Slides

Section 4 : Variables and data types

Variables and Data Types

int main(){}

Entry Point

Statements

Functions

Errors

Warnings

Input

Output

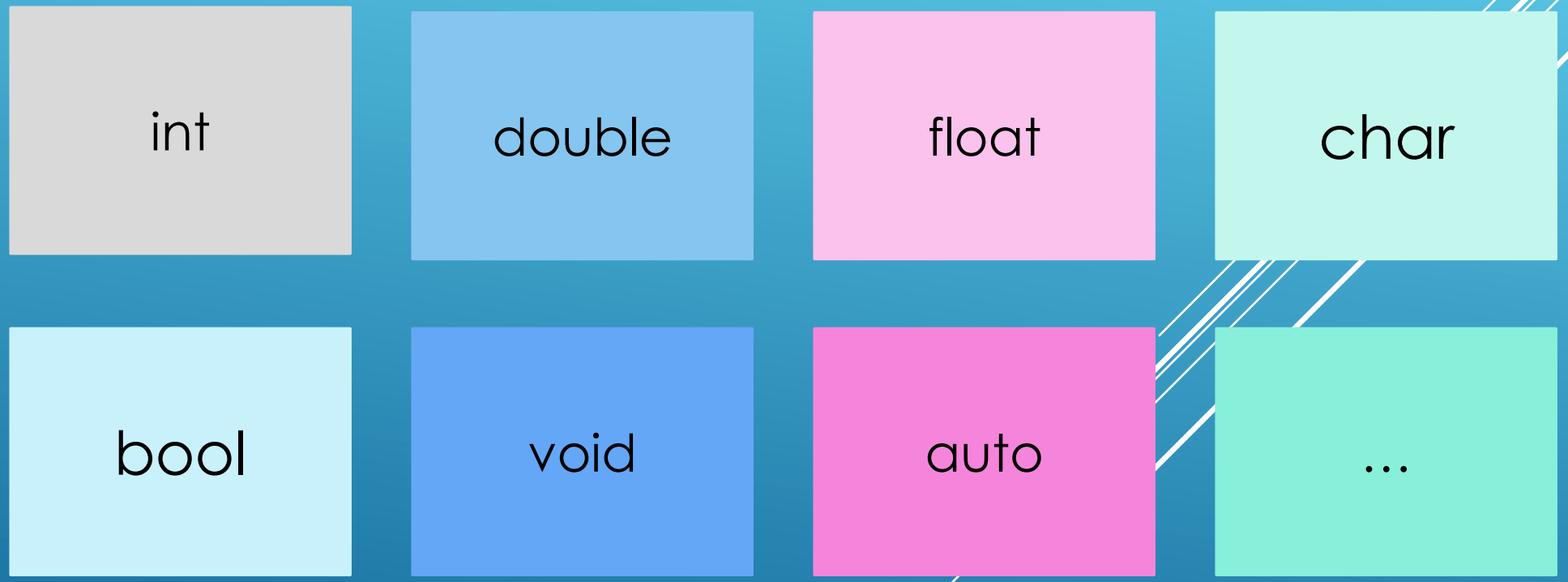
Comments

Dev Workflow

Memory Model

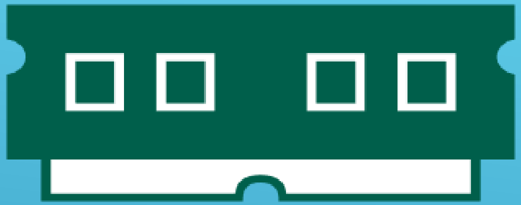
Execution Model

```cpp
int main(int argc, char **argv)
{
    //Compiler syntax error : missing semicolon
    std::cout << "Hello World in C++20!" << std::endl;

    int a {4};
    int b {4};

    //Runtime error
    int c = 10/ (a -b);
    std::cout << "The value of c is : " << c << std::endl;

    //Warnings
    20/0; // This throws a warning on gcc10.
    return 0;
}
```

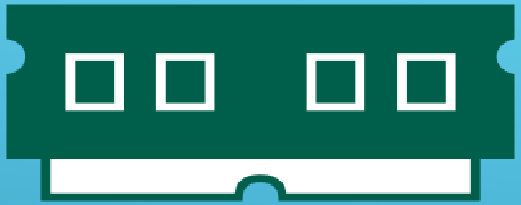

int
double
float
char
bool
void
auto
...

