Titolo nota 20/10/2015

PUNTATORI

- Per dichiarare un puntatore usiamo la sintassi

  T * int *p; char *q; float *p1;

- Per denotare l'indirizzo di una variabile usiamo l'operatore &

  &x è l'indirizzo (la locazione) di x

  quindi se x è di tipo T, &x è un valore (indirizzo) di tipo T*

  ```
  int *p; int x;
  p = &x;
  ```
  ← è un assegnamento lecito

- Per denotare la variabile PUNTATA da p (di tipo T*) usiamo l'operatore *

  quindi se p è di tipo T*, *p è una variabile di tipo T

  ```
  int *p; int x;
  p = &x;
  *p = 10
  ```
  ← assegna il valore 10 alla variabile x

Titolo nota 20/10/2015

In `T *p;` T può essere a sua volta `T' *`

```
int x;
int *p;      → p può avere come valori INDIRIZZI di variabili intere
int **q;     → q "  "  "  "  "  di puntatori a variabili intere
```

`T *p;` → `int *p;`

`T *q;` → `int **q;`

```
p = &x;
q = &p;
*p = 10;
*(*q) = 50;
```

q	l_3
p	l_2
x	l_1

ℓ

l_1	~~10~~ ? ~~50~~
l_2	l_1
l_3	l_2

μ

Titolo nota 20/10/2015

Per ora abbiamo visto che a variabili di tipo puntatore possiamo assegnare indirizzi di altre variabili (mediante &)

```
int x = 10;   int y = 20;
int *p;       int *q;
p = &x;       q = &y;
```

```
x = y;
p = q;
```

```
y = *p + *q;
    ↓     ↓
  (è y) (è y)
```

q	ℓ_4
p	ℓ_3
y	ℓ_2
x	ℓ_1

ρ

ℓ_4	ℓ_2
ℓ_3	~~ℓ_1~~ ℓ_2
ℓ_2	20
ℓ_1	~~10~~ 20

μ

No! `y = p + q;` è un'espressione di tipo `int *` (in aritmetica dei puntatori)

Titolo nota 20/10/2015

ARRAY (tabelle)

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

dimensione dell'array

In `a[exp]`

indice

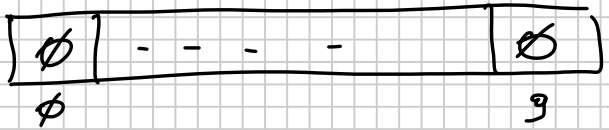
10 elementi, 10 variabili di tipo int

Ciascuna è individuata da NOME dell'array e (tra parentesi quadre) da un INDICE (una espressione di tipo int che DEVE avere un valore compreso nell'intervallo $[0, 9]$ $[0, 10)$

`a[exp]` il valore di exp deve appartenere a $[0, 10)$.

Titolo nota 20/10/2015

Voglio scrivere una porzione di codice che dichiara un array e inizializza tutti i suoi elementi al valore 0




```
int a[10];
a[0] = 0;
a[1] = 0;
  ⋮
a[9] = 0;
```

≈

```
int a[10];
int i = 0;
while (i < 10)
  { a[i] = 0;
    i = i+1;
  }
```

La dimensione di un array deve essere una COSTANTE

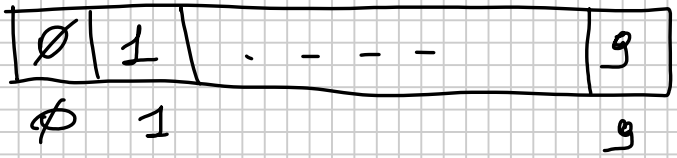
Non è lecita una sequenza di dichiarazioni fatta così:

```
int x = 10;
int a[x];
```

NO !!

Titolo nota 20/10/2015

Assegnare ad ogni elemento di un array il valore del suo indice




```
int a[10];
int i = 0;
while (i < 10)
  { a[i] = i;
    i = i+1;
  }
```

COSTRUTTO FOR

```
for (i=0; i<10; i=i+1)
    a[i] = i;
```

primo valore

da 1 in 1

"fino a 10 escluso"

i++ equivale a i=i+1;

Titolo nota 20/10/2015

Stampare il quadrato dei primi N numeri

```
int i;
for (i=0; i<N+1; i=i+1)
    printf("%d\n", i*i);
```

ITERAZIONE DETERMINATA

for

```
int x;
int i;
⋮
for (i=0; i<x; i++)
    { corpo }
```

ITERAZIONE INDETERMINATA

while

Titolo nota 20/10/2015

SINTASSI del FOR

```
for ( C1 ; exp ; C2 )
      C3
```

$\approx\approx$

```
C1 ;
while (exp)
   { C3
     C2
   }
```

```
for ( i = a ; i < b ; i++ )
      corpo (dipende da i)
```

"per tutti i valori di i nell'intervallo [a, b) esegui il corpo"

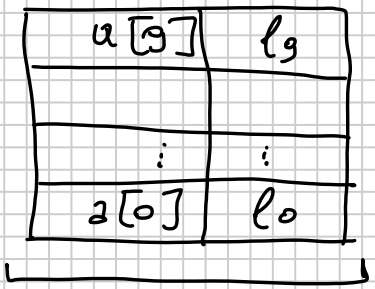
Titolo nota 20/10/2015

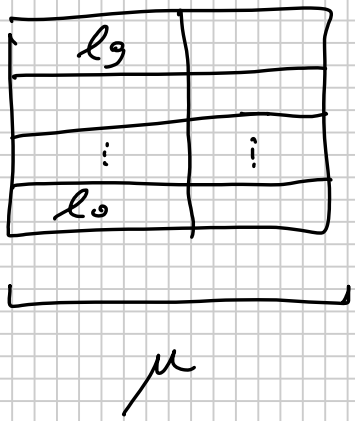
Aumentare di 1 tutti gli elementi con indice pari di un array

```
int a[10];
int i;
  .
  .
  .
for (i=0; i<10; i=i+2)
    a[i] = a[i] + 1;
```

variabile di controllo del for

È come se:





Titolo nota 20/10/2015

Come si usano gli array come parametri di procedure / funzioni

```
void azzera (int v[], int dim)
    { int i;
      for (i=0; i<dim; i++)
          v[i]=0;
    }

main()
  { int a[10];
    :
    azzera(a,10);
  }
```

è un puntatore al primo elemento dell'array !!