Random Numbers

This week

- **Project 2 Interviews:** until Nov 4th
- **Homework 7: Nov** 4th
- **Quiz 7:** due Sunday, **Nov 7th**

- **Project 3 - CYO**
  - Design Meeting: Nov 7th – Nov 9th
  - Code Skeleton due on **Nov 10th**

Undergrads, join our community!

Apply to be a Learning Assistant (LA) in Spring 2023

Get paid to collaborate with faculty to support in-class student learning and success!

Engage in training and support to help you do this important, fun, and challenging work!

Hiring Departments:

APPM	CSCI	MATH
ASTR	COMM	MCDB
BCHM	ENGL	NRSC
BCOR	ENVS	PSYC
CHEM	GRMN	PHYS
	HIST	WRTG

*Some appointments in Continuing Education Online Learning

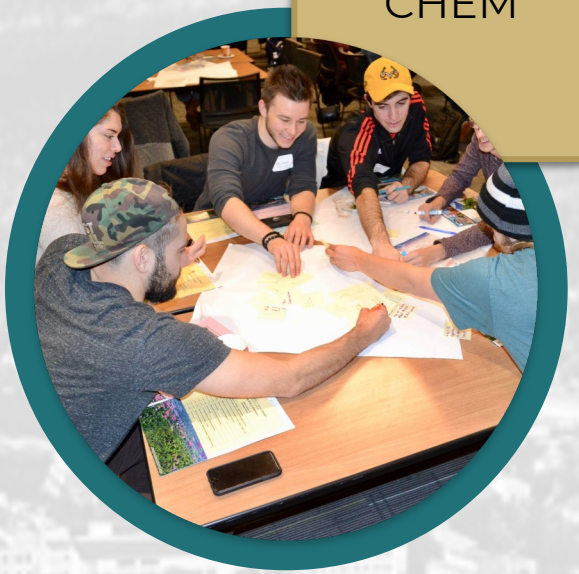
Applications open:

October 28, 2022

Priority application deadline:

November 7, 2022

for full consideration



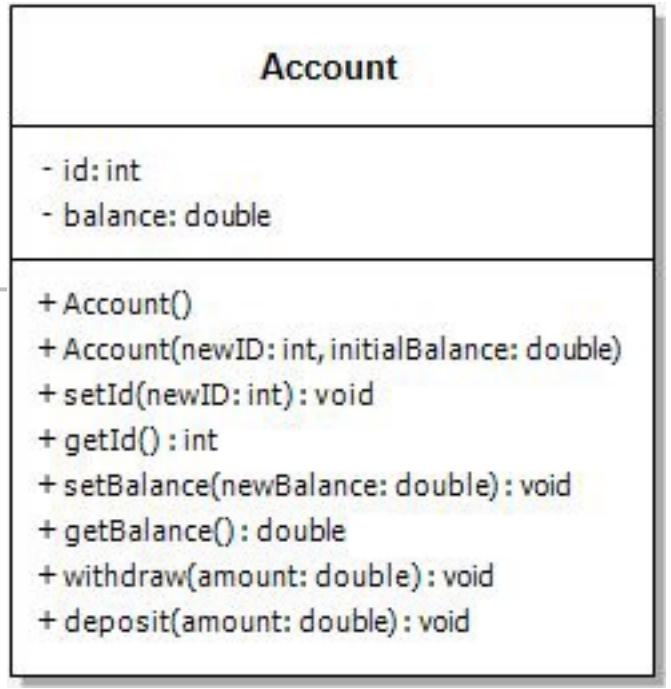
Scan me to learn more!

Learning Assistant Program

UNIVERSITY OF COLORADO **BOULDER**

https://www.colorado.edu/program/learningassistant/

Design Meetings



- Your idea: what's the story (at least a written outline)
- **Classes skeleton:**
  - Data members
  - Member function stubs
- **Main class:** "Game, World, Universe, …"
- How you will meet requirements
  - What will you read from a file?
  - Where do the data come from/look like?
  - What will you write to a file?
- **Code skeleton: pseudocode**
  - Game flow -- what functions?
- Optional: Class diagrams

Objectives

- To understand the potential applications of simulation as a way to solve real-world problems.
- To understand pseudorandom numbers and their application in simulations.
- To understand and be able to apply top-down and spiral design techniques in writing complex programs.

Based on “racquetball” simulation: Zelle, *Python Programming*, 3e

Simulating Tennis Game

- *Simulation* can solve real-world problems by modeling real-world processes to provide otherwise unobtainable information.

- What are simulations used for?

Simulating Tennis Game

- *Simulation* can solve real-world problems by modeling real-world processes to provide otherwise unobtainable information.

- Computer simulation is used to predict the weather, design aircraft, create special effects for movies, etc.

