086180 - INFORMATICA APPLICATA

A.A. 2011-12

2° semestre

M05a – le funzioni

Dichiarazioni e Definizioni

- Una dichiarazione è
  - Il punto in cui ad un nome viene associato un tipo
- Una definizione è
  - È anche una dichiarazione, ma viene creato l'oggetto riservandogli memoria
  - Una definizione di funzione ha anche un body nella forma di una frase composta

Parametri e argomenti

- I parametri formali
  - sono i nomi usati all'interno della funzione per riferirsi agli argomenti
- Argomenti attuali
  - Sono i valori usati nel momento in cui si chiama la funzione
  - Sono di fatto i valori che vengono assegnati ai parametri formali all'ingresso della funzione

- **Dichiarazione**
  - `double  aax1 (float);`
- **Uso**
  - `main() {`
    - `double  aax1 (float);`
    - `double  y;  float z;`
    - `y = aax1(z);`
    - `exit (EXIT_SUCCESS);`
  - `}`
- **Definizione**
  - `double aax1(float k) {`
    - *`Bla bla bla;`*
  - `return (3.5 * k);`
  - `}`

La frase "return"

- Ogni funzione, eccettuate quelle che ritornano "void" devono avere almeno un return per mostrare quale è il valore che si deve ritornare
- Se una funzione ritorna un "void" allora può non esserci una frase return
- La sintassi è
  - `return (espressione) ;`

```
#include <stdio.h>

main() {int car;
int non_space(void);
while(non_space() != '.')
        printf("\n%c",car)
}

int non_space(void){
int c;
while ( (c=getchar() ) ==
  '\t' ||                c ==
  '\n' || c==' ');
/* empty statement */
      return (c);
}
```

Argomenti

- Il valore del parametro attuale viene assegnato al parametro formale nel momento della chiamata
- Come in un assegnamento
  - Se la funzione è
  - `f(int x, int y) { …. }`
  - La chiamata `f(12,k)` equivale a
  - `X=12; y=k;`
  - seguito da chiama `f`

Il prototipo

- È la dichiarazione di un nome di funzione che fornisce informazioni circa il numero e il tipo di argomenti
  - **Es.** `void pmax (int,int);`
- Nel caso in cui la funzione non avesse argomenti
  - `int f_name (void);`

Un ultimo esempio

- Il fattoriale di N è definito come
  1. N * N-1 * N-2 * ……. *1
- Ovvero come
  2. N * fattoriale di (N-1)

Si scriva una funzione

```
int fattoriale (int)
```

In base alla definizione 1

```
1. /*    N * N-1 * N-2 * ..... *1 */

2. int fattoriale1 (int N) {
3.   int i,risultato=1;
4.
5.   for(i=N; i>=1;i-- )
       {
       risultato= risultato * i;
       };
6.   return risultato;
7. }
```

In base alla definizione 2

```
1. /*  N * fattoriale di (N-1) ; fattoriale(1)=1
   */
2.
3. int fattoriale2 (int N) {
4. if (N==1) return (1);
5. else return (N*fattoriale2(N-1));
6. }
```

- La funzione richiama sé stessa!!!

```c
#include <stdio.h>
int main()
{long int fattoriale (long int);
long int n=1;
while(n>0)  {
    scanf("%ld",&n);
printf("\n\n Fattoriale di %ld  =  %ld\n\n" , n, fattoriale(n));
};
    return 0;
}

long int fattoriale (long int N) {
if (N==1) return (1);
else return (N*fattoriale(N-1));
}
```

```
5

 Fattoriale di 5   =  120

9

 Fattoriale di 9   =  362880

11

 Fattoriale di 11   =  39916800
```