

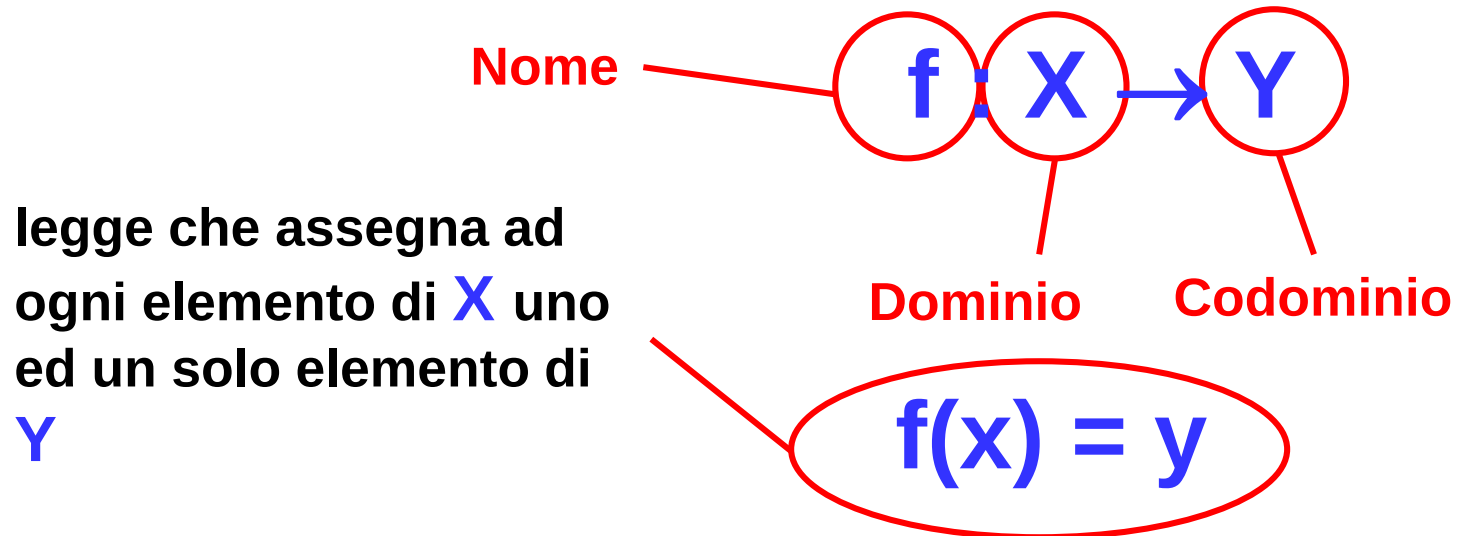
Programmazione e Laboratorio di Programmazione

Lezione VI

Le funzioni

Le funzioni della matematica

- Definizione di una funzione:

