

C*versus CIntegerLiterals

- C* integer literals are unsigned 64-bit
- C integer literals are signed 32-bit
- For example, $1 / -1 == 0$ in C* but $1 / -1 == -1$ in C
- And, $1 \% -1 == 1$ in C* but $1 \% -1 == 0$ in C
- Also, $1 < -1$ and $1 <= -1$ hold in C* but $1 > -1$ and $1 >= -1$ do not whereas the opposite is true in C
- The semantics of $/$ and $\%$ as well as $<$, $<=$, $>$, and $>=$ is different for signed and unsigned integers!

C* Characters and Strings

- C* characters are ASCII-encoded.
- C* character literals are characters in code like 'c'.
- C* strings are stored as null-terminated sequences of characters.
Alternatively the end of a string could be identified by storing the number of characters at its beginning.
- C* string literals are strings in code like "this".
- The difference between 'a' and "a".



ascii representation
(**numerical value**)
in memory



pointer to first
word of where the
string is stored

C* versus C Integer Literals

- C* integer literals are unsigned 64-bit
- C integer literals are signed 32-bit
- For example, $1 / -1 == 0$ in C* but $1 / -1 == -1$ in C
- And, $1 \% -1 == 1$ in C* but $1 \% -1 == 0$ in C
- Also, $1 < -1$ and $1 \leq -1$ hold in C* but $1 > -1$ and $1 \geq -1$ do not whereas the opposite is true in C
- The semantics of $/$ and $\%$ as well as $<$, \leq , $>$, and \geq is different for signed and unsigned integers!