Array di puntatori,

puntatori a funzioni,

array di funzioni

Conversioni

(anno,mese, giorno) ----> giorno dell'anno

(anno, giorno dell'anno) ----> (mese e giorno)

Array multidimensionali

```
static int gior_mese[2][13]={
{0,31,28,31,30,31,30,31,31,30,31,30,31},{0,31,29,31,30,31,30,31,31,30,31,30,31}};
int giorno_anno(int anno, int mese, giorno)
{
        int   i, bisest;
        bisest = anno % 4 == 0 && anno %100 != 0 || anno %400 == 0;
        for(i = 1; i < mese; i++)
                giorno += gior_mese[bisest][i];
        return(giorno);
}

mese_giorno(int anno, int annogior, int  *pmese, int *pgiorno)
{
        int       i, bisest;
        bisest = anno % 4 == 0 && anno %100 != 0 || anno %400 == 0;
        for(i = 1; annogior > gior_mese[bisest][i]; i++)
                annogior -= gior_mese[bisest][i];
        *pmese = i;
        *pgiorno = annogior;
}
```

Array multidimensionali

```
static int gior_mese[2][13]={
{0,31,28,31,30,31,30,31,31,30,31,30,31},{0,31,29,31,30,31,30,31,31,30,31,30,31}};
int giorno_anno(int anno, int mese, giorno, int (*gior_mese)[13])
{
        int   i, bisest;
        bisest = anno % 4 == 0 && anno %100 != 0 || anno %400 == 0;
        for(i = 1; i < mese; i++)
                giorno += gior_mese[bisest][i];
        return(giorno);
}

mese_giorno(int anno,int annogior,int  *pmese,int *pgiorno, int gior_mese[][13])
{
        int       i, bisest;
        bisest = anno % 4 == 0 && anno %100 != 0 || anno %400 == 0;
        for(i = 1; annogior > gior_mese[bisest][i]; i++)
                annogior -= gior_mese[bisest][i];
        *pmese = i;
        *pgiorno = annogior;
}
```

Array di puntatori

- Possono essere usati in diverse situazioni
- Utili per la trattazione di stringhe
  - *char *a[5]* contiene 5 puntatori a carattere
  - *char *a[5]* può essere visto come un array di stringhe

Array di puntatori

```
char *nome[]={
                "mese scorretto",
                "Gennaio",
                "Febbraio",
                "Marzo",
                …
                …
                "Dicembre"};

char  * nome_mese(int n) /* ritorna il nome dell' n-esimo
                                                mese */
{


        return((n < 1 || n > 12) ? nome[0] : nome[n]);
}
```

Array di puntatori

```
char *nome[]={
                "mese scorretto",
                "Gennaio",
                "Febbraio",
                "Marzo",
                …
                …
                "Dicembre"};
mese_giorno(int anno, int annogior, char  **pmese, int *pgiorno)
{
        int     i, bisest;
         bisest = anno % 4 == 0 && anno %100 != 0 || anno %400 == 0;
        for(i = 1; annogior > gior_mese[bisest][i]; i++)
                annogior -= gior_mese[bisest][i];
        *pmese = nome[i];
        *pgiorno = annogior;
}
```

Un programma che ordina le parole

```
#define N 20
#define MAXW 50
void InWords(char *[ ],int);
void SortWords (char *[ ], int);
void swap (char **, char **);
void PrintWords(char *[ ], int);
main()
{
char *w[N];
printf("inserisci %d parole di max 50 caratteri\n",
N);
InWords(w,N);
SortWord (w, N);
PrintWords(w,N);
}
```

- Prototipi:
  - InWords prende in input N parole e le inserisce in un array
  - SortWords ordina le parole
  - swap è utilizzata da SortWords per scambiare due parole
  - PrintWords stampa le parole

La funzione *InWords()*

```
void InWords(char *a[ ],int n)
{
int i;
for (i=0;i < n; i++)
{
 a[i] = calloc (MAXW, sizeof(char));
scanf ("%s", a[i]);
}
```

- alloca memoria per una stringa di MAXW caratteri
- a[i] punta all'inizio della stringa

