Ricorsione, Puntatori e Array

Quarto Laboratorio

16 DICEMBRE 2011
SCADENZA
TERZA ESERCITAZIONE

FUNZIONI RICORSIVE

ESERCIZIO

- Scrivere una funzione ricorsiva che, assegnati due interi N1 ed N2, restituisca la somma di tutti gli interi compresi tra N1 ed N2.

- Schema Ricorsione
  - Prima versione
    - F(N1, N2) = N1 se N1 = N2
    - F(N1, N2) = N1 + F(N1 + 1, N2) se N1 < N2
    - F(N1, N2) = F(N2, N1) se N1 > N2
  - Seconda versione
    - F(N1, N2) = N1 se N1 = N2
    - F(N1, N2) = N1 + N2 se N1 = N2 - 1
    - F(N1, N2) = N1 + F(N1 + 1, N2 - 1) + N2 se N1 < N2 - 1
    - F(N1, N2) = F(N2, N1) se N1 > N2

SOLUZIONE

```
int somma(int N1, int N2){
   if(N2 < N1) return somma(N2, N1);
  if(N1 == N2) return N1;
  return N1 + somma(N1 + 1, N2);
}

int main(){
  int N1 = getInput(), N2 = getInput();
  int somma = somma_ric(N1, N2);
  printf("Somma dei numeri tra N1 ed N2 = %d",
    somma);
}
```

ESERCIZIO

- Scrivere una funzione che riceve in input una sequenza di interi positivi (num negativo per terminare) e poi stampa a video i numeri inseriti in ordine inverso:

- Esempio:
  - 2 3 4 5 -3
  - Stampa
  - 5 4 3 2

SOLUZIONE

```
void rev(){
  int x;
  scanf("%d",&x);

  if(x<0) return;
  rev(x);
  printf("%d \n",x);
}
```

ESERCIZIO

- Scrivere un programma che tramite l'utilizzo di una funzione ricorsiva calcoli il MCD tra due numeri con il metodo delle sottrazioni successive

SOLUZIONE

```
int mcd (int x, int y) {
  if (x == y) return x;
  else if (x > y) return mcd (x-y, y);
  else return mcd (x, y-x);
}
```

ESERCIZIO

- Richiedi e controlla un numero intero strettamente positivo e trova i divisori non banali stampandoli a video

SOLUZIONE

```
void divisori(int n, int k){
    if(n <= 0 || k <= 0) return;
  if(k == 1) return;
  if(n != k && n % k == 0){
    printf("Un divisore %d", k);
    return divisori(k, k-1);
  }
  return divisori(n, k - 1);
}

divisori(n, n - 1){
```

ESERCIZIO

- Richiedi e controlla un numero intero strettamente positivo e trova se è un numero primo

PUNTATORI

I PUNTATORI

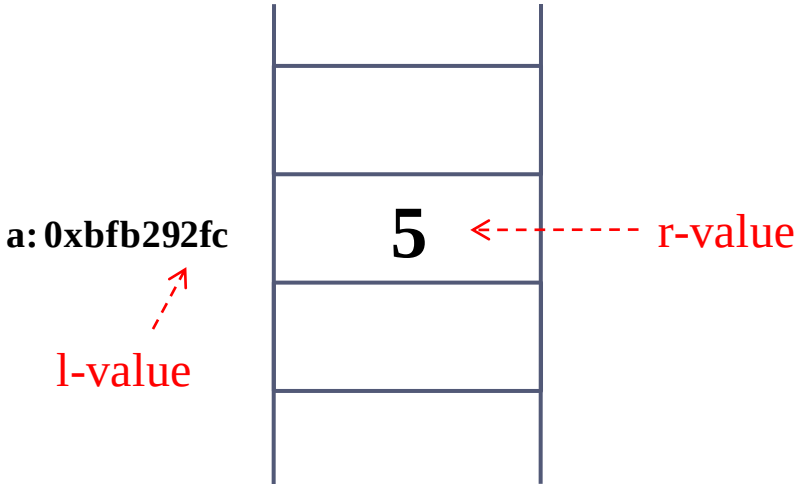
- *Il puntatore è un tipo di variabile che contiene l'indirizzo di memoria di un'altra variabile.*

