Introduzione al C

Alcuni difetti del linguaggio della macchina di von Neumann…

Meglio questo:

```
0)    READ
1)    STORE  101
2)    LOAD=  0
3)    STORE  102
4)    LOAD   101
5)    BEQ    13
6)    READ
7)    ADD    102
8)    STORE  102
9)    LOAD   101
10)   SUB=   1
11)   STORE  101
12)   BR     4
13)   LOAD   102
14)   WRITE
15)   END
```

... o questo?

```
              READ
              STORE     CONTATORE
              LOAD=     0
              STORE     SOMMA
CICLO_SOMMA:  LOAD      CONTATORE
              BEQ       STAMPA_FINALE
              READ
              ADD       SOMMA
              STORE     SOMMA
              LOAD      CONTATORE
              SUB=      1
              STORE     CONTATORE
              BR        CICLO_SOMMA
STAMPA_FINALE: LOAD     SOMMA
              WRITE
              END
```

Alcuni difetti del linguaggio della macchina di von Neumann…

- (a+b)*(c+d) diventerebbe ad esempio:
  - LOAD A
  - ADD B
  - STORE TEMP
  - LOAD C
  - ADD D
  - MULT TEMP

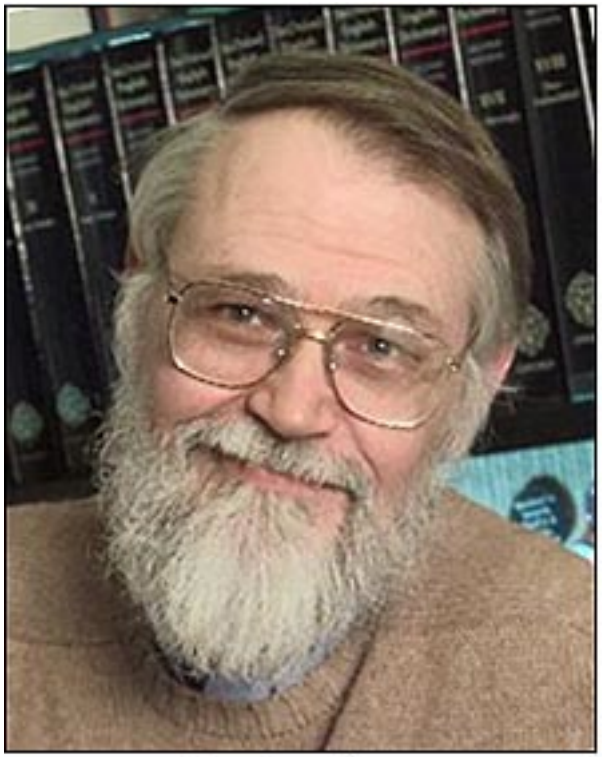
- Ripetere una serie di operazioni (cicli) è laborioso..
  - BEQ, BR, ….
- Più facile usare etichette con nomi intuitivi

Linguaggio C

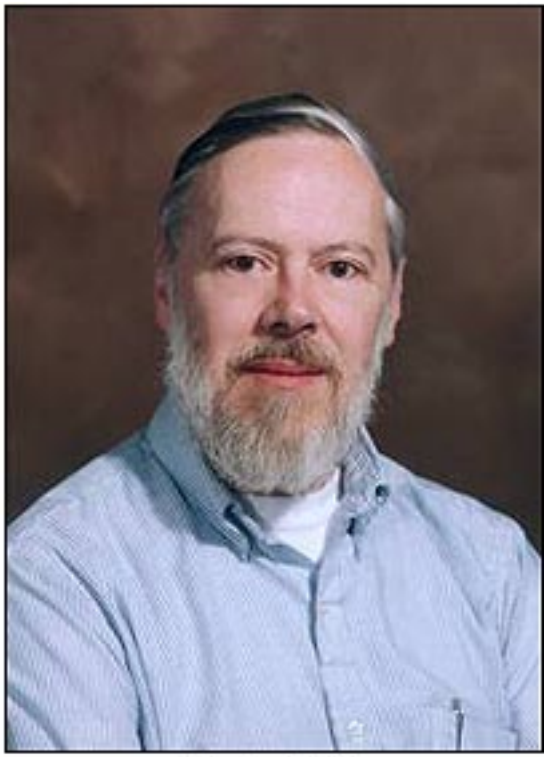
Verso l'infinito e oltre!