A Simulation Problem

- Susan often plays tennis with players who are slightly better than she is.
- Susan usually loses her matches!
- Shouldn't players who are *a little* better win *a little* more often?
- Michelle suggests that they write a simulation to see if slight differences in ability can cause such large differences in scores.

Analysis and Specification

- Tennis is played between two players using a racquet to hit a ball in a court, across the net.
- One player starts the game by putting the ball in motion – *serving*.
- Players try to alternate hitting the ball to keep it in play, referred to as a *rally*. The rally ends when one player hits a shot out, or in the net.

Analysis and Specification

- Players alternate serve. First to win 6 games wins the set.
- There must be at least a 2 games gap. 6-5 would not be a valid set score.
- Tiebreak, or 2 games advantage, i.e 8-6
- The first player to win 3 sets wins the match

Analysis and Specification

- In our simulation, the ability level of the players will be represented by the probability that the player wins their serve when they serve.
- Example: Players with a 0.60 probability win 60% of their serves.
- The program will prompt the user to enter the service probability for both players and then simulate multiple matches of tennis.
- The program will then print a summary of the results.

Analysis and Specification

Input:

The program prompts for and gets the service probabilities of players A and B. The program then prompts for and gets the number of matches to be simulated.

Analysis and Specification

Output: The program then prints out a nicely formatted report showing the number of games simulated and the number of wins and the winning percentage for each player.

```
Matches simulated: 500
Wins for A: 268 (53.6%)
Wins for B: 232 (46.4%)
```

Notes:

- All inputs are assumed to be legal numeric values, no error or validity checking is required.
- In each simulated game, player A serves first.

PseudoRandom Numbers

- When we say that player A wins 50% of the time, that doesn't mean they win every other game. Rather, it's more like a coin toss.
- Overall, half the time the coin will come up heads, the other half the time it will come up tails, but one coin toss does not effect the next (it's possible to get 5 heads in a row).

PseudoRandom Numbers

- A similar approach is used to generate random (technically *pseudorandom*) numbers.
- A pseudorandom number generator works by starting with a *seed* value. This value is given to a function to produce a “random” number.
- The next time a random number is required, the current value is fed back into the function to produce a new number.

PseudoRandom Numbers

What do we need:

```
#include <stdlib.h>   // for the rand() function
#include <time.h>    // for time() function

rand()
srand()
```

