IEIM

Esercitazione III

2014-2015

Alessandro A. Nacci
alessandro.nacci@polimi.it - alessandronacci.it

Nuovo sito:
alessandronacci.it

Nuovo indirizzo mail:
alessandro.nacci@polimi.it


READY...
SET...
GO

Esercizio 1

LEZIONE PRECEDENTE

- Scrivere un programma che
  - legga due array di interi da tastiera
  - dica quale dei due array ha valor medio più alto
  - calcoli l'array concatenato tra i due array in ingresso
  - trovi tutti i numeri primi inseriti
  - trovi il massimo ed il minimo tra tutti i valori inseriti
  - dica in quale dei due array sono presenti i due valori di massimo e minimo
  - calcoli l'array riversato

In questa soluzione, non usare le funzioni!

Esercizio 1: dichiarazione variabili

LEZIONE PRECEDENTE

```
int i,j;
int a[DIM];
int b[DIM];

int ab[DIM2];
int ab_r[DIM2];

float medio_a, medio_b;
float somma = 0;
bool primo = 1;              //booleana
```

Esercizio 1: lettura array

LEZIONE PRECEDENTE

```
// LETTURA VALORI
// -------------------------------------

// Leggo l'arrayA
for (i = 0; i < DIM; i++){
     printf("\nInserisci i valore %d per l'array a: ", i);
     scanf("%d",&a[i]);
}

// Stampo l'array A
printf("\narray a = ");
for (i = 0; i < DIM; i++)
     printf("%d ", a[i]);

// Leggo l'array B
for (i = 0; i < DIM; i++){
     printf("\nInserisci i valore %d per l'array b: ", i);
     scanf("%d",&b[i]);
}

// Stampo l'array B
printf("\narray b = ");
for (i = 0; i < DIM; i++)
     printf("%d ", b[i]);
```

Esercizio 1: calcolo del valor medio

LEZIONE PRECEDENTE

```
// VALOR MEDIO
// ------------------------------------

// Calcolo valor medio si A

for (i = 0; i < DIM; i++){
    somma += a[i];
}
medio_a = somma / DIM;
```

```
// Calcolo valor medio si B
somma = 0;
for (i = 0; i < DIM; i++){
    somma += b[i];
}
medio_b = somma / DIM;

if(medio_a == medio_b)
    printf("L'array a e l'array b hanno lo stesso valor medio.\n");
else if (medio_a > medio_b)
    printf("L'array a ha valor medio piu' alto dell'array b.\n");
else if (medio_a < medio_b)
    printf("L'array b ha valor medio piu' alto dell'array a.\n");
```

Esercizio 1: concatenazione

LEZIONE PRECEDENTE

```
// CONCATENAZIONE
// ----------------------------------

for (i=0; i < DIM; i++)
     ab[i] = a[i];

for (i=0; i < DIM; i++)
     ab[i+DIM] = b[i];


// Stampo l'array AB
printf("\narray a.b = ");
for (i = 0; i < DIM2; i++){
     printf("%d ", ab[i]);
}

printf("\n");
```

Esercizio 1: numeri primi

```
// NUMERI PRIMI
// ------------------------------------

printf("I numeri primi inseriti sono: ");

for (i = 0; i < DIM2; i++){

    primo = 1; // Inizializzo la variabile

    for (j = 2; j < ab[i]; j++)
        if ((ab[i] % j) == 0 ){
            primo = 0;
            break;
        }

    if (primo)
        printf("%d ", ab[i]);
}

printf("\n");
```

Esercizio 1: massimo e minimo

```
// MAX e MIN
// ---------------------------------

int max = 0;
int min = 2147483647;   //2^(n-1) - 1 :::: CPL2 max value
                              //n = 32 poiché int è rappresentato
                              //con 32 bit.

for (i = 0; i < DIM2; i++){

      if (ab[i] > max)
            max = ab[i];

      if (ab[i] < min)
            min = ab[i];

}

printf("MIN = %d\n", min);
printf("MAX = %d\n", max);
```

Esercizio 1: massimo e minimo

```
// TROVARE DOVE SONO MAX e MIN
// -----------------------------------


bool max_in_array_a = 0;
bool min_in_array_a = 0;

for (i = 0; i < DIM; i++){
    if (a[i] == max)
        max_in_array_a = 1;

    if(a[i] == min)
        min_in_array_a = 1;
}

printf("\n");

if (max_in_array_a)
    printf("Il valore massimo %d e' nell'array a", max);
else
    printf("Il valore massimo %d e' nell'array b", max);

printf("\n");

if (min_in_array_a)
    printf("Il valore minimo %d e' nell'array a", min);
else
    printf("Il valore minimo %d e' nell'array b", min);


printf("\n");
```

||

Esercizio 1: array riversato

```
// ARRAY RIVERSATO
// ---------------------------------

for (i = 0; i < DIM2; i++)
     ab_r[DIM2 - i - 1] = ab[i];
```

```
printf("\narray (a.b)_r = ");
for (i = 0; i < DIM2; i++){
     printf("%d ", ab_r[i]);
}

printf("\n");
printf("\n");
```

MATRICI E FUNZIONI

Il passaggio di una matrice ad una funzione e' sempre per indirizzo e mai per copia.

