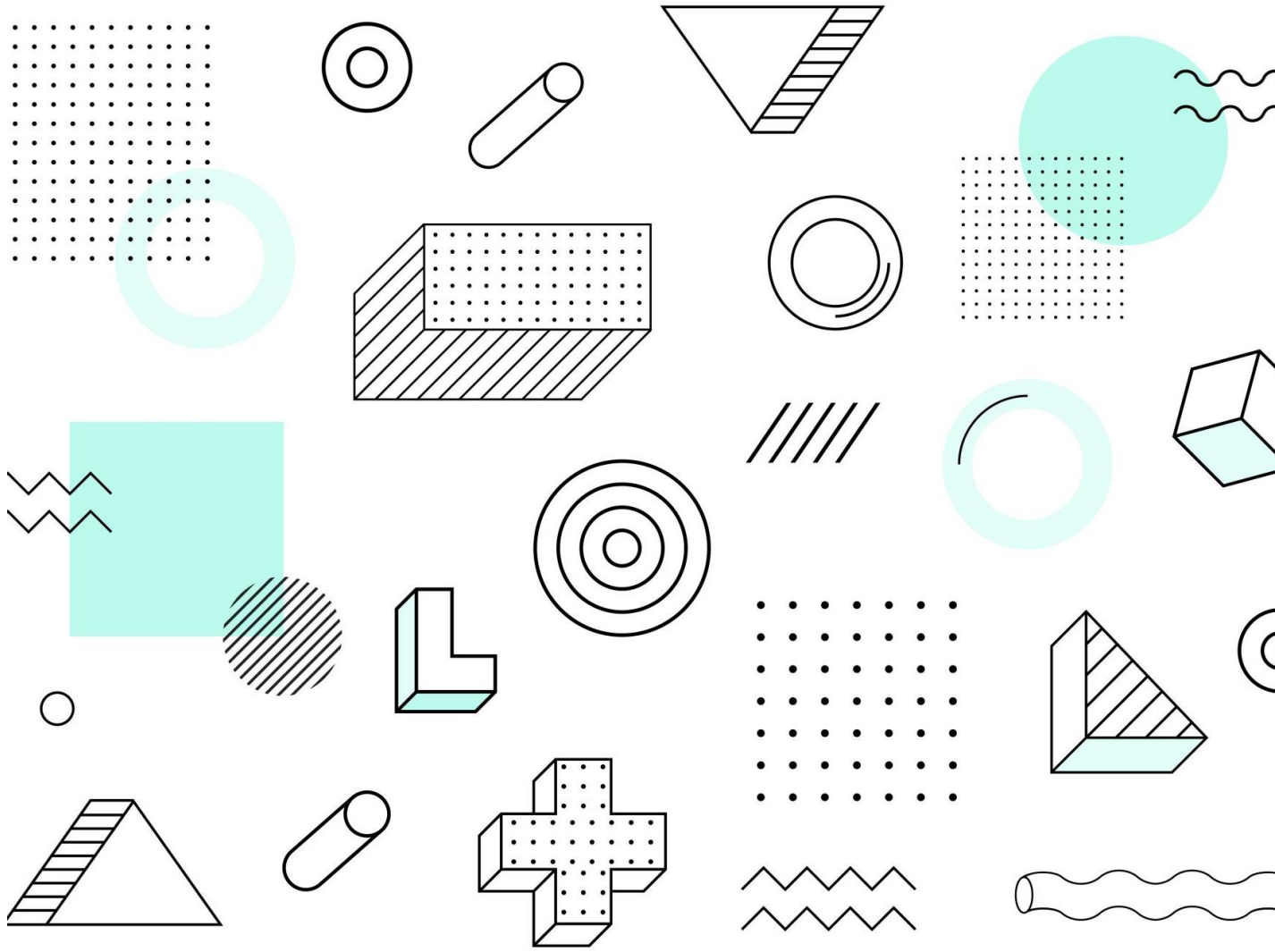
Laboratorio

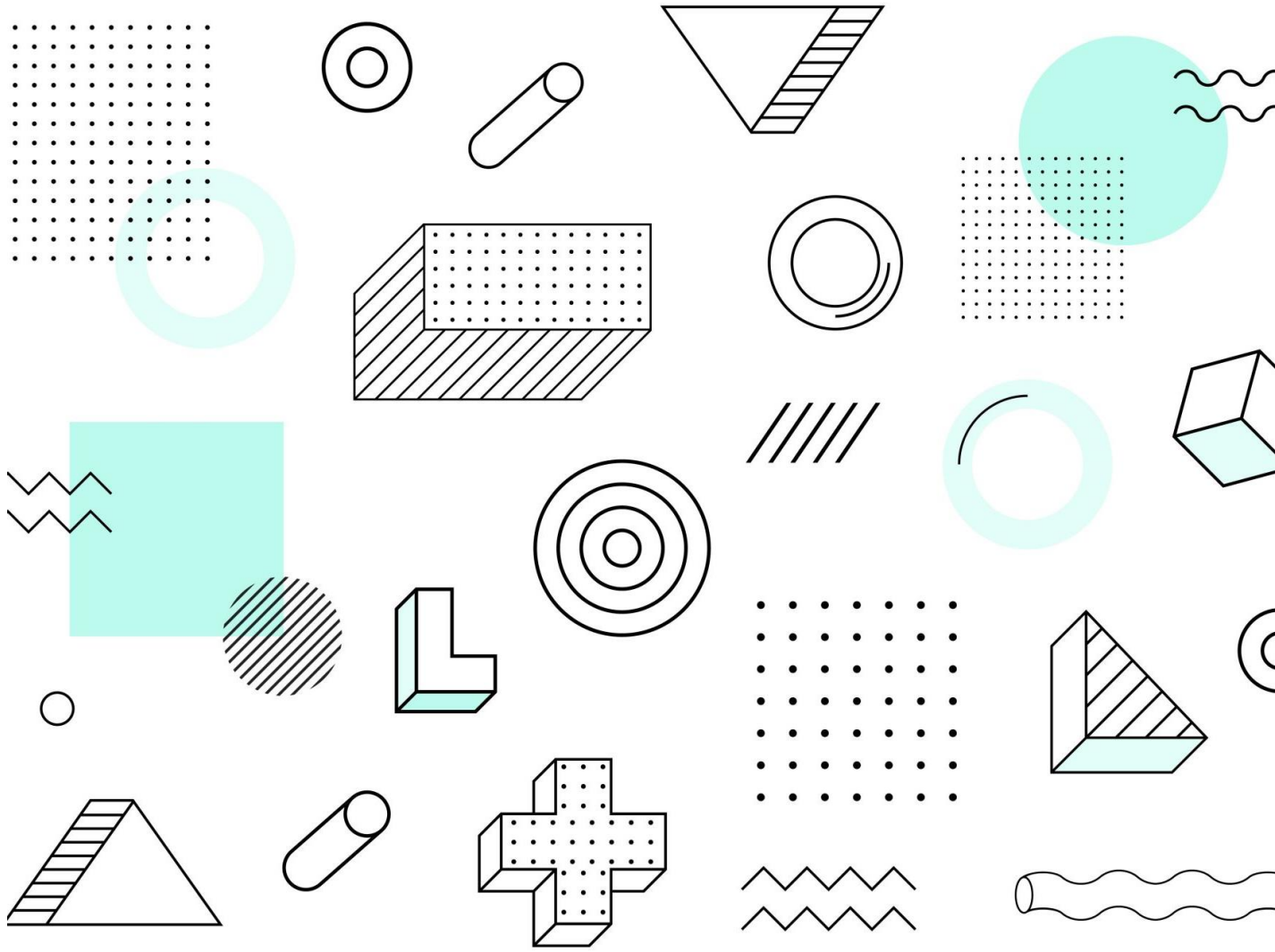
12-11-2024

Informatica A

Ingegneria Fisica



Operatori Condizionali



Recap (1)

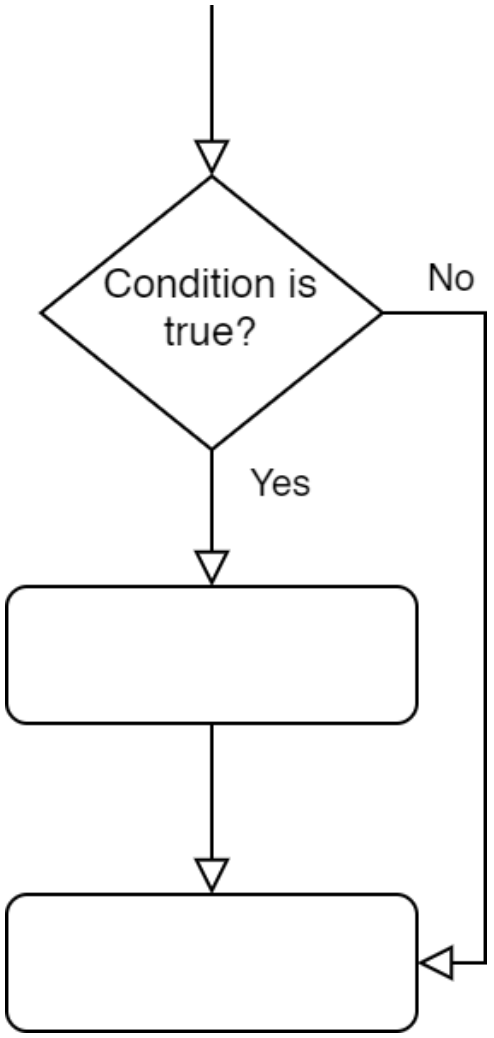
L'operatore condizionale **`if`** permette di eseguire un blocco di codice solo se una certa condizione è vera.

```
if (<condizione>)
{
