‘’

Data Type	Size (bytes)	Range
int	**2 or 4**	**-32,768 -to 32,767**
float	4	-3.4×10^{-38} to 3.4×10^{38}
double	8	-1.7×10^{-308} to 1.7×10^{308}
char	**1**	**-128 to 127**
bool	undefined	true/false

C++ Basics (Abdul Bari)

Wills Liou

today

Abstract

Summary of C++ Basics by Abdul Bari (Udemy); This took a really long time.

cout ¡¡ "Hello World" ¡¡ endl; #include ¡iostream¿ iostream is a library cout stands for "console out" COUT and CIN are OBJECTS

¡¡ is an insertion operator :: is a scope resolution

= is an "ASSIGNMENT OPERATOR" == is a "comparison operator"

¡ is less than 5 ¡ 3 five is less than 3

¿ is greater than 2 ¿ 1 2 is greater than 1

Program is set of two ingredients:

1. data 2. operations performed on data

1 byte = 8 bits

Maximum integer in 1 byte (8 bits) is 255.

Formula for maximum binary value for N bits

Formula to calculate maximum decimal value for N bits.

$$M = 2^N - 1$$

Example:

$$255 = 2^8 - 1$$

25. Primitive Data Types

16 15

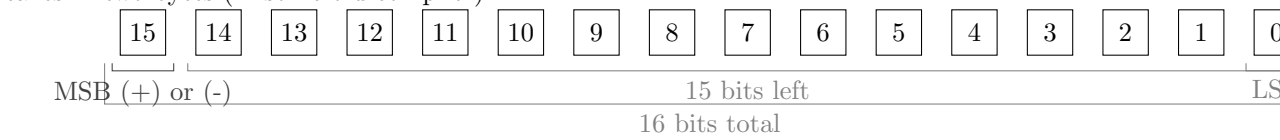
How to memorize range for ?? floats and ?? doubles
Realize that doubles have an extra **zero** in the power
floats are 10^{38} and doubles are 10^{308}

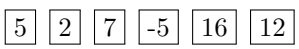
Float range: -3.4×10^{-38} to 3.4×10^{38}
Double range: -1.7×10^{-308} to 1.7×10^{308}
multiply double's range and get float's number;
$1.7 \times 2 = 3.4$ Other data types: 1. long 2.long long (for really long decimals)

1 Proving the int range is -32,768 -to 32,767

This is our array for 2 bytes (16 bits)
Assume the numbers inside each index are it's index value
Notation: Most significant byte (MSB), Least significant byte (LSB)
-ve stands for negative +ve stands for positive

Known: 1 byte = 8 bits.
If we have two bytes, we will have 16 bits. Let's assume that our int data type takes in two bytes (in some old compiler).





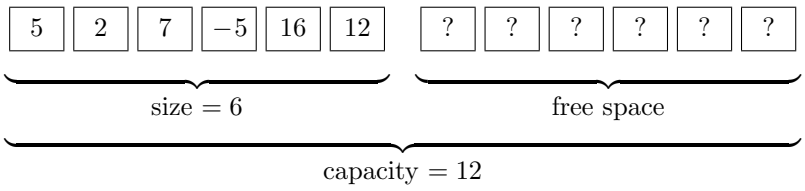
1. Assume 2 bytes (16 bits)
2. Most significant bit (MSB) reserves the first spot for the sign; In other words, MSB takes one spot for a (1) -ve or (0) +ve
3. Excluding the MSB, there are 15 bits left for the integer.
4. Therefore $2^{15} = 32768$
   (a) $32,768$ is the maximum number from the *range 1 to 32,768*
   (b) So, our int data type can take in a max number, 32,768.
5. If we want a range starting from 0, our range is now *range of 0 to 32,767*
6. Range visually represented as: 0 ... 32,767
7. Reminder: we know that one bit is reserved at MSB (first index) for positive and negative signs.
8. So if we include the negative side from -32767 to -0, our range looks like this

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

9. -32,767 -0 0 32,767
10. But realize that -0 does not exist; we add one more value to the negative side to account for our -0
11. -32,767 + 1 0 32,767
12. -32,768 0 32,767
13. Therefore our int data type range is from *-32,768 to 0 to 32,767*

Question at step #9: How do we have enough memory for the negative side if our maximum number is 32,768?
Answer: Look at two's complement; when we reach our one more than maximum value of 32,768, (e.g. 32,769), we actually overflow and get -1.
Two's Complement

In practice, it looks like this:



2 Proving the char range is -128 to 127

1. The data type, char, is 1 byte or 8 bits.
2. We reserve the first bit (MSB) for the positive and negative signs.
3. Therefore, we are left with 7 bits.
4. So we have $2^7 = 128$
   (a) our values for char is the range *1 to 128*
5. When we begin at 0, our range becomes *0 to 127*
6. To account for the negative side, we have *-127 -0 0 127*
7. To fix the -0, we add one more value to our negative side, our final range is *-127 0 127*

This had many similiar to our int range proof.

For unsigned(only positive) integers, add back the first bit that stores the negative and positive values. Aka. use all 16 bits (2 bytes in integer data type) One way:

1. Instead of $2^{15} = 32,768$, we will have $2^{16} = 65,536$.
2. With a range starting from 0, *0 to 65,535*
3. Therefore, unsigned int has a range from *0 to 65,535*

Other way: Add the min and max values of an int range *-32,768 to 0 to 32,767*

1. $32spoq, 768 + 32,767 = 65,535$
2. Therefore, unsigned int has a range from *0 to 65,535*

3 *** IMPORTANT REMEMBER THESE: ***

a = 97 ... z = 122; 0 -¿ 49 ... 9 -¿ 57

A = 65 a = 97 '0' = 48
B = 66 b = 98 '1' = 49
.
.
.
Z = 90

Data Types

Data Type
-37,768 to 37,767 (15 bits used with 1 reserved)
char
0 to 65,535 (all 16 bits or 2 bytes are used) unsigned char
if int is 2 bytes, then long int is 4 bytes or (if int is 4 bytes, then long int is 8 bytes long float (does not exis