FUNZIONI & ARRAY IN C

Prof. Fischetti Pietro

Una funzione e' un blocco di istruzioni con un nome, che puo' o meno restituire un valore e che puo' o meno accettare un certo numero di parametri (racchiusi tra parentesi tonde) passati per valore e/o riferimento.

Esempio:

funzione che calcola e restituisce la somma di 2 numeri interi:

```
int somma(int a , int b)
{
        int c=0;
        c=a+b;
        return c;
}
```

Il blocco e' tutto cio' che e' contenuto tra le parentesi graffe, il nome del blocco in questo caso e' 'somma', i parametri sono a e b passati per valore (vuol dire che viene passata una copia, un po' come dare la fotocopia della propria carta d'identita'), il tipo di ritorno e' un int in questo caso (nel caso in cui la funzione non ritorni nulla si mette return void).

Il compilatore C deve conoscere preventivamente la dichiarazione della funzione una volta che viene incontrata la chiamata. Ci sono 2 modi per farlo:

1)mettere la funzione prima del main

```
#include <stdio.h>
int somma(int a , int b)
{
        int c=0;
        c=a+b;
        return c;
}
int main()
{
        int r;
        r=somma(3,2);
}
```

2) mettere prima del main (o in un file esterno con estensione .h richiamato con la #include) la sola dichiarazione:

```
#include <stdio.h>
int somma(int a , int b);

Int main()
{
        int r;
        r=somma(3,2);
}

int somma(int a , int b)
{
        int c=0;
```

```
        c=a+b;
        return c;
}
```

Notare che anche main e' una funzione, e' speciale perche' il nome deve essere esattamente scritto cosi' (in minuscolo) in quanto viene chiamata dal sistema operativo quando lancio il programma.

ARRAY

Un array statico e' una porzione di memoria contigua identificata da un nome contenente elementi tutti dello stesso tipo. Esempio:

```
int v[5];
```

dichiara un blocco di memoria di 5 interi (5*32=160 bytes su una macchina a 32 bit) contigui, con nome v. Il contenuto non viene inizializzato dal C (a differenza di altri linguaggi), quindi deve essere fatto esplicitamente.

```
int v[5]= {0};//Inizializza a zero l'intero vettore
```

```
int m[3][5]= {0};//Inizializza a zero l'intera matrice
```

Si puo' dichiarare un array e contemporaneamente inizializzarlo (nell'esempio sotto non e' necessario impostare la lunghezza in quanto viene calcolata dal C), altrimenti le dimensioni dell'array vanno specificate come costanti:

```
int arr[] = {1, 2, 1, -4, 5};
```

oppure:

```
int arr[5];
arr[0]=1;
arr[1]=2;
…
```

NB!: Una stringa e' un vettore di caratteri terminato con il carattere speciale '\0' (NULL)

Esempio:

"salve"

viene memorizzata internamente come (6*8=48 bytes nel caso di codifica ASCII, N.B. 6 Bytes!!!):

s	a	l	v	e	'\0'

Quindi attenzione:

```
char s[]="salve";
```

e' profondamente diverso da:

```
char v[]={'s','a','l','v','e'};
```

per renderli equivalenti devo aggiungere il carattere di fine stringa:

```
char v[]={'s','a','l','v','e','\0'};
```

Per sapere velocemente quanto e' grande un'array si puo' utilizzare la funzione del C sizeof.

Esempio:

int v[5];

printf("%d",sizeof(v)/sizeof(v[0]));//stampa 5

Il C, a differenza di altri linguaggi in cui gli array sono oggetti, non memorizza la lunghezza massima di un array, questo rende il C da un lato piu' veloce ma ne complica l'utilizzo. In pratica nelle applicazioni scritte in C in cui occorre trattare con array, si dimensiona inizialmente l'array (ripeto con valore costante) ad un valore che si presume sufficiente a contenere tutti i dati (questo puo' comportare uno spreco di memoria) e si memorizza in una variabile quanti dati sono stati effettivamente inseriti. Nell'esempio seguente si crea un vettore che puo'contenere al massimo 100 interi, poi si passa in un while in cui l'utente inserisce i dati fino a che decide di terminare inserendo la combinazione di tasti CONTROL+D (scanf ritorna un valore >0 se ha letto almeno un dato del tipo specificato), il numero di dati effettivamente letti vengono memorizzati nella variabile n, (inserire un controllo nel caso in cui n superi NMAX per evitare spiacevoli sorprese):

```
#define NMAX 100
int v[NMAX];
int n=0,i=0;
//legge
while (scanf("%d",&v[n])>0)
        n++;

//stampa il vettore
for(i=0;i<n;i++)
        printf ("%d ",v[i]);
```

FUNZIONI CON ARRAY

Per quanto visto nel paragrafo precedente, nei parametri di funzioni che sono array occorre obbligatoriamente specificare le dimensioni, a meno che che non siano stringhe, in questo caso si puo' anche omettere la dimensione facendo grande attenzione in quanto la stringa deve essere terminato con il NULL.
Esempio:

```
#include <stdio.h>
#define NMAX 100

//riempie un vettore v di lunghezza n con il valore passato in a
int v1d(int v[],int n,int a)
{
 int i=0;
 for (i=0; i < n; i++)
        v[i]=a;
 return i;
}

//riempie una matrice m [nr x nc] con il valore passato in a, deve essere specificato il numero di colonnemax.
int v2d(int m[][NMAX],int nr,int nc,int a)
{
 int i=0,j=0;
 for (i=0; i < nr; i++)
        for (j=0; j < nc; j++)
                m[i][j]=a;
 return i+j;
```

```
}
//funzione che conta quanti caratteri presenti nella stringa s sono uguali al parametro a
int s1d(char s[], char a)
{
 int i=0,k=0;
 while(s[i++]!='\0')
        if (s[i]==a)
                k++;
 return k;
}
int main()
{
        #define NR 100
        #define NC 100
        int V[NMAX] ={0},M[NMAX][NMAX]={0},i=0,j=0;
        char S[NMAX]="salve a tutti";

        v1d(V,NR,1);
        for (i=0; i< NR;i++)
                printf("%d ",V[i]);

        printf ("\n\n");

        v2d(M,NR,NC,1);
        for (i=0; i< NR;i++)
        {
                for (j=0; j< NC;j++)
                        printf("%d ",M[i][j]);
                printf("\n");
        }
        printf ("\n\n");

        printf("%d",s1d(S,'a'));

        return 0;
}
```