

Approfondimenti sul Linguaggio C

Punti da analizzare:

1. Prime Inizializzazioni
2. Operazioni dirette sui bit
3. Accesso agli indirizzi delle variabili
4. Allocazione dinamica

1. Prime Inizializzazioni

Esempio : La somma di 3 numeri

```
#include <stdio.h>
main()
{ /* inizio */
  float a,b,c,d;
  /* Inizializzazione di a, b, c */
  scanf("%f %f %f",&a,&b,&c);
  a=5.; b=7.;
  /* Calcolo della somma tra a, b e c */
  d=a+b+c;
  printf("SOMMA= %f \n",d); /*
  Stampa del risultato*/
} /* fine */
```

#include<stdio.h>

Utilizzo degli header file

Prima dell'istruzione **main()** e' presente una direttiva del tipo:

#include <"file.h">

file.h contiene:

- specifiche di funzioni;
- definizioni di strutture di dati;
- definizioni di parametri;
- ...

Per eseguire Operazioni di I/O ⇒ #include<stdio.h>

Per utilizzare Funzioni Matematiche ⇒ #include<math.h>

Gestione Stringhe ⇒ #include<string.h>

...

Esempi di funzioni definite nel file stdio.h

- `int scanf(const char *format ,)` *Lettura da tastiera*
- `int fscanf(FILE *stream, const char *format, ...)` *Lettura da file*
- `int printf(const char *format ,)` *Stampa a video*
- `int fprintf(FILE *stream, const char *format, ...)` *Stampa su file*
- `FILE *fopen (cont char *filename, cost char *mode)` *Apertura file*
- `FILE *fclose (FILE * stream)` *Chiusura file*
- ...

Esempi di funzioni definite nel file math.h

Gli argomenti x ed y sono di tipo *double*, n è di tipo *int* e tutte le funzioni restituiscono un *double*.

- `double sin(x)` seno di x ;
- `double cos(x)` coseno di x ;
- `double tan(x)` tangente di x ;
- `double exp(x)` funzione esponenziale e^x ;
- `double log(x)` logaritmo naturale $\ln(x)$, con $x > 0$;
- `double log10(x)` logaritmo in base 10 $\log(x)$, con $x > 0$;
- `double pow(x,y)` potenza x^y ;
- `double sqrt(x)` radice quadrata di x , con $x > 0$;
- ...

1. Prime Inizializzazioni

Esempio : La somma di 3 numeri

```
#include <stdio.h>
main()
{ /* inizio */
    float a,b,c,d;
    /* Inizializzazione di a, b, c */
    scanf("%f %f %f",&a,&b,&c);
    a=5.; b=7.;
    /* Calcolo della somma tra a, b e c */
    d=a+b+c;
    printf("SOMMA= %f \n",d); /*
    Stampa del risultato*/
} /* fine */
```

- L'inizio di un programma comincia dalla **prima parentesi {**
- La fine di un programma è determinata dall'**ultima parentesi }**

```
main( )
{ /* inizio */
    .....
} /* fine */
```

- Non ci sono limitazioni sulle modalità di scrittura del codice (linguaggio a formato libero);
- Possibilità di scrivere più istruzioni sulla stessa linea;
- C'è differenza tra lettere minuscole e maiuscole.

```
/* Inizializzazione di a, b, c */
```

La stringa di caratteri di un commento è racchiusa tra

```
/* ..... */
```

un commento può occupare anche solo **parte di una linea**.

```
float a,b,c,d;
```

Si utilizza SOLO DICHIARAZIONE ESPLICITA

```
printf("SOMMA= %f \n",d);
```

Le operazioni di I/O sono eseguite da funzioni di libreria

```
scanf( ..... );
```

```
printf( ..... );
```

A. Murli - Calcolo Parallelo e Distribuito

9

Linguaggio C

Punti da analizzare:

1. Prime Inizializzazioni

2. Operazioni dirette sui bit

3. Accesso agli indirizzi delle variabili

4. Allocazione dinamica

A. Murli - Calcolo Parallelo e Distribuito

10

2. Operazioni dirette sui bit

Problema

Verificare se un numero naturale
è PARI o DISPARI

```
#include <stdio.h>
main()
{
    int n,resto;
    printf("inserisci numero\ n");
    scanf("%d",&n);
    resto= n & 1;
    if(resto==0) {
        printf("numero pari\n");
    }
    else {
        printf("numero dispari\n");
    }
}
```

1 confronto bit a bit.