- Esempio dichiarazione:
  - // puntatore (*****) ad una variabile di tipo intero (**int**)
  - int * p;
- Esempio inizializzazione:
  - int a = 5;
  - // inizializzo il puntatore **p** con l'indirizzo di memoria (**&**) di **a**
  - int * p = &a;

R-value e L-value

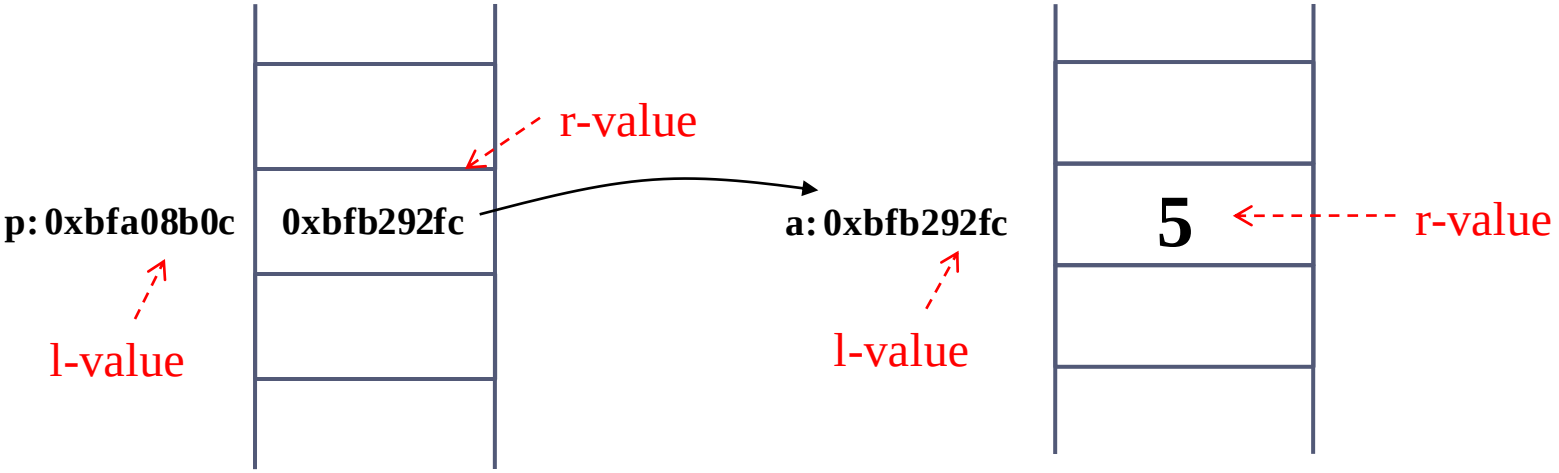
- r-value
  - il valore della variabile definito con l'inizializzazione, assegnamenti o non definito
- l-value
  - l'indirizzo dell'area di memoria allocata per la variabile

- Esempio:
  - int a = 5;



