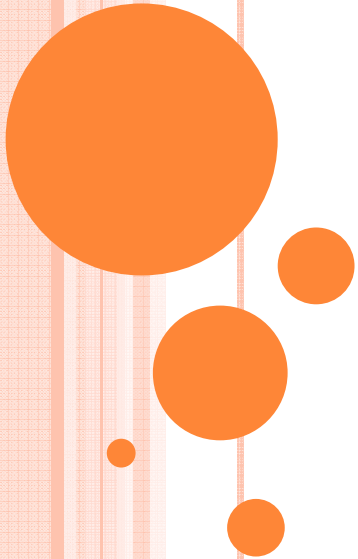
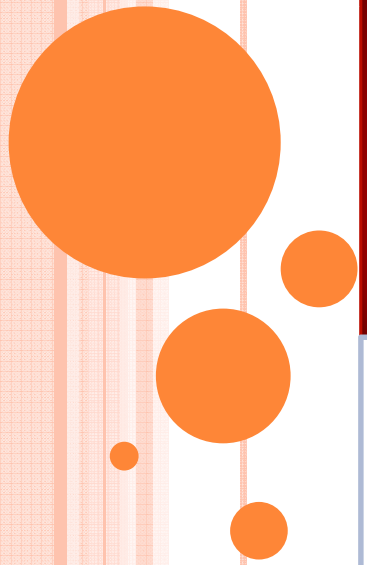


Esercitazione N.2
Linguaggio C

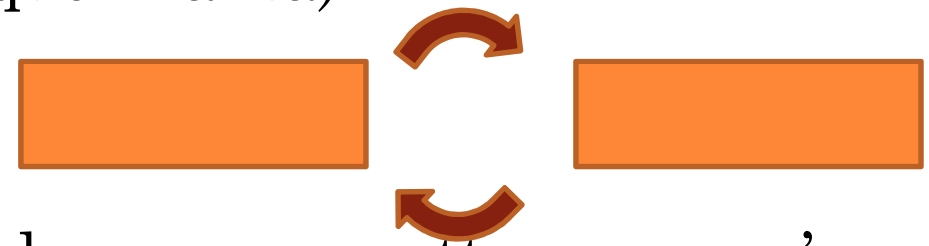
In preparazione alla prova in Itinere



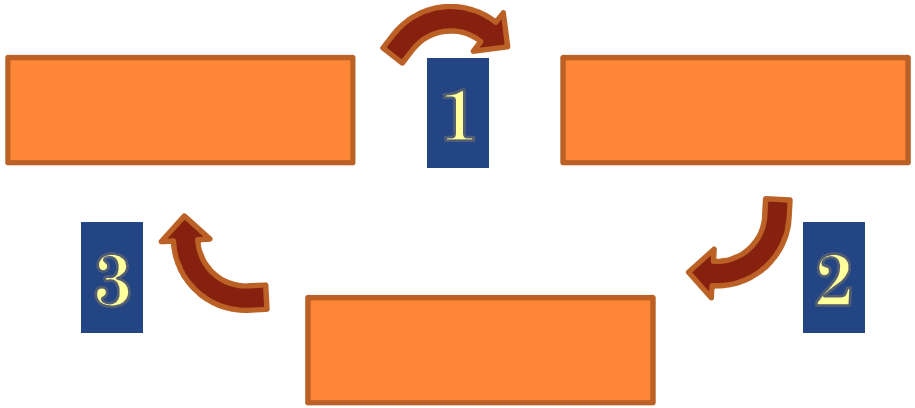
Ordinare un array

Invertire il valore di due variabili

- Le due operazioni di assegnamento non possono essere contemporanee (deve esserci sequenzialità)



- Si deve passare attraverso un'area temporanea eseguendo tre operazioni in sequenza



INVERSIONE DI DUE NUMERI

ordina1allaVolta.c | primomaggioreINTEGRO.c | **invers.c**

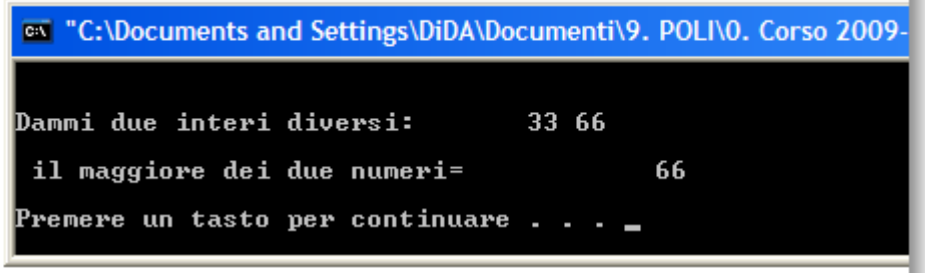
```
1  /* Inverte i valori di due variabili
2  */
3  #include <stdio.h>
4  int main(int argc, char *argv[])
5  {int I,J, temp;
6
7  printf("\n\nDammi due interi diversi:\t");
8  scanf("%d%d",&I,&J);
9        temp = J;
10       J = I;
11       I = temp;
12 printf("\n i due numeri scambiati\t\t%d\t%d\n\n",I,J);
13
14       return 0;
15 }
16
17
```

```
"C:\Documents and Settings\DiDA\Documenti\9. POLI\0. Cor

Dammi due interi diversi:       33 88
 i due numeri scambiati         88      33
Premere un tasto per continuare . . .
```