11

Regole di composizione dell'operatore &

&	0	1
0	0	0
1	0	1

ESEMPIO 1 (a & 1)

```
a=6510=10000012 ⇒ a & 1 = 10000012 &
110=00000012          00000012 =
                        00000012
```

a & 1 = 1

ESEMPIO 2 (a & b)

```
a=4010=1010002 ⇒ a & b = 1010002 &
b=3410=0110002          0110002 =
                        0010002
```

a & b = 001000₂ = 8₁₀

A. Murli - Calcolo Parallelo e Distribuito

12

In particolare ...

Nell'esempio, $n \& 1$ equivale al resto della divisione di N per 2

Il resto della divisione per 2 è il bit meno significativo

0 se il numero è pari

1 se è dispari

Es. $n=12_{10}=1100_2 \Rightarrow r = n \& 1 = 1100_2 \&$

$0001_2 =$

0000_2

Es. $n=13_{10}=1101_2 \Rightarrow r = n \& 1 = 1101_2 \&$

$0001_2 =$

0001_2

$r = n \& 1$; $\& \Rightarrow$ AND bit a bit

A. Murli - Calcolo Parallelo e Distribuito

13

ALTRE OPERAZIONI bit a bit

Divisione di un numero intero N per 2

dividere per due significa spostare di un posto verso destra la rappresentazione binaria

Es. $n=13_{10}=1101_2 \Rightarrow n/2=6_{10}=110_2$

$n >> 1$; $>> \Rightarrow$ SHIFT verso destra

Moltiplicazione di un numero intero N per 2

Moltiplicare per 2 significa spostare di 1 posto verso sinistra la rappresentazione binaria

Es. $n=13_{10}=1101_2 \Rightarrow n * 2 = 1101_2 * 0010_2 = 11010_2 = 26_{10}$

$n << 1$; $<< \Rightarrow$ SHIFT verso sinistra

A. Murli - Calcolo Parallelo e Distribuito

14

Operatori sui bit in C

$\&$	\rightarrow	and
$>>$	\rightarrow	shift a destra
$<<$	\rightarrow	shift a sinistra
$ $	\rightarrow	or inclusivo
\wedge	\rightarrow	or esclusivo
\sim	\rightarrow	complemento a 1

Si utilizzano solo con operandi di tipo intero e carattere

A. Murli - Calcolo Parallelo e Distribuito

15

Linguaggio C

Punti da analizzare:

1. Prime Inizializzazioni
2. Operazioni dirette sui bit
3. Accesso agli indirizzi delle variabili
4. Allocazione dinamica

A. Murli - Calcolo Parallelo e Distribuito

16

Memorizzazione di un dato di tipo intero

In Memoria									
Indirizzi									
1 0 0 0	0	0	0	0	0	0	1	0	y = 2
1 0 0 1	0	0	0	0	0	0	1	1	x = 3
1 0 1 0	0	0	0	0	0	1	1	1	t = 7
...									
1 1 1 1									

A. Murli - Calcolo Parallelo e Distribuito

17

Memorizzazione di un dato di tipo reale

ESEMPIO $z = (.1172 \times 10^2)_{10} = (.1011110 \times 2^{100})_2$

0	1	0	1	1	1	1	1	0	0	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---

↓

Segno della mantissa

↓

Segno dell'esponente

Indirizzi	In Memoria												
1 0 0 0	0	1	0	1	1	1	1	1	0	0	1	0	0
1 0 0 1	1	1	1	0	1	1	1	1	0	0	0	1	1
1 0 1 0													
1 1 1 1													
	0	0	0	0	1	0	0	0	0	0	0	0	0

