Dati strutturati in linguaggio C: array

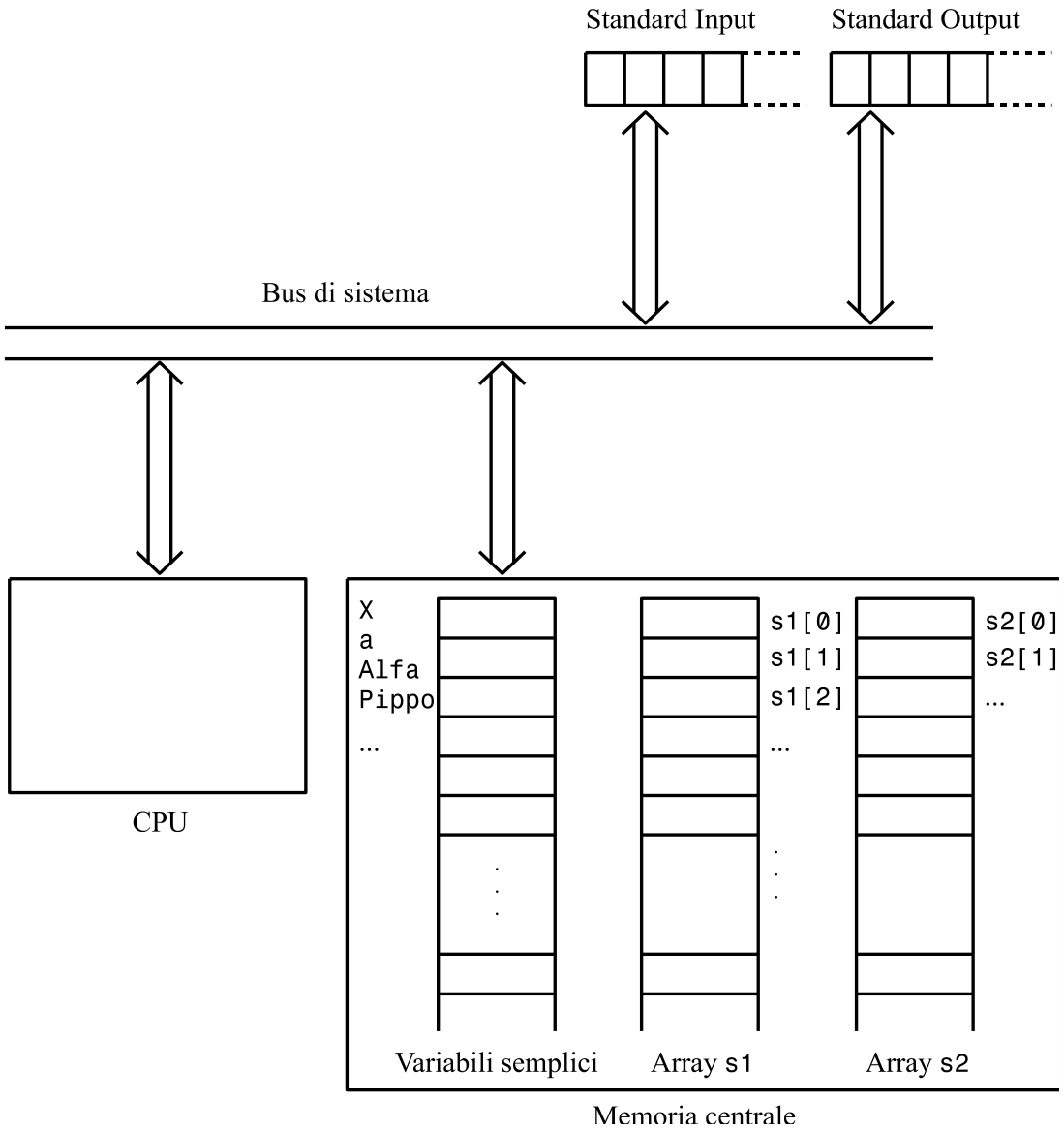
Problema

- Leggere una sequenza di dati in ingresso e riscriverli in ordine inverso in uscita
  - I dati sono conclusi dal carattere ‘%’
- Lo abbiamo già visto…
  - Il problema richiede la memorizzazione di tutti i dati **in celle di memoria diverse** prima di cominciare a scriverli
  - Non sappiamo a priori di quante celle avremo bisogno!