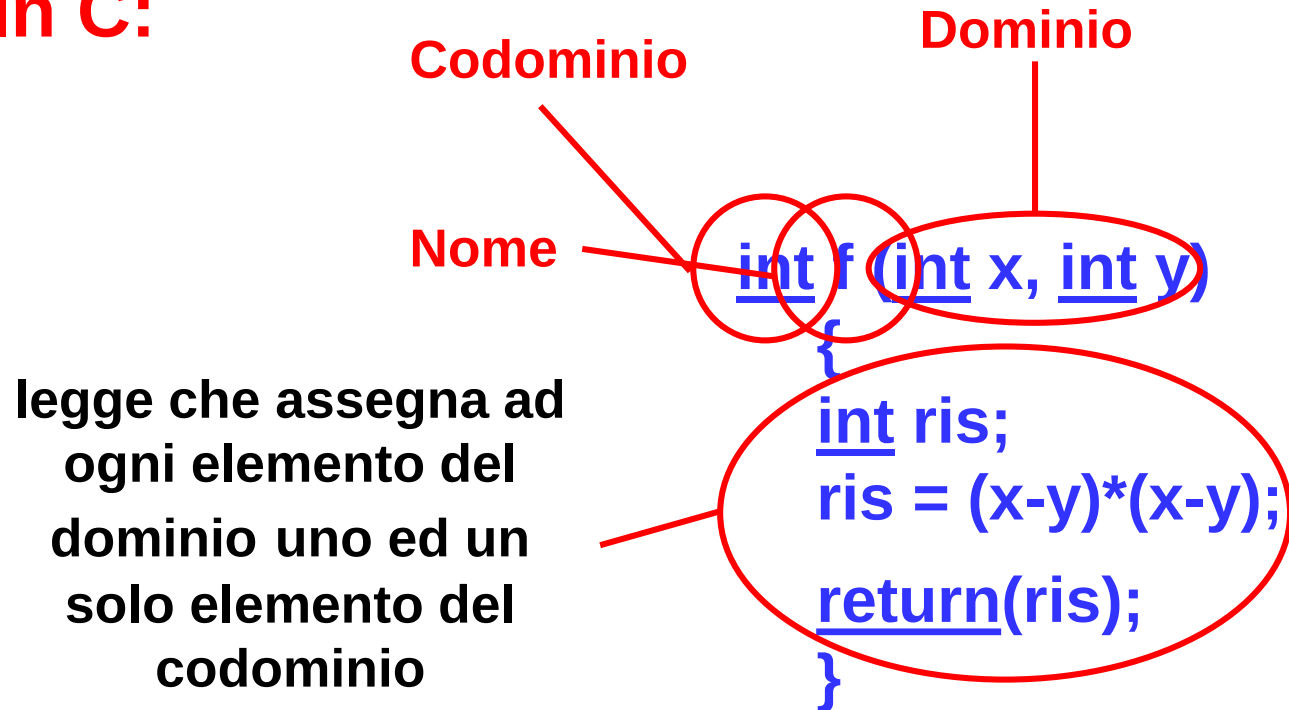


- In matematica:

$$f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$
$$f(x, y) = (x - y)^2$$

Le funzioni del C

- In C:

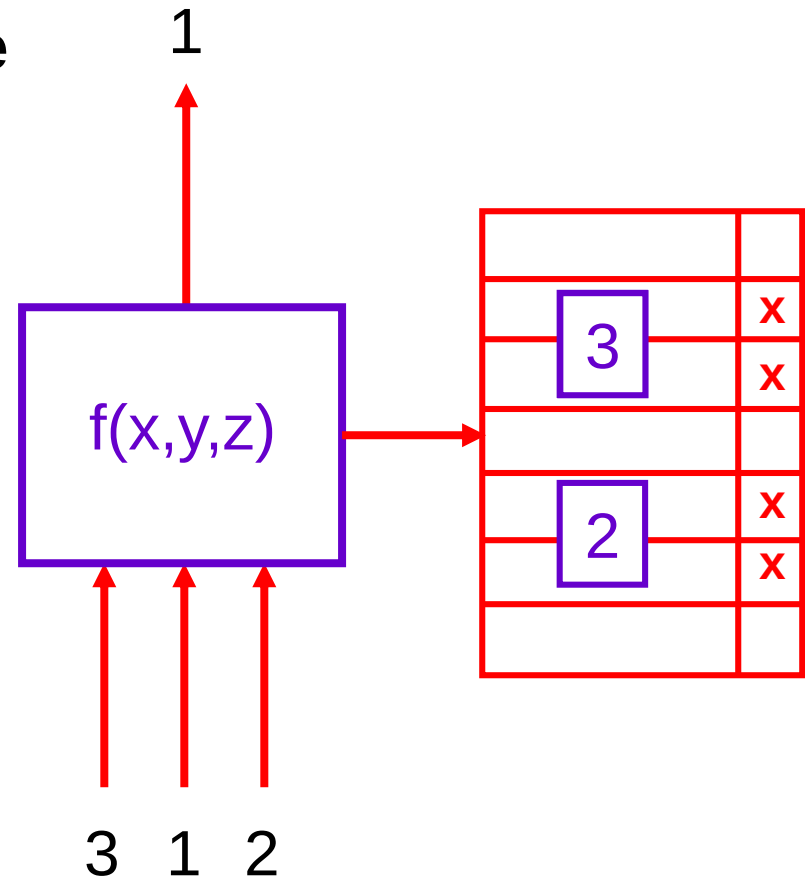


- Ma non c'è proprio nessuna differenza?

La differenza ...

