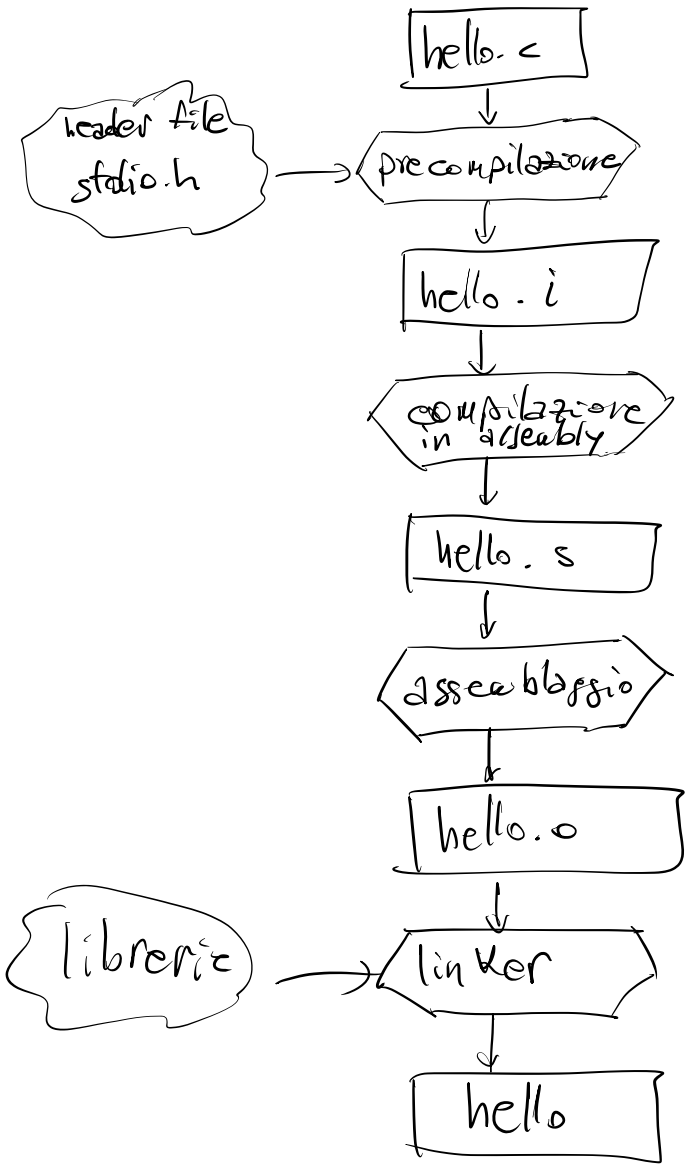

hello.c
header file
stdio.h
precompilazione
hello.i
compilazione in assembly
hello.s
assemblaggio
hello.o
librerie
linker
hello
gcc -E hello.c
gcc -S hello.c
gcc -c hello.c
gcc hello.c -o hello

SYNTACTIC SUGAR

DICHIARAZIONI DI VARIABILI

Go	C
var x,y int	int x,y;
var f float64	double f;

DICHIARAZIONI DI FUNZIONI

```
func f(x int, y float64) int {
    ...
}
func g(x int) {
    ...
}
```

```
int f(int x, double y){
    ...
}
void g(int x){
    ...
}
```

PERVASIVITÀ DELLE ESPRESSIONI

GO:

```
x = (y+3) * 2
z = f(x) + g("ciao", y) - 3
```

(espressioni)

C:

```
x+3;
x = y+3;
x = (y = 5+z) * z;
x = (y++) * (--z * 6) - (a=7);
x = y>3 ? x+7 : z--;
```

(I, II)

CASTING (CONVERSIONE DI TIPO)

```
double f,g;
int x,y;
f = x+y+g;
```

```
x = f;
x = (int)(f);
```

(espr.)

```
f = x/y;
```

←

```
f = (double)x / y
```

NO

```
f = (double)(x/y)
```

TIPI COMPOSTI

GO	C
puntatori	sì, MOLTO pervasivi
struct	sì
array	sì, con alcune differenze
slice	no
map	no
tipi funzione	sì

PUNTATORI

- Stessa sintassi

```
var p *int
var x int

p = &x
```

```
int *p;
int x;

p = &x;
```

```
type data struct {
    g, m, a int
}

var p *data
    ⋮
p.g = 11
p.m = 1
p.a = 2023
```

```
typedef struct {
    int g, m, a;
} data;

data *p;
    ⋮
p->g = 11;
p->m = 1;
p->a = 2023;
```

ARRAY

```
var x int [10]
var y data [100]
```

COSTANTE

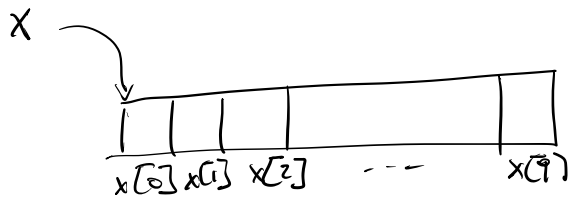
```
int x[10];
data y[100];
```

COSTANTE IN C89
VARIABILE IN C99

ARRAY E PUNTATORI

- Il nome di un array è il puntatore al suo elemento iniziale

```
int x[10];
```

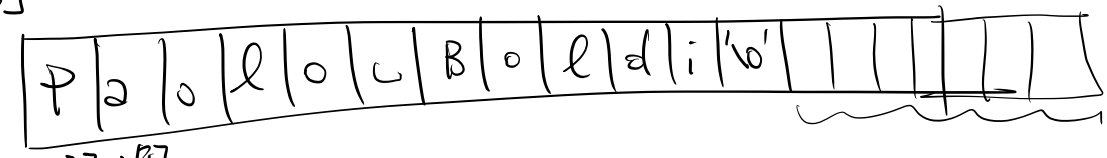



```
void stampa(int *a, int n) {
    int i;
    for (i=0; i<n; i++)
        printf("%d\n", a[i]);
}
int main() {
    int x[100];
    ...
    stampa(x, n)
}
```

ARRAY E STRINGHE

STRINGHE = array di char terminato da 0

char s[∞]



s[0] s[1]

include <string.h> ←

```
int strlen (char *s) {
    int i;
    i=0;
    while (s[i]) i++;
    return i;
}
```