    //DO SOMETHING
}
```

Per utilizzare condizioni più complesse, ci affidiamo agli operatori logici e relazionali, che ritornano un valore di verità {0, 1}.

`==`	uguale a
`!=`	diverso da
`>`	maggiore di
`<`	minore di
`>=`	maggiore o uguale a
`<=`	minore o uguale a

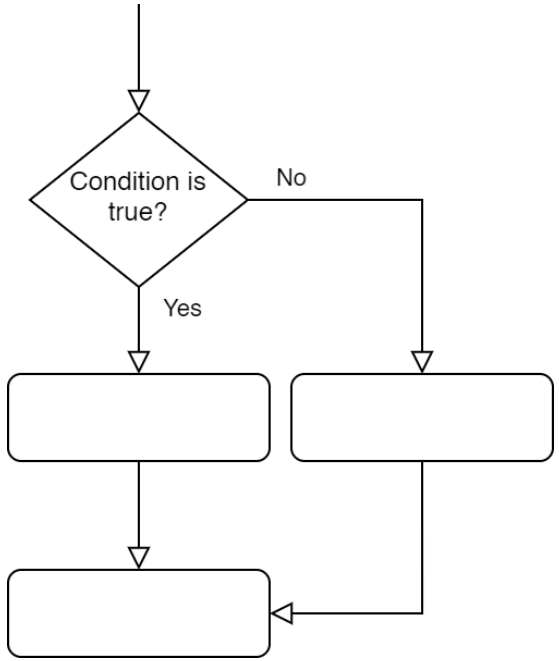
`&&`	AND logico
`||`	OR logico
`!`	NOT logico