Linguaggio C

- Il primo passo verso la programmazione di **alto livello**



Brian Kernighan



Dennis Ritchie

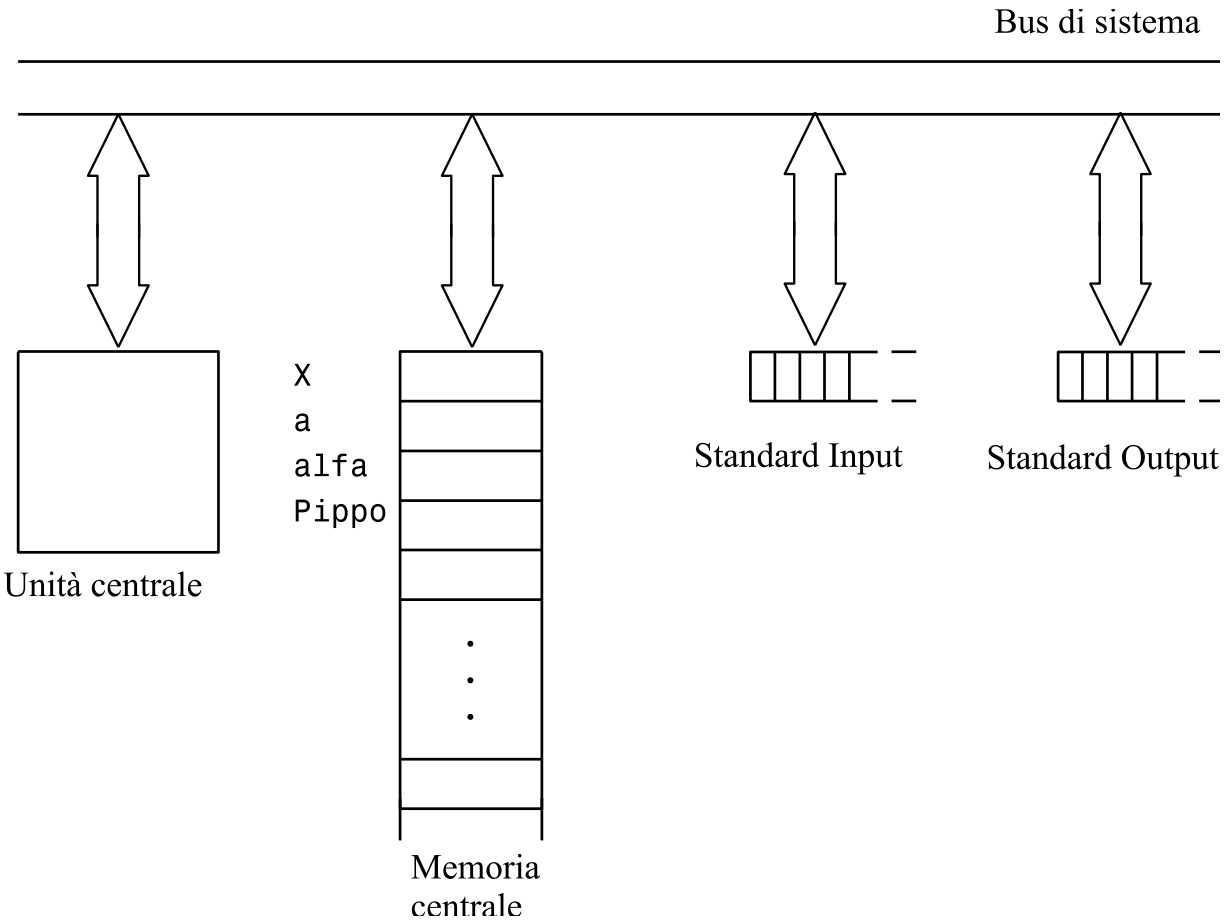
If programming languages were vehicles:

C was the great all-arounder: compact, powerful, goes everywhere, and reliable in [situations where your life depends on it.]

Alcune applicazioni

- Calcolo numerico e scientifico
- Applicazioni gestionali
- Servizi telematici (Internet)
- Automazione industriale
- Realtà virtuale
- …

La macchina (astratta) del C



Elementi essenziali

- Standard Input, Standard Output (come i "nastri" della macchina di von Neumann) e la memoria sono divisi in **celle elementari**, contenenti ciascuna un **dato singolo**
- Per il momento, **non poniamo limiti** fisici alla dimensione della memoria né dei supporti di ingresso e uscita:
  - numero di celle illimitato e ogni singola cella può contenere un qualsiasi valore numerico (sia intero sia reale) o un qualsiasi carattere
- **Stringa**: una successione finita di caratteri (per esempio Giorgio, ieri, alfa-beta…)
  - è immagazzinata in celle consecutive, ciascuna contenente un singolo carattere della stringa
- Le celle di memoria vengono chiamate anche **variabili**
  - diverso dall'omonimo concetto matematico!

