è case-sensitive, quindi la variabile Pippo è diversa dalla variabile pippo
- Per quanto riguarda la lunghezza occorre tenere presente che soltanto i primi 32 caratteri sono significativi, anche se nelle versioni del C meno recenti questo limite scende a 8 caratteri
- Non è possibile usare parole chiave riservate del linguaggio come nomi delle variabili (o costanti)

Variabili - La memoria

Una variabile rappresenta un’area di memoria alla quale il nostro programma accede per leggere o scrivere valori.

Il programma accede (in lettura e scrittura) alla cella desiderata per mezzo dell’indirizzo.

(per questo motivo la memoria accessibile a un programma si dice anche spazio di indirizzamento)

Ogni riferimento al nome della variabile è in realtà un riferimento al valore in essa contenuto.

File Header

La prima riga del programma (#include) è una direttiva al preprocessore C.

Il suo compito è di dare indicazioni al compilatore C di includere i riferimenti alla libreria standard di input/output, in pratica, vengono resi disponibili all'interno del codice che stiamo scrivendo tutte le funzioni definite nella libreria.

Tutto il codice presente nel file stdio.h viene inserito nella posizione dell'include.

Altre librerie comuni sono:

- stdlib.h
- time.h
- string.h

i Commenti

I commenti sono porzioni di testo che vengono ignorate dal compilatore in fase di compilazione.

I commenti sono importanti per descrivere cosa stiamo facendo all'interno del codice.

Inserire i commenti nel nostro codice è importante per spiegare ad altri e ricordare a noi stessi cosa stiamo facendo in quel punto del codice.

- Su più righe:

```
/* Questo commento
    è su
  più righe */
```

- Su una sola riga:

```
// Questo su una sola
```

Funzione main()

Quando un programma scritto in C viene eseguito, il kernel “cerca” questa funzione come punto di partenza dell’esecuzione del programma.

```
int main(void){ }
```

TUTTI i programmi C devono contenere una ed una sola funzione main.

Il valore di ritorno della funzione main() informa il kernel dello stato di esecuzione del programma:

= 0 → Il programma ha terminato l’esecuzione con **successo**

≠ 0 → Il programma ha terminato l’esecuzione con **un errore**

Variabili - Dichiarazione

All’interno della funzione main() sono presenti le dichiarazioni delle variabili necessarie ad eseguire il nostro programma:

```
int area, base, altezza;
```

Nel nostro caso area, base e altezza sono variabili di tipo intero (int).

Anche le dichiarazioni così come le altre istruzioni (con alcune rare eccezioni) devono terminare con un punto e virgola.

Assegnare un valore

Effettuata la dichiarazione, la variabile può essere utilizzata.

Ad una variabile definita si può assegnare un valore (di un tipo coerente) con l'operatore di assegnamento (=)

Il valore può essere il risultato di una espressione:

```
area = base * altezza;
```

Riassumendo

Formalmente,le variabili sono aree riservate di memoria, identificate da un *nome* e da un *tipo*.

- **Nome** utilizzato per accedere alle variabili in lettura e scrittura

  (in C lettera o underscore seguito da un numero qualsiasi di lettere, cifre o underscore)

- **Tipo** ne determina la dimensione in memoria, i valori ammissibili e le operazioni possibili

  (es. int, float, struct, puntatore, etc.)

➔ Il contenuto di una variabile in un dato momento è detto valore della variabile.

Output di un programma

La funzione `printf()` visualizza a schermo una stringa formattata.

La formattazione della stringa avviene utilizzando i formattatori che sono specificati come parti della stringa.

Alcuni esempi:

```
printf ( " Hello World " );
printf ( " Il risultato è: %d " , area );
```

Nella seconda printf() è stato inserito il formattatore %d il quale indica che al posto dello stesso deve essere inserito un valore intero che viene specificato alla fine dell'istruzione dopo la chiusura degli apici.

Esempio

```
#include <stdio.h>

int main(void)
{
int area, base, altezza;

/* I valori di base e altezza sono fissati */
base = 3;
altezza = 5;

/* Assegna alla variabile area il prodotto di base e altezza */
area = base * altezza;

/* Stampa il risultato a video */
printf("Area = %d\n", area);

return 0;
}
```

La funzione printf()

Tipo di dato	Formattatore
intero	`%d`
float	`%f`
double	`%f` (o `%lf`)
char	`%c`

Visualizzare i risultati

```
printf(“Base:%d”, base);

printf(“Base: %d”, base);

printf(“Base: %d,\n Altezza: %d”,base,altezza);

printf(“Base: %d,\t Altezza: %d”,base,altezza);
```

Grazie per l’attenzione

Riferimenti

Il corso di programmazione per il primo anno della Laurea Triennale in Matematica nasce con l'intento di unire ai principi di programmazione una conoscenza basilare di uno degli strumenti software più diffusi nell'ambito matematico: Matlab.

La prima parte del corso pertanto è una rielaborazione del programma di Programmazione per la Laurea Triennale in Informatica 15/16 (in particolare) e 16/17.

Parte del materiale originale da cui ho ricavato il percorso didattico, alcuni approfondimenti ed integrazioni possono essere trovati in:

- Lezioni di Programmazione 15/16 CdS Informatica - Giacomo Piva
- Lezioni di Programmazione 16/17 CdS Informatica - Marco Alberti