$z = .1172 \times 10^2$

$h = -.6625 \times 10^1$

1 1 1 1													
	0	0	0	0	1	0	0	0	0	0	0	0	0

$q = .8 \times 10^1$

A. Murli - Calcolo Parallelo e Distribuito

18

In C è possibile definire una **variabile (di tipo intero)** che contiene l'indirizzo di un'altra variabile di qualunque tipo.

Il puntatore e' una variabile che contiene l'indirizzo di un'altra variabile

A. Murli - Calcolo Parallelo e Distribuito

19

I PUNTATORI : DEFINIZIONE

In Memoria															
Indirizzi															
1 0 0 0	0	1	0	1	1	1	1	0	0	1	0	0	z = .1172x10 ² h = -.6625x10 ¹		
1 0 0 1	1	1	1	0	1	1	1	0	0	0	1	1			
1 0 1 0															
...															
...															
1 1 1 1	0	0	0	0	0	0	0	0	1	0	0	0	p = 8		

A. Murli - Calcolo Parallelo e Distribuito

20

ESEMPI...

x = 3 variabile di tipo intero

x: 0 0 0 0 0 0 1 1 Indirizzo: 1 0 1 0

Px contiene l'indirizzo di x

Px: 0 0 0 0 1 0 1 0

z variabile di tipo floating-point

z: 0 1 0 1 1 1 1 0 1 0 0 Indirizzo: 1 1 1 1

Pz contiene l'indirizzo di z

Pz: 0 0 0 0 1 1 1 1

A. Murli – Calcolo Parallelo e Distribuito 21

COME SI DICHIARA UN PUNTATORE?

int *p
p è un puntatore ad una variabile intera.

float *p
p è un puntatore ad una variabile float.

A.

A. Murli – Calcolo Parallelo e Distribuito 22

Esempio

```
...
int x ,*p;
p=&x; /*Assegnazione dell'indirizzo di x a p */
*p=3; /* L'istruzione assegna il valore 3 alla
       variabile x puntata da p */
...
```

Diagramma: Memoria con variabile p (indirizzo 100) e variabile x (valore 3). Una freccia verde indica che p punta a x.

(P punta a x)

A. Murli – Calcolo Parallelo e Distribuito 23

Linguaggio C

Punti da analizzare:

1. Prime Inizializzazioni
2. Operazioni dirette sui bit
3. Accesso agli indirizzi delle variabili
4. Allocazione dinamica

A. Murli – Calcolo Parallelo e Distribuito 24

Si ricordi che per l'allocazione **statica** di una matrice:

A[100][100]

Viene riservata una memoria di 100 × 100 elementi

Problema:

Utilizzo solo della memoria necessaria.

Risposta:

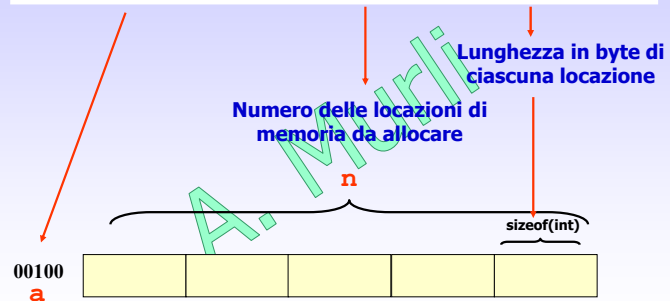
Le dimensioni di un array sono stabilite
in runtime

(ALLOCAZIONE DINAMICA)

```
#include <stdio.h>
main()
{ int *a,n,x,i;
  scanf("%d",&n);
  a=(int*)malloc(n*sizeof(int));
  ...
  for (i=0; i<n; i++)
  { a[i]= i; }
  ... }
```