...continua

- Le variabili, le istruzioni e altri elementi del programma che saranno introdotti più avanti sono indicati tramite **identificatori simbolici**
  - identificatore simbolico: una successione di lettere e cifre, in cui al primo posto vi è una lettera
  - il carattere speciale "_" viene considerato come cifra
  - le lettere maiuscole sono distinte dalle corrispondenti lettere minuscole
    - Var1, var1 e VAR1 sono tre diversi identificatori
- Esempi di identificatori C:
  - a, x, alfa, pippo, a1, xy23, Giuseppe, DopoDomani….
- Per evitare ambiguità **non** è possibile usare lo stesso identificatore per indicare diversi elementi né usare diversi identificatori per lo stesso elemento

...continua

- Alcuni identificatori sono **predefiniti e riservati**, nel senso che sono associati a priori a qualche elemento del linguaggio e pertanto non possono essere usati dal programmatore con significati differenti da quello predefinito
  - per esempio, **printf** indica un'operazione elementare di ingresso/uscita e non può essere impiegata in un programma C per indicare, per esempio, una variabile
- **Parola chiave:** parola predefinita del linguaggio di programmazione; anch'essa riservata; non può fungere da normale identificatore
- Per comodità di lettura (umana, non del calcolatore!) le parole chiave saranno scritte in neretto

Struttura sintattica di un programma C

- Un programma C è composto da:
  - un'**intestazione** seguita da
  - una **sequenza di istruzioni** racchiusa tra i simboli { e }
- L'**intestazione** è costituita dall'identificatore predefinito **main** seguito da una coppia di parentesi ( )
- Le **istruzioni** sono frasi del linguaggio di programmazione; ognuna di esse termina con il simbolo ';'

Istruzioni di assegnamento

- Assegna a una **variabile** il valore di un'**espressione**
  - consiste nel simbolo = preceduto dall'identificatore di una cella di memoria e seguito da un'espressione che definisce un valore
- L'espressione può essere costituita da valori costanti, identificatori di variabili, o una loro combinazione ottenuta mediante i normali operatori aritmetici (+, -, *, /) e parentesi, come nelle consuete espressioni aritmetiche
- Esempi:
  - x = 23;
    w = 'a';
    r3 = (alfa*43–xgg)*(delta–32*ijj);
    x = x+1;
- Se la cella a contiene il valore 45 e la cella z il valore 5, l'istruzione
  - x = (a–z)/10
  - fa sì che nella cella x venga immagazzinato il valore 4
- NB: per distinguere il valore carattere a dall'identificatore della variabile a, il primo viene indicato tra **apici** (similmente per le stringhe, vedremo tra breve).

Istruzioni di ingresso e uscita

- Consistono negli identificatori predefiniti **scanf** o **printf** seguiti da una coppia di parentesi che racchiude l'identificatore di una variabile
- Determinano la lettura o scrittura del valore di una variabile dallo Standard Input o sullo Standard Output in modo del tutto identico alle corrispondenti istruzioni del linguaggio della macchina di von Neumann

- Alcune comode abbreviazioni:
  - printf((a–z)/10);
  - abbreviazione per temp = (a-z)/10; printf(temp);
  - dove temp denota una variabile non usata altrimenti nel programma.
- Invece, printf("alfa");
  - abbreviazione per: printf('a'); printf('l'); printf('f'); printf('a');
- Qual'è la differenza tra l'istruzione printf(2) e l'istruzione printf('2')?

Primo programma C

- Stampiamo la media di due numeri inseriti da tastiera

```
main()
{
  scanf(x);
  scanf(y);
  z = (x+y)/2;
  printf(z);
}
```

- Qual è il suo equivalente nel linguaggio della macchina di von Neumann?

Istruzioni composte

- Istruzioni condizionali
- Istruzioni iterative

- Sono equivalenti ai vari BR, BEQ, … visti nella macchina di von Neumann, solo **molto** più potenti e semplici da usare

Istruzioni composte

- Le istruzioni composte producono effetti diversi a seconda che siano verificate o meno certe condizioni sul valore delle variabili
- **Condizione** (o espressione) **booleana**: un'espressione il cui valore può essere vero o falso
- Essa è costruita mediante
  - i normali operatori aritmetici,
  - gli operatori di relazione (==, !=, <, >, <=, >=),
  - gli operatori logici (!, ||, &&), corrispondenti, nell'ordine, alle operazioni logiche NOT, OR, AND
- Esempi di condizioni:
  - x == 0
    alfa > beta && x != 3
    !((a + b)*3 > x || a < c)
- Se le variabili x, alfa, beta, a, b e c contengono rispettivamente i valori 0, 1, 2, 3, 4 e 5, la valutazione delle tre condizioni precedenti dà come risultato, rispettivamente: V, F e F
- Esistono regole di precedenza tra gli operatori logici; per esempio, nell'espressione
  - x > 0 || y == 3 && z > w
  - l'operatore && deve essere eseguito prima dell'operatore ||, (in analogia con a + b * c)

