UCLA

CS 31:
Introduction To Computer Science I

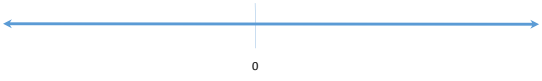
Howard A. Stahl

Understanding Numeric Representations

- Pardon The Slight Diversion Into The Realm Of Mathematics…

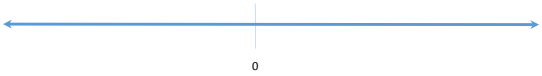
Let's Go Visit The Number Line

- Please Think About All The Whole Numbers On The Number Line



Let's Go Visit The Number Line

- Please Think About All The Whole Numbers On The Number Line
- Please Answer The Question:
  How Many Whole Number Are There On The Number Line?



Let's Go Visit The Number Line

- Please Think About All The Whole Numbers On The Number Line
- Please Answer The Question:
  How Many Whole Number Are There On The Number Line?
  The Answer: There Are An INFINITE Number Of Whole Numbers On The Number Line

Let's Go Revisit The DataType Chart

- This Comes From The Textbook

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED
short (also called short int)	2 bytes

Let's Go Revisit The DataType Chart

- This Comes From The Textbook

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED
short (also called short int)	2 bytes

- 1 Byte = 8 bits 2 Bytes = 16 bits

Let's Go Revisit The DataType Chart

- This Comes From The Textbook

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED
short (also called short int)	2 bytes

- 1 Byte = 8 bits 2 Bytes = 16 bits
- How Can 16 Bits Be Used To Store An Infinite Number Of Different Possibilities?

Let's Go Revisit The DataType Chart

- This Comes From The Textbook

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED
short (also called short int)	2 bytes

- 1 Byte = 8 bits 2 Bytes = 16 bits
- How Can 16 Bits Be Used To Store An Infinite Number Of Different Possibilities?
  The Answer: **It CAN'T!**

Let's Go Revisit The DataType Chart

- This Comes From The Textbook

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED
short (also called short int)	2 bytes

- 1 Byte = 8 bits 2 Bytes = 16 bits
- How Can 16 Bits Be Used To Store An Infinite Number Of Different Possibilities?
  The Answer: **It CAN'T!** 16 bits Can Only Hold 2^16 – 1 Different Values

Let's Go Revisit The DataType Chart

- This Comes From The Textbook

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED
short (also called short int)	2 bytes

- 1 Byte = 8 bits 2 Bytes = 16 bits 2 ^ 16 – 1 = 65,535
- How Can 16 Bits Be Used To Store An Infinite Number Of Different Possibilities?
  The Answer: **It CAN'T!** 16 bits Can Only Hold 2^16 – 1 Different Values

Let's Go Revisit The DataType Chart

- This Comes From The Textbook

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED	SIZE RANGE
short (also called short int)	2 bytes	–32,768 to 32,767

- 1 Byte = 8 bits 2 Bytes = 16 bits 2 ^ 16 – 1 = 65,535
- How Can 16 Bits Be Used To Store An Infinite Number Of Different Possibilities?
  The Answer: **It CAN'T!** 16 bits Can Only Hold 2^16 – 1 Different Values

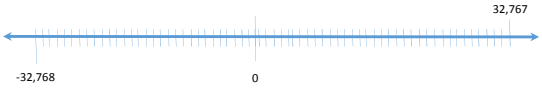
Let's Go Revisit The DataType Chart

- This Comes From The Textbook

- 1 Byte = 8 bits 2 Bytes = 16 bits 2 ^ 16 – 1 = 65,535
- How Can 16 Bits Be Used To Store An Infinite Number Of Different Possibilities?
  The Answer: **It CAN'T!** 16 bits Can Only Hold 2^16 – 1 Different Values

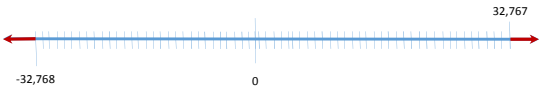
Working With `short`

- Only A Portion Of The Number Line Is A Possibility
- However, Each Value Can Be Represented Exactly



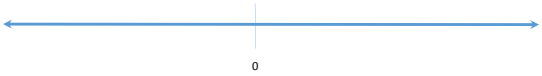
Working With `short`

- Only A Portion Of The Number Line Is A Possibility
- However, Each Value Can Be Represented Exactly
- What About The Red Part Of The Number Line?



Working With Floating Point Values

- Please Think About All The Real Numbers On The Number Line



Working With Floating Point Values

- Please Think About All The Real Numbers On The Number Line
- Please Answer The Question:
  How Many Real Numbers Are There Between 0 And 1 On The Number Line?

Working With Floating Point Values

- Please Think About All The Real Numbers On The Number Line
- Please Answer The Question:
  How Many Real Numbers Are There Between 0 And 1 On The Number Line?
  Answer: An Infinite Number Of Real Numbers Exist Between 0 And 1

Let's Go Revisit The DataType Chart

- This Comes From The Textbook

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED
float	4 bytes

- 1 Byte = 8 bits 4 Bytes = 32 bits
- How Can 32 Bits Be Used To Store An Infinite Number Of Different Possibilities?

Let's Go Revisit The DataType Chart

- This Comes From The Textbook

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED
float	4 bytes

- 1 Byte = 8 bits 4 Bytes = 32 bits
- How Can 32 Bits Be Used To Store An Infinite Number Of Different Possibilities?
  The Answer: **It CAN'T!**

Let's Go Revisit The DataType Chart

- This Comes From The Textbook

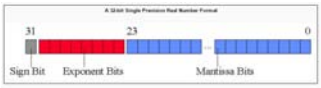
Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED	SIZE RANGE
float	4 bytes	approximately 10^{-38} to 10^{38}

- 1 Byte = 8 bits 4 Bytes = 32 bits
- How Can 32 Bits Be Used To Store An Infinite Number Of Different Possibilities?
  The Answer: **It CAN'T!**

IEEE Floating Point Formats

- A 32-Bit Floating Point Value Has Three Components



- A Sign Bit: High-Order Bit, Set To Represent A Negative Number
- The Exponent: Represents The Location Of The Decimal Point
- The Mantissa: Represents All The Significant Digits

An Example

- 1.2345678901234567890
- 12.345678901234567890
- 123.45678901234567890
- 1234.5678901234567890
- 12345.678901234567890

Working With `float`

- Only 32 Bits For Any Given Value
- 1 Bit Reserved For The Sign Bit
- 8 Bits Reserved For The Exponent
  - The Location Of The Decimal Point
- JUST 23 Bits Reserved For All The Significant Digits

Working With `float`

- Floating Point Values Will Always Be Approximations
  - Sometimes, Better Than Others…
- Only The First Leading Digits Will Be Correct…

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED	SIZE RANGE	PRECISION
`float`	4 bytes	approximately 10^{-38} to 10^{38}	7 digits

An Example

- 1.2345678901234567890
- 12.345678901234567890
- 123.45678901234567890
- 1234.5678901234567890
- 12345.678901234567890