Soluzione?

- Nel linguaggio della macchina di von Neumann avevamo risolto il problema mediante l'**indirizzamento indiretto**
- Il linguaggio C introduce il concetto di **dato strutturato**, ossia informazione che a sua volta è composta di informazioni più elementari
- Il primo tipo di dato strutturato che vediamo è l'**array**: sequenza lineare di celle omogenee e consecutive che di per sé costituiscono **una sola variabile**

Macchina astratta C e array



Array

- Un array viene identificato come qualsiasi altra variabile
  - Però anche i suoi elementi sono variabili
- Ad essi si accede mediante un **indice**:
  - Ad esempio: scanf(s[2]);
- Nel linguaggio C, il primo elemento di un array è lo 0-esimo
  - Se un array ha 10 elementi il suo indice può assumere i valori interi tra 0 e 9
  - Per il momento non ci preoccupiamo del numero di elementi di un array
- Esempi:
  - a[3] = s[1] + x;
  - **if** (a[4] > s[1] + 3) s[2] = a[2] + a[1];
- … ma anche
  - x = a[i];
  - a[i] = a[i+1];
  - a[i*x] = s[a[j+1]–3]*(y – a[y]);

Torniamo al problema iniziale…

```
main()
{
        indice = 0;
        scanf(x);
        while (x != '%')
        {
                sequenza[indice] = x;
                indice = indice + 1;
                scanf(x);
        }
        while (indice > 0)
        {
                indice = indice - 1;
                printf(sequenza[indice]);
        }
}
```

Il loop continua a ripetersi fintanto che il carattere letto in ingresso è diverso da %.

La variabile `indice` è usata per scorrere l'array, così come si usava una cella di supporto che "scorreva" la memoria mediante indirizzamento indiretto nella macchina di von Neumann.

La variabile `indice` scorre l'array all'indietro per riscrivere la sequenza in ordine inverso; il loop continua ad eseguire fintanto che `indice` non è arrivato all'inizio dell'array.

Un problema meno banale

- Supponiamo di avere sullo Standard Input una serie di dati relativi a delle fatture
- Per ogni fattura una cella dello Standard Input ne contiene l'importo e le tre successive la data di emissione, nella forma "giorno" (un numero compreso tra 1 e 31), "mese", "anno"
- Il carattere '%' è posto dopo l'anno dell'ultima fattura
- Si vogliono stampare, nell'ordine, sullo Standard Output:
  - la dicitura: "IMPORTI FATTURE EMESSE";
  - la sequenza di tutti gli importi, nello stesso ordine di ingresso, preceduti dal carattere $;
  - la dicitura: "TOTALE FATTURE EMESSE";
  - il totale delle fatture stesse, precedute dal carattere $;
  - la dicitura: "DATE DI EMISSIONE";
  - la sequenza delle date di emissione, nello stesso ordine di ingresso
    - I tre elementi di ogni data devono essere separati da una /; alla fine di ogni data va scritto il carattere #

Conviene andare per gradi

- Dividiamo l'algoritmo in due **macro-passi**:
  1) raccolta degli input
  2) produzione degli output
- Un primo ciclo esegue la lettura dei vari dati, quattro per volta
  - infatti ogni fattura è descritta da quattro elementi
- Ognuno dei quattro dati letti viene memorizzato, rispettivamente, in una cella di **quattro diversi array**, denominati `fatture`, `giorno`, `mese` e `anno`
- L'indice di ognuno di questi array viene incrementato di 1 a ogni iterazione del ciclo
- Durante questo ciclo viene anche calcolato l'importo totale delle varie fatture

Primo macro-passo

```
main()
{
        contatore = 0; totale = 0;
        scanf(dato);
        while (dato != '%')
        {
                fatture[contatore] = dato;
                totale = totale + dato;
                scanf(dato); giorno[contatore] = dato;
                scanf(dato); mese[contatore] = dato;
                scanf(dato); anno[contatore] = dato;
                contatore = contatore + 1;
                scanf(dato);
        }
        …
}
```

Inizializziamo le variabili che andremo ad aggiornare durante le ripetizioni del ciclo.