Memory is a grid of cells that contain either 1 or 0 in them.
Each cell is a bit

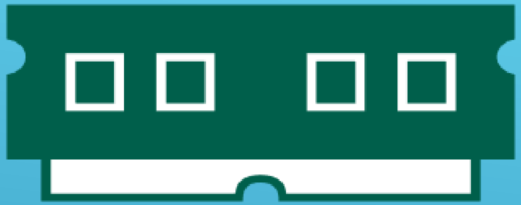
We group them in groups of 8, 16,32, 64 cells depending on the size of Data we want to store



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

22

23.8

“Steve”

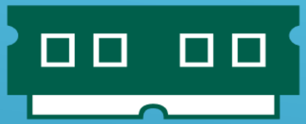
...

Slide intentionally left empty

Number Systems

- Binary
- Octal
- Hexadecimal

Number Systems



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

22

23.8

“Steve”

...

Base 10

2371

$2 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 1 \times 10^0$

924

$9 \times 10^2 + 2 \times 10^1 + 4 \times 10^0$

47

$4 \times 10^1 + 7 \times 10^0$

15

Base 2

100101

$1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

10010

$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

111

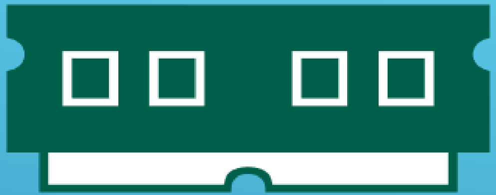
$1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

16

3 Digits

Binary	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

3 Digits
in memory

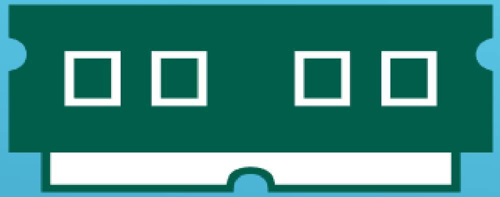


0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

4 Digits

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

4 Digits
in memory



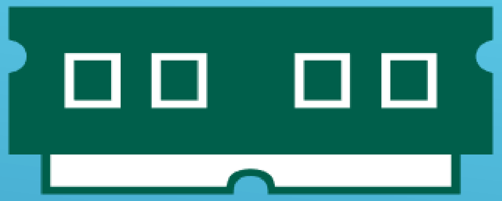
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

5 Digits

Binary	Decimal
00000	0
00001	1
00010	2
00011	3
00100	4
00101	5
00110	6
00111	7
01000	8
01001	9
01010	10
01011	11
01100	12
01101	13
01110	14
01111	15

Binary	Decimal
10000	16
10001	17
10010	18
10011	19
10100	20
10101	21
10110	22
10111	23
11000	24
11001	25
11010	26
11011	27
11100	28
11101	29
11110	30
11111	32

5 Digits
in memory



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

Generalization

Digits	Data Range
1	0 ~1
2	0~3
3	0~7
4	0~15
5	0~31
…	…
n	$0\sim2^{n-1}$

In practice

Digits	Bytes	Data Range
8	1	0 ~255
16	2	0~65,535
32	4	0~34,359,738,367
64	8	0~18,446,744,073,709,551,615