TROVARE IL MAGGIORE DUE NUMERI

ordina1allaVolta.c | primomaggioreINTEGRO.c | invers.c | **maggiore.c**

```c
/* stampa il maggiore di due numeri
*/
#include <stdio.h>
int main(int argc, char *argv[])
{int I,J, temp,maggiore;

printf("\n\nDammi due interi diversi:\t");
scanf("%d%d",&I,&J);
if(I>J)
        {maggiore=I;
        }
    else
        {maggiore=J;
        }
    printf("\n il maggiore dei due numeri=\t\t%d\t\n\n",maggiore);
    return 0;
}
```



TROVARE IL MAGGIORE DI UNA SERIE

- Il programma si divide in tre parti
  1. Input della serie di NElem elementi
  2. Calcolo del massimo
  3. Output del risultato

- Una variante è quella di mettere il massimo nella prima posizione della tabella … quindi:

  1. Input della serie di NElem elementi
  2. Calcolo del massimo
  3. Output della serie di NElem elementi

TROVARE IL MASSIMO DI UNA SERIE

ordina1allaVolta.c | primomaggioreINTEGRO.c | invers.c | maggiore.c | **maxDIserie.c**

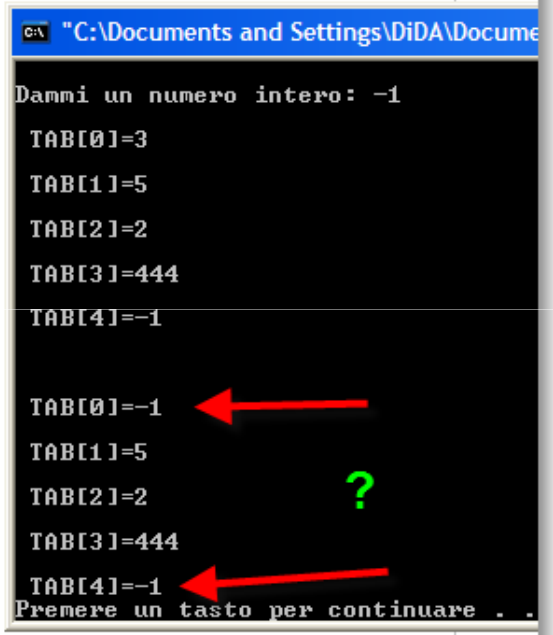
```c
/* stampa il maggiore di una serie di numeri
*/
#include <stdio.h>
#define NElem 5

int main(int argc, char *argv[])
{int I,TAB[NElem], maggiore;

for(I=0;I<NElem ;I++ )
    {printf("\n\nDammi un numero intero:\t");
     scanf("%d",&TAB[I]);
    }
/*si suppone che il maggiore sia il primo */
maggiore=TAB[0];
for(I=1;I<NElem  ;I++ )
    {if(maggiore<TAB[I])
            {maggiore=TAB[I];
            };
    }
printf("\n\nil maggiore della tabella = %d\n",maggiore);
    return 0;
}
```

```
"C:\Documents and Settings\DiDA\Document

Dammi un numero intero: 44

Dammi un numero intero: 5

Dammi un numero intero: 222

Dammi un numero intero: -1

Dammi un numero intero: 3

il maggiore della tabella = 222
Premere un tasto per continuare . . .
```

TROVARE IL MINIMO DI UNA SERIE E METTERLO NELLA PRIMA POSIZIONE

ordina1allaVolta.c | primomaggioreINTEGRO.c | invers.c | maggiore.c | maxDIserie.c | **primominore.c**

```
1  /* stampa il minore di una serie di numeri */
2  #include <stdio.h>
3  #define NElem 5
4  int main(int argc, char *argv[])
5  {int I,TAB[NElem];
6  for(I=0;I<NElem ;I++ )
7      {printf("\n\nDammi un numero intero:\t");
8       scanf("%d",&TAB[I]);
9  };
10 for(I=0;I<NElem ;I++ )
11 {printf("\n TAB[%d]=%d\n",I,TAB[I]);
12 };
13 /*si suppone che il minore sia il primo ovvero TAB[0]*/
14 for(I=1;I<NElem  ;I++ )
15     {if(TAB[0]>TAB[I])
16             TAB[0]=TAB[I];
17     };
18 printf("\n\n");
19 for(I=0;I<NElem ;I++ )
20 {printf("\n TAB[%d]=%d\n",I,TAB[I]);
21 };
22     return 0;
```