R-value e L-value con i puntatori

- Esempio:
  - `int a = 5;`
  - `int * p = &a;`



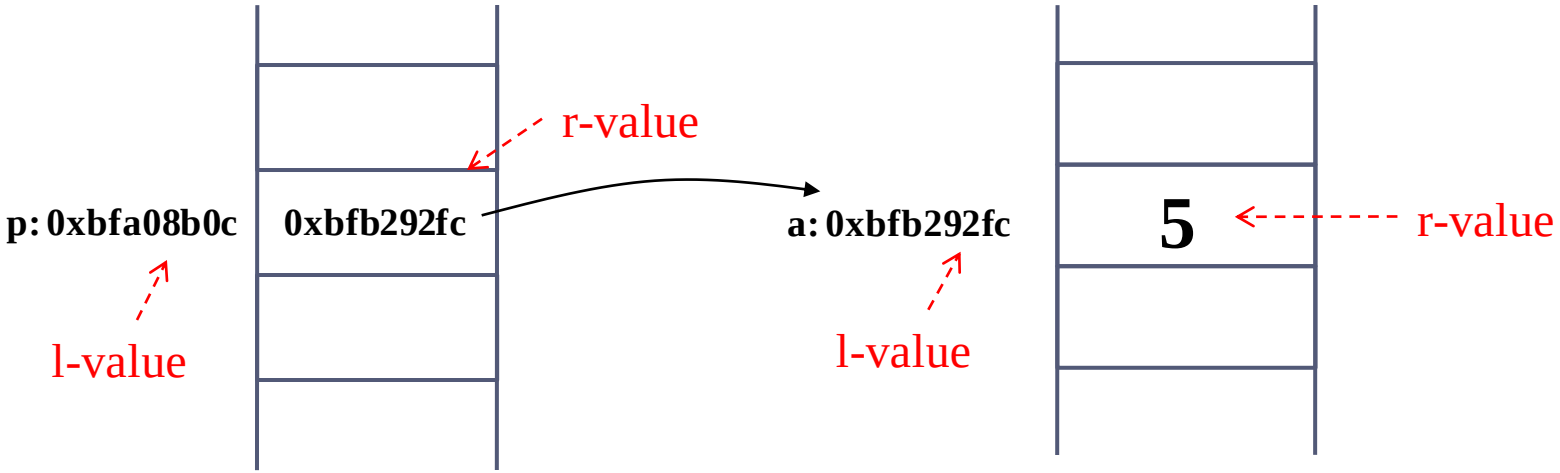
- `printf("r-value a = %d l-value &a = %p \n", a, &a);`
  - r-value a = 5 l-value &a = 0xbfb292fc
- `printf("r-value p = %p l-value &p = %p \n", p, &p);`
  - r-value p = 0xbfb292fc l-value &p = 0xbfa08b0c

OSSERVAZIONI

- Ad un puntatore si assegna **SOLO** un **INDIRIZZO di MEMORIA**
  - `int a = 5;`
  - `int * p = a;` **// ERRORE**
  - *In function 'main':*
  - *warning: initialization makes pointer from integer without a cast*
- Ad un puntatore si assegna **SOLO** un **INDIRIZZO di MEMORIA** di una VARIABILE dello **STESSO TIPO**
  - float a = 5;
  - int * p = &a; **// ERRORE**
  - *In function 'main':*
  - *warning: initialization from incompatible pointer type*

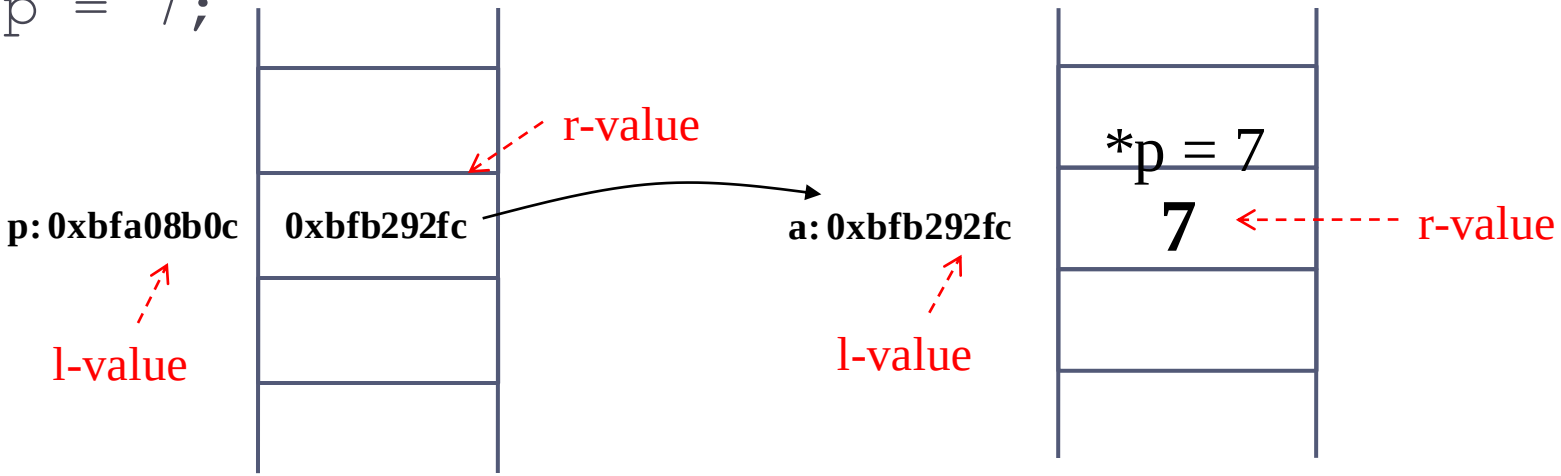
MANIPOLARE I PUNTATORI (1)