Allocazione dinamica C

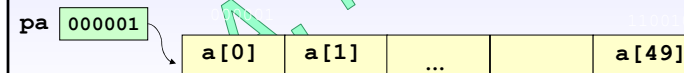
```
int *a;
a = (int *) malloc (n * sizeof(int));
```



Array e puntatori in C

Il nome di un array e' **l'indirizzo della prima componente**

```
...
int a[50];
int *pa;
pa = &a[0]; /* Definizione di pa */
...
```



pa contiene l'indirizzo del
primo elemento di **a**

Problema:

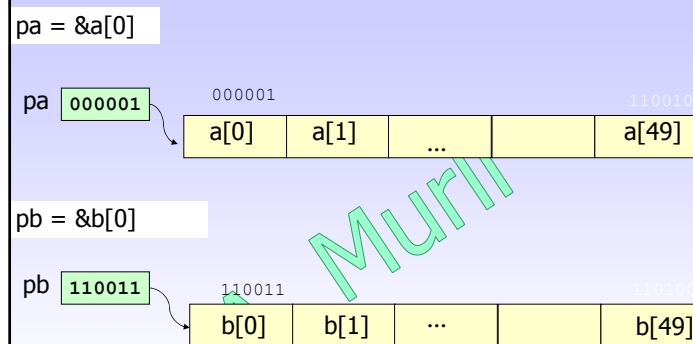
Dati 2 array di ordine N scambiare
il loro contenuto

Soluzione:

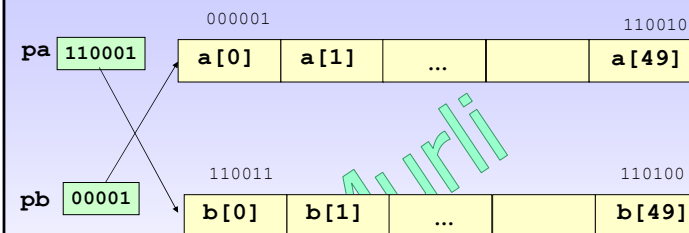
```
#include <stdio.h>
main()
{
    int a[50], b[50], *tmp, *pa, *pb;
    ...
    /* pa e pb contengono indirizzo del primo
    elemento di a di b */
    pa = &a[0];
    pb = &b[0];
    /* scambio di indirizzi */
    tmp = pa;
    pa = pb;
    pb = tmp;
    ...
}
```

1 SCAMBIO

In memoria....

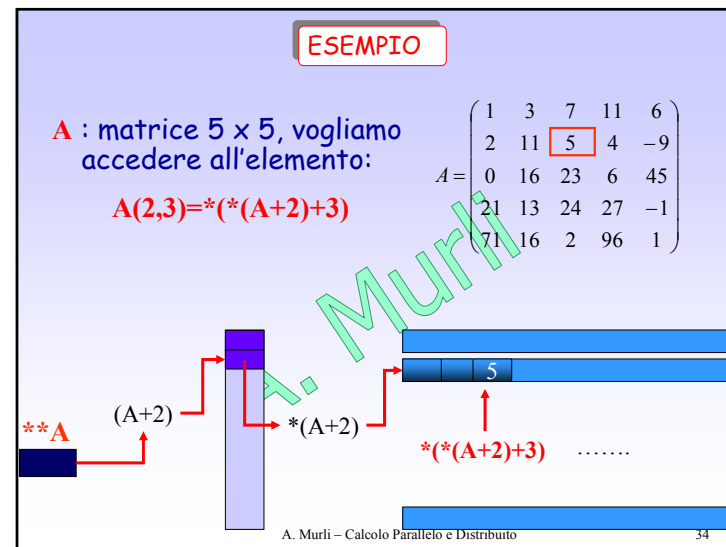
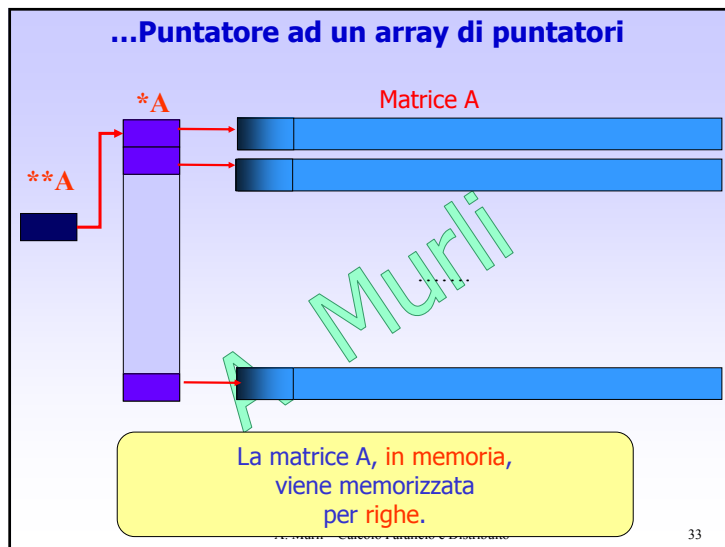


