Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

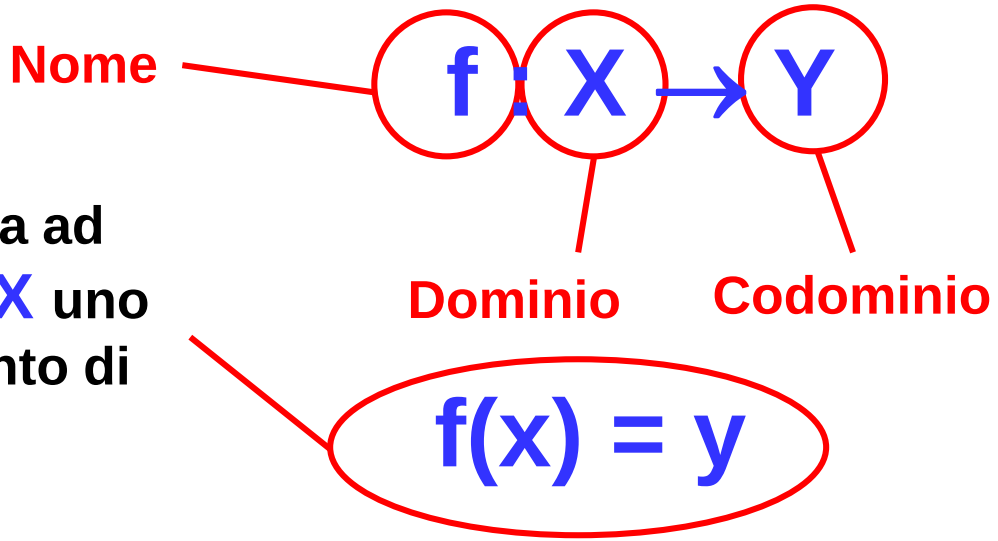
Programmazione e Laboratorio di Programmazione