TROVARE IL MINIMO DI UNA SERIE E METTERLO NELLA PRIMA POSIZIONE

SENZA DISTRUGGERE ELEMENTI

ordina1allaVolta.c | primominoreINTEGRO.c | invers.c | maggiore.c | maxDIserie.c | primominore.c

```c
/* stampa il minore di una serie di numeri*/
#include <stdio.h>
#define NElem 5
int main(int argc, char *argv[])
{int I,TAB[NElem],temp;
for(I=0;I<NElem ;I++ )
    {printf("\n\nDammi un numero intero:\t");
     scanf("%d",&TAB[I]);
};
for(I=0;I<NElem ;I++ )
{printf("\n TAB[%d]=%d\n",I,TAB[I]);
};
/*si suppone che il minore sia il primo ovvero TAB[0]*/
for(I=1;I<NElem  ;I++ )
    {if(TAB[0]>TAB[I])
    /* per preservare i valori si deve fare uno scambio */
        {temp=TAB[0];
         TAB[0]=TAB[I];
         TAB[I]=temp;
        };
    };
printf("\n\n");
for(I=0;I<NElem ;I++ )
{printf("\n TAB[%d]=%d\n",I,TAB[I]);
};
return 0;
}
```

```
"C:\Documents and Settings\DiDA\Doc

Dammi un numero intero: 3

Dammi un numero intero: 6

Dammi un numero intero: 3

Dammi un numero intero: 7

Dammi un numero intero: -5
 TAB[0]=3
 TAB[1]=6
 TAB[2]=3
 TAB[3]=7
 TAB[4]=-5


 TAB[0]=-5
 TAB[1]=6
 TAB[2]=3
 TAB[3]=7
 TAB[4]=3
Premere un tasto per continuare
```

Ordinamento della tabella

- Ma se allora siamo capaci di mettere nella prima posizione il minimo
- ------- allora saremo capaci di mettere nella seconda posizione il minimo dei restanti
- -------- e nella terza …. Ecc ecc

- L'ordinamento è composto di Nelem-1 operazioni di "minimo"
- Quindi le operazioni di minimo saranno ripetute più volte

ORDINAMENTO DELLA TABELLA

ordina1allaVolta.c | primominoreINTEGRO.c | invers.c | maggiore.c | maxDIserie.c | primominore.c

```
1 /* Ordina un elemento alla volta */
2 #include <stdio.h>
3 #define NElem 5
4 int main(int argc, char *argv[])
5 {int I,TAB[NElem],temp,Imin;
6     for(I=0;I<NElem ;I++ )
7         {printf("\n\nDammi un numero intero:\t");
8          scanf("%d",&TAB[I]);
9     };
10    for(I=0;I<NElem ;I++ )
11    {printf("\n TAB[%d]=%d\n",I,TAB[I]);
12    };
13 /*si mette in TAB[0] il minore
14 quindi si ripete per TAB[1} e così via */
15 for(Imin=0;Imin<NElem-1;Imin++)
16 {
17    for(I=Imin;I<NElem   ;I++ )
18        {if(TAB[Imin]>TAB[I])
19            {   temp=TAB[Imin];
20                TAB[Imin]=TAB[I];
21                TAB[I]=temp;
22            };
23        };
24 };
25 printf("\n\n");
26 for(I=0;I<NElem ;I++ )
27 {printf("\n TAB[%d]=%d\n",I,TAB[I]);
```

```
"C:\Documents and Settings\DiDA\Docume

Dammi un numero intero: 33

Dammi un numero intero: 22

Dammi un numero intero: -123

Dammi un numero intero: 432

Dammi un numero intero: 0
 TAB[0]=33
 TAB[1]=22
 TAB[2]=-123
 TAB[3]=432
 TAB[4]=0


 TAB[0]=-123
 TAB[1]=0
 TAB[2]=22
 TAB[3]=33
 TAB[4]=432
Premere un tasto per continuare . .
```

UN ALGORITMO PIÙ "FURBO"

