INFORMATICA A

A.A. 2019-20

Laboratorio n°3

Dott. Michele Zanella

Ing. Gian Enrico Conti

Puntatori

- I **puntatori** sono delle variabili che contengono l'indirizzo di memoria di un'altra variabile

```
int* nome_puntatore;
```

- "*" è chiamato **operatore di deferenziazione** e restituisce il contenuto dell'oggetto puntato dal puntatore; mentre l'operatore "&" restituisce l'indirizzo della variabile.

```
int var=0, var2, ind;

int* puntatore;

puntatore = &var;  //puntatore punta a var
var2 = *puntatore; //var2 == var
ind = puntatore;   //ind contiene l'indirizzo di var
*puntatore = 5     //var == 5
```

Operazioni sui puntatori

- È possibile effettuare operazioni sui puntatori.
  - Spostamento in avanti: *p* + *i*, sposta il puntatore di *i* posizioni avanti
  - Spostamento indietro: *p* – *i*, sposta il puntatore di *i* posizioni indietro
- **Un puntatore esiste sempre in funzione del tipo di oggetto puntato.**
  - Un puntatore ad *int* ha un blocco di memoria di 4 byte, a *char* di 1 byte, ecc…
  - Se sommo al puntatore un intero o utilizzo l'operatore **++,** il puntatore si sposterà in avanti di tanti byte quanti ne prevede il tipo di variabile puntata.
    Ad esempio: *p* + *i* equivale a *posizione* + *(i*dimensione tipo puntato)*
- Se ad una funzione passo dei puntatori eventuali operazioni su questi parametri saranno effettuate sulle variabili originali!
- È possibile agire sugli array come se si stesse agendo su un puntatore poiché entrambi sono blocchi contigui di memoria. **Attenzione:** devo comunque assegnare la prima cella dell'array ad un puntatore.
- Posso utilizzare i puntatori per puntare a strutture, in questo caso per accedere ai campi della *struct* utilizzo l'operatore **"->"**

Puntatori e tipi

- Puntatori ed array

```
char stringa[20];
char* pointer;

pointer= &stringa[0]; //pointer punta a stringa[0]
pointer=stringa;      //Uguale alla riga sopra
pointer++;            //Ora pointer punta a stringa[1]
```

- Nel dettaglio

```
int a[20]; // a è puntatore alla prima cella dell'array => a[0]
           // a[1] = *(a+1)
           // &[1] = (a+1)

scanf("%d", (a+1))     // &a[1] = &*(a+1)=(a+1)

for(int i=0; i<20; i++){
    printf("%d", a[i]); // Le due printf stampano la stessa cosa
    printf("%d", i[a]); // a[i] = *(a+i) = *(i+a) = i[a]!!!
}
```

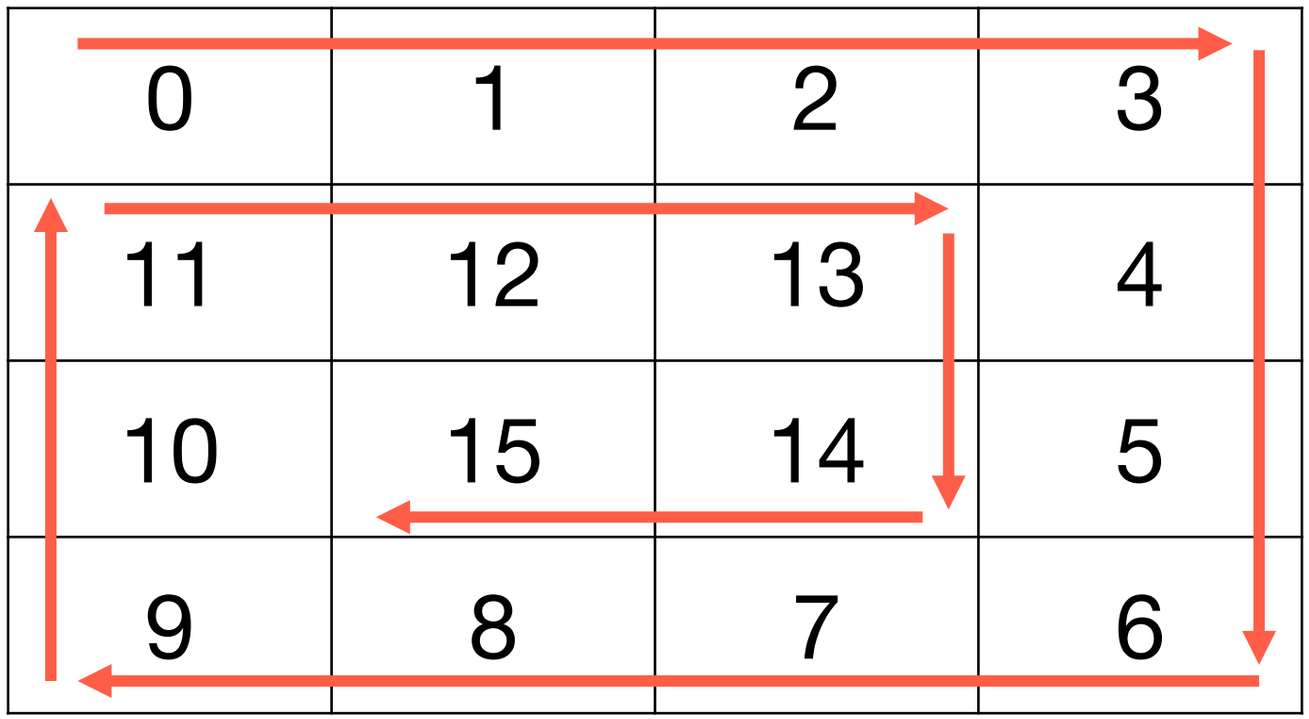
Puntatori e tipi

- Puntatori e struct

```
struct punto {
    int x;
    int y;
} puntoA;

struct punto* pointer;

pointer = &puntoA; //pointer punta a
puntoA

pointer->x = 6;     //Assegno il campo x
pointer->y = 7;     //Assegno il campo y
```