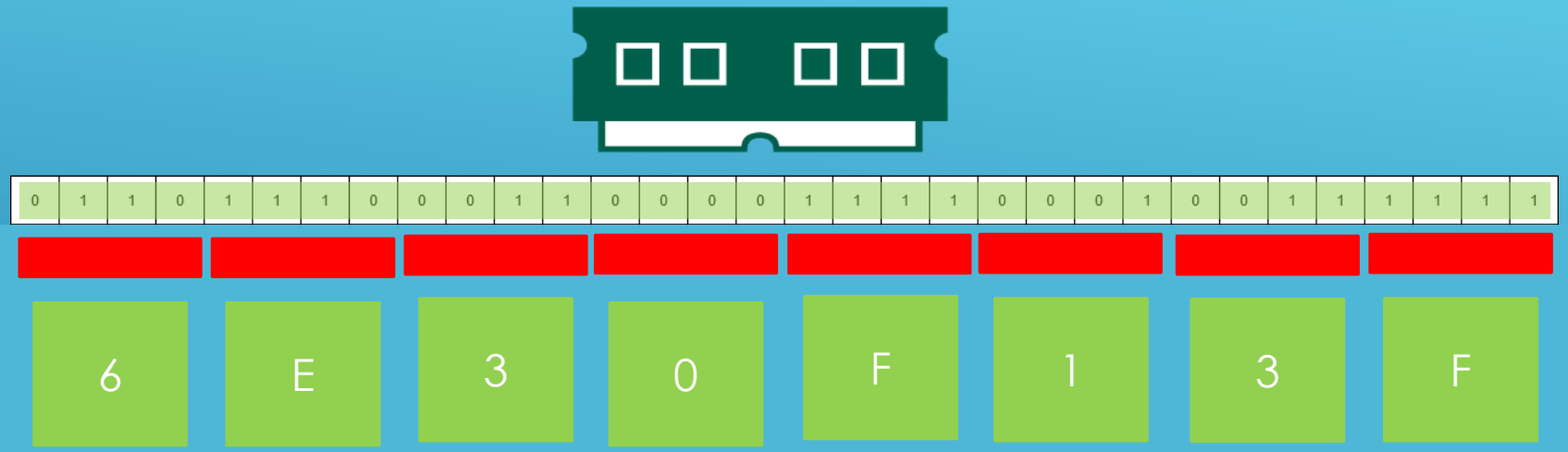
Hexadecimal System

0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

Hexadecimal System

Binary	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Hexadecimal System



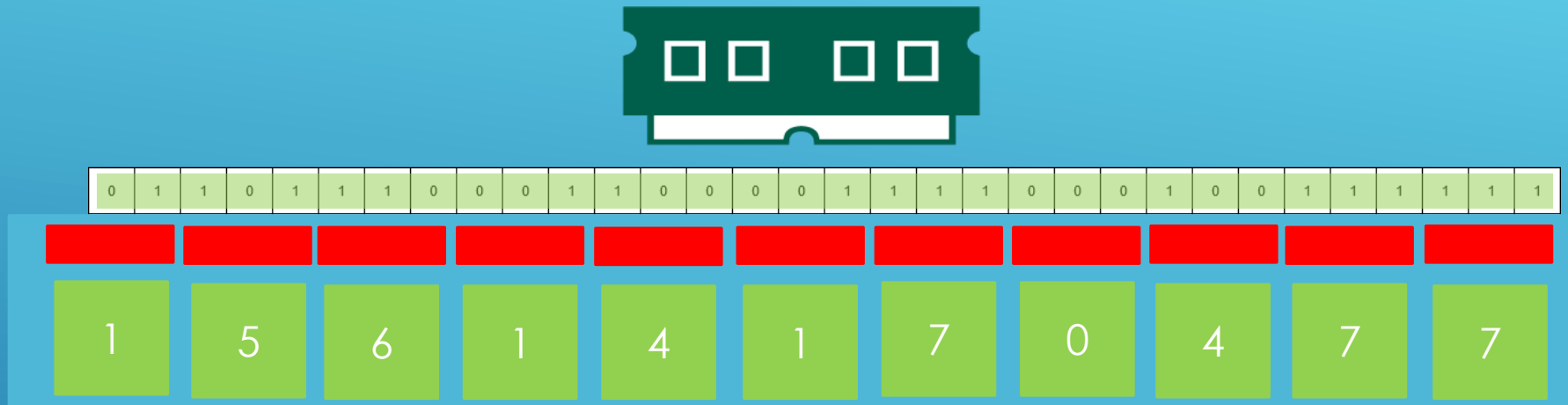
0x 6E30F13F

Padding

Octal System

Binary	Decimal	Oct
000	0	0
001	1	1
010	2	2
011	3	3
100	4	4
101	5	5
110	6	6
111	7	7