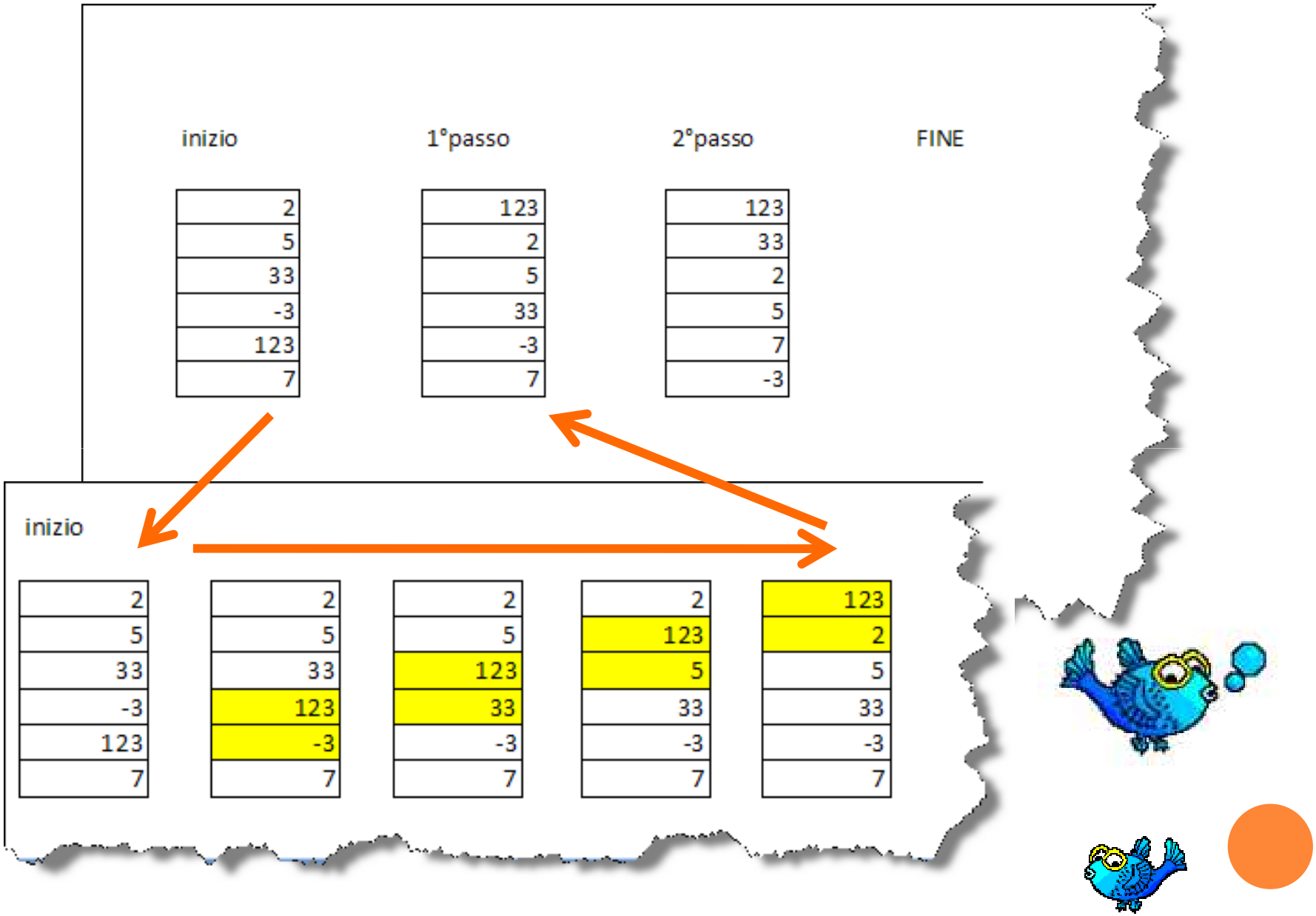
- L'algoritmo scritto è corretto ma ha un difetto
- Nel caso in cui la tabella sia già ordinata, anche parzialmente, esegue sempre un numero uguale di operazioni
- Spesso la tabella da ordinare è parzialmente già ordinata:
  - es. si tratta di aggiungere elementi nuovi ad una tabella già ordinata
- Si possono pensare algoritmi diversi che tengono conto di parziali ordinamenti

BUBBLE SORT

- Si confrontano i primi due numeri
  - Se sono ordinati si confrontano il 2° e il 3à
  - Se non sono ordinati si invertono
- L'operazione viene eseguita dal primo al penultimo numero

- I numeri bassi posti all'inizio si spostano sempre verso la fine
- I numeri alti posti alla fine si spostano sempre verso l'inizio
- COME LE BOLLE NELL'ACQUA!!!

