

# SPLICE: Self-Paced Learning in an Inverted Classroom Environment

Matt Boutell and Curt Clifton

Computer Science and Software Engineering, Rose-Hulman Institute of Technology

## Problem

Learning to program is hard. Students benefit from the help of an expert coach. Finding **time** for this is an issue, because we must also present concepts, show examples, and model problem solving. **Pace** is also an issue because some of our students arrive with confidence and prior experience, while others become overwhelmed.

## Solution?

We created **video modules** for a C programming unit. Students watch the videos to learn new concepts, see concrete examples of them, and observe an expert solving real problems. You can't do that in a book!

Students stay engaged while watching the videos by solving on-paper **active learning problems** and doing **follow-along coding exercises**.

## Every Class a Lab

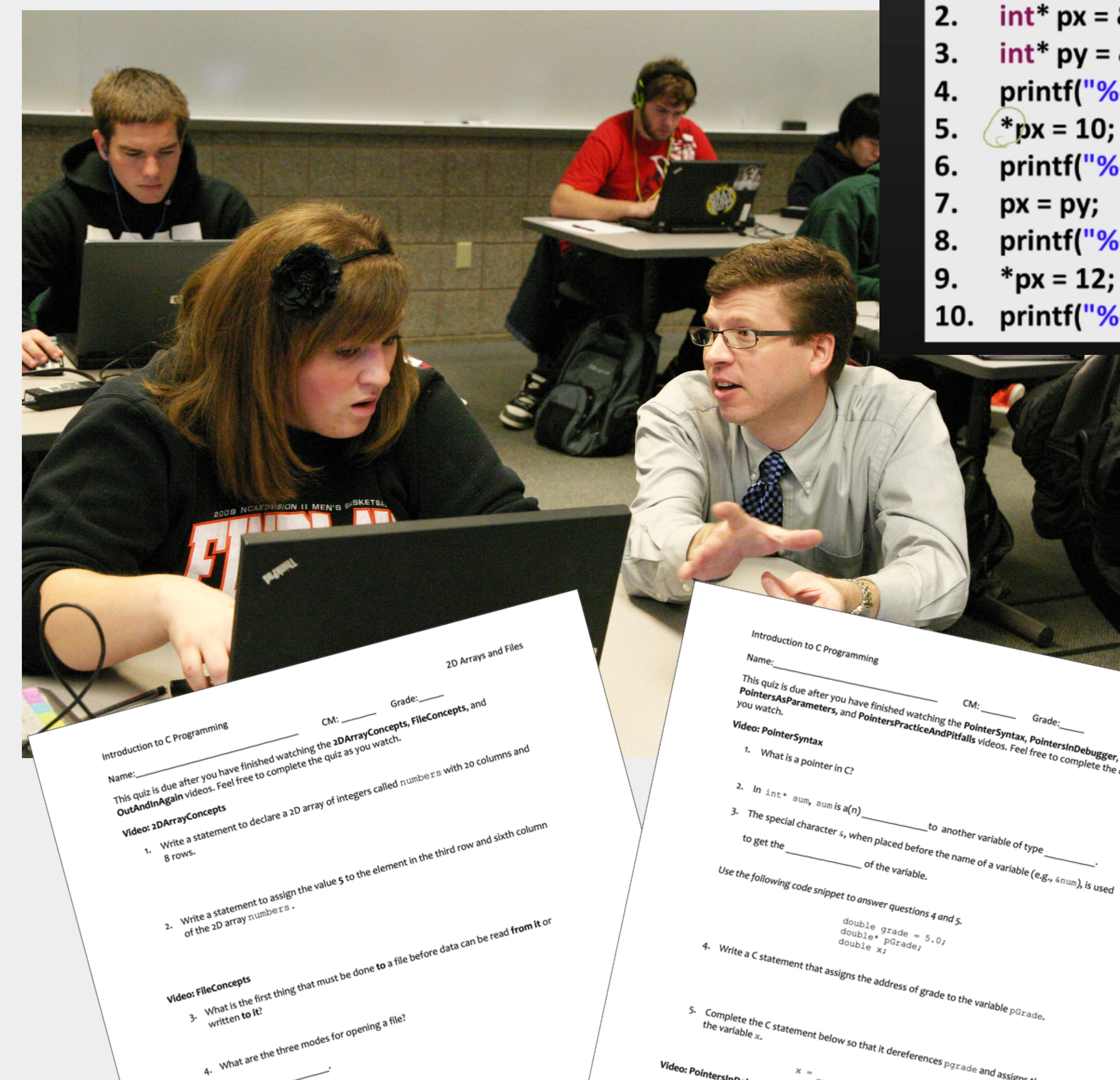
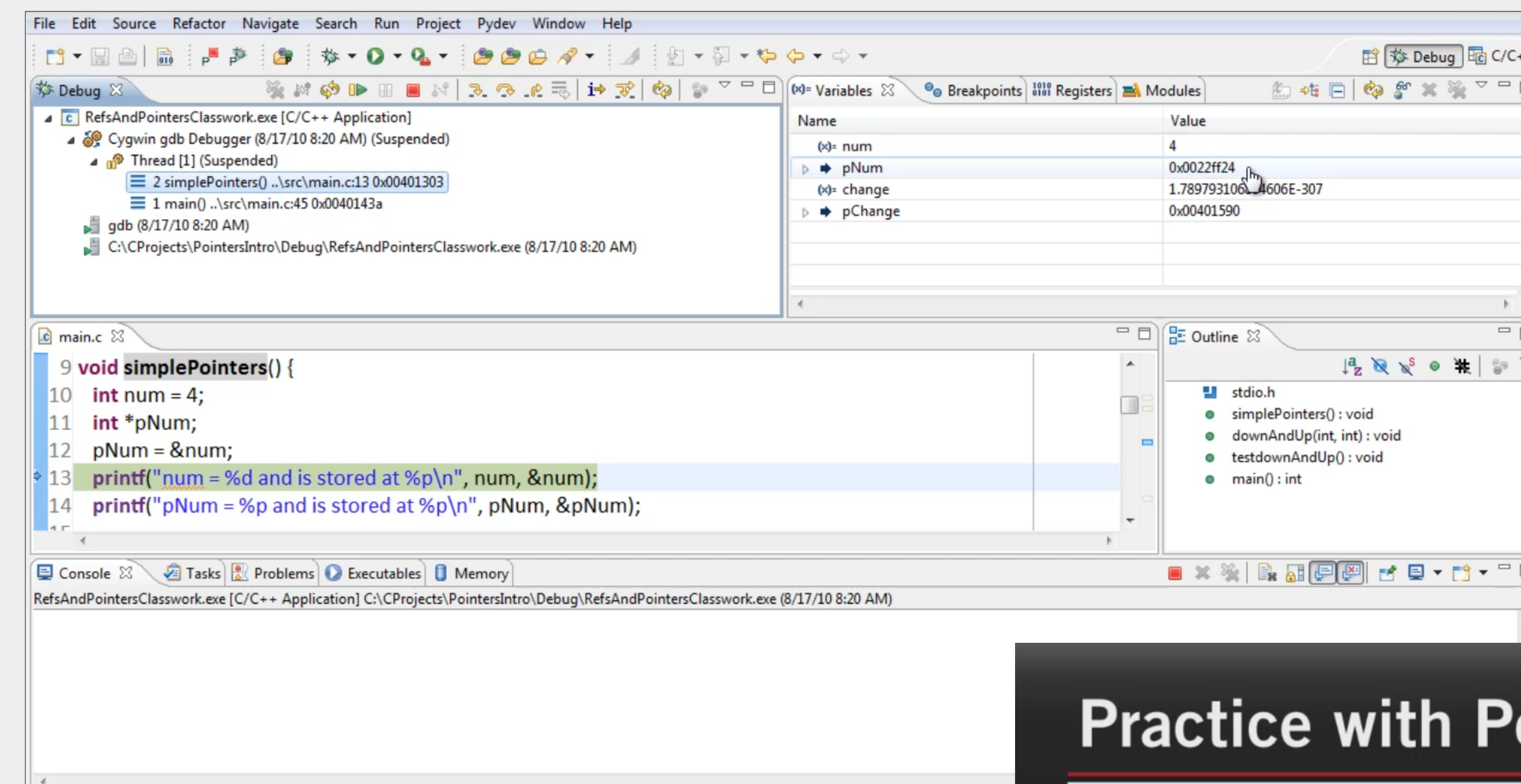
Our students come to class primed to **put what they've learned into practice**. We open the floor for questions at the start of class. Students then spend the rest of the class period solving problems either alone or in pairs. Students get expert coaching, receive more individual attention, and set their own pace. Students focus on **confronting their misconceptions** because we are immediately able to help with tool problems and obscure error messages.