Octal System



0 15614170477

Binary Table	https://kb.iu.edu/d/afdl

```cpp
int number1 = 15; //Decimal
int number2 = 017;//Octal.
int number3 = 0x0f; //Hexadecimal
int number4 = 0b00001111; //Binary - C++14

std::cout << "Hello World !" << std::endl;

std::cout << "Number1 is : " << number1 << std::endl;
std::cout << "Number2 is : " << number2 << std::endl;
std::cout << "Number3 is : " << number3 << std::endl;
std::cout << "Number4 is : " << number4 << std::endl;
```

- All data is represented by a bunch of grouped cells of 0's and 1's in memory
- As the range of your data grows, so will the number of digits you need to represent the data in memory
- Hexadecimal system makes it a little easier for humans to handle streams of data with 1's and 0's
- Octal has the same goal as Hexadecimal, but it's almost no longer used in modern times. It's just mentioned here for your awareness

0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

22

23.8

“Steve”

…

Slide intentionally left empty

Integers

0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

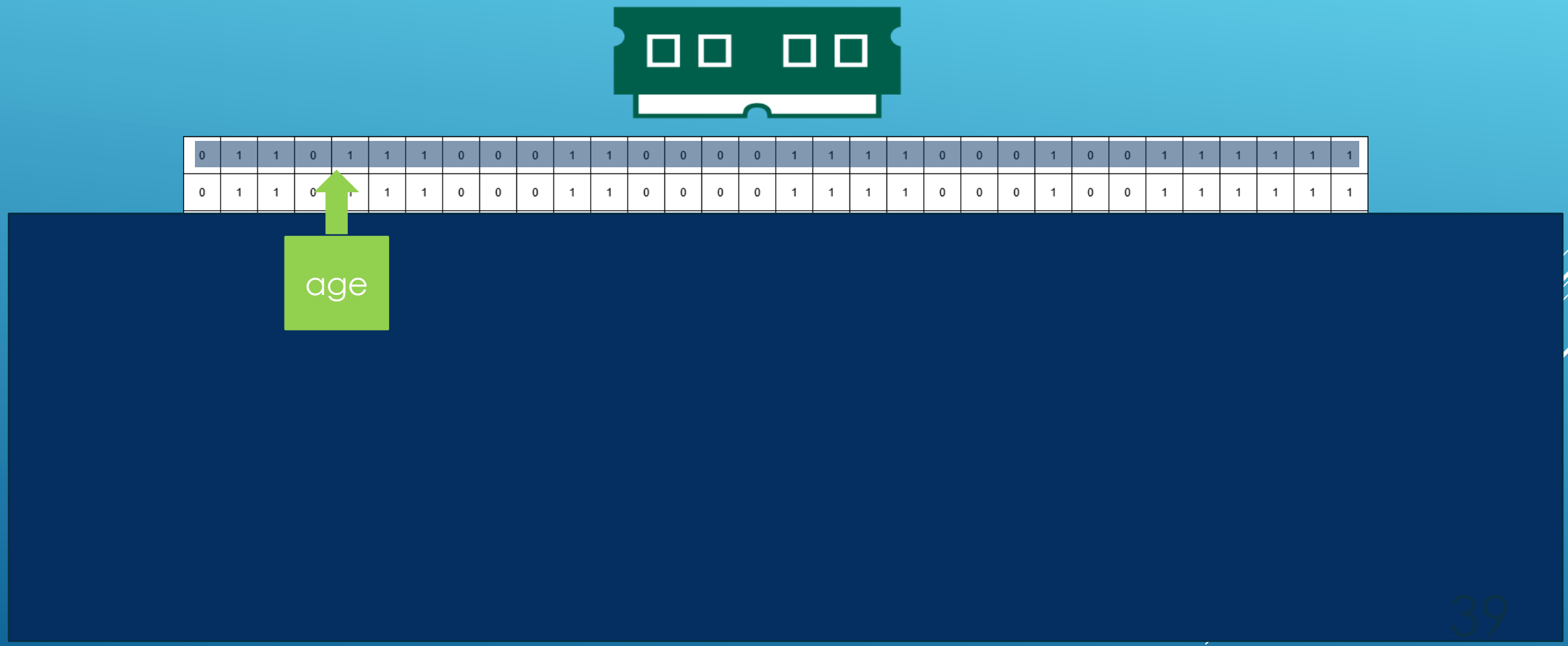
22

37

55939

472299

variable	A named piece of memory that you use to store specific types of data.


0 1 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1 1 1 0 0 0 1 0 0 1 1 1 1 1 1
0 1 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1 1 1 0 0 0 1 0 0 1 1 1 1 1 1
age

int

- Stores decimals
- Typically occupies 4 bytes or more in memory

Variable braced initialization

```
//Variable may contain random garbage value . WARNING
int elephant_count;

int lion_count{};//Initializes to zero