Recap (2)

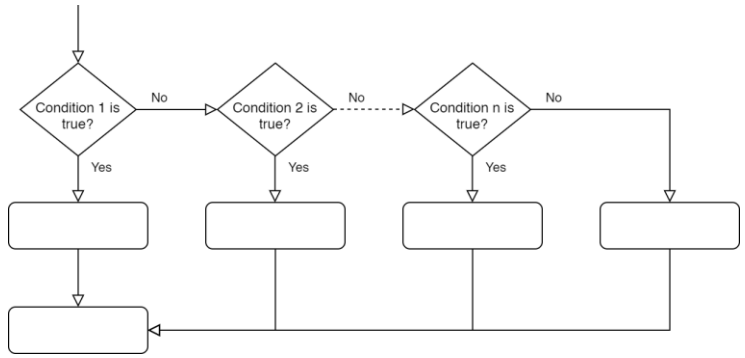
L'operatore **`if`** può essere esteso per comprendere vari rami del flusso di esecuzione, condizionatamente a più di una condizione.

```
if(<condizione>)
{
    //DO SOMETHING
}
else
{
    //DO SOMETHING
}
```



Solo un ramo in una catena di **`if, else if, else`** può essere eseguito. Anche se più condizioni sono verificate, solo il ramo corrispondente alla prima condizione vera sarà eseguito.

```
if(<condizione 1>)
{
    //DO SOMETHING
}
else if(<condizione 2>)
{
    //DO SOMETHING
}
...
else if(<condizione n>)
{
    //DO SOMETHING
}
else
{
    //DO SOMETHING
}
```



Esercizio 1

Scrivere un programma che chiede che all'utente due numeri interi, e che stampa il più grande tra i due.

Esercizio 2

Scrivere un programma che chiede in input due numeri interi positivi e stabilisce se sono oppure no uno il multiplo dell'altro.

Esercizio 3

Scrivere un programma che presi in input due numeri interi positivi o nulli indica all'utente se il loro rapporto è maggiore, minore o uguale a 1.

CHALLENGE: risolvere senza far calcolare al programma il rapporto tra i numeri.

Esercizio 4

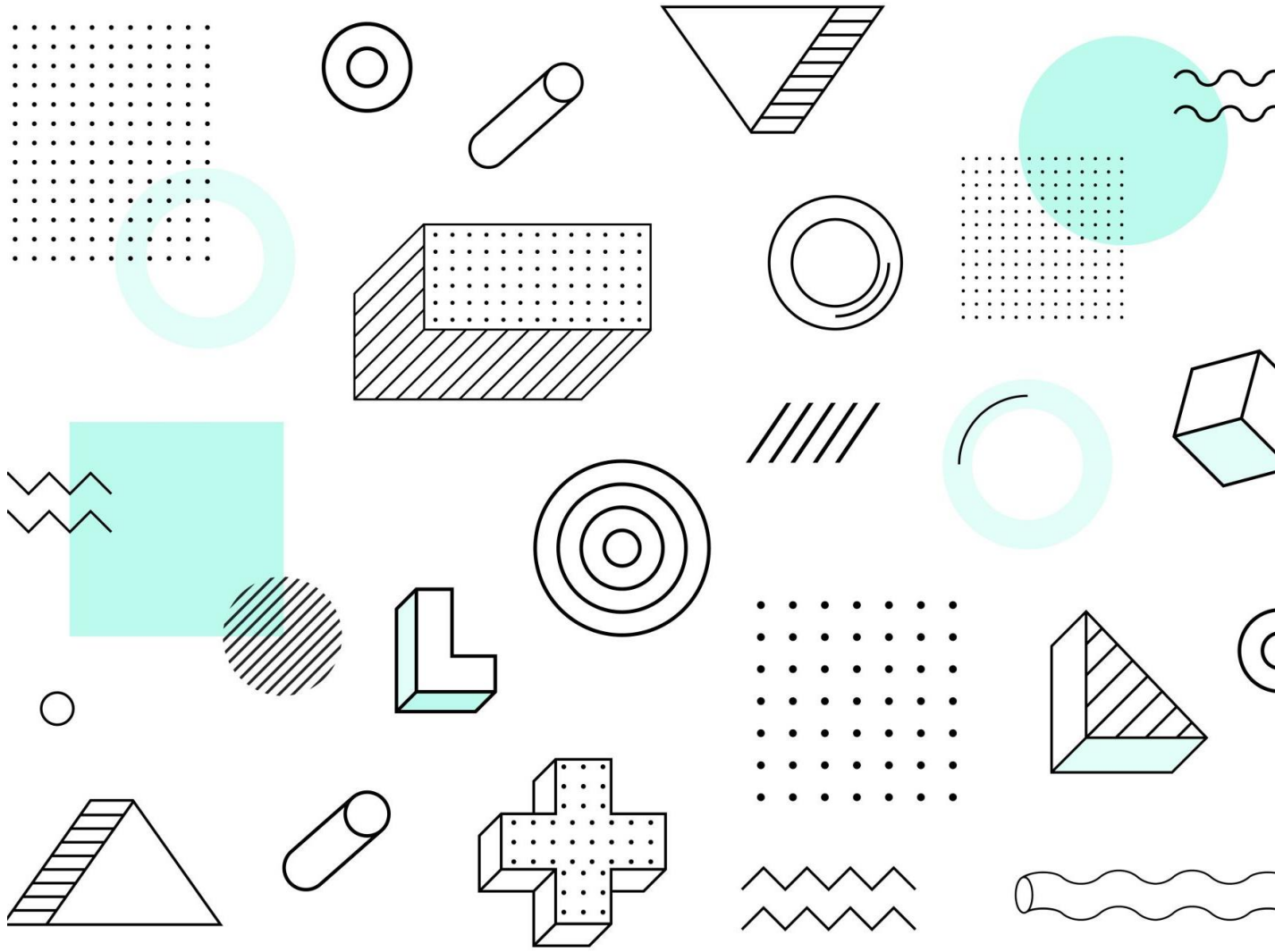
Scrivere un programma che chiede all'utente tre numeri interi, e che stampa i tre numeri in ordine decrescente.

