


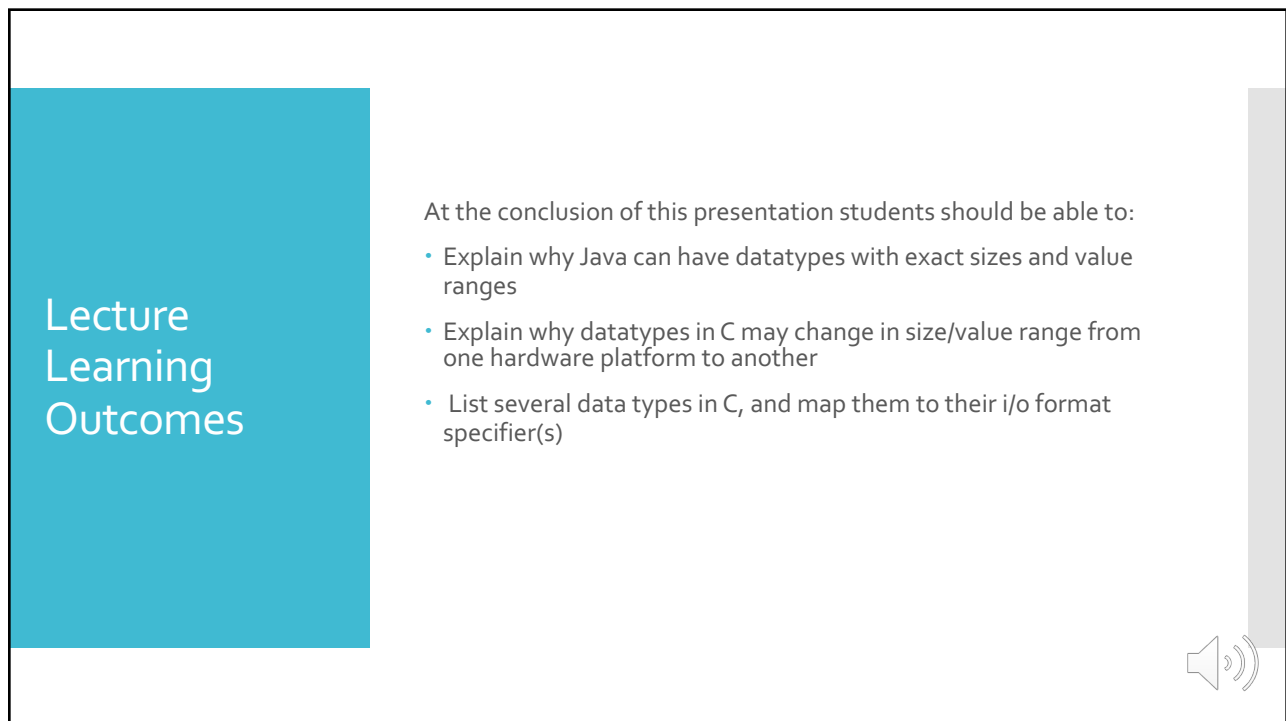

The slide features a large blue rectangle on the right side containing the title 'Datatypes' in white. To the left of this rectangle is a vertical grey bar. Below the blue rectangle is a dark grey bar containing the text 'CS2263 – Systems Software Development' in white. A small speaker icon is located in the bottom right corner of the slide.

Datatypes

CS2263 – Systems Software Development



1




The slide has a blue rectangle on the left side containing the text 'Lecture Learning Outcomes' in white. To the right of this rectangle is a vertical grey bar. The main content area on the right contains a list of learning outcomes. A small speaker icon is located in the bottom right corner of the slide.

Lecture Learning Outcomes

At the conclusion of this presentation students should be able to:

- Explain why Java can have datatypes with exact sizes and value ranges
- Explain why datatypes in C may change in size/value range from one hardware platform to another
- List several data types in C, and map them to their i/o format specifier(s)



2

What Are Datatypes?

"a particular kind of data item, as defined by the values it can take, the programming language used, or the operations that can be performed on it" —OED

What we think they are

- Places to store values
- Types correspond to things we're familiar with
 - Integers
 - Floating point values
 - characters

What they also are

- Specific sizes (bytes)
- Ways of interpreting the bits
 - (signed/unsigned) integers
 - IEEE 754 floating point
 - Characters
 - US-ASCII
 - Extended ASCII
 - UTF-8, UTF-16, UTF-32
 - EBCDIC



3

Implications for Java

- Java runs on a virtual machine
- The virtual machine is standard across all hardware platforms (that it runs on)
 - Ability to standardize the datatypes across the hardware platforms



4

Implications for C

- C runs on bare hardware
- C can only support the datatypes directly supported by the hardware
- Although the representation of datatypes can be standardized, the storage sizes cannot, since the hardware can limit representation
- Range of values can differ across hardware platforms
 - `<limits.h>`
 - `<float.h>`
 - Located in
 - `/usr/include/limits.h`
 - `/usr/include/float.h`
 - preprocessor macro `sizeof()` can also help



5

Why Might These Things Matter?

Consider I/O

- Printing to the terminal: conversion of binary to characters
 - Consider `first.c`:
 - `'%d'` specifies a conversion from decimal (integer representation) to character
- Reading from the keyboard: conversion of characters to binary
 - Consider: `scanf("%d", &value)`
 - `'%d'` specifies a conversion from character to decimal (integer representation)

```
// first.c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char ** argv)
{
    int a = 5;
    int b = 17;
    printf("main: a = %d, b = %d, argc = %d\n", a, b, argc);
    return EXIT_SUCCESS;
}
```



6

What About Other Datatypes?

- Check the L2Resources for links to a complete list.
- Here are common specifiers
 - `%d` decimal integer
 - `%f` float
 - `%lf` double
 - `%c` character
 - `%s` string
 - `%p` pointer
 - `%x` hexadecimal
 - `%o` octal
- As well, the exact display format can be specified
 - `%4d` display 4 digits (`7`)
 - `%7.2f` display 7 digits, to two "decimal places" (`9999.99`)



7

Notes about scanf () Format String

- Whitespace is a separator; multiple whitespaces are a separator:
 - Blanks, tabs, linefeeds
- Ordinary characters (not `%`) are expected to match next non-white space character
 - `scanf("%d,%lf", &i, &d);`
- Input type must be consistent with the conversion specifications (e.g. `%d`)



8

Return Values

- `printf`: Number of characters printed
- `scanf`: Number of arguments successfully converted, or EOF if no items are read and EOF has been encountered
- How to signal EOF?
- Consider `scanf("%d%d", &i, &j)`



9

Idioms: (formulaic language)

- Read a single value:


```
n = scanf("%d", &i);
if(n != 1){
    printf("invalid
    int\n");
    return
    EXIT_FAILURE;
}
```

- Read until end-of-file (EOF)

```
n = scanf("%d", &i);
while(n == 1){
    /* Do stuff */
    n = scanf("%d",
    &i);
}
```



10

forNextDay()

- Write a short C program that prints out the integer value 65 as `%d`, `%4d`, `%x`, `%o`, and `%c` (you'll need to cast it to a `char`)
- Write another short C program that declares a `char`, `int`, `float`, and `double`, and then prints out their sizes (as an integer) using the `sizeof()` macro. What do these values mean?
- Look at the `<limits.h>` include file on your lab machine and report the minimum and maximum values for a `signed int`.