int dog_count {10}; //Initializes to 10

int cat_count {15}; //Initializes to 15

//Can use expression as initializer
int domesticated_animals { dog_count + cat_count };

// Won't compile, the expression in the braces uses undeclared variables
int bad_initialization { doesnt_exist1 + doesnt_exist2 };

// 2.9 is of type double, with a wider range than int.ERROR OR WARNING.
int narrowing_conversion {2.9};
```

Functional variable initialization

```
int apple_count(5);

int orange_count(10);

int fruit_count (apple_count + orange_count);

int bad_initialization ( doesnt_exist3 + doesnt_exist4 );

//Information lost. less safe than braced initializers
int narrowing_conversion_functional (2.9);
```

Assignment Initialization

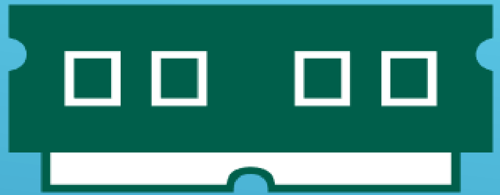
```
int bike_count = 2;

int truck_count = 7;

int vehicle_count = bike_count +truck_count;

int narrowing_conversion_assignment = 2.9;
```

- Braced Initialization
- Functional Initialization
- Assignment Initialization

Size of a type in memory

```
std::cout << "sizeof int : " << sizeof(int) << std::endl;
std::cout << "sizeof truck_count : " << sizeof(truck_count) << std::endl;
```

Int's
in memory



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

```
typename variable_name {initializer_value};
```

Slide intentionally left empty

Integer Modifiers

0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

22

37

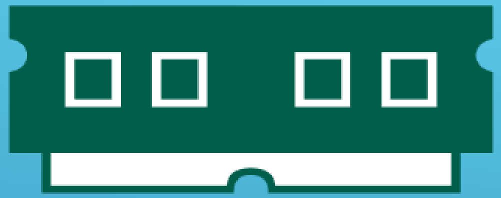
55939

472299

Positive and negative numbers

```
int value1 {10};
int value2 {-300};
```

```
signed int value1 {10};
signed int value2 {-300};
```