Esercizio 5

Scrivere un programma che richiede in input la lunghezza dei tre lati di un triangolo. Verificare se i numeri inseriti posso effettivamente rappresentare i lati di un triangolo (suggerimento: cosa dice la disuguaglianza triangolare?). In caso negativo, visualizzare un messaggio di errore. In caso affermativo, stabilire se il triangolo è equilatero, isoscele o scaleno, e calcolarne il perimetro.

Operatori Condizionali

Esercizi Aggiuntivi



Esercizio 6

Scrivere un programma che chiede in input all'utente un numero e che indica se:
1.Il numero è zero
2.Il numero è positivo
3.Il numero è pari
4.Il numero è a tre cifre
5.L’ultima cifra è maggiore di 5

Esercizio 7

Scrivere un programma per aiutare i furbi del cartellino.
Il programma acquisisce il numero di ore effettuate e stampa a video “Numero di ore di lavoro:” con il numero di ore effettive pari a n +3. Per non farsi beccare, però, se il numero di ore è inferiore a 4, il numero viene aumentato solo del 10%

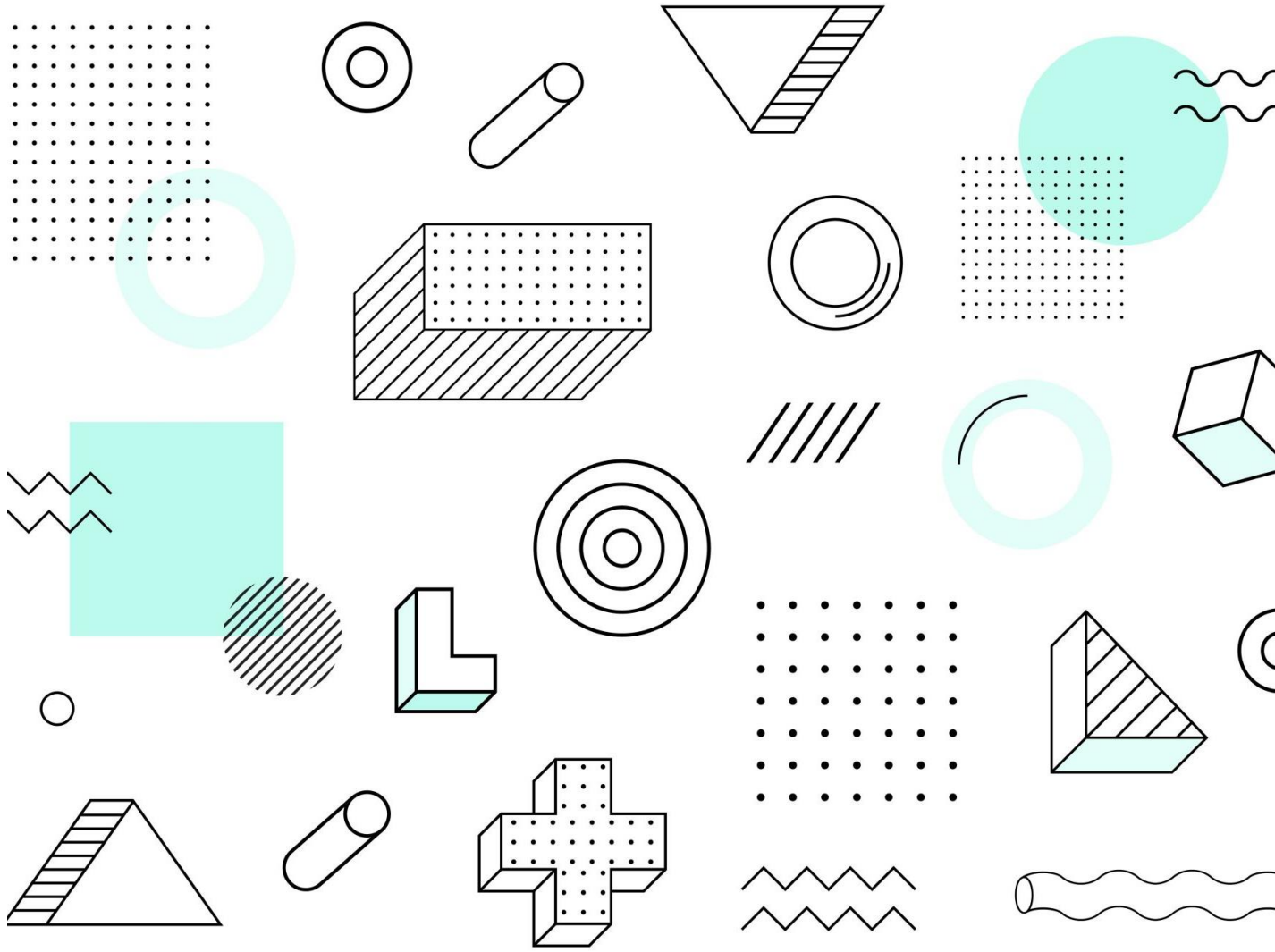
Il risultato deve essere stampato in formato orario, quindi 4,5 ore diventa 4:30

Esercizio 8

Scrivere un programma che, acquisitor un anno dall'utente, stabilisca se questo è bisestile.