- Attraverso l'operatore di dereferenziazione (*) si può accedere all'area di memoria della variabile puntata e manipolarne il valore
- Esempio:
  - `int a = 5;`
  - `int * p = &a;`



MANIPOLARE I PUNTATORI (2)

- Attraverso l'operatore di dereferenziazione (*) si può accedere all'area di memoria della variabile puntata e manipolarne il valore
- Esempio:
  - `int a = 5;`
  - `int * p = &a;`
  - `*p = 7;`

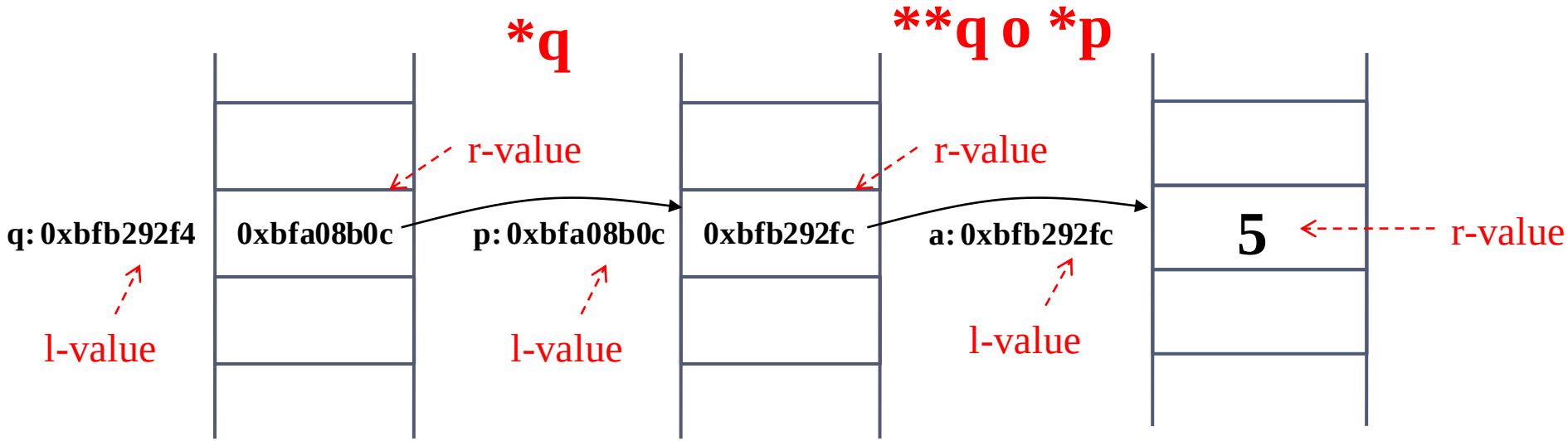


MANIPOLARE I PUNTATORI (3)

