```c
#include <stdint.h>
#include <stdio.h>

int8_t is_ascii(char c) {
    return c <= 128;
}

void test_is_ascii(char c) {
    printf("is_ascii(%c %d) = %s\n", c, c, is_ascii(c) ? "true" : "false");
}
void tests_is_ascii() {
    test_is_ascii('a');
    test_is_ascii(0b10000000);
    test_is_ascii(0b01111111);
    for(int i = 0b00000000; i <= 0b11111111; i += 1) {
        test_is_ascii(i);
    }
}

int main() {
    tests_is_ascii();
}
```

synonym for char

```
$ gcc numbers.c -o numbers
$ ./numbers
is_ascii(a 97) = true
is_ascii(€ -128) = true
is_ascii( 127) = true
is_ascii( 0) = true
is_ascii( 1) = true
is_ascii( 2) = true
... lots of output ...
is_ascii(z 122) = true
is_ascii({ 123) = true
is_ascii(| 124) = true
is_ascii(} 125) = true
is_ascii(~ 126) = true
is_ascii( 127) = true
is_ascii(€ -128) = true
is_ascii(€ -127) = true
... lots more output
is_ascii(€ -1) = true
```

char in C is a signed type

The most significant bit has a special meaning

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$ unsigned interpretation

$-2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$ ↳ signed interpretation

0b1000 0000

unsigned: $2^7 * 1 + 0 \ldots = 128$

signed: $-2^7 * 1 + 0 \ldots = -128$

0b1000 0001

unsigned: $2^7 * 1 + 0 \ldots + 2^0 * 1 = 129$

signed: $-2^7 * 1 + 0 \ldots + 2^0 * 1 = -127$

0b0000 0000 0b0111 1111 0b1111 1111

u: 0 1 ... 127 128 129 130 255

s: 0 1 ... 127 -128 -127 -126 .... -1

2's complement representation

64*64

why *64? *4096?

int32_t - what's that?

é has 2 byte UTF-8 rep

```c
#include <stdio.h>
#include <stdint.h>

int32_t code_point2(char c1, char c2) {
    return (c1 & 0b00011111) * 64 + (c2 & 0b00111111);
}

int32_t code_point3(char c1, char c2, char c3) {
    return (c1 & 0b00001111) * 4096 + (c2 & 0b00111111) * 64 + (c3 & 0b00111111);
}

int main() {
    char joseph[] = "Joséph";
    printf("Code point: %d\n", code_point2(joseph[3], joseph[4]));

}
```

233 (unicode cp for é)

2 byte — continuation

é 0b110 00011 | 0b10 101001

masking

& 0b00011111 | 0b00111111

0b00000011 | 0b00101001

*

64

0b11000000 + 0b00101001

0b00011101001 = 233

0b00011 101001

6 bits "to the left"