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

unsigned range	$[0 \sim 2^n - 1]$
signed range	$[- 2^{n-1} \sim 2^{n-1} - 1]$

* n is the number of bits for a type in memory

Type with modifier	Bytes in memory	Range
unsigned int	4	[0, 4,294,967,295]
signed int	4	[-2,147,483,648, 2,147,483,647]


short
long

```
short short_var {-32768} ; //  2 Bytes
short int short_int {455} ; //
signed short signed_short {122}; //
signed short int signed_short_int {-456}; //
unsigned short int unsigned_short_int {456};

int int_var {55} ; // 4 bytes
signed signed_var {66};//
signed int signed_int {77};//
unsigned int unsigned_int{77};

long long_var {88}; // 4 OR 8 Bytes
long int long_int {33};
signed long signed_long {44};
signed long int signed_long_int {44};
unsigned long int unsigned_long_int{44};

long long long_long {888};// 8 Bytes
long long int long_long_int {999};
signed long long signed_long_long {444};
signed long long int signed_long_long_int{1234};
unsigned long long int unsigned_long_long_int{1234};
```

Note	These modifiers only apply to integral types : those in which you can store decimal numbers

Slide intentionally left empty

Fractional Numbers

0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

22.37

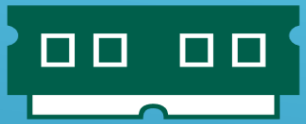
37.076

55.939

4722.99

Floating Point Types

Used to represent numbers with fractional parts in C++



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

22.37

37.076

55.939

4.72299

Type	Size	Precision	Comment
float	4	7	-
double	8	15	Recommended default
long double	12	> double	

Precision

1.23456700001

```cpp
//Declare and initialize the variables
float number1 {1.123456789012345678901234567890f};
double number2 {1.123456789012345678901234567890};
long double number3  {1.123456789012345678901234567890L};

//Print out the sizes
std::cout << "sizeof float : " << sizeof(float) << std::endl;
std::cout << "sizeof double : " << sizeof(double) << std::endl;
std::cout << "sizeof long double : " << sizeof(long double) << std::endl;
```

Precision

```
std::cout << std::setprecision(20); // Control the precision from std::cout.
std::cout << "number1 is : " << number1 << std::endl; //7 digits
std::cout << "number2 is : " << number2 << std::endl; // 15'ish digits
std::cout << "number3 is : " << number3 << std::endl; // 15+ digits
```

Precision gone wrong

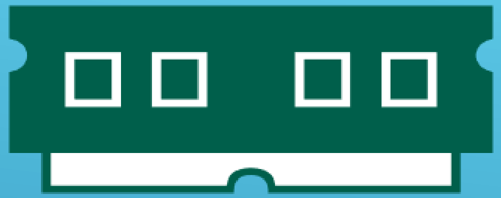
```
float number4 {192400023};
```

Problem not caught at compile time

```
float number4 (192400023);
number4 = number4 + 1;
std::cout << "number4 is : " << number4 << std::endl;
```

Scientific Notation

```
double number5 {192400023};
double number6 {1.92400023e8};
double number7 {1.924e8};
double number8 {0.00000000003498};
double number9 {3.498e-11};
```



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

IEEE_754

Floating Point Numbers Memory Representation

https://en.wikipedia.org/wiki/IEEE_754

n(floating point)/ 0	Infinity(+/-)
0.0/ 0.0	NaN

```
double number10{ 5.6 };
double number11{};//Initialized to 0
double number12{};  //Initialized to 0

//Infinity
double result { number10 / number11 };


std::cout << number10 << "/" << number11 << "  yields " << result << std::endl;
std::cout << result << " + " << number10 << " yields " << result + number10 << std::endl;

//NaN
result = number11 / number12;


std::cout << number11 << "/" << number12 << " = " << result << std::endl;
```

- Remember the suffixes when initializing floating point variables, otherwise the default will be double
- Double works well in many situations, so you will see it used a lot