- Eseguendo le stampe del contenuto delle variabili a e p, prima e dopo l'istruzione *p = 7 si otterranno
- Prima di *p = 7;
  - printf("r-value a = %d l-value &a = %p \n", a, &a);
    - r-value a = 5 l-value &a = 0xbfb292fc
  - printf("r-value p = %p l-value &p = %p \n" *p = %d, p, &p, *p);
    - r-value p = 0xbfb292fc l-value &p = 0xbfa08b0c *p = 5
- Dopo *p = 7;
  - printf("r-value a = %d l-value &a = %p \n", a, &a);
    - r-value a = 7 l-value &a = 0xbfb292fc
  - printf("r-value p = %p l-value &p = %p \n" *p = %d, p, &p, *p);
    - r-value p = 0xbfb292fc l-value &p = 0xbfa08b0c *p = 7

Più Livelli di Dereferenziazione (1)

- Esempio
  - `int a = 5;`
  - `int  * p = &a;`
  - `int ** q = &p;`



Più Livelli di Dereferenziazione (2)

- Stampando a video le variabili si ottiene:
  - printf("r-value a = %d l-value &a = %p \n", a, &a);
    - *r-value a = 5 l-value &a = 0xbfb292fc*
  - printf("r-value p = %p l-value &p = %p *p = %d \n", p, &p, *p);
    - *r-value p = 0xbfb292fc l-value &p = 0xbfb292f8, *p = 5*
  - printf("r-value q = %p l-value &q = %p *q = %p **q = %d \n", q, &q, *q, **q);
    - *r-value q = 0xbfb292f8 l-value &q = 0xbfb292f4, *q = 0xbfb292fc, **q = 5*

ESERCIZIO

```
main (){
        int a = 5;
        int* p = &a, **q = &p;
        printf ("r- value a = %d l- value &a = %p \n", a, &a);
        printf ("r- value p = %p l- value &p = %p, *p = %d \n", p, &p, *p);
        printf ("r- value q = %p l- value &q = %p, *q = %p **q = %d \n", q,
        &q, *q, **q);

        **q = *p + 1;
        printf ("r- value a = %d l- value &a = %p \n", a, &a);
        printf ("r- value p = %p l- value &p = %p \n", p, &p);
        printf ("r- value q = %p l- value &q = %p, *q = %p **q = %d \n", q,
        &q, *q, **q);
}
```

Array

ARRAY STATICI

- *Un array è una collezione di oggetti dello stesso tipo, molto utile per memorizzare un insieme di elementi in memoria*.

- Esempio dichiarazione:

```
int A [10];

const int N = 10;
int AA[N];

int B[] = {1, 2, 3}; // B ha 3 elementi,

int C[3] = {1, 2, 3}; // C ha 3 elementi

int D[5] = {1, 2, 3}; // D ha 5 elementi
```

OSSERVAZIONI

- Se viene specificata, la **DIMENSIONE** di un array deve essere **MAGGIORE** o **UGUALE** al **NUMERO** di **ELEMENTI** inseriti **nell'INIZIALIZZAZIONE**
  - **int E[3] = {1, 2, 3, 4}; // ERRORE**

  In function 'main':
  warning: excess elements in array initializer
  warning: (near initialization for 'E')

- **NON** è possibile dichiarare un array definendone la **DIMENSIONE** attraverso un **INTERO**
  - **int N = 10;**
  - **int E[N];**

Come si Accede agli Elementi di un Array

- L'operatore di *indicizzazione* o *subscripting* ([]) permette di accedere all'area di memoria in cui è memorizzato l'elemento desiderato

- Esempio:

```
main (){
   int A [10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
   printf ("Il primo elemento dell 'array `e %d \n", A [0]) ;
}
```

OSSERVAZIONE

- Un array di dimensione **N** è **INDICIZZATO** da **0 a N-1**
- Fare molta attenzione a **NON INDICIZZARE** l'array in una **POSIZIONE NON DEFINITA**. Il linguaggio non fornisce alcun controllo dell'indice nè a tempo di compilazione, nè a tempo di esecuzione.
- Esempio:

```
int A[2] = {1, 2};
A[3] = 12;
```

- Un programmatore potrebbe superare i limiti di memoria di un array, senza causare errori, ma con un'esecuzione errata del programma.

