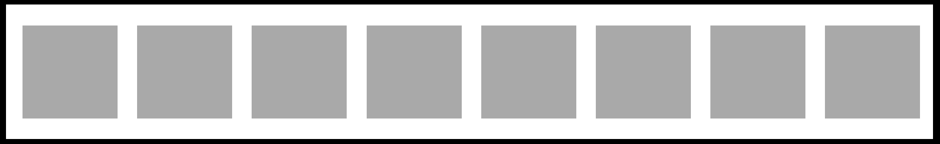
THINK IN C

SESSION 2

Let’s Byte in a bit

- Bit - binary digit   0 or 1
- 8 bits = 1 byte



PRIMARY DATA TYPES

char - 8 bits

int - 32 bits

float - 32 bits

double - 64 bits

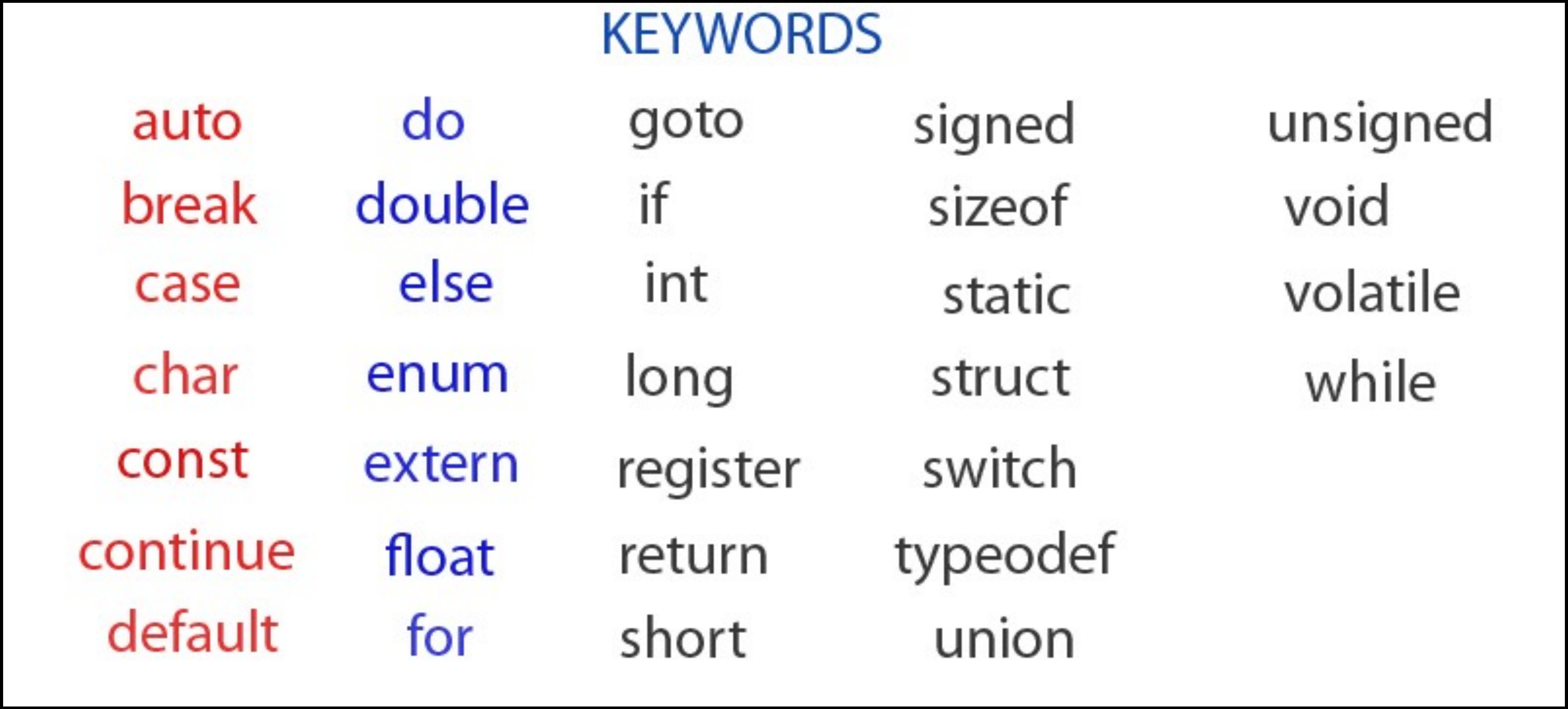
long

short

signed

unsigned

OTHER KEYWORDS



KEYWORDS

auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	
continue	float	return	typeodef	
default	for	short	union	

THUMB RULES

- Each instruction is a separate statement
- Statements must appear in the same order as we wish them to be executed

;

MAIN FUNCTION

Statements inside this block is executed first

VARIABLE USAGE

DECLARE

VARIABLE USAGE

INITIALISE

STATEMENTS

LHS - Variable

LHS = RHS;

RHS != LHS;

RHS - Variable or Some for Computation

++ - - += -=

MIND YOUR BRACKETS

CODE INTERACTION

```
#include <stdio.h>

printf();

scanf();
```

SYNTAX

printf(“<format string>”, <list of variables>);

scanf(“<format string>”, &<list of variables>);