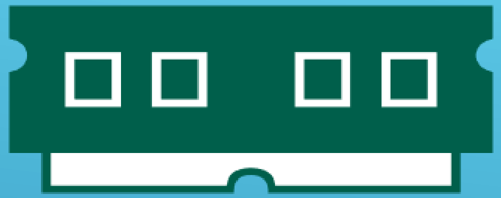
Slide intentionally left empty

Booleans


true
false

```cpp
bool red_light {true};
bool green_light{false};

if(red_light == true){
    std::cout << "Stop!" << std::endl;
}else{
    std::cout << "Go through!" << std::endl;
}


if(green_light){
    std::cout << "The light is green!" << std::endl;
}else{
    std::cout << "The light is NOT green!" << std::endl;
}
```

```cpp
//Printing out a bool
//1 -->> true
//0 -->> false
std::cout << std::endl;
std::cout << "red_light : " << red_light << std::endl;
std::cout << "green_light : " << green_light << std::endl;

//Print out true and false
std::cout << std::endl;
std::cout << std::boolalpha; // Forces the output format to true/false
std::cout << "red_light : " << red_light << std::endl;
std::cout << "green_light : " << green_light << std::endl;
```



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

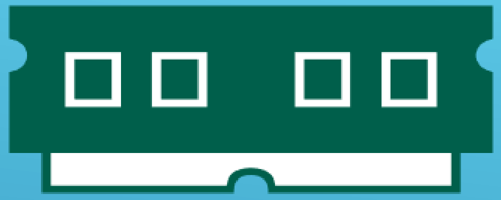
- A byte can store 256 different values
- Using it just to cover two states (true/false) is wasteful, especially for devices with hard memory constraints (think embedded devices)
- There are techniques to pack even more data into a byte . We'll learn more about these in a few upcoming chapters

Slide intentionally left empty

Characters and Text

```
char character1 {'a'};
char character2 {'r'};
char character3 {'r'};
char character4 {'o'};
char character5 {'w'};
```

```
std::cout << character1 << std::endl;
std::cout << character2 << std::endl;
std::cout << character3 << std::endl;
std::cout << character4 << std::endl;
std::cout << character5 << std::endl;
```



0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

1 byte : 256 Possible values	Each matched to some character

ASCII Table

'5' has the int value 53
if we write '5'-'0' it evaluates to 53-48, or the int 5
if we write char c = 'B'+32; then c stores 'b'

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

* Source : https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

It is possible to assign a valid ASCII code to a char variable, and the corresponding character will be stored in. You can choose to interpret that either as a character or a regular integral value

```cpp
char value = 65 ; // ASCII character code for 'A'
std::cout << "value : " << value << std::endl;
std::cout << "value(int) : " << static_cast<int>(value) << std::endl;
```

- ASCII was among the first encodings to represent text in a computer.
- It falls short when it comes to representing languages other than English and a few western languages. Think Arabic, East Asian Languages like Japanese, Chinese ,…
- There are better ways to represent text that is meant to be seen in different languages, one of the most common being Unicode
- The details of Unicode are out of scope for this course, just know that it's a robust way to represent text in different languages for a computer

Slide intentionally left empty

Auto


auto
Let the compiler deduce the type

```
auto var1 {12};
auto var2 {13.0};
auto var3 {14.0f};
auto var4 {15.0l};
auto var5 {'e'};

//int modifier suffixes
auto var6 { 123u}; // unsigned
auto var7 { 123ul}; //unsigned long
auto var8 { 123ll}; // long long
```

```
std::cout << "var1 occupies : " << sizeof(var1) << " bytes" << std::endl;
std::cout << "var2 occupies : " << sizeof(var2) << " bytes" << std::endl;
std::cout << "var3 occupies : " << sizeof(var3) << " bytes" << std::endl;
std::cout << "var4 occupies : " << sizeof(var4) << " bytes" << std::endl;
std::cout << "var5 occupies : " << sizeof(var5) << " bytes" << std::endl;
std::cout << "var6 occupies : " << sizeof(var6) << " bytes" << std::endl;
std::cout << "var7 occupies : " << sizeof(var7) << " bytes" << std::endl;
std::cout << "var8 occupies : " << sizeof(var8) << " bytes" << std::endl;
```