BUBBLE SORT


inizio
1°passo
2°passo
FINE
2
5
33
-3
123
7
123
2
5
33
-3
7
123
33
2
5
7
-3
inizio
2
5
33
-3
123
7
2
5
33
123
-3
7
2
5
123
33
-3
7
2
123
5
33
-3
7
123
2
5
33
-3
7

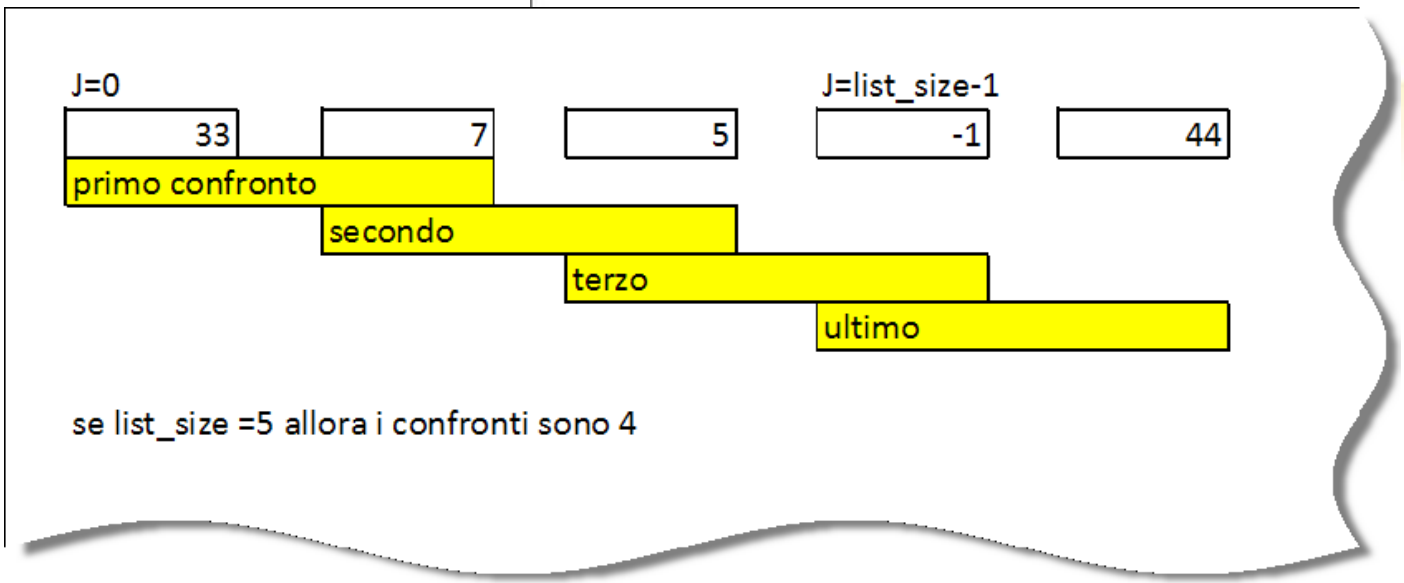
Si organizza il Bubble Sort in una funzione

sort.c*

```
1  /* BUBBLE SORT */
2
3
4  #include <stdio.h>
5
6  int main(int argc, char *argv[])
7
8  {void bubble_sort( int [], int  );  /* prototipo  */
9
10   static int list[] = { 13, 56, 23, 1, 89, 58, 20, 125, 86, 3 };
11
12           bubble_sort( list, sizeof(list)/sizeof(list[0]));
13
14   return ( 0 );
15 }
```

IL LOOP DI SCAMBIO

```
1 #define FALSE 0
2 #define TRUE 1
3 int j, k, temp,;
4
5
6 for (j = 0; j < list_size - 1; j++) {
7     if (list[j] > list[j+1])
8             {
9             temp = list[j];
10            list[j] = list[j+1];
11            list[j+1] = temp;
12            }
```



Si deve predisporre la fine dell'algoritmo

- Gli scambi devono essere fatti sino a quando non ce n'è più bisogno
- Cioè sino a quando la tabella non sia già completamente ordinata

- Si deve poter sapere se c'è stato almeno uno scambio durante un ciclo

- Si usa una variabile SORTED che può avere valori VERO e FALSO

CICLO ESTERNO

```
16 #define FALSE 0
17 #define TRUE 1
18 /* funzione di ordinamento  */
19 void bubble_sort( int list[], int list_size )
20 {
21 int j, k, temp, sorted = FALSE;
22 while( !sorted )
23 {
24   sorted = TRUE; /* assume list is sorted */
25
26
27
28
29
30   for (j = 0; j < list_size - 1; j++) {
31     if (list[j] > list[j+1])
32       {
33     /* At least 1 element is out of order */
34       sorted = FALSE;
35       temp = list[j];
36       list[j] = list[j+1];
37       list[j+1] = temp;
38       }
39     } /* end of for loop */
40   } /* end of while loop */
41 }
```