## Assessment Plan

We are interested in these questions:

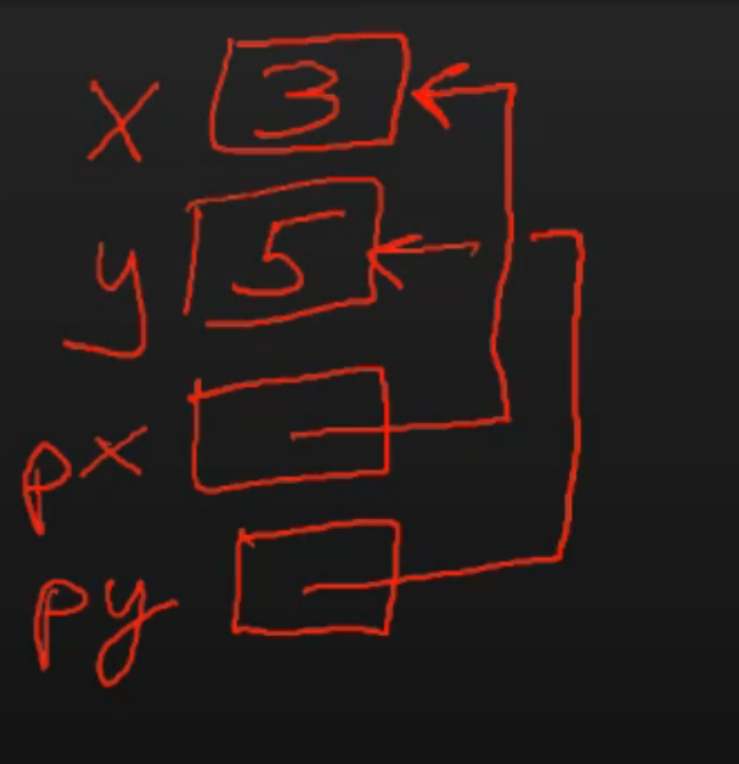
- How does the new format affect **perceptions** of class time and instructor effectiveness?
- How does the new format affect **performance** on assignments and exams?
- Do assignment and exam performance correlate with **video viewing behavior**?
- Do students in the lower half of the class **benefit** more from extra examples and mentored practice?

We are assessing nine sections of CS1 this year—four taught traditionally and five via videos—using **surveys**, **grade data**, and **video download logs**.



### Practice with Pointers

1. `int x = 3, y = 5;`
2. `int* px = &x;`
3. `int* py = &y;`
4. `printf("%d %d\n", x, y);`
5. `*px = 10;`
6. `printf("%d %d\n", x, y);`
7. `px = py;`
8. `printf("%d %d\n", x, y);`
9. `*px = 12;`
10. `printf("%d %d\n", x, y);`



## What We Did

We began with classroom-tested examples and slides, but simplified the slides to reduce distraction.

We created short video modules to:

- **Introduce concepts** and
- **Model the problem solving process**

To help students wanting more practice, we created extra problem solving videos beyond what we assigned.

Our recording environment included:

- Camtasia Studio
- PowerPoint
- Eclipse
- Wacom Cintiq interactive pen display

We produced **233 minutes of video**, which we are using for the final third of CS1.

To produce each video, we:

- Converted existing in-class quizzes
- Created presentation slides
- Recorded separate video segments for slides and coding
- Combined and edited video segments, adding music and credits (sometimes re-recording short segments)
- Rendered MPEG-4 videos for posting

**Fifteen minutes of finished video took about 3 hours** of effort from initial planning to final production.

## Current Status

Feedback from three video-based sections was mixed. Instructors all found it beneficial. One wrote, "**They were a HUGE win for me** and fit my style of teaching perfectly...I don't intend to do anything live in the spring in C." Students liked the **convenience**, setting their own **pace**, and the additional work **time** in class. However, they also disliked **not getting immediate answers** to their initial questions.

## Future Work

In the coming months, we will continue to collect assessment data and will post the videos to YouTube.

Next, we plan to disseminate the results of the first year's study and add closed captioning to the videos.

Future directions include inverting a full course, trying a fully self-paced version, and assessing the effectiveness of videos as review material.

## References

- Carlisle, M., 2010. Using YouTube to enhance student class preparation in an introductory Java course. *SIGCSE Bull.*, Milwaukee, WI, Mar. 2010.
- Day, J. and Foley, J. 2006. Evaluating a web lecture intervention in a human-computer interaction course. *IEEE T. Educ.* 49, 4 (Nov. 2006), 420-431.
- Gannod, G. C., Burge, J. E., and Helmick, M. T. 2008. Using the inverted classroom to teach software engineering. *30th Proc. Int. Conf. Software Eng.*, Leipzig, Germany, May 2008, 777-786.
- Kaner, C. and Fiedler, R. 2005. Blended learning: A software testing course makeover. *11th Sloan-C Int. Conf. on Asynchronous Learning Networks*, Orlando, FL, Nov. 2005.
- Lage, M., Platt, G., and Treglia, M. 2000. Inverting the classroom: A gateway to creating an inclusive learning environment. *J. Econ. Educ.* 31, 1 (Winter, 2000), 30-43.

## Acknowledgments

This work was funded by a Rose-Hulman Summer Professional Development grant. Thanks to Claude Anderson, Delvin Defoe, Dave Fisher and David Mutchler for work on the materials from which we developed our videos.