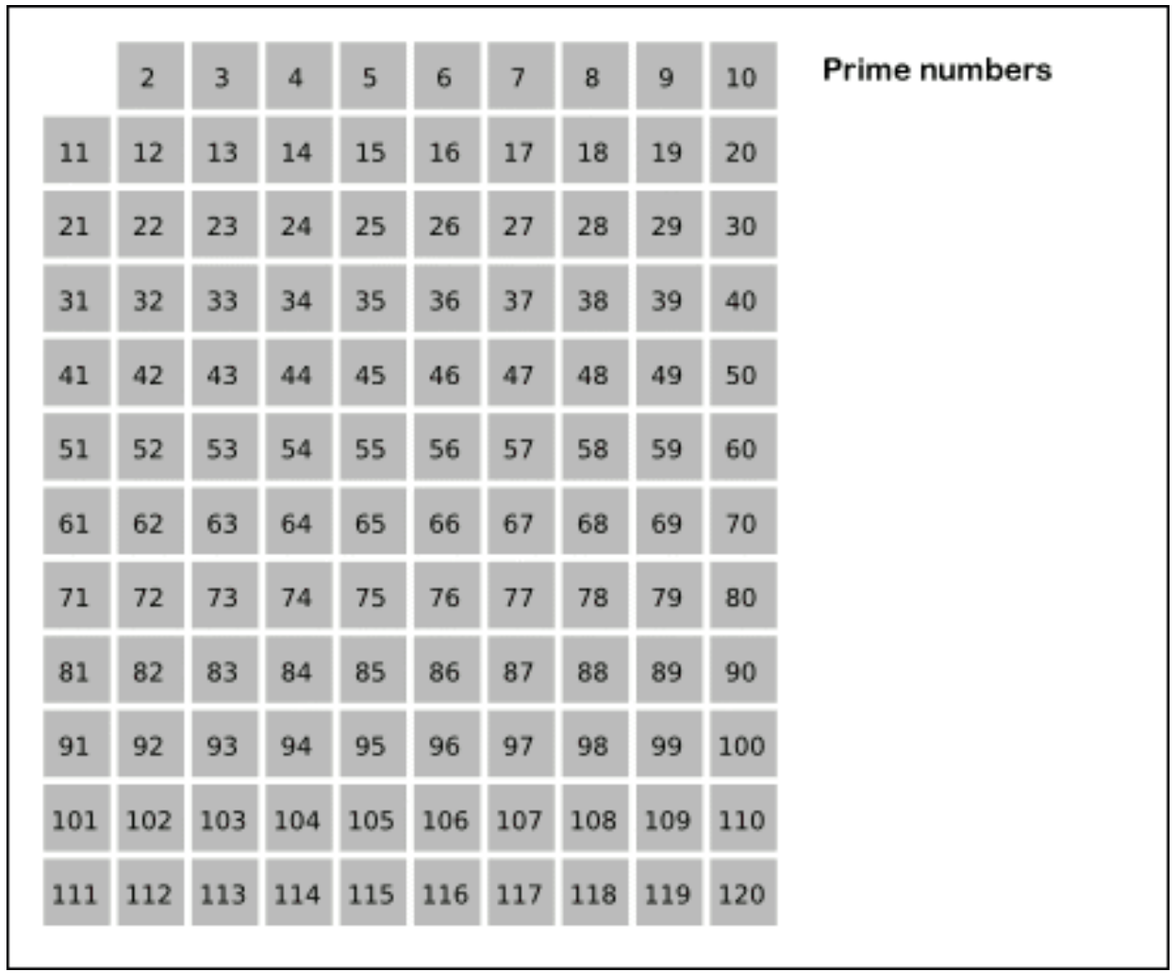
Esercizio 3.1: Matrice - Riempimento concentrico

Scrivere un programma che riempia una matrice in modo concentrico, ricevendo gli interi da I/O, e che stampi la matrice formattata.



Esercizio 3.2: Crivello di Eratostene

Scrivere un programma che implementi il crivello di Eratostene, antico algoritmo per il calcolo dei primi *n* numeri primi.



Esercizio 3.3: Confronto tra stringhe

Scrivere un programma che, date due stringhe verifichi se sono uguali o in caso negativo quale viene prima in ordine alfabetico.

Il programma deve stampare:

- 0: le stringhe sono uguali
- -1: la `stringa1` è minore della `stringa2`
- 1: la `stringa2` è maggiore della `stringa2`

Esercizio 3.4: Parentesi

Scrivere un programma che richiede all'utente una sequenza di caratteri (stringa) che non contiene spazi, li registra in una struttura dati appropriata e verifica se la sequenza è ben parentesizzata".

- Una sequenza si dice ben "parentesizzata" se le parentesi, per semplicità consideriamo solo quelle tonde, sono aperte e chiuse correttamente.
- Ad esempio sono ben parentesizzate" le sequenze:

  Bla

  (bla)

  (bla(bla))

  (bla()(bla)())

- **Hint:** si tenga conto man mano di quante sono le parentesi già aperte, e si decida di conseguenza. Si arresti la scansione non appena si scopre che la sequenza è illegale, anche prima di arrivare alla fine.

Esercizio 3.5: Array puntatore

Si implementi un programma che salvi n elementi in un array e li stampi a video scorrendo l'array come un puntatore.