La funzione *PrintWords()*

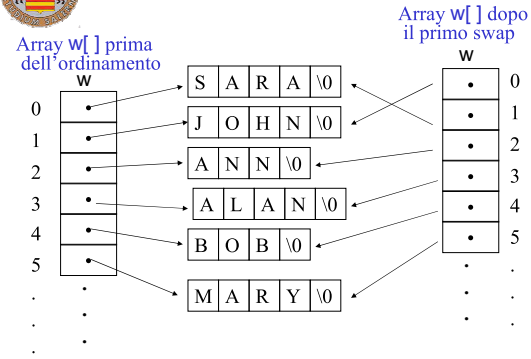
```
void PrintWords (char *a[ ],int n)
{
int i;
for (i=0;i < n; i++)
{
 printf("%s\n", a[i]);
}
```

Le funzioni *SortWords()* e *swap()*

```
 void SortWords(char *a[ ], int n)
{
int i,j;
for (i=0; i < n; i++)
  for (j=i+1; j< n; ++j)
    if (strcmp (a[i], a[ j ]) > 0)
      swap (&a[i], &a[ j ]);
}
void swap (char ** p, char **q)
{
char *tmp;
tmp = *p;
*p =*q;
*q = tmp;
}
```

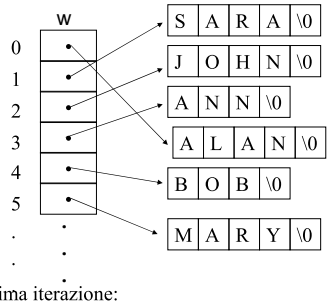
- *SortWords()*
  - confronta ogni parola con tutte le successive
  - se necessario scambia le parole confrontate
- Funzione *swap()*
  - ha per argomenti indirizzi di puntatori a carattere

Esempio



Esempio

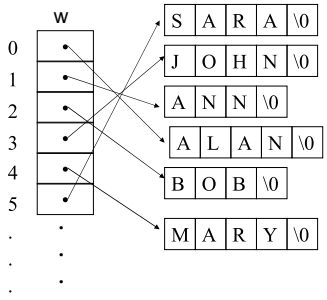
Array w[ ] dopo la prima iterazione del for esterno



Ordine dopo la prima iterazione:
SARA JOHN ANN ALAN BOB MARY

Esempio

Array w[ ] dopo l'ordinamento



Parametri di *main()*

- *main()* rappresenta una funzione avente particolari caratteristiche:
  - viene eseguita per prima
  - riceve i parametri dalla riga di comando
- Per passare a *main* i parametri dalla riga di comando si specificano due argomenti:
  - *argc* rappresenta il numero di parametri
  - argv[ ] e` un array di stringhe ognuna delle quali e` una parola sulla riga di comando

Passaggio di parametri a *main*

```
int main(int argc, char  *argv[ ])
{
int i;
printf ("argc=%d\n", argc);
for (i = 0; i < argc ; i++)
printf ("%s ", argv[i]);
}
```

- Se si digita
  a.out pippo topolino e pluto
  - il programma stampa
    5
    a.out
    pippo
    topolino
    e
    pluto
- Se non si passa alcun parametro
  - *argc* vale 1 e quindi stampa solo
    1
    a.out

Funzioni come parametri di funzione

- Il C permette ad una funzione di avere come argomenti una o più funzioni da applicare a dati
- Esempio di prototipo di funzione avente una funzione come parametro
  - int g (int f(int) , int n);
- Nelle dichiarazioni dei parametri le funzioni vengono interpretate come *puntatori a funzioni*
  - int g (int f(int) , int n); e`equivalente a
    int g (int (*f)(int) , int n);

Un esempio: somma_funz

- Obiettivo: scrivere una funzione che effettui la somma dei valori ottenuti applicando una certa funzione agli elementi di un array di *n* interi
- Esempi di uso:
  - somma delle radici quadrate degli elementi dell'array
- Parametri in input della funzione *somma_funz:*
  - l'array di *n* interi
  - la dimensione *n* dell'array
  - la funzione f da applicare