Procedura per determinare se un anno è bisestile:
1.Se è divisibile per 4, vai allo step 2,altrimenti non è bisestile
2.Se è divisibile per 100, vai allo step 3, altrimenti è bisestile
3.Se è divisibile per 400, è bisestile, altrimenti non è bisestile

Cicli



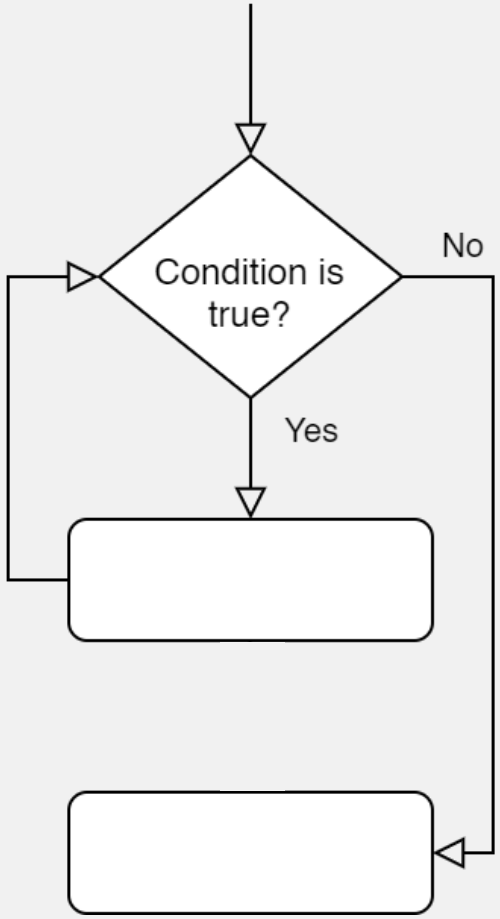
Recap (1)

Molte volte è richiesto ripetere le stesse operazioni (o gruppo di operazioni) per più volte.

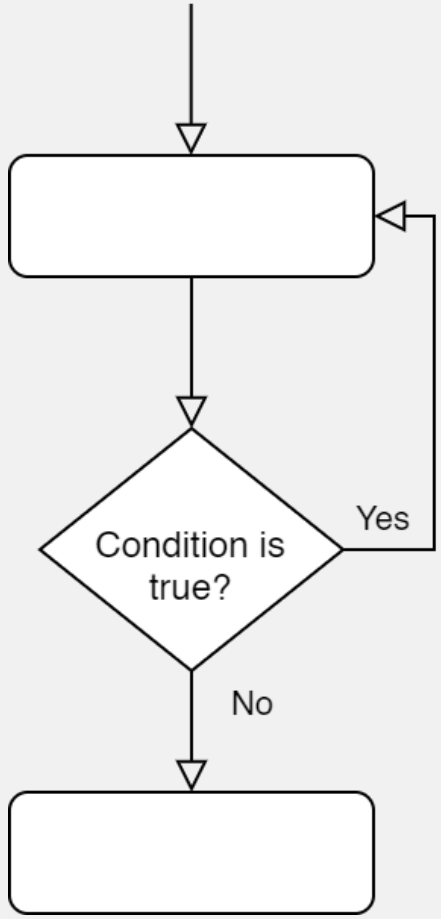
1 I *cicli* ci permettono di ripetere tutte le istruzioni all'interno di un blocco di codice.

Nei cicli di tipo *while* la condizione viene controllata prima di eseguire le istruzioni del blocco. Le istruzioni potrebbero non essere mai eseguite

2 Nei cicli di tipo *do-while* la condizione viene controllata dopo aver eseguito le istruzioni del blocco.
Le istruzioni vengono quindi eseguite almeno una volta



1



2

Recap (2)

Ciclo while

Il ciclo *while* permette di eseguire un blocco di istruzioni finché una determinata condizione è vera.

```
while(<condizione>)
{
    //DO SOMETHING
}
```

Ciclo for

Il ciclo for è un ciclo di tipo *while*, quindi la condizione viene verificata prima di eseguire le istruzioni del blocco.

```
int i;
for(A;B;C)
{
    //DO SOMETHING
}
```

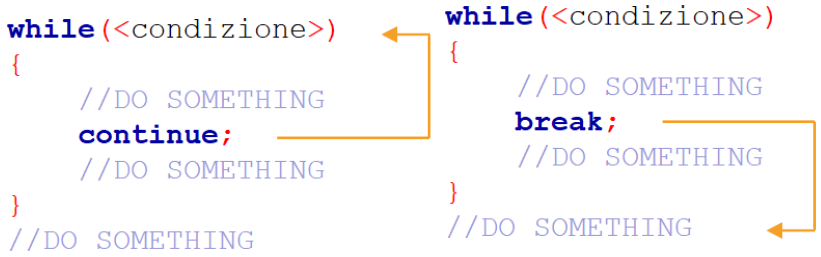
Ciclo do-while

Il ciclo *do-while* permette di eseguire un blocco di istruzioni finché una determinata condizione è vera.

```
do
{
    //DO SOMETHING
}
while(<condizione>);
```

ℹ Alcune parole chiave ci permettono di influenzare il flusso di esecuzione all'interno dei cicli.

break	termina l'esecuzione del ciclo istantaneamente e salta all'istruzione dopo il ciclo.
continue	termina l'iterazione corrente del ciclo e salta alla prima istruzione del ciclo (che per i cicli di tipo while è il controllo della condizione).