Lezione VI
Le funzioni

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

Le funzioni della matematica

- **Definizione di una funzione:**



legge che assegna ad ogni elemento di X uno ed un solo elemento di Y

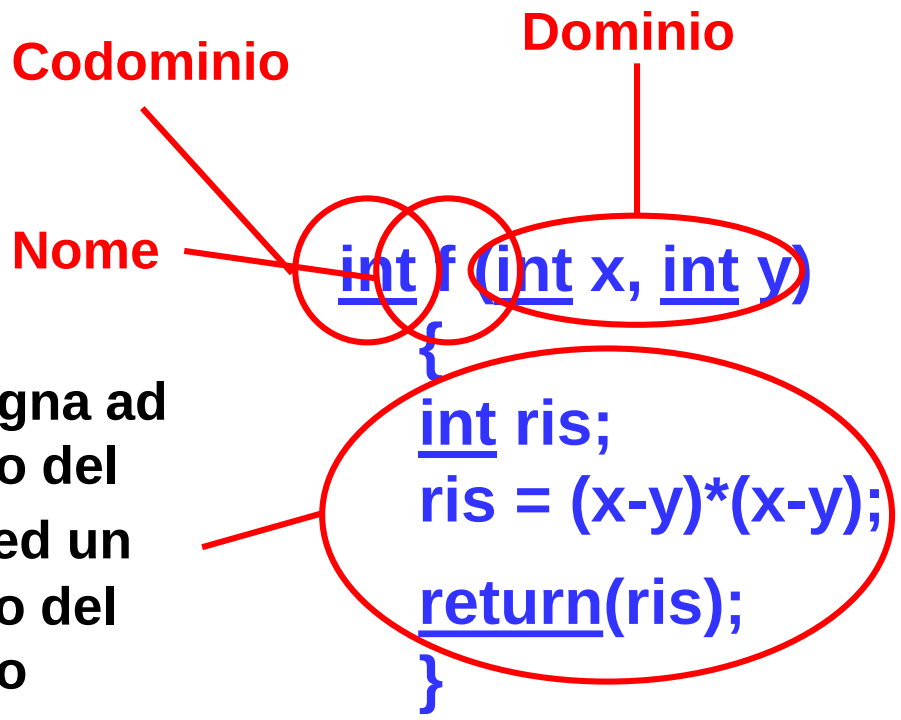
- **In matematica:**

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

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

Le funzioni del C

- **In C:**



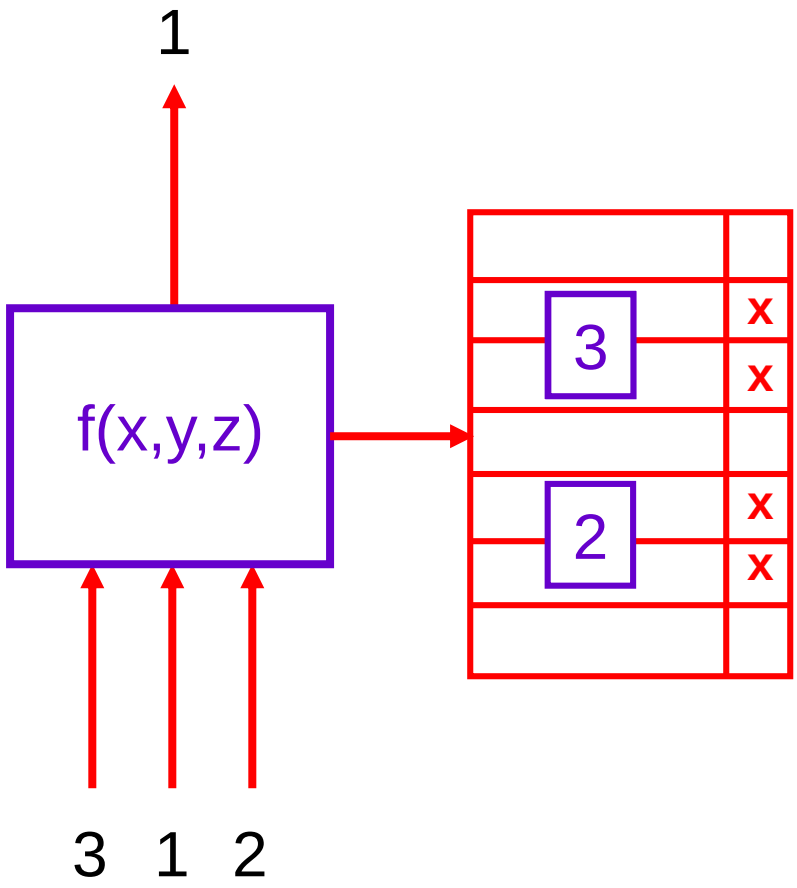
legge che assegna ad ogni elemento del dominio uno ed un solo elemento del codominio

- **Ma non c'è proprio nessuna differenza?**

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

La differenza …

- **Le funzioni del C:**
  - **possono calcolare valori**
  - **ma possono anche modificare lo stato della memoria**



Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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**

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

Definizione di funzione

- **Funzioni che calcolano un valore:**

$tipo_F\ nome_F\ (tipo_1\ par_1, \ldots, tipo_k\ par_k)$
{
…
return(espressione);
}

il valore di espressione è il valore "restituito " dalla funzione. Il suo tipo "deve " concidere con $tipo_F$

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

void $nome_F\ (tipo_1\ par_1, \ldots, tipo_k\ par_k)$
{
…
~~return(espressione);~~
}

corpo

intestazione

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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);
        }
```

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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**

$$nome_F\ (espr_1, \ldots, espr_k)$$

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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:**

  $$\text{nome\_variabile} = \text{nome}_F\ (\text{espr}_1, \ldots, \text{espr}_k);$$

  con il tipo di nome_variabile coincidente con quello della funzione

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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);
    }
```

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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”**

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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