Può leggere un nuovo importo oppure il carattere '%'

…proseguiamo per gradi

- Dopo aver raccolto tutti i dati ed aver calcolato l'importo totale…
- Viene stampata la dicitura "IMPORTI FATTURE EMESSE" mediante un'unica istruzione printf
- Un primo ciclo di scrittura stampa nell'ordine tutti gli elementi dell'array fatture intercalandoli con il simbolo $
- Viene stampata la dicitura "TOTALE FATTURE EMESSE" seguita da $ e dal valore del totale precedentemente calcolato
- Viene stampata la dicitura "DATE DI EMISSIONE"
- Un secondo ciclo di scrittura stampa le varie terne di valori "giorno, mese, anno", prelevandole nell'ordine appropriato dai corrispondenti array e introducendo
  - il carattere '/' tra giorno e mese e tra mese e anno,
  - il carattere # tra l'anno della data corrente e il giorno della terna successiva

Secondo macro-passo

```
        ...
        NumFatture = contatore;
        contatore = 0;
        printf("IMPORTI FATTURE EMESSE");
        while       (contatore < NumFatture)  {
                printf('$'); printf(fatture[contatore]); contatore = contatore + 1;
        }
        printf("TOTALE FATTURE EMESSE"); printf('$'); printf(totale);
        printf("DATE DI EMISSIONE"); contatore = 0;
        while (contatore < NumFatture) {
                printf(giorno[contatore]); printf('/');
                printf(mese[contatore]); printf('/');
                printf(anno[contatore]); printf('#');
                contatore = contatore + 1;
        }
}
```

Dovendo ri-utilizzare `contatore` per il ciclo, abbiamo bisogno di una nuova variabile che memorizzi il numero totale di fatture inserite.

Problemi sempre meno banali…

- Si vuole costruire un analizzatore di testo che sostituisca sistematicamente una parola con un'altra
- Si supponga che lo Standard Input della macchina abbia il contenuto seguente
  - In primo luogo è scritta (carattere per carattere) una parola (per "parola" intendiamo qui una sequenza di caratteri alfabetici)
  - Segue il carattere $; poi un'altra parola seguita dal carattere #
  - Successivamente vi è una sequenza di parole separate tra di loro da uno spazio e chiusa da un '%' (cioè, dopo l'ultima parola c'è uno spazio seguito dal terminatore finale '%')
- Il programma deve riscrivere su Standard Output il testo costituito dalla sequenza di parole dopo il #, sostituendo a ogni occorrenza della prima parola dello Standard Input la seconda
  - Se per caso la prima parola mancasse, il programma deve stampare un messaggio che avverte l'utente dell'errore e sospendere l'esecuzione
  - Al contrario, può essere mancante la seconda parola: in tal caso si ottiene l'effetto di cancellare ogni occorrenza della prima parola
  - E' ammesso il caso particolare che l'intero testo da riscrivere sia assente.

- Per la prima volta ci preoccupiamo della possibilità di dati errati o, più in generale ancora, di condizioni eccezionali

Procediamo per gradi

- Prima specifichiamo a grandi linee e informalmente **in italiano** l'algoritmo
  - Cioè, cerchiamo di ottenere un primo **pseudocodice**

- Verifica se prima del carattere $ esiste una parola
  - In caso negativo stampa il messaggio MANCA LA PAROLA DA SOSTITUIRE; in caso positivo procedi
- Memorizza la prima parola in un array
- Memorizza la seconda parola in un altro array
- A questo punto fai la scansione dell'intero testo parola per parola fino a trovare il carattere %
  - Memorizza ogni parola in un array
  - Confronta la parola letta con la prima parola: se le due parole coincidono, scrivi nello Standard Output la seconda parola, altrimenti scrivi la parola appena letta
- **Non tutti i termini utilizzati**, seppur chiari, (ad esempio: memorizza parola) corrispondono a operazioni elementari del C: occorre analizzarli separatamente come **sottoproblemi**
- Per non fare tutta la fatica in un colpo solo, codifichiamo "in parte" l'algoritmo

Dallo pseudocodice al C

```
main()
{
        scanf(carattere);
        if (carattere == '$') printf("MANCA LA PAROLA DA SOSTITUIRE");
        else  {
        [memorizza nell'array PrimaParola la sequenza di caratteri fino a '$'];
        [memorizza nell'array SecondaParola la sequenza di caratteri seguenti '$' fino a '#'];
                scanf(carattere);
                while (carattere != '%') {
                [memorizza nell'array ParolaCorrente la sequenza di caratteri fino al
                prossimo spazio];
                [confronta PrimaParola con ParolaCorrente];
                if      ([PrimaParola == ParolaCorrente])
                        printf(SecondaParola);
                else
                        printf(ParolaCorrente);
                printf(' ');
                scanf(carattere);
                }
        }
}
```

Leggo il primo carattere della prossima parola, oppure '%'.