Lo scambio....



Dopo lo scambio :

pb contiene l'indirizzo del primo elemento di a
pa contiene l'indirizzo del primo elemento di b



Sintassi per definire un puntatore a puntatori

Definizione del puntatore a puntatori di float:

```
float **a;
```

Definizione del puntatore a puntatori di interi:

```
int **a;
```

A. Murli - Calcolo Parallelo e Distribuito 35

Sintassi per allocare un puntatore a puntatori

costruzione di una matrice n x m di float:

```
a=(float**)calloc(n,sizeof(float*));
for (i=0;i<n;i++){
a[i]=(float*)calloc(m,sizeof(float)); }
```

costruzione di una matrice n x m di interi:

```
a=(int**)calloc(n,sizeof(int*));
for (i=0;i<n;i++){
a[i]=(int*)calloc(m,sizeof(int)); }
```

A. Murli - Calcolo Parallelo e Distribuito 36

Esempio: trasposta di una matrice

```
#include <stdio.h>
/* Prototipo della funzione trasp */
Void trasp(float **,int );
main() {
    float **a;
    int n,i,j;
    printf("introdurre n \n");
    scanf("%d",&n);
    /* Allocazione dei puntatori alle righe */
    a=(float**)calloc(n,sizeof(float*));
    /* Allocazione delle righe */
    for (i=0;i<n;i++){
        *(a+i)=(float*)calloc(n,sizeof(float)); }
    printf("introdurre la matrice \n");
    For (i=0; i<n; i++)
        For (j=0; j<n; j++)
            scanf("%f",((a+i)+j)); /*
    &a[i][j] */
    trasp(a,n);
    printf("la trasposta è: \n");
    printf( "*(a+i)+j" ); }
```

Programma

Chiamante A. Murli – Calcolo Parallelo e Distribuito

Routine che esegue
la trasposta

37

PROBLEMA

Scambio a 2 a 2 delle righe di una matrice quadrata

Esempio: A di dimensione 4×4

$$A = \begin{pmatrix} 3 & 4 & 11 & 6 \\ 27 & 5 & 22 & 1 \\ 13 & 9 & 15 & 81 \\ 73 & 17 & 21 & 33 \end{pmatrix} \rightarrow A = \begin{pmatrix} 27 & 5 & 22 & 1 \\ 3 & 4 & 11 & 6 \\ 73 & 17 & 21 & 33 \\ 13 & 9 & 15 & 81 \end{pmatrix}$$