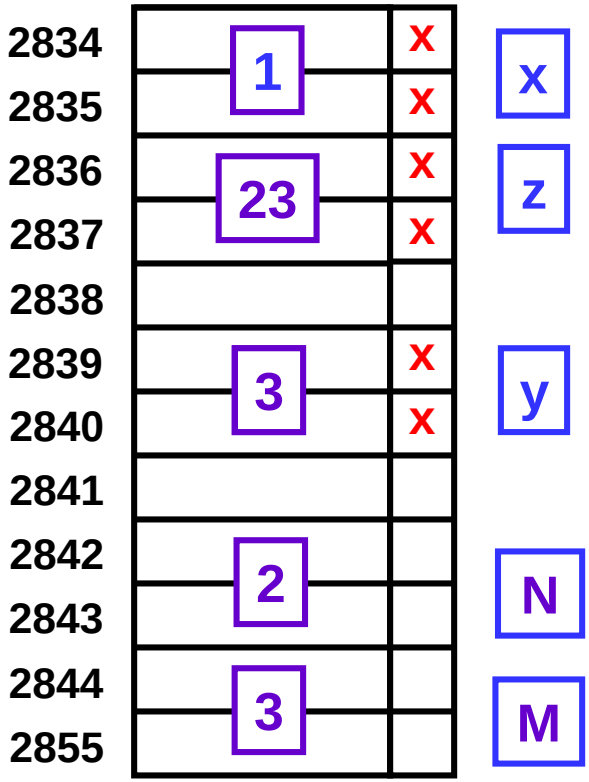
Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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);
```



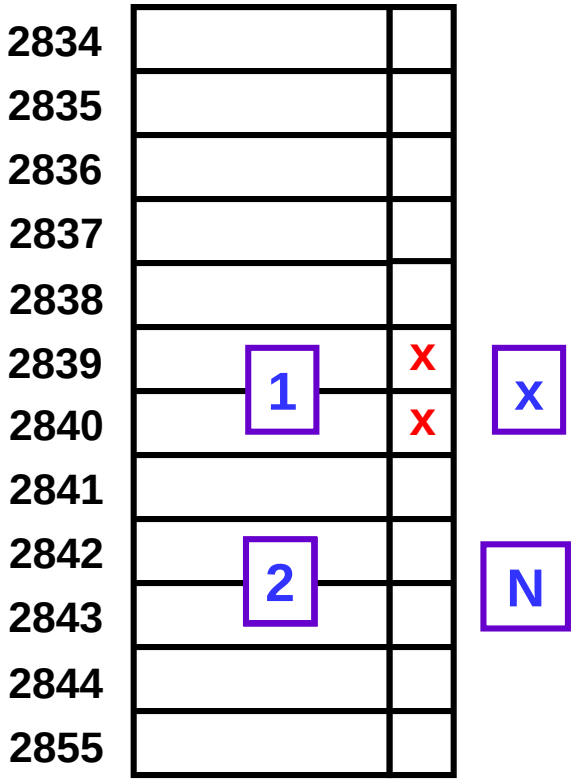
Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

Modalità di passaggio dei parametri

- **Abbiamo un problema!!!**

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

Add(x);
```



Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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);
  };
```

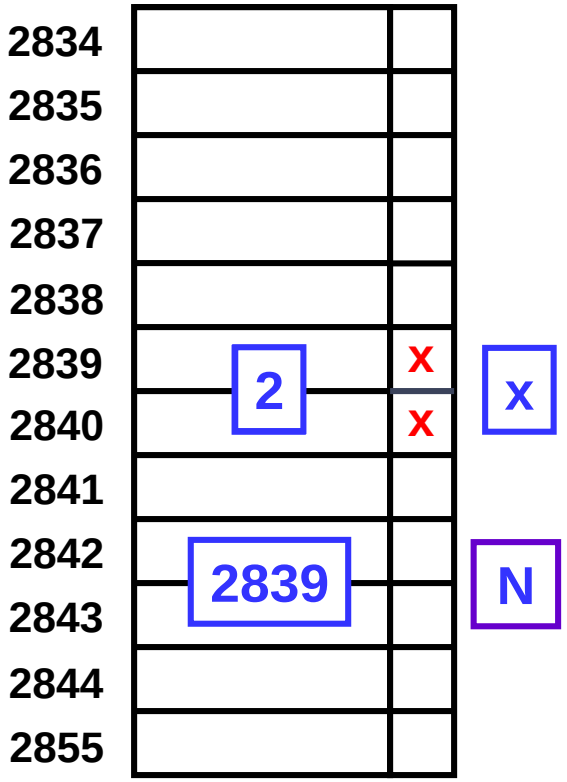
Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

Modalità di passaggio dei parametri

- **Ma abbiamo anche la soluzione!!!**

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

Add(&x);
```



Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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);
  }
```

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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**

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

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;
    }
```

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

Per definire l'intestazione di 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);
   }
```

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

Per definire l'intestazione di 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;
   };
```

Carlo Gaibisso
Bruno Martino
Marco
Pietrosanto

Per definire l'intestazione di 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);
   };
```