- **Le funzioni del C:**

- possono calcolare valori
- ma possono anche modificare lo stato della memoria



Definizione di una funzione

- **Specificare:**
 - a) il nome
 - b) se calcola o meno un valore, e, nel caso, il tipo del valore calcolato
 - c) il nome e il tipo delle variabili di ingresso (**parametri formali**)
 - d) le modalità secondo le quali (**algoritmo**) calcola il valore restituito e/o modifica lo stato della memoria a partire dai parametri formali

Definizione di funzione

- **Funzioni che calcolano un valore:**

```
tipoF nomeF (tipo1 par1, ..., tipok park)  
{  
  ...  
  return(espressione);  
}
```

il valore di
espressione
è il valore
"restituito"
dalla
funzione

- **Funzioni che si limitano a modificare
lo stato della memoria:**

```
void nomeF (tipo1 par1, ..., tipok park)  
{  
  ...  
  return(espressione);  
}
```

intestazione

corpo

Definizione di funzione

- **Esempio:**

```
/* definizione della funzione che calcola la somma di 4 numeri interi */  
int somma (int num1, int num2, int num3, int num4)  
{  
    /* definizione della variabile per la somma */  
    int totale;  
    /* calcola la somma dei 4 interi */  
    totale = num1 + num2 + num3 + num4;  
    /* restituisce il valore cosi' calcolato */  
    return(totale);  
}
```

```
/* definizione della funzione che calcola la somma di 4 numeri interi */  
int somma (int num1, int num2, int num3, int num4)  
{  
    /* restituisce la somma dei 4 interi */  
    return(num1 + num2 + num3 + num4);  
}
```

Chiamata di funzione

- **E' necessario specificare:**
 - a) il nome della funzione
 - b) una lista di espressioni, una per ognuno dei parametri formali (**parametri attuali**)
 - c) parametri formali e espressioni corrispondenti devono essere dello stesso tipo

nome_F (espr₁, ..., espr_k)

Chiamata di funzione

- **Funzioni che calcolano un valore:**

può comparire ovunque può comparire
un'espressione dello stesso tipo di quello
assegnato alla funzione nella sua definizione

- **Esempio:**

`nome_variabile = nome_F (espr1, ..., esprk);`

con il tipo di `nome_variabile` coincidente con
quello della funzione

Chiamata di funzione

- **Esempio**

```
/* sorgente: DefChiamata.c */
/* definizione della funzione per la somma di 4 numeri interi */
#include <stdio.h>

int somma (int num1, int num2, int num3, int num4)
{
    /* restituisce la somma dei 4 interi */
    return(num1+num2+num3+num4);
}

int main()
{
    /* definizione delle variabili per i 4 valori e loro acquisizione */
    int n1, n2, n3, n4;
    printf("\nFornire i 4 valori: ");
    scanf("%d %d %d %d", &n1, &n2, &n3, &n4);
    /* chiamata della funzione e visualizzazione del valore calcolato */
    printf("\nLa somma e': %d", somma(n1, n2, n3, n4));
    return(0);
}
```

Modalità di passaggio dei parametri

- **Modalità di passaggio dei parametri:**
modalità in accordo alle quali i parametri attuali sono “legati” ai parametri formali
- **In modo informale (ma non troppo), per ognuno dei parametri formali:**
 1. si alloca memoria per il parametro formale
 2. si valuta il corrispondente parametro attuale
 3. il risultato di tale valutazione viene assegnato al corrispondente “parametro formale”
 4. si esegue il corpo della funzione
 5. si rilascia la memoria allocata per i “parametri formali”

Modalità di passaggio dei parametri

- **Corrispondenza tra parametri attuali e parametri formali:**

stabilita sulla base dell'ordine con cui questi compaiono nella chiamata e nella intestazione della funzione, rispettivamente

Modalità di passaggio dei parametri

- Esempio

```
int max (int N, int M)
{
  if (N > M)
    return(N);
  else
    return(M);
} ;
```

y = max((x+1), (z%4));

2834	1	x	x
2835		x	
2836	23	x	z
2837		x	
2838			
2839	3	x	y
2840		x	
2841			
2842	2		N
2843			
2844	3		M
2855			

Modalità di passaggio dei parametri

- Abbiamo un problema!!!

```
void Add (int N)
```

```
{  
  N = N+1;  
};
```

```
Add(x);
```

2834			
2835			
2836			
2837			
2838			
2839	1	x	x
2840		x	
2841			
2842	2		N
2843			
2844			
2855			

Abbiamo un problema ...