Slide intentionally left empty

Assignments

```cpp
int var1{123}; // Declare and initialize
std::cout << "var1 : "  << var1 << std::endl;

var1 = 55; // Assign
std::cout << "var1 : "  << var1 << std::endl;
```

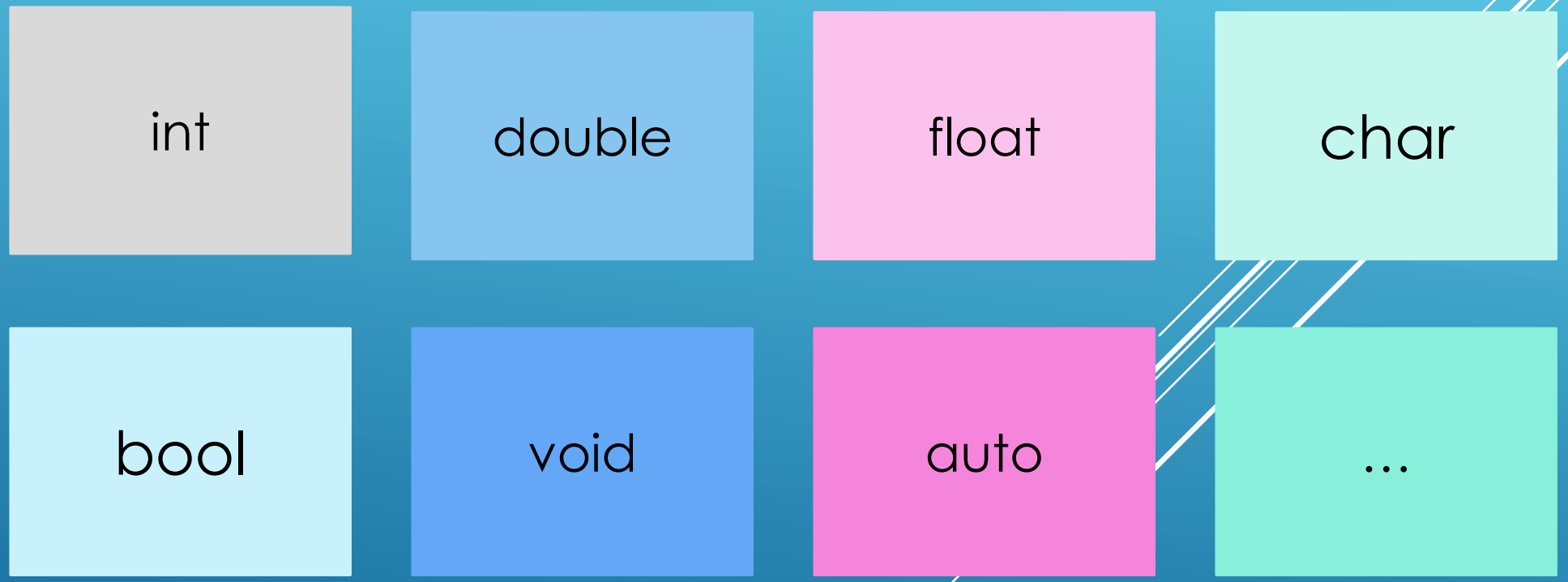
```
double var2 {44.55}; // Declare and initialize
std::cout << "var2 : " << var2 << std::endl;

var2 = 99.99; // Assign
std::cout << "var2 : " << var2 << std::endl;
```

```cpp
bool state{false}; // Declare and initialize
std::cout << std::boolalpha;
std::cout << "state : " << state << std::endl;

state = true; // Assign

std::cout << "state : " << state << std::endl;
```

```
//Careful about auto assignments
auto var3 {333u}; // Declare and initialize with type deduction

var3 = -22; // Assign negative number. DANGER!

std::cout << "var3 : " << var3 << std::endl;
```

Slide intentionally left empty

Variables and data types


int
double
float
char
bool
void
auto
...

0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1

22

23.8

“Steve”

...

Number Systems

- Binary
- Octal
- Hexadecimal

Slide intentionally left empty