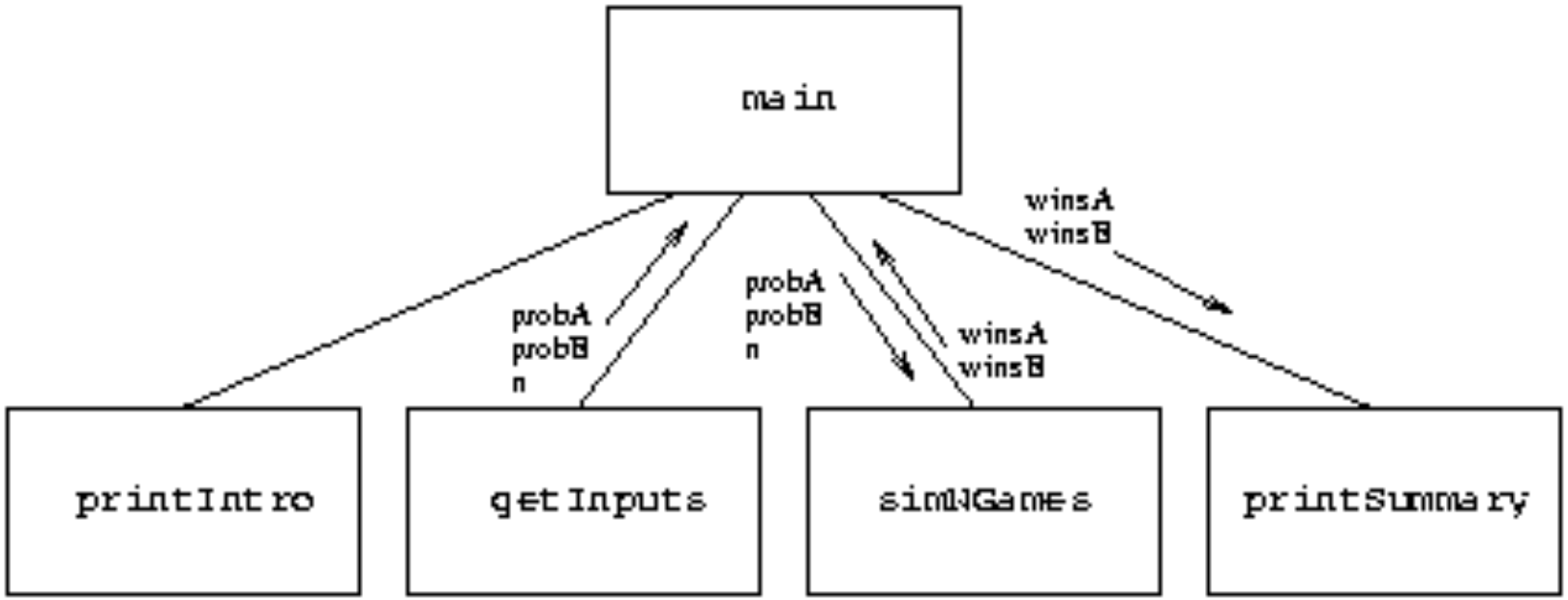
PseudoRandom Numbers

- Example: tennis_v0.cpp

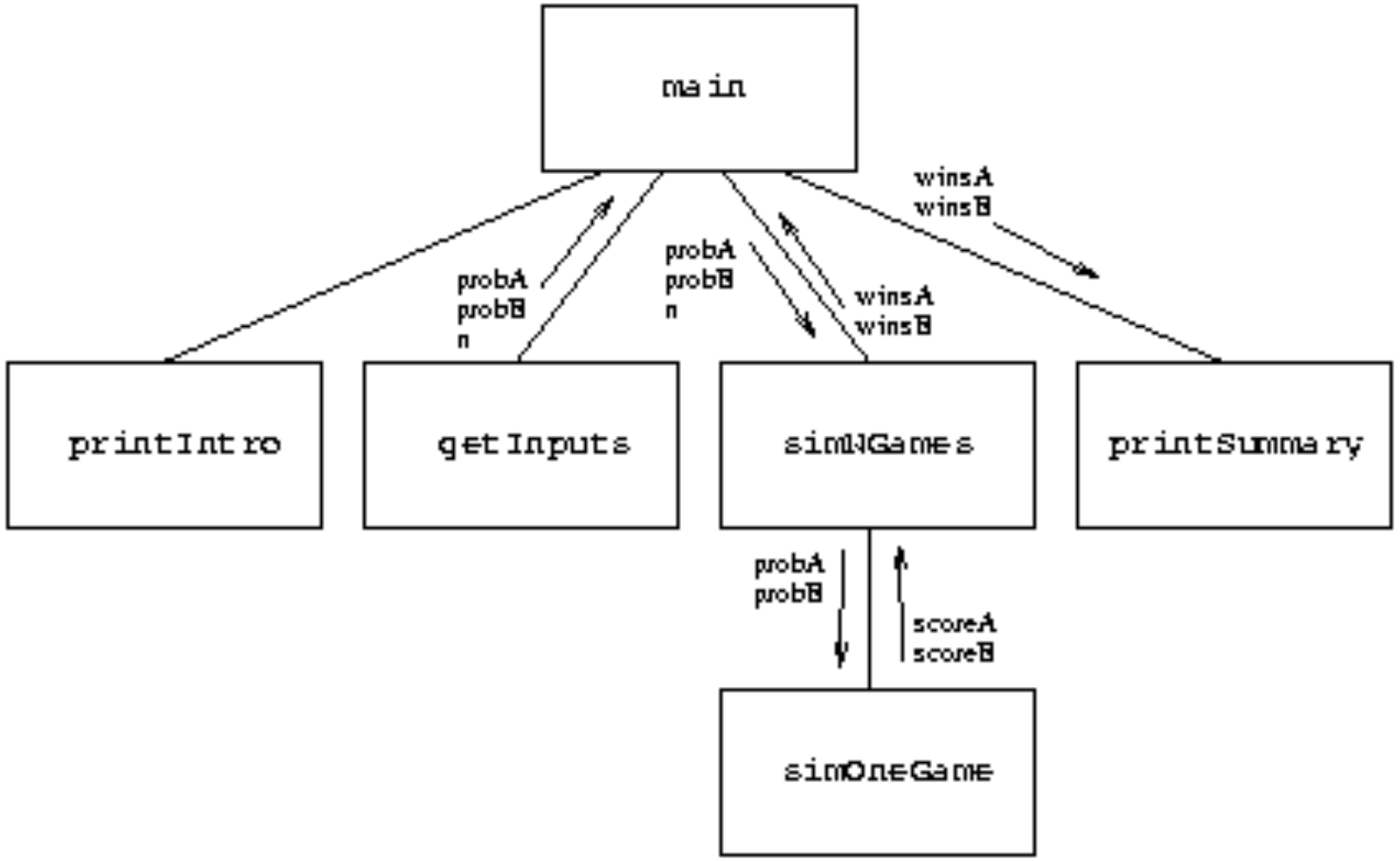
Top-Down Design

- In *top-down design*, a complex problem is expressed as a solution in terms of smaller, simpler problems.
- These smaller problems are then solved by expressing them in terms of smaller, simpler problems.
- This continues until the problems are trivial to solve. The little pieces are then put back together as a solution to the original problem!