FORMAT SPECIFIERS

%d - int

%f - float/double

%c - char

%ld - long int

%lld - long long int

%lu - long unsigned

SIMPLE INTEREST

To Do or Not to

That is the question

if-else

for

while

do-while

switch

SYNTAX

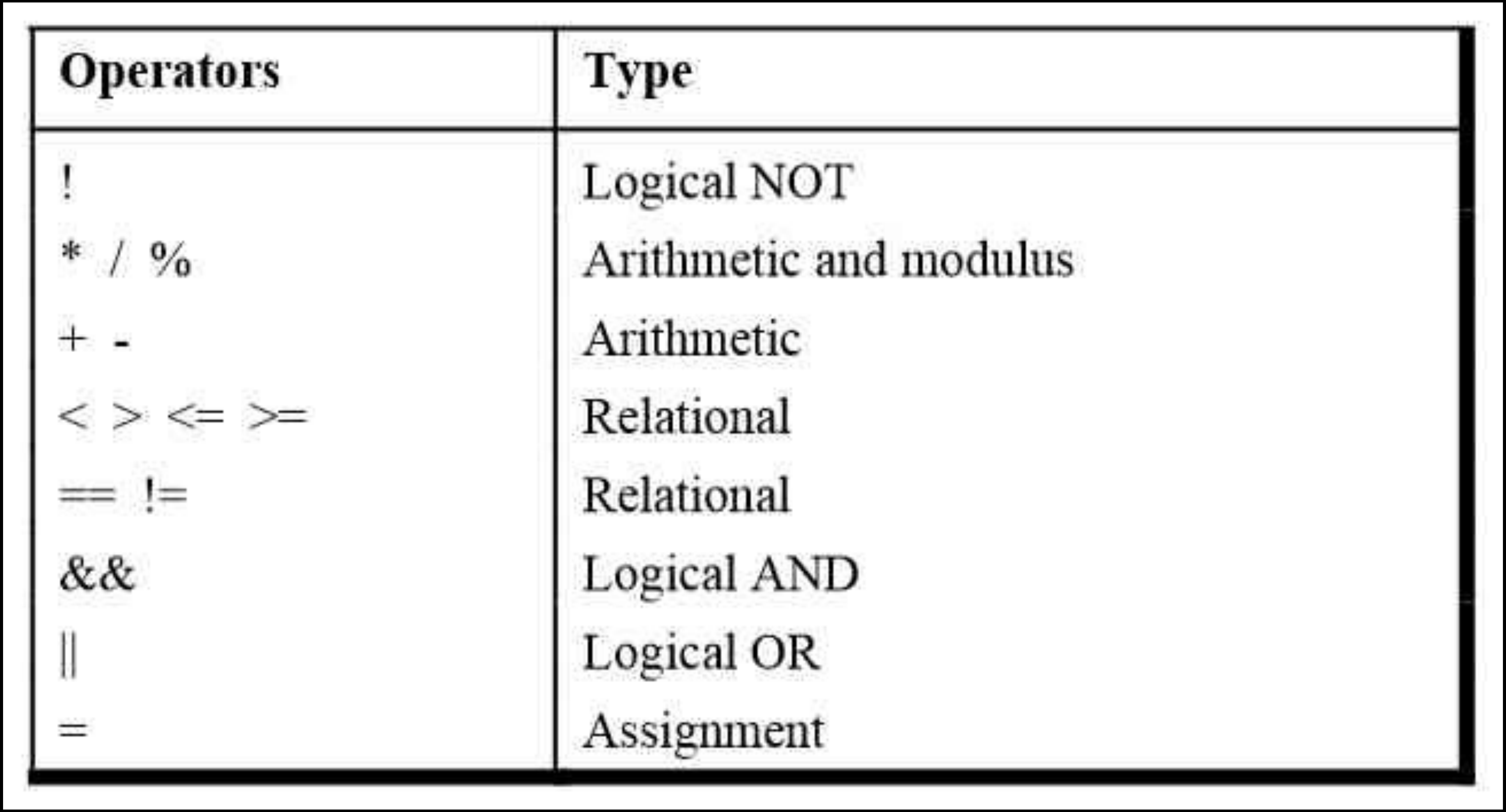
```
if (condition) {
   statements;
}
else if (condition) {
   statements;
}
else {
   statements;
}
```

```
do {
   statements;
}
while (condition);
```

```
while (condition){
   statements;
}
```

```
for (<initialisation>; <condition>; <increment>) {
   statements;
}
```

Condition	Symbol	Example
Less than	<	(a < b)
Greater than	>	(a > b)
Less than or equal to	<=	(a <= b)
Greater than or equal to	>=	(a >= b)
Equal to	==	(a == b)
Not equal to	!=	(a != b)



Operators	Type
!	Logical NOT
* / %	Arithmetic and modulus
+ -	Arithmetic
< > <= >=	Relational
== !=	Relational
&&	Logical AND
||	Logical OR
=	Assignment

JUMP STATEMENTS

break - jump out of the current loop
continue - bypass statements inside to restart loop

SYNTAX

```
switch (integer expression) {
    case constant:
       statement;
        //optional break
    case constant1:
       statement;
        //optional break
    default:
        statement;
}
```

LET’S PRACTICE