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo1(int *mat)
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  *(mat+i*C+j) = (i+j) * 2;
}
```

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

Puntatore ad intero!

```c
void foo1(int *mat)
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  *(mat+i*C+j) = (i+j) * 2;
}
```

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo1(int *mat)
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  *(mat+i*C+j) = (i+j) * 2;
}
```

Puntatore ad intero!

Calcolo manuale dello spiazzamento

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo1(int *mat)
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  *(mat+i*C+j) = (i+j) * 2;
}
```

Puntatore ad intero!

Calcolo manuale dello spiazzamento

Valore contenuto all'indirizzo tra parentesi

```c
*(mat+i*C+j)
```

INDIRIZZO

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);


int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo2(int mat[R][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 3;
}
```

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);


int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

Esplicito che il puntatore punta al primo elemento di una matrice RxC

```c
void foo2(int mat[R][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 3;
}
```

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo2(int mat[R][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 3;
}
```

Esplicito che il puntatore punta al primo elemento di una matrice RxC

Accesso comodo alla matrice!

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
     int mat[R][C];
     foo1(mat);
     foo2(mat);
     foo3(mat);
}
```

```c
void foo3(int mat[][C])
{
     int i, j;
     for(i=0;i<R;i++)
          for(j=0;j<C;j++)
               mat[i][j] = (i+j) * 4;
}
```

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
     int mat[R][C];
     foo1(mat);
     foo2(mat);
     foo3(mat);
}
```

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

```c
void foo3(int mat[][C])
{
     int i, j;
     for(i=0;i<R;i++)
          for(j=0;j<C;j++)
               mat[i][j] = (i+j) * 4;
}
```

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo3(int mat[][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 4;
}
```

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

Accesso comodo alla matrice!

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo3(int mat[][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 4;
}
```

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

Accesso comodo alla matrice!

NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

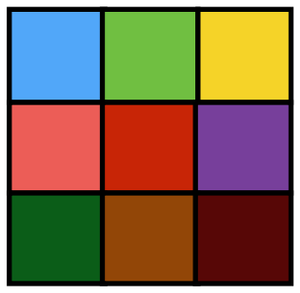
Esercizio 2: Matrici e Funzioni

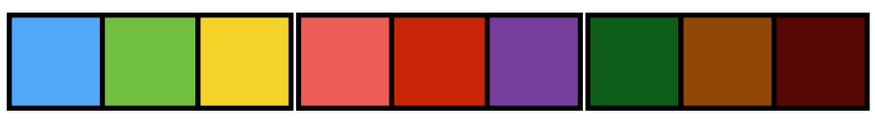
```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo3(int mat[][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 4;
}
```

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

Accesso comodo alla matrice!





NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

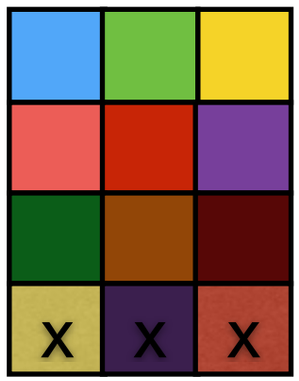
Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo3(int mat[][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 4;
}
```

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

Accesso comodo alla matrice!



NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

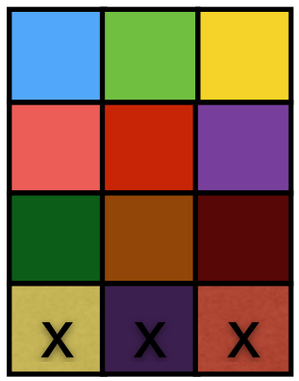
Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

```c
void foo3(int mat[][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 4;
}
```

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

Accesso comodo alla matrice!



NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

Esercizio 2: Matrici e Funzioni

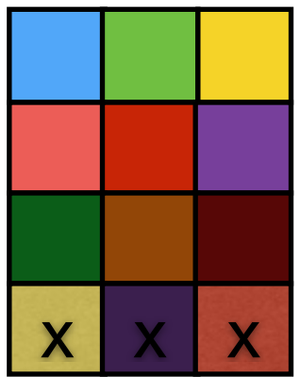
```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

Il valore di mat cambia

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

```c
void foo3(int mat[][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 4;
}
```

Accesso comodo alla matrice!



NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

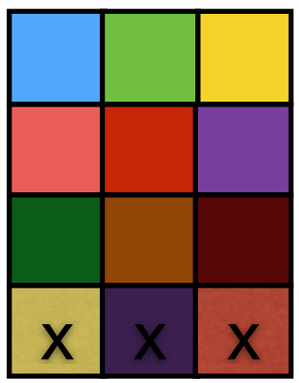
Il valore di mat cambia

Il valore di mat cambia

