Lezione 8: le Funzioni

Laboratorio di Elementi di Architettura e Sistemi Operativi

3 Aprile 2013

Funzioni

- Un programma C consiste di una o più funzioni
  - Almeno `main()`
- Funzionamento simile ai metodi di Java
- Definizione delle funzioni
  - Prima della definizione di `main()`
  - Dopo la definizione di `main()` $\Rightarrow$ necessario premettere in testa al file il *prototipo* della funzione:
    - Nome
    - Argomenti
    - Tipo del valore di ritornato (`void` se la funzione non ritorna valori)
  - `return` per terminare e ritornare un valore
  - Le funzioni ausiliarie si possono raccogliere in un file da includere con la direttiva `#include`

Funzioni e prototipi: esempio

```
double f(int x)
{
  ...
}

int main ()
{
  ...
  z = f(y);
  ...
}
```

```
double f(int);

int main ()
{
  ...
  z = f(y);
  ...
}

double f(int x);
{
  ...
}
```

Dichiarazione

Definizione

Passaggio dei parametri

- In C, il passaggio dei parametri avviene per *valore*
  - **Significato:** Il valore dei parametri attuali viene copiato in variabili locali della funzione
- Implicazione

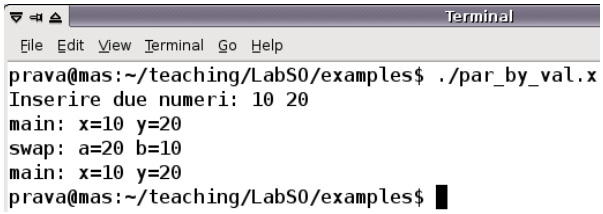
  - I parametri attuali non vengono MAI modificati dalle istruzioni della funzione

- *Eccezione:*
  - Stringhe e vettori sono passati per *indirizzo*:
    * possono quindi essere modificati dalle istruzioni della funzione.

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

void swap(int a,int b) {
  int tmp;
  tmp = a;
  a = b;
  b = tmp;
  printf("swap: a=%d
          b=%d\n",a,b);}
```

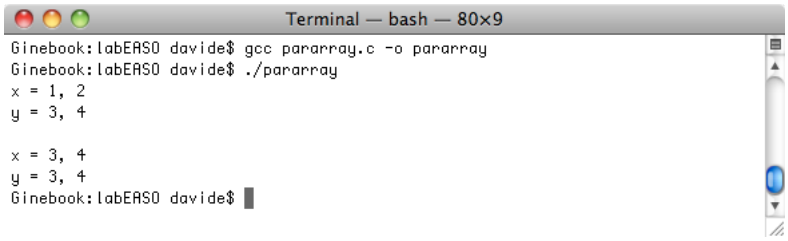
```
int main()
{
  int x,y;
  printf("Inserire due
          numeri: ");
  scanf("%d %d",&x,&y);
  printf("main: x=%d
          y=%d\n",x,y);
  swap(x,y);
  /* x e y NON VENGONO
     MODIFICATI */
  printf("main: x=%d
          y=%d\n",x,y);}
```




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

void copia(int a[],
   int b[], int n) {
  int i;
  for(i=0; i<n; i++) {
    a[i] = b[i];
  }
}
```

```
int main()
{
  int x[] = {1, 2};
  int y[] = {3, 4};
  printf("x = %d, %d\n", x[0], x[1]);
  printf("y = %d, %d\n\n", y[0], y[1]);
  copia(a, b, 2);
  printf("x = %d, %d\n", x[0], x[1]);
  printf("y = %d, %d\n", y[0], y[1]);
}
```

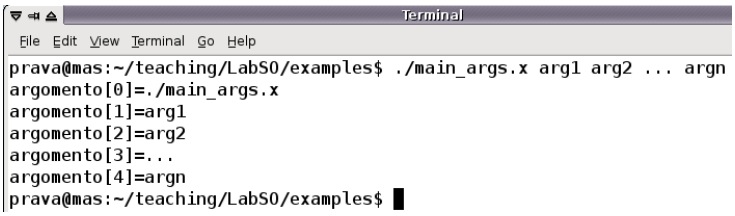


Parametri del `main`

- Come in JAVA è possibile passare dei parametri direttamente dalla linea di comando
- Sintassi:
  - Numero di parametri: `int argc`
  - Vettore di parametri: `char *argv[]`
  - `argv[0]` = *nome del programma*
- il `main` può ritornare un valore `int` come exit status del programma