Lo spazio serve per separare le parole sullo Standard Output.

Sottoproblemi

- Esaminiamo ora i sottoproblemi:
  1) memorizzazione di una parola in un array
  2) scrittura di una parola
  3) confronto tra due parole
- Ognuna di queste operazioni richiede a sua volta un "**sottoalgoritmo**" per la sua realizzazione
- Concentriamo l'attenzione sul problema meno banale: il confronto
- Decidere se due parole memorizzate rispettivamente nei due array `PrimaParola` e `ParolaCorrente` coincidono:
  - Se le lunghezze delle due parole sono diverse, allora le parole sono diverse
  - In caso contrario: fai la scansione dei due array carattere per carattere fino a quando o non trovi due caratteri diversi o l'indice dell'array ha superato la lunghezza di una delle due parole
  - Nel primo dei due casi precedenti concludi che le due parole sono diverse; nel secondo caso concludi che le due parole coincidono

Ora codifichiamo l'algoritmo separatamente ...

Supponiamo di avere già questa informazione: sarà un nuovo sottoproblema da risolvere separatamente!

```
if (LunghPrimaPar == LunghParCorr) {
        contatore = 0;
        while (contatore < LunghPrimaPar &&
                        PrimaParola[contatore] == ParolaCorrente[contatore]) {
                        contatore = contatore + 1;
        }
        if      (contatore >= LunghPrimaPar)
                printf("Le due parole coincidono");
        else
                printf("Le due parole sono diverse");
}
else printf("Le due parole sono diverse");
```

Mettiamo tutto insieme: lettura delle due parole iniziali

```
main()  {
        scanf(carattere);
        if (carattere == '$') printf ("MANCA LA PAROLA DA SOSTITUIRE");
        else  {
                contatore = 0;
                while (carattere != '$') {
                        PrimaParola[contatore] = carattere;
                        contatore = contatore + 1;
                        scanf(carattere);
                }
                LunghPrimaPar = contatore;

                scanf(carattere); contatore = 0;
                while (carattere != '#')
                {
                        SecondaParola[contatore] = carattere;
                        contatore = contatore + 1;
                        scanf(carattere);
                }
                LungSecPar = contatore;
```

Memorizzazione della prima parola carattere per carattere.

Memorizzazione della seconda parola carattere per carattere.

Scansione del testo seguente…

```
scanf(carattere);
while (carattere != '%') {
    contatore = 0;
    while  (carattere != ' ' ) {
        ParolaCorrente[contatore] = carattere;
        contatore = contatore + 1;
        scanf(carattere);
    }
    LungParCorr = contatore;

    if (LunghPrimaPar == LunghParCorr) {
        contatore = 0;
        while (contatore < LunghPrimaPar &&
                PrimaParola[contatore] ==   ParolaCorrente[contatore])
            contatore = contatore + 1;
        if (contatore >= LunghPrimaPar) {
            contatore = 0;
            while (contatore < LungSecPar)  {
                printf(SecondaParola[contatore]);
                contatore = contatore + 1;
            }
        } else ....   {
```

Memorizzazione della parola corrente carattere per carattere.

Confronto della parola corrente con la prima, visto prima come "sottoalgoritumo"

La parola corrente è uguale alla prima, copiamo sullo Standard Output la seconda parola.

…e termine

```
                contatore = 0;
                while (contatore < LungParCorr) {
                    printf(ParolaCorrente[contatore]);
                    contatore = contatore + 1;
                }
            }
        } else  {
            contatore = 0;
            while (contatore < LungParCorr) {
                printf(ParolaCorrente[contatore]);
                contatore = contatore + 1;
            }
        }
        printf(' ');
        scanf(carattere);
        }
    }
}
```

…altrimenti, ricopiamo la parola corrente (arriviamo qui se le parole sono diverse ma della stessa lunghezza)!

…ricopiamo di nuovo la parola corrente (ma arriviamo qui se le parole sono di diversa lunghezza)!

…inseriamo uno spazio per separare le parole ed iniziamo la lettura della parola successiva (o del carattere ‘%’)