```c
void foo3(int mat[][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 4;
}
```

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

Accesso comodo alla matrice!



NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

Esercizio 2: Matrici e Funzioni

```c
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
      int mat[R][C];
      foo1(mat);
      foo2(mat);
      foo3(mat);
}
```

Il valore di mat cambia

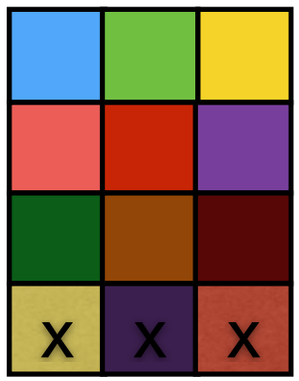
Il valore di mat cambia

Il valore di mat cambia

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

```c
void foo3(int mat[][C])
{
      int i, j;
      for(i=0;i<R;i++)
            for(j=0;j<C;j++)
                  mat[i][j] = (i+j) * 4;
}
```

Accesso comodo alla matrice!



NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

Esercizio 3 (a)

- Quale è l'ouput del seguente codice?

```
#include <stdio.h>

   typedef enum{bianco,azzurro_chiaro,giallo,rosso,verde_scuro,rosa,
      azzurro_scuro,verde_chiaro,nero,marrone} colore;

int main(){



   int auto_codice[3];          //ATTENZIONE QUI!
   colore auto_colore[3];       // tipo_di_dato nome_varibile[]
   int i;

   auto_codice[0] = 1000;
   auto_colore[0] = giallo;

   auto_codice[1] = 4000;
   auto_colore[1] = rosa;

   auto_codice[2] = 8000;
   auto_colore[2] = nero;



   for (i = 0; i < 3; i++ )
      printf("L'auto con codice %d e' di colore %d\n",
         auto_codice[i], auto_colore[i]);

   return 0;
}
```

Esercizio 3 (a)

- Quale è l'ouput del seguente codice?

```c
#include <stdio.h>

   typedef enum{bianco,azzurro_chiaro,giallo,rosso,verde_scuro,rosa,
      azzurro_scuro,verde_chiaro,nero,marrone} colore;

int main(){



   int auto_codice[3];          //ATTENZIONE QUI!
   colore auto_colore[3];       // tipo_di_dato nome_varibile[]
   int i;

   auto_codice[0] = 1000;
   auto_colore[0] = giallo;

   auto_codice[1] = 4000;
   auto_colore[1] = rosa;

   auto_codice[2] = 8000;
   auto_colore[2] = nero;



   for (i = 0; i < 3; i++ )
      printf("L'auto con codice %d e' di colore %d\n",
         auto_codice[i], auto_colore[i]);

   return 0;
}
```

```
Terminal — bash — 80x24
> ./es3
> L'auto con codice 1000 e' di colore 2
> L'auto con codice 4000 e' di colore 5
> L'auto con codice 8000 e' di colore 8
```

Esercizio 3 (a)

- Quale è l'ouput del seguente codice?

```
#include <stdio.h>

   typedef enum{bianco,azzurro_chiaro,giallo,rosso,verde_scuro,rosa,
      azzurro_scuro,verde_chiaro,nero,marrone} colore;

int main(){



   int auto_codice[3];          //ATTENZIONE QUI!
   colore auto_colore[3];       // tipo_di_dato nome_varibile[]
   int i;

   auto_codice[0] = 1000;
   auto_colore[0] = giallo;

   auto_codice[1] = 4000;
   auto_colore[1] = rosa;

   auto_codice[2] = 8000;
   auto_colore[2] = nero;



   for (i = 0; i < 3; i++ )
      printf("L'auto con codice %d e' di colore %d\n",
         auto_codice[i], auto_colore[i]);

   return 0;
}
```

```
Terminal — bash — 80x24
> ./es3
> L'auto con codice 1000 e' di colore 2
> L'auto con codice 4000 e' di colore 5
> L'auto con codice 8000 e' di colore 8
```

MA QUINDI A COSA SERVE L'ENUM?

Solo a semplificare la vita al programmatore! Non ti devi ricordare la corrispondenza numerica!a

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

DATO UN PROGRAMMA C, COME SI COMPORTA LA MEMORIA?

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO
9	

ATTENZIONE!

Il comportamento della memoria mostrato in questo esercizio non è del tutto coerente con quanto avviene su un reale calcolatore. L'esempio mostrato è però funzionale alla spiegazione del comportamento dei puntatori in C.

```
}
```

DATO UN PROGRAMMA C, COME SI COMPORTA LA MEMORIA?

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

DATO UN PROGRAMMA C, COME SI COMPORTA LA MEMORIA?

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	~	b
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	~	mat
5	~	mat
6	~	mat
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	~	mat
5	~	mat
6	~	mat
7	~	c
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	~	c
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	5	c
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	5	c
8	3	d
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	5	c
8	3	d
9	~	e

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```


That's all Folks!

Tutte il materiale sarà disponibile sul mio sito internet!

alessandronacci.it