Esercizio 9

Scrivere un programma che chiede all'utente un numero intero e che stampa tutti i numeri positivi minori o uguali al numero dato.

Esercizio 10

Scrivere un programma che chiede all'utente un numero intero e che stampa la somma di tutti i numeri interi positivi minori o uguali al numero dato.

Esercizio 11

Scrivere un programma che stampi a schermo un triangolo rettangolo composto da asterischi, di altezza definita dall'utente mediante un numero intero

Esercizio 11 Pro

Estendere il programma dell'es. 4 per poter disegnare anche un rettangolo ed il perimetro di un rettangolo. Permettere di scegliere all'utente la forma desiderata.

```
****        ****        *
****        *  *        **
****        *  *        ***
****        ****        ****
```

Esercizio 12

Scrivere un programma che chiede in input un numero e stampa tutte le potenze di 2 minori o uguali del numero dato.

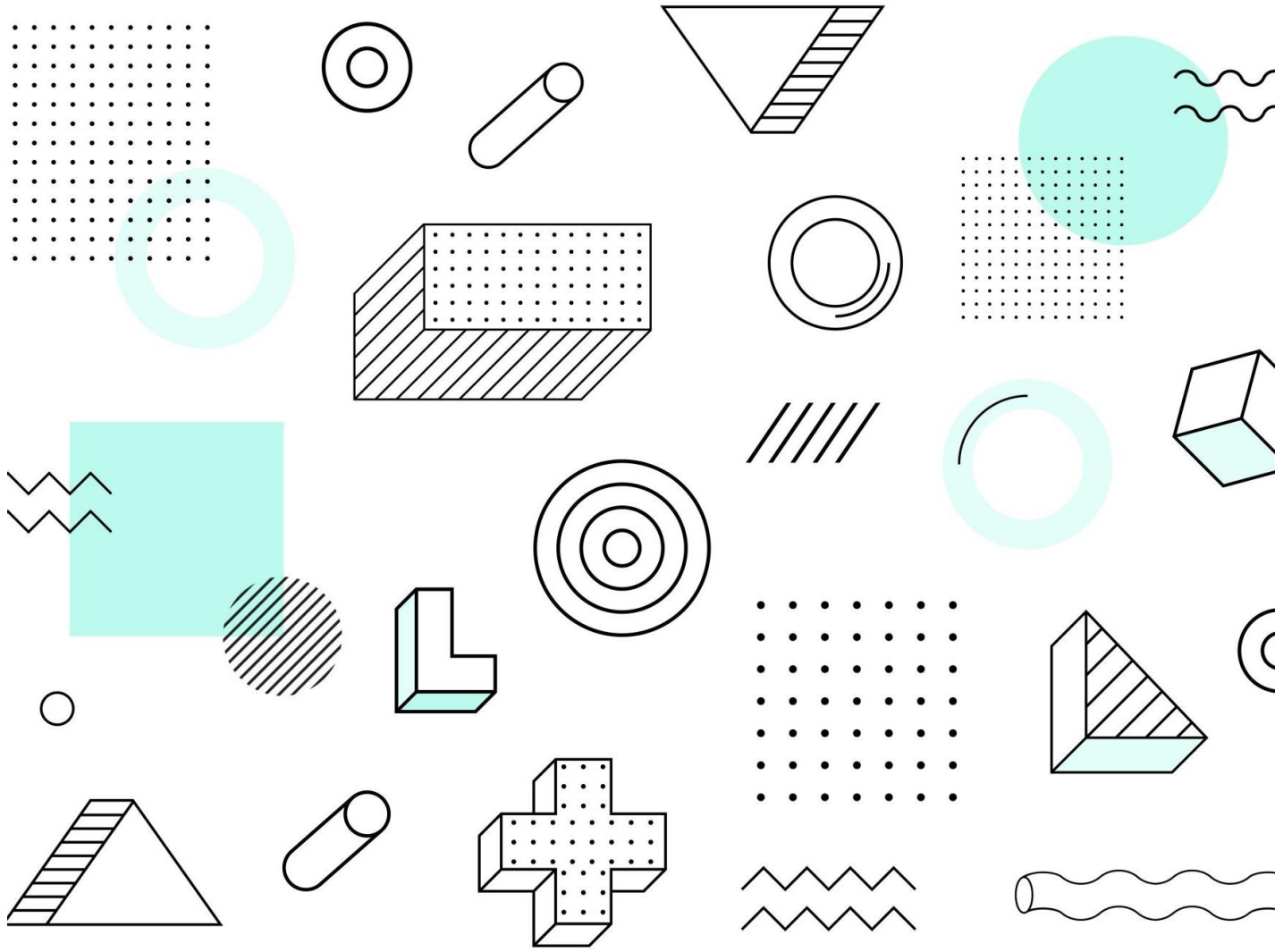
CHALLENGE: fare due versioni, una con un ciclo for e una con un ciclo while.