La funzione completa

```
18 void bubble_sort( int list[], int list_size )
19 {
20 int j, k, temp, sorted = FALSE;
21 while( !sorted )
22 {
23   sorted = TRUE; /* assume list is sorted */
24   /* Print loop: not part of bubble sort algorithm */
25   for (k = 0; k < list_size; k++) {
26     printf ("%d\t", list[k]);
27     } ;
28     printf ("\n");/* End of print loop */
29
30   for (j = 0; j < list_size - 1; j++) {
31     if (list[j] > list[j+1])
32       {
33     /* At least 1 element is out of order */
34       sorted = FALSE;
35       temp = list[j];
36       list[j] = list[j+1];
37       list[j+1] = temp;
38       }
39     } /* end of for loop */
40   } /* end of while loop */
```

stampa tabella

In esecuzione

sort.c | Untitled2.c* | Untitled3.c*

```
16
17 /* funzione di ordinamento  */
18 void bubble_sort( int list[], int list_size )
19 {
20 int j, k, temp, sorted = FALSE;
21 while( !sorted )
22 {
23   sorted = TRUE; /* assume list is sorted */
24   /* Print loop: not part of bubble sort algorithm */
25   for (k = 0; k < list_size; k++) {
26     printf ("%d\t", list[k]);
27     } ;
28     printf ("\n");/* End of print loop */
29
30   for (j = 0; j < list_size - 1; j++) {
31     if (list[j] > list[j+1])
32       {
33     /* At least 1 element is out of order */
34       sorted = FALSE;
35       temp = list[j];
36       list[j] = list[j+1];
37       list[j+1] = temp;
38       }
39     } /* end of for loop */
40   } /* end of while loop */
41 }
```

"C:\Documents and Settings\DiDA\Documenti\9. POLI\0. Corso 2009-10 MN\samples\8-Eserc...

```
13      56      23      1       89      58      20      125     86      3
13      23      1       56      58      20      89      86      3       125
13      1       23      56      20      58      86      3       89      125
1       13      23      20      56      58      3       86      89      125
1       13      20      23      56      3       58      86      89      125
1       13      20      23      3       56      58      86      89      125
1       13      20      3       23      56      58      86      89      125
1       13      3       20      23      56      58      86      89      125
1       3       13      20      23      56      58      86      89      125
Premere un tasto per continuare . . .
```

Convertire stringhe in alfabeto Morse e viceversa

Es.: Alfabeto Morse: numeri

- **1:** · – – – –
  2: · · – – –
  3: · · · – –
  4: · · · · –
  5: · · · · ·
  6: – · · · ·
  7: – – · · ·
  8: – – – · ·
  9: – – – – ·
  0: – – – – –

Morse code program.
Enter a number (1 digit) and find out what it is in Morse code

```
#include <stdio.h>
#include <string.h>
/************************************
   */
main ()
{ short digit;  char string[10];


printf ("Enter any digit in the range
   0..9\t");
scanf ("%d",&digit);
if ((digit < 0) || (digit > 9))
   {
   printf ("\nNumber was not in range
   0..9");
   return (0);
   }
printf ("\nThe Morse code of that
   digit is \t");
ToMorse (digit);
 }
```

```
/************************************
   */
ToMorse ( digit)        /* print out
   Morse code */
{
  static char code[][6]=
     {
     "-----",
     ".----",
     "..---",
     "...--",
     "....-",
     ".....",
     "-....",
     "--...",
     "---..",
     "----.",
     };
printf ("%s\n",code[digit]);
};
```

Dato il codice Morse trovare la lettera rappresentata

- Per trovare la corrispondenza tra un codice Morse e la lettera (numero) rappresentata si deve fare una ricerca
- Si deve scandire la tabella e i risultati possono essere
  - Trovato il carattere in una certa posizione
  - Non trovato il carattere in nessuna posizione

SERVE L'ALGORITMO DI RICERCA (LOOKUP)

- Data una tabella con dei valori, determinare se un valore X è nella tabella e dove

- Si scandisce tutta la tabella sino a quando si determina che X == TAB[i]
- Nel caso affermativo si può rispondere affermativamente e l'indice "i" è la posizione dell'elemento.
- Nel caso negativo si può rispondere negativamente e l'indice non è valido (es. -1)