ESERCIZIO: CICLO FOR E ARRAY

- Scrivere un programma che dichiari un array di 10 elementi e richieda all'utente di inserirli da tastiera.
- Stampi a video gli elementi contenuti nell'array sia nell'ordine di inserimento, sia nell'ordine inverso

Ciclo For e Array (1)

- *dichiarare un array di 10 elementi, senza inizializzarlo*

```
int a[10];
```

- *richiedere all'utente di inserire da tastiera i 10 elementi e riempire con questi l'array*

```
int i;
// l'array è indicizzato da 0 a 9
for (i=0; i<10; i++){
        printf (" Inserisci il valore di a[%d]:",i);
        scanf ("%d", &a[i]);
}
```

Ciclo For e Array (2)

- *mostrare a video i valori memorizzati sia nell'ordine di inserimento sia in quello inverso*
  - `// ordine di inserimento`

```
for (i=0; i<10; i++)
        printf ("a[%d]=% d ",i, a[i]);
printf ("\n");
```

  - // ordine inverso di inserimento

```
for (i=9; i >=0; i --)
        printf ("a[%d]=% d ",i, a[i]);
printf ("\n");
```

Ciclo For e Array (3)

```
main (){
     int a[10];
      int i;
     for (i=0; i<10; i++){
           printf (" Inserisci il valore di a[%d]:",i);
           scanf ("%d", &a[i]);
     }
     for (i=0; i<10; i++)
           printf ("a[%d]=% d ",i, a[i]);
     printf ("\n");
     for (i=9; i >=0; i --)
           printf ("a[%d]=% d ",i, a[i]);
     printf ("\n");
}
```

ESERCIZIO: CONVERSIONE IN BINARIO

- *Scrivere un programma che legga in input un valore positivo e ne stampi la sua conversione in binario.*

ESERCIZIO: CONVERSIONE IN BINARIO (1)

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

main (){
// dichiarazione numero intero
        int num ;

// lettura numero intero positivo
     do{
            printf (" inserisci un numero >0 :");
            scanf ("%d", &num);
     } while (num <0);
```

ESERCIZIO: CONVERSIONE IN BINARIO(2)

```
// dichiarazione e inizializzazione variabili ausiliarie
int i;
int cifre_usate = 0;
short binario [N];

// calcolo del numero binario
do{
    binario [ cifre_usate ] = num %2;
    cifre_usate ++;
    num = num / 2; // num /=2;
} while (num !=0) ;
```

ESERCIZIO: CONVERSIONE IN BINARIO(3)

```
printf ("Il valore binario e ': ");
for(i= cifre_usate -1; i >=0; i --){
    printf ("%d", binario [i]);
}
printf ("\n");
}
```

ESERCIZIO: ARRAY E FUNZIONI

- Dato un array scrivere due funzioni
  - una funzione che calcola il valore massimo nell'array,
  - una funzione che calcola la posizione del massimo nell'array

- Prototipi

```
int max_array(int* a, int lung);
int max_ind(int* a, int lung);
```

Array e Funzioni (1)

```
# include <stdio .h>

int max_array(int* a, int lung);
int max_ind(int* a, int lung);

main(){
    int arr[]={3,5,2,8,56,4}; //sono 6
    printf("il massimo e' %d\n",
max_array(arr, 6));
    printf("il massimo e' %d\n",
arr[max_ind(arr, 6)]);
}
```

Array e Funzioni (2)

```
// si devono passare l'array a e la sua lunghezza lung
int max_array(int* a, int lung){
     // inizializzo con il primo elemento dell'array
     int max = a[0]; // *a;

     // scorro tutti gli elementi dell'array
     int i;
     for(i=1; i<lung;i++) {
            if(*(a+i) > max) //if(a[i] > max)
                  max = *(a+i); // max=a[i];
     }