- **Esempio**

```
/* sorgente: PassParNo.c */  
/* esempio che dimostra come non sia possibile  
** modificare il valore dei parametri attuali */  
  
#include <stdio.h>  
  
/* definizione della funzione che vorrebbe aggiungere 1 ad una variabile */  
void add (int n)  
{  
    n++;  
};  
  
/* chiamante */  
int main()  
{  
    /* definizione e inizializzazione della variabile di prova */  
    int prova=1;  
    printf("\nValore prima della chiamata: %d", prova);  
  
    /* chiamata della funzione che dovrebbe aumentarne il valore e verifica  
    ** del suo effetto */  
    add(prova);  
    printf("\nValore successivo alla chiamata: %d", prova);  
    return(0);  
};
```

Modalità di passaggio dei parametri

- Ma abbiamo anche la soluzione!!!

```
void Add (int *N)  
{  
    *N = *N+1;  
};
```

```
Add(&x);
```

2834			
2835			
2836			
2837			
2838			
2839	2	x	x
2840		x	
2841			
2842	2839		N
2843			
2844			
2855			

E abbiamo la soluzione ...

• Esempio

```
/* sorgente: PassParSi.c */
/* modalita' di modifica dello stato della memoria tramite una funzione */
#include <stdio.h>
/* definizione della funzione che aggiunge 1 al valore di una variabile */
void add (int *n)
{
    /* aggiunge 1 alla variabile puntata dal parametro formale */
    *n=*n+1;
}
int main()
{
    /* definizione e inizializzazione della variabile di prova */
    int prova=1;
    printf("\nValore prima della chiamata: %d", prova);
    /* chiamata della funzione che ne aumenta il valore e verifica del suo effetto */
    add(&prova);
    printf("\nValore successivo alla chiamata: %d", prova);
    return(0);
}
```

Per definire l'intestazione di una funzione ...

- **Devo capire:**
 1. se la funzione calcola un valore e/o modifica lo stato della memoria
 2. se la funzione calcola un valore:
 - 2.1 qual è il tipo del valore calcolato
 - 2.2 quali e di che tipo sono i valori in ingresso a partire dai quali tale valore è calcolato
 3. se la funzione modifica lo stato della memoria:
 - 3.1 quali e di che tipo sono le variabili modificate

Per definire l'intestazione di una funzione ...

- **Esempio:**

```
/* sorgente: scambia.c */  
#include <stdio.h>  
/* funzione che scambia il contenuto di due variabili intere */  
void scambia (int *var1, int *var2)  
{  
    int temp;  
    temp = *var1;  
    *var1 = *var2;  
    *var2 = temp;  
}
```

Quando definisco una funzione ...

• Esempio:

```
/* chiamante */  
int main ()  
{  
    /* definizione e acquisizione delle variabili */  
    int A, B;  
    printf (“\nDammi il valore della I variabile: “);  
    scanf (“%d”, &A);  
    printf (“\nDammi il valore della II variabile: “);  
    scanf (“%d”, &B);  
    printf(“\nI variabile prima dello scambio: %d\n”, A);  
    printf(“II variabile prima dello scambio: %d\n”, B);  
  
    /* chiama la funzione che scambia le variabili e ne  
    ** verifica l’effetto */  
    scambia(&A, &B);  
    printf(“\nI variabile dopo lo scambio: %d\n”, A);  
    printf(“II variabile dopo lo scambio: %d\n”, B);  
    return(1);  
}
```

Quando definisco una funzione ...

- **Esempio:**

```
/* sorgente: somma2in1.c */  
#include <stdio.h>  
/* funzione che somma il contenuto di due variabili  
** in una terza variabile */  
void somma_in (int s1, int s2, int *dest)  
{  
    *dest = s1+s2;  
};
```

Quando definisco una funzione ...

- **Esempio:**

```
/* chiamante */  
int main ()  
{  
    /* definizione e acquisizione delle variabili */  
    int A, B, somma;  
    printf (“\nDammi il valore della I variabile: “);  
    scanf (“%d”, &A);  
    printf (“\nDammi il valore della II variabile: “);  
    scanf (“%d”, &B);  
    /* chiamata della funzione che somma i due input  
    ** in una terza variabile e verifica del suo effetto */  
    somma_in(A, B, &somma);  
    printf(“\nSomma: %d”, somma);  
    return(1);  
};
```