Confronta ogni posizione
Fa "break" quando trova l'eguaglianza

```c
#define FALSE 0
#define TRUE 1
#include <stdio.h>

int main(int argc, char *argv[])
{int  X, I,k,presente, list_size;
  static int list[] = { 13, 56, 23, 1, 89, 58, 20, 125, 86, 3 };
    list_size=sizeof(list)/sizeof(list[0]);
  for (k = 0; k < list_size; k++) {
        printf ("%d\t", list[k]);
        } ;
    printf ("\n");/* End of print loop */

 printf("\n\nDammi un numero compreso tra i precedenti:\t");
 scanf("%d",&X);

 presente=FALSE;
 for(I=0;I<list_size ;I++ )
 {
 if(X==list[I] )
    {presente=TRUE;
    break; }
 };
  if(presente==TRUE) printf("\n Il numero %d si trova in posizione %d\n",X,I);
    else printf("\n %d Non si trova nella tabella\n",X);
  return ( 0 );
```

```
"C:\Programmi\C-Free Standard\temp\Untitled4.exe"
13      56      23      1       89      58      20      125     86      3


Dammi un numero compreso tra i precedenti:      89

 Il numero 89 si trova in posizione 4
Premere un tasto per continuare . . .
```

Morse code program.
Enter a Morse code and find out the number

```
#include <stdio.h>
#include <string.h>
/************************************
   */
main ()
{ short digit;  char string[10];


printf ("Adesso dammi una stringa
   Morse\t");


scanf("%s",string);
printf("\n\t%s",string);


FromMorse(string);
 }
```

```
FromMorse ( char string[])
{ int I;
static char letter[]=
  { '0‘, '1‘, '2‘, '3‘,'4‘,'5',
     '6‘,'7‘, '8‘, '9‘, };
static char code[][6]=
   { "-----“, ".----“, "..---",
     "...--“, "....-“, ".....",
     "-....“, "-...-“, "---..",
     "----.“       };


for(I=0;I<10 ;I++ )
{
if (strcmp (string, code[I])==0 )
   break;
};


printf ("il carattere morse %s
   corrisponde a %d\n",  code[I] ,I
   );
}
```

Modificare il programa per riconoscere una frase

- Estendere le tabelle con le lettere dell'alfabeto
- Scrivere prima il ToMorse esteso
  - Input carattere
  - Output Morse
- Poi modificare il FromMorse
  - Leggere una sequenza di stringhe (ognuna è un codice Morse)
  - Ogni volta cercare e stampare il carattere relativo
  - Eseguire le operazioni in loop sino a che si inserisce un codice errato

Alfabeto Morse: lettere

- **A:** · –
  B: – · · ·
  C: – · – ·
  D: – · ·
  E: ·
  F: · · – ·
  G: – – ·
  H: · · · ·
  I: · ·
  J: · – – –
  K: – · –
  L: · – · ·
  M: – –
- **N:** – ·
  O: – – –
  P: · – – ·
  Q: – – · –
  R: · – ·
  S: · · ·
  T: –
  U: · · –
  V: · · · –
  W: · – –
  X: – · · –
  Y: – · – –
  Z: – – · ·

Codice Morse Completo

Lettere, numeri e punteggiatura

Lettere	Codice	Lettere	Codice	Numeri	Codice	Punteg.	Codice
A	•—	N	—•	0	—————	•	•—•—•—
B	—•••	O	———	1	•————	,	——••——
C	—•—•	P	•——•	2	••———	:	———•••
D	—••	Q	——•—	3	•••——	?	••——••
E	•	R	•—•	4	••••—	=	—•••—
F	••—•	S	•••	5	•••••	-	—••••—
G	——•	T	—	6	—••••	(	—•——•
H	••••	U	••—	7	——•••	)	—•——•—
I	••	V	•••—	8	———••	"	•—••—•
J	•———	W	•——	9	————•	'	•—————
K	—•—	X	—••—			/	—••—•
L	•—••	Y	—•——			Sottolineato	•——•—
M	——	Z	——••			@	•——•—•

LE TABELLE DI CONVERSIONE SONO DUE

```
1    /* Morse code program.     */
2 #include <stdio.h>
3 #include <string.h>
4 #define ERRCODE -1
5 /*************************************************/
6  static char codeAlfaNum[][6]=
7       {
8       "-----",
9       ".----",
10      "..---",
11      "...--",
12      "....-",
13      ".....",
14      "-....",
15      "--...",
16      "---..",
17      "----.",
18      ".-",
19      "-..."
20     ,"-.-."
21     ,"-.."
22     ,"."
```

la tabella dei codici morse deve avere lo stesso ordine della tabella dei caratteri

Se si devono inserire dei nuovi simboli
Bisogna farlo in entrambe le tabelle
Nello stesso ordine

```
    ,"--.."
     };
    static char alfanum[]=
 {
    '0',
    '1',
    '2',
    '3',
    '4',
    '5',
    '6',
    '7',
    '8',
    '9',
    'A',
    'B',
    'C',
    'D',
    'E',
    'F',
    'G',
    'H',
```