La funzione somma_funz

```
double somma_funz(double f (double), int dim, double arr[ ])
{
int i;
double somma = 0;
for (i = 0; i< dim; i++)
somma + = f (arr[i]);
return (somma);
}
```

• Parametri
– funzione *f* che accetta un double e restituisce un intero
– intero che specifica al dimensione dell'array
– array

Il programma che usa *somma_funz*

• Dichiara un array di 10 numeri reali
  – inizializzato a 1.4, 2.4, 3.8, 4, 5, 6.9, 7.2, 8, 9.4, 10
• Prende in input sulla riga di comando la funzione che si vuole applicare agli elementi dell' array
• Chiama somma_funz passandole la funzione specificata sulla riga di comando

Il programma che usa*somma_funz*

```
#include<string.h>
#include<stdio.h>
double sqrt(double);
double log(double);
double somma_funz(double f (double), int, double []);
double main(int argc, char  *argv[ ])
{
double[10]= {1.4, 2.4, 3.8, 4, 5, 6.9, 7.2, 8, 9.4, 10};
char *scelta;
if (argc > 1)
 scelta = argv[1];
if(!strcmp(scelta , "radici_quadrate"))
  printf ("somma delle radici quadrate =%lf\n", somma_funz(sqrt,10,a);
else
  printf("somma dei logaritmi =%lf\n", somma_funz(log,10,a);
}
```

Puntatori a funzioni

• Una funzione che compare come parametro viene interpretata come un *puntatore ad una funzione*

• All'interno di somma_funz f puo` essere trattato come una funzione o puo` essere trattato come un puntatore:
  – somma += f (arr[i]);
  – somma += (*f) (arr[i]);

Puntatori a funzioni

• Il compilatore C tratta il nome di una funzione come un puntatore

• Quando una funzione g viene passata come parametro ad un'altra funzione in realta` viene passato l'indirizzo di g

• E` possibile fare assegnamenti a puntatori a funzioni:

  – Esempio:
    e` possibile dichiarare un puntatore a funzione
    `double (*pf)(double);`
    e assegnargli l'indirizzo di una funzione
    `pf = somma_funz; (o anche pf= &somma_funz;)`

Array di puntatori a funzioni

• Si possono dichiarare array di puntatori a funzioni
• Dichiarazione di una array di due funzioni che
  – hanno come parametri due *double*
  – restituiscono un *double*

`double (*funz[2])(double,double);`

• Ogni elemento può essere assegnato:
`funz [0] = pow;  (oppure funz [0] = &pow;)`

Uso di *somma_funz* con un array di funzioni

```
double somma_funz(double f (double), int, double []);
double sqrt(double);
double log(double);
int main(int argc, char *argv[])
{
double a[10]= {1.4, 2.4, 3.8, 4, 5, 6.9, 7.2, 8, 9.4, 10};
char *scelta =“log”;
double(*funz[2])(double);     /*dich. Array di puntatori a funzioni */
funz[0] = sqrt;
funz[1] = log;
if (argc > 1)
 scelta = argv[1];
if(*scelta == “radici_quadrate”)
  printf (“somma delle radici quadrate =%lf\n", somma_funz(funz[0],10,a);
else
  printf(”somma dei logaritmi =%lf\n", somma_funz(funz[1],10,a);
}
```

Esercizio 1

- scrivere una funzione *applica()* che effettui un’operazione su un array di n interi
- esempi :
  - somma degli n interi a[0]+a[1]+a[2]+a[3]+a[4]
  - prodotto degli n interi a[0]*a[1]*a[2]*a[3]*a[4]

- Parametri in input della funzione *applica*:
  - l’array di n interi
  - la dimensione dell’array
  - la funzione da applicare

- La funzione *applica()* restituisce un intero

Esercizio 2

- Scrivere un programma che usa *applica*
- Dichiara due funzioni
  - *int somma (int, int)* che somma due interi
  - *int prodotto (int, int)* che moltiplica due interi
- Dichiara un array di 10 interi
  - inizializzato a 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- Chiama *applica* passandole o la funzione somma o la funzione prodotto

Altri esercizi

- 38 , 42 e 43 del cap.6