     // ritorno il massimo trovato
     return max;
}
```

Array e Funzioni (3)

```
//ritorna l'indice del massimo nell'array a
int max_ind(int* a, int lung){
     // indice inizializzato al primo elemento dell'array
     int imax = 0;

     // scorro tutti gli elementi dell'array
     int i;
     for(i=1; i<lung;i++){
            if(*(a+i) > a[imax]) //if(a[i] > *(a+imax))
                  imax = i; // max=a[i];
     }

     // ritorno l'indice trovato
     return imax;
}
```

ARRAY E FUNZIONI (4)

```
//ritorna il puntatore al massimo
nell'array a
int* max_pun(int* a, int lung){
    // puntatore inizializzato al primo
elemento dell'array
    int * pmax = a;

    // scorro tutti gli elementi dell'array
    int* p;
    for(p=a; p<a+lung;p++){
        if(*p > *pmax)
            pmax = p;
    }

    // ritorno il puntatore trovato
    return pmax;
}
```

Quarta Esercitazione

Quarta Consegna

- Consegna dovrà essere eseguita entro il giorno 27 dicembre 2011
- Le consegne effettuate successivamente non verranno considerate...

- Comando per la consegna:

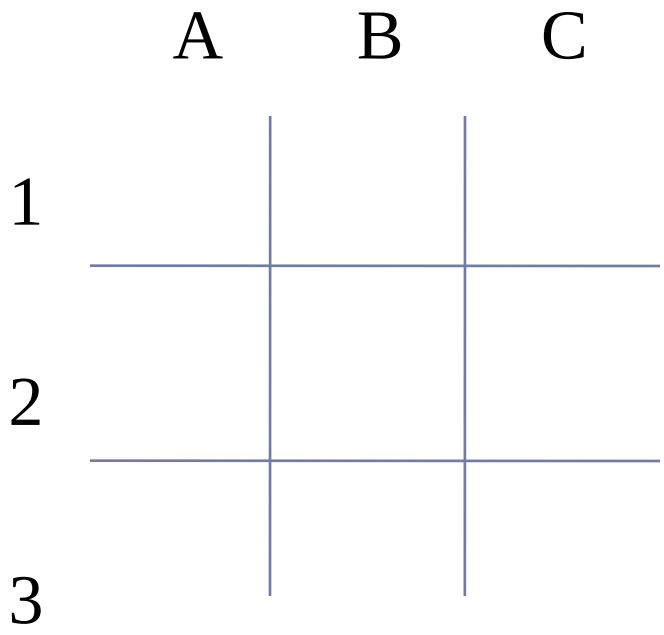
consegna consegna4

ESERCIZIO 1

- Scrivere un programma che riceve in input un intero N grande, e stampa a video tutti i numeri perfetti compresi tra 0 e N.
- Un numero si dice **perfetto** quando è uguale alla somma dei suoi divisori propri.

ESERCIZIO 2

- Tris-game: 2 giocatori
  - Si stampa a video lo schema del gioco del tris:



ESERCIZIO 2

- Si chiedono da tastiera i nomi di due giocatori
  - es. U e D
- Si chiedono i relativi simboli che utilizzeranno
  - es simbolo “o” per U e simbolo “x” per D
- A turno i giocatori inseriscono le coordinate della cella dove vogliono mettere il proprio simbolo
  - es per U la giocata vale A 1
- Si stampa a video lo schema del gioco aggiornato
- Così via finchè uno dei due giocatori non realizza un tris per riga, colonna o diagonale, vincendo.
- Il programma deve controllare che le celle inserite dai due giocatori esistano e siano vuote.

ESERCIZIO 3

- **Versione easy**:

  Scrivere un programma che riceva in input una stringa e tramite l'uso di una funzione ricorsiva, stampi a video se la stringa è palindrome.

- **Versione less easy**:

  Scrivere un programma che riceva in input una stringa e che tramite l'uso di una funzione ricorsiva (o più), stampi a video la stringa palindrome di massima lunghezza contenuta nella stringa.