OPERAZIONI PRELIMINARI

morse-codice-alfanum-bilaterale-3.c

```
84 int numElem;
85
86 main ()
87 { char digit;  char string[6]; int indice;
88
89 int ToMorse(char);
90 int FromMorse(char[]);
91
92 numElem = sizeof(alfanum);
93
94 /*  stampa di tutto il codice  */
95 {  int I;
96 printf("\n numero elementi =  %d  e stringa = %s\n\n\n",numElem,st
97     for(I=0; I<numElem; I++)
98      {printf("\n %c     %s  ",alfanum[I],codeAlfaNum[I]);
99      } ;
100};
101/****************acquisisci un alfanumerico***********************/
102
103printf ("\n\n\nEnter any letter in the range 0..9 A..Z\t\t\t");
104 anf ("%c",&digit);
```

paradigmi delle funzioni

calcolo delle dimensioni

```
numero elementi =  36

0     -----
1     .----
2     ..---
3     ...--
4     ....-
5     .....
6     -....
7     --...
8     ---..
9     ----.
A     .-
B     -...
C     -.-.
D     -..
E     .
F     ..-.
G     --.
H     ....
I     ..
      .---
```

Data una lettera o numero si converte in Morse (conversione a maiuscolo!!)

```
101 /*****************acquisisci un alfanumerico************************/
102
103 printf ("\n\n\nEnter any letter in the range 0..9 A..Z\t\t\t");
104 scanf ("%c",&digit);
105
106 digit=toupper(digit); /* ATTENZIONE converte in maiuscolo */
107
108 indice = ToMorse(digit);
109  if(indice<0)   {
110    printf ("\nERROR   Letter was not in correct range \n\n");
111    return (ERRCODE);
112    }
113
114 printf("\nil carattere  %c corrisponde al morse %s\n",digit,codeAlfaNum[indice]);
```

Data una stringa Morse (corrispondente ad un carattere) si esegue la conversione

```
115
116 /*****************acquisisci un codice Morse************************/
117 printf ("\nAdesso dammi una stringa Morse\t");
118 scanf("%s",string);
119
120 indice=FromMorse(string);
121  if(indice<0)  {
122    printf ("\nERROR   Il codice Morse non era corretto \n\n");
123    return (ERRCODE);
124    }
125 printf ("\nil simbolo morse %s corrisponde a %c\n\n\n\n",string,alfanum[indice]);
126 ;
127 return(0);
128 }
129
```

Le funzioni che effettuano la conversione - ToMorse

```
130
131 /*************************************************/
132
133 int ToMorse ( char digit)          /* calcola codice Morse */
134                                  /* se non lo trova ritorna -1 */
135 {int I;
136
137 for(I=0; I<numElem;I++ )
138     {if (digit==alfanum[I]) return(I);
139     }
140 return (-1);
141
142 };
143 /* ********************************************** */
```

Le funzioni che effettuano la conversione - FromMorse

```
143 /* ********************************************** */
144
145 int FromMorse ( char string[])          /* print out Morse code */
146 {int I;
147
148     for(I=0; I<numElem;I++ )
149         {if (strcmp(string,codeAlfaNum[I])==0) return(I);
150         }
151 return (-1);
152 }
```

Lo stesso esempio può essere eseguito ciclicamente con alcune modifiche

```
03 printf ("\n\n\nEnter any letter in the range 0..9 A..Z\t\t\t");
04 scanf ("\n%c",&digit);
05 digit=toupper(digit); /* converte in maiuscolo  */
06 printf("\n%d ",digit);
07
08 /*il ciclo termina quando si immette un carattere @ */
09 while (digit!='@')
10 {
11     indice = ToMorse(digit);
12      if(indice<0)  {
13        printf ("\nERROR   Letter %c was not in correct range \n",digit);
14        return (ERRCODE);
15  ciclo  }
16
17     printf("\nil carattere  %c corrisponde al morse %s\n",digit,codeAlfaNum[indice]);
18     scanf ("\n%c",&digit);
19     digit=toupper(digit);        /* converte in maiuscolo  */
20     printf("\n%d ",digit);
21 }
```

lettura del dato

Lo schema precedente è importante

- Tutte le volte in cui si deve scandire una tabella (array o stringa o altro) di N elementi in cui N sia ignoto si devono effettuare
  - N+1 letture del dato da elaborare
  - N elaborazioni
- Quindi apparentemente c'è un controsenso poiché il ciclo deve essere eseguito o "N" o "N+1" volte

- La lettura deve essere eseguita anche quando non c'è più nulla da fare, quindi N+1