Esercizio 13

Scrivere un programma che chiede all'utente un numero e che accetta solo un numero pari

Cicli

Esercizi Aggiuntivi



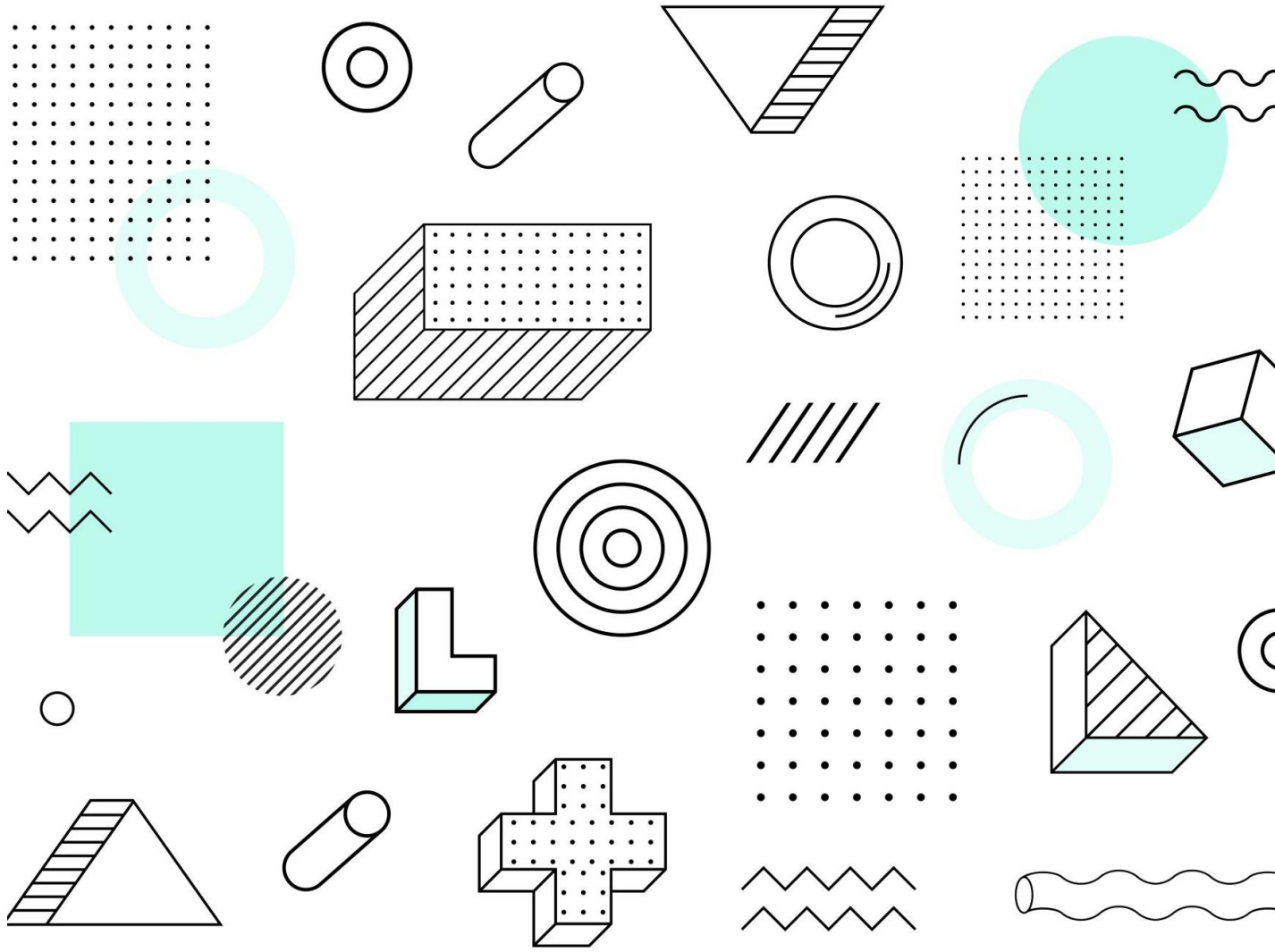
Esercizio 14

Scrivere un programma che chiede all'utente due numeri `n`, `k` e che calcola n^k.

Esercizio 15

Scrivere un programma che chiede all'utente un numero e che stabilisce se questo è primo.

Vettori, stringhe e Matrici



Recap (1)

Un insieme di variabili ordinate

I vettori (array) sono variabili che al loro interno contengono più celle di un certo tipo.

Accedere alle celle

Le varie celle sono identificate dalla loro posizione all'interno del vettore

⚠ Attenzione, la prima cella si trova in posizione **0** (si dice che gli array in C sono zero-based). Non sarà lo stesso in MATLAB.
L'ultima posizione di un vettore è dunque quella in posizione **n-1**, dove **n** è la dimensione del vettore.

```c
int primes[5] = {2,3,5,7,11};
int i;

for(i=0;i<5;i++)
{
    printf("%d\n", primes[i]);
}
```

Recap (2)

Vettori di vettori

Le matrici, come i vettori, contengono un insieme di variabili dello stesso tipo. A differenza di questi, le celle nelle matrici sono identificate da due indici anziché uno solo.

```c
void main()
{
    int mat[5][5];

    printf("Inserisci in posizione 0,0: ");
    scanf("%d", &mat[0][0]);
}
```

Vettori di caratteri

Le stringhe in C son definit come vettori di caratteri. Ogni cella del vettore contiene un carattere della stringa.

Poiché non possiamo sapere a priori quanto sia lunga una stringa, il carattere terminatore '\0' viene utilizzato per indicarne il termine.

```c
char stringa[100];
```

c	i	a	o	!	\0

La stringa 'ciao' ha bisogno di un array di almeno 5 caratteri per poter essere salvata.

```c
char stringa[5] = "ciao";
```

Esercizio 16

Scrivere un programma che chiede all'utente cinque numeri interi e che li stampa in ordine inverso.

Esercizio 17

Scrivere un programma che chiede all'utente cinque numeri interi tra 0 e 100 e che ne stampa il massimo, il minimo e la media.

Esercizio 18

Scrivere un programma che chiede all'utente cinque numeri interi positivi e che li stampa in ordine decrescente.

Esercizio 19

Scrivere un programma che data la stringa «stampami», salvata in un vettore di 100 caratteri, la stampi carattere per carattere.