A. Murli – Calcolo Parallelo e Distribuito

38

Scambio a 2 a 2 delle righe di una matrice quadrata

```
#include <stdio.h>
/* Prototipo della funzione scambio */
Void scambio(float **,int );
main() {
    float **a;
    int n,i,j;
    printf("introdurre n \n");
    scanf("%d",&n);
    /* Allocazione dei puntatori alle righe */
    a=(float**)calloc(n,sizeof(float*));
    /* Allocazione delle righe */
    for (i=0;i<n;i++){
        *(a+i)=(float*)calloc(n,sizeof(float)); }
    printf("introdurre la matrice \n");
    For (i=0; i<n; i++)
        For (j=0; j<n; j++)
            scanf("%f",((a+i)+j));
    scambio(a,n);
    printf("RISULTATO: \n");
    printf( "*(a+i)+j" ); }
```

Programma
Chiamante

A. Murli – Calcolo Parallelo e Distribuito

Routine che esegue
lo scambio

39

In C per ogni coppia di righe bisogna effettuare solo
lo scambio dei loro indirizzi

$$A = \begin{pmatrix} 3 & 4 & 11 & 6 \\ 27 & 5 & 22 & 1 \\ 13 & 9 & 15 & 81 \\ 73 & 17 & 21 & 33 \end{pmatrix} \rightarrow A = \begin{pmatrix} 27 & 5 & 22 & 1 \\ 3 & 4 & 11 & 6 \\ 73 & 17 & 21 & 33 \\ 13 & 9 & 15 & 81 \end{pmatrix}$$

Scambi effettuati 2

In C vengono effettuati $n/2$
scambi!!

A. Murli – Calcolo Parallelo e Distribuito

40

Un breve sommario...

- Operatori sui bit e strutture di controllo;

Esercizio 1 (conversione da base 10 a base 2)

- Conversione di tipi ed utilizzo di funzioni di libreria matematiche

Esercizio 2 (somma di n numeri in valore assoluto)

- Utilizzo di funzioni che non ritornano alcun valore e utilizzo puntatori

Esercizio 3 (ordinamento per selezione)

Esercizio 1

Convertire un numero intero n da base 10 a base 2.

Pascal-like

```
begin CONVERSIONE
  Var: r:array(100) of integer
  Var: n,base,quoz,i: integer
  begin
    read n
    base:=2
    i:=0
    repeat
      i:=i+1
      quoz:=n/base
      r(i):= n-quoz*base
      n:=quoz
    until n=0
    for n=i,1,step -1 do
      print(r(n))
    endfor
  end
```

1/1

Esercizio 1

C

```
#include <stdio.h>

main()
{
  int n,base,quoz,i
  int *r; /* puntatore ad intero*/
  scanf("%d", &n);
  base=2;
  i=0;
  /* allocazione dinamica di r */
  r=(int*)calloc(n,sizeof(int));
  do{
    quoz =n>>1; /* divisione di n per 2*/
    r[i]= n-quoz<<1; /* moltiplicazione di n per 2*/
    n=quoz;
    i++; /* incremento del contatore */
  }while (n!=0); /* != per il confronto */
  for (n=i; n >= 0; i--)
    printf("%d \n",r[n]);
}
```

1/1

Esercizio 2

Calcolare la somma del valore assoluto di n numeri reali.

Pascal-like

```
procedure SOMMABS(a,n,sum)
  Var: a:array(100) of real
  Var: sum, tmp : real
  Var: n,i: integer
  begin
    sum:=0
    for i=1,n,step 1 do
      tmp:= abs(a(i))
      sum = sum + tmp
    endfor
  end
end procedure
```

1/1

Esercizio 2



```
float sommabs(float *a,int n)
{ float sum;
  double tmp;
  sum=0;
  for (i=0; i<n; i++) {
    /* *(a+i) equivale a[i] */
    /* Conversione di tipi (cast) */
    tmp = (double) (*(a+i));
    sum += fabs(tmp);
  }
  return sum;
}
```

```
#include <stdio.h>
#include <math.h>
float sommabs(float *,int );
main()
{
  int n,i;
  float *a; /* puntatore a reale*/
  scanf("%d ", &n);
  /* allocazione dinamica di a */
  a=(float*)calloc(n,sizeof(float));
  for (i=0; i<n; i++) {
    /* a+i equivalente &a[i] */
    scanf("%f", a+i);
    /* compattezza nella scrittura */
    printf("somma=%f \n", sommabs(a,n));
  }
}
```

double fabs(double x)