Istruzioni condizionali

- Consente di eseguire in alternativa due diverse sequenze di istruzioni sulla base del valore di verità di una condizione
- E' costituita dalla parola chiave **if**, seguita da
  - una condizione racchiusa tra parentesi tonde,
  - dalla prima sequenza di istruzioni racchiusa in parentesi graffe,
  - (eventualmente) dalla parola chiave **else**,
  - dalla seconda sequenza di istruzioni racchiusa in parentesi graffe
- Il "ramo **else**" dell'istruzione può essere assente
- Le parentesi graffe vengono in genere omesse quando la successione di istruzioni si riduce a un' istruzione singola

Esempi di istruzioni condizionali

- if(x == 0) z = 5; else y = z + w*y;
  if(x == 0) {z = 5;} else {y = z + w*y;}
  if ((x+y)*(z-2) > (23+v)) {z = x + 1; y = 13 + x;}
  if ((x == y && z >3) || w != y) z = 5; else {y = z + w*y; x = z;}

- Semantica di un'istruzione condizionale:
  - primo, la macchina valuta la condizione, cioè stabilisce se il suo valore è vero o falso
  - nel caso "vero" esegue solamente la prima sequenza di istruzioni,
  - nel caso "falso" esegue la seconda sequenza di istruzioni,
  - se manca il ramo **else** e la condizione è falsa, la macchina prosegue con l'istruzione successiva all'istruzione condizionale

Secondo programma C

- Stampiamo il maggiore tra due numeri inseriti da tastiera

```
main()
{
  scanf(x);
  scanf(y);
  if (x > y) z = x;
  else z = y;
  printf(z);
}
```

- Qual è il suo equivalente nel linguaggio della macchina di von Neumann?

Istruzioni iterative

- Permettono la **ripetizione** dell'esecuzione di una sequenza di istruzioni ogni volta che una certa condizione è verificata
- Esempio: la parola chiave **while**, seguita dalla condizione racchiusa tra parentesi tonde, come per l'istruzione condizionale, e da una sequenza di istruzioni fra parentesi graffe
  - la sequenza di istruzioni è detta **corpo del ciclo**
- Esempi:
  - while (x >= 0) x = x – 1;
    while (z != y) {y = z – x; x = x*3;}
- Semantica di un'istruzione iterativa **while**:
  - la macchina valuta la condizione
  - se questa è falsa non viene eseguito il corpo del ciclo e si passa direttamente all'istruzione successiva
  - altrimenti si esegue una prima volta il corpo del ciclo; si valuta ancora la condizione e, nuovamente, si esegue il corpo del ciclo se essa è risultata vera
  - quando la condizione risulta falsa si esce dal ciclo, ovvero si passa all'istruzione successiva all'istruzione iterativa
  - in altre parole, il ciclo viene ripetuto finché la condizione rimane vera

Terzo programma C

- Stampiamo la somma di n numeri interi strettamente positivi inseriti da tastiera
  - il numero zero conclude la sequenza

```
main()
{
 somma = 0;
 scanf(addendo);
 while (addendo>0) {
        somma = somma + addendo;
        scanf(addendo);
 }
 printf(somma);
}
```

- Qual è il suo equivalente nel linguaggio della macchina di von Neumann?

Un'osservazione importante

- In un'istruzione ciclica, l'esecuzione potrebbe **non terminare mai**!

```
main()
{
  boo = 10;
  while (boo>0) {
        printf(boo);
  }
}
```

- Come "correggere" questo loop infinito?

Istruzioni composte

- L'istruzione condizionale e l'istruzione iterativa sono dette **istruzioni composte** perché esse sono costruite componendo istruzioni più semplici; contengono quindi altre istruzioni al proprio interno
  - caratteristica profondamente diversa dal linguaggio di von Neumann
  - molto utile per la costruzione di programmi complessi (vedremo in seguito)
  - un'istruzione composta può contenere al suo interno una qualsiasi altra istruzione, eventualmente essa stessa composta

Istruzioni composte

```
main()
{
  somma = 0;
  scanf(addendo);
  while (addendo>0) {
        somma = somma + addendo;
        scanf(addendo);
  }
  printf(somma);
}
```

A livello logico, agisce come una singola istruzione!

Esercizio

- Scrivere un programma che riceva come primo input un numero intero strettamente positivo N
- Il numero N ricevuto come primo input corrisponde al numero di interi strettamente positivi che verranno ricevuti successivamente dallo standard input
- Il programma deve stampare il maggiore di questi N interi strettamente positivi