(Suggerimento: usare la funzione **`strlen`**)

La funzione **`strlen`** nella libreria **`string.h`** ritorna la lunghezza della stringa data.

Esercizio 20

Scrivere un programma che chiede ripetutamente un numero **n**. Se questo è pari, il programma stampa i primi **n** valori della sequenza di fibonacci. Se invece **n** è dispari o **0**, il programma si arresta. Con:

```
FYI:= fib(i-1)+fib(i-2)
fib(i)
fib(0)= 0
fib(1)= 1
```

Cosa succede se **n** è negativo? Cambia qualcosa se è pari o dispari?

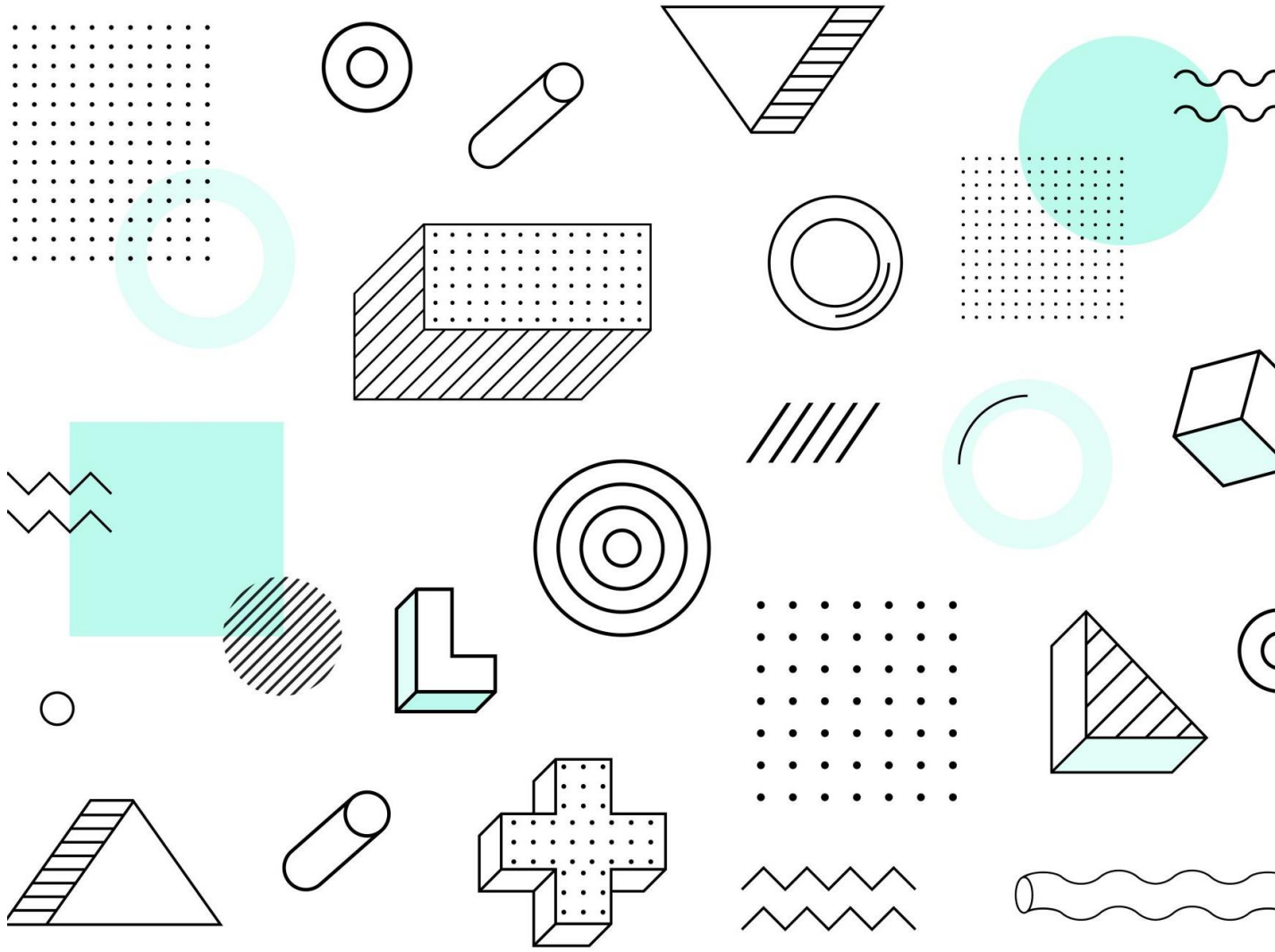
Esercizio 21

Scrivere un programma che, data una matrice 8×8 rappresentante una scacchiera sulla quale sono presenti un Re ed una Regina, stampi «sì» se la Regina può mangiare il Re, «no» altrimenti.

SUGGERIMENTO: potete indicare il Re con un `1` e la Regina con un `2`, mentre lo `0` può indicare le caselle vuote.

Vettori, stringhe e Matrici

Esercizi Aggiuntivi



Esercizio 22

Scrivere un programma che, data una stringa ed un numero intero, codifichi la stringa secondo il cifrario di Cesare.

Suggerimento: fare uno shift sull'alfabeto di ogni lettera di tanti posti quanti indica il numero (la chiave).

Esercizio 23

Scrivere un programma che, date due stringhe, codifichi la prima utilizzando il cifrario "snake cypher" con la seconda come chiave.

Chiave

a	b	c	d

Testo

c	i	a	o	m	o	n	d	o

a	b	c	d	a	b	c	d	a

+

c	i	a	o	m	o	n	d	o

=

c	j	c	r	m	p	p	g	o