- Esempio:

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



Funzioni di libreria

- Il C prevede numerose funzioni predefinite per scopi diversi
- Particolarmente utili sono
  - Funzioni matematiche
  - Funzioni di utilità
- Definite in specifiche *librerie*
- Tutte descritte nel `man`

Funzioni matematiche

- Utilizzabili con `#include <math.h>`

funzione	*definizione*
`double sin (double x)`	seno
`double cos (double x)`	coseno
`double tan (double x)`	tangente
`double asin (double x)`	arcoseno
`double acos (double x)`	arcoseno
`double atan (double x)`	arcotangente
`double atan2 (double y, double x)`	atan (y / x)
`double sinh (double x)`	seno iperbolico
`double cosh (double x)`	coseno iperbolico
`double tanh (double x)`	tang. iperbolica

funzione	definizione
`double pow (double x, double y)`	x^y
`double sqrt (double x)`	radice quadrata
`double log (double x)`	logaritmo naturale
`double log10 (double x)`	logaritmo decimale
`double exp (double x)`	e^x
`double ceil (double x)`	ceiling(x)
`double floor (double x)`	floor(x)
`double fabs (double x)`	valore assoluto
`double fmod (double x, double y)`	modulo

Funzioni di utilità

- Classificazione caratteri: `#include <ctype.h>`

funzione	definizione
`int isalnum (char c)`	Se c è lettera o cifra
`int isalpha (char c)`	Se c è lettera
`int isascii(char c)`	Se c è lettera o cifra
`int islower(char c)`	Se c è minuscola
`int isdigit (char c)`	Se c è cifra
`int isupper (char c)`	Se c è maiuscola
`int isspace(char c)`	Se c è spazio,tab,\n
`int iscntrl(char c)`	Se c è di controllo
`int isgraph(char c)`	Se c è stampabile, non spazio
`int isprint(char c)`	Se c è stampabile
`int ispunct(char c)`	Se c è di interpunzione

- Funzioni matematiche intere: `#include <stdlib>`

funzione	definizione
`int abs (int n)`	valore assoluto
`long labs (long n)`	valore assoluto
`div_t div (int numer, int denom)`	quoto e resto della divisione intera
`ldiv_t ldiv(long numer, long denom)`	quoto e resto della divisione intera

- Stringhe: `#include <string.h>`

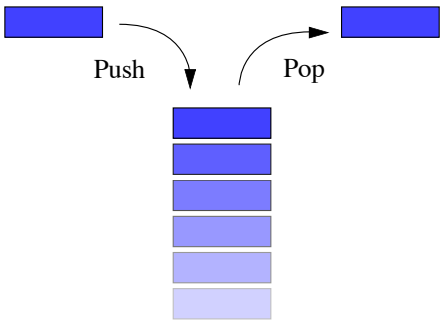
funzione	*definizione*
`char* strcat (char* s1, char* s2);`	concatenazione di s1 e s2
`char* strchr (char* s, int c);`	trova c dentro s
`int strcmp (char* s1, char* s2);`	confronto
`char* strcpy (char* s1, char* s2);`	copia s2 in s1
`int strlen (char* s);`	lunghezza di s
`char* strncat (char* s1,char* s2,int n);`	concat. n car. max
`char* strncpy (char* s1,char* s2,int n);`	copia n car. max
`int strncmp(char* dest,char* src,int n);`	cfr. n car. max

Esercizio 3

1. Realizzare un insieme di funzioni per gestire una pila (stack) di caratteri. In particolare, si implementino le operazioni di inserimento (`pop`) ed estrazione (`push`) di un valore nella pila, ed il controllo di pila vuota (`isempty`). Si assuma che la pila possa contenere al massimo 80 valori.
2. Utilizzare le funzioni di gestione della pila per controllare il bilanciamento delle parentesi di una stringa di testo.
...

Le pile

- Una pila (o stack) è un tipo di dato astratto, in cui i dati vengono estratti in ordine rigorosamente inverso rispetto a quello in cui sono stati inseriti.
- Una pila si manipola tramite due operazioni:
  - `push`: inserimento di un nuovo dato in cima alla pila
  - `pop`: estrazione del dato in cima alla pila



- *Implementazione:* un vettore per contenere gli elementi, ed un indice all'ultimo elemento inserito.