La funzione valore assoluto **fabs** necessita della direttiva

#include <math.h>

- In **input** richiede un **double**
- In **output** restituisce un **double**



Conversione di tipo (CAST)

In C e' possibile convertire una **variabile t** di assegnato tipo in un **qualsiasi altro tipo definito** dal linguaggio mediante l'istruzione

(tipo_variabile).t

ESEMPIO

```
main()
{ int a,b,c;
  float d, *pc;
  char nome;
  ...
  d =(float) a; /* a e' di tipo float */
  nome =(char) b; /* b e' di tipo carattere */
  pc=(float *) c; /* c e' di tipo puntatore a float */
  ...
}
```

Nel Esercizio 2 ...

E' necessario un'operazione di **cast**

da **float** a **double** ;

per memorizzare nella variabile **tmp** di tipo **_double** ;

```
tmp = (double) (*(a+i));
```

float

double

NON è necessario il **cast**

da **double** a **float** ;

per memorizzare nella variabile **sum** di tipo **_float** ;

```
sum += fabs(tmp);
```

double

Esercizio 2 bis

Calcolare 2^n con n numero intero

```
#include<stdio.h>
#include<math.h>

main()
{ int n,result;

    scanf("%d",&n);          /*Lettura di n */

    /* necessità del cast sulle variabili intere */
    result = (int) power((double) 2,(double) n);

    printf("risultato=%d\n",result);
}
```

Nel calcolo di 2^n e' necessario fare il cast delle variabili passate a `power` e del risultato restituito dalla funzione

Esercizio 3

Ordinamento per selezione di un array di reali di dimensione n

Pascal-like

```
procedure ORD(v,n)
Var: n,i,k,m:integer
Var: a:array (n) of real
Var: t:real
Var: l:boolean
begin
    k:=n
    l:=.true.
    repeat
        m:=1
        for i=2, k do
            if (a(m)<a(i)) then
                m:=i
            endif
        endfor
        if m=k then
            l:=.false.
        else
            t:=a(k)
            a(k):=a(m)
            a(m):=t
        endif
        k:=k-1
    until k:=1 ∨ ¬l
    return a
end
```

Esercizio 3

```
#include <stdio.h>

void ordina(float *,int );
main()
{ int n,i
  float *a; /* puntatore a reale*/
  scanf("%d ", &n);
  /* allocazione dinamica di a */
  a=(float*)calloc(n,sizeof(float));
  for (i=0; i<n; i++) {
      /* a+i equivalente &a[i] */
      scanf("%f", a+i);}
  /* compattezza nella scrittura */
  ordina(a,n);
  for (i=0; i<n; i++)
      printf(" a[%d]=%f \n",i,a[i] );
}
```

```
void ordina(float *a,int n)
{ float tmp;
  int m,i,k,l;
  k=n-1;
  l=0;
  do( /* inizio ciclo repeat */
      m=0;
      for (i=1; i<k; i++) {
          if ( a[m]<a[i] )
              m=i;
      }
      if ( m == k )
          !l; /* l=1 */
      else
      {
          tmp =a[k];
          a[k]=a[m];
          a[m]=t ;
      }
      k=k-1;
  }while(k==0 || !l);
  return;
}
```

Funzioni VOID

I sottoprogrammi che non ritornano alcun valore al programma chiamante si ottengono in C attraverso funzioni VOID

```
void ordina(float *a,int n)
```

```
return;
```

La VOID non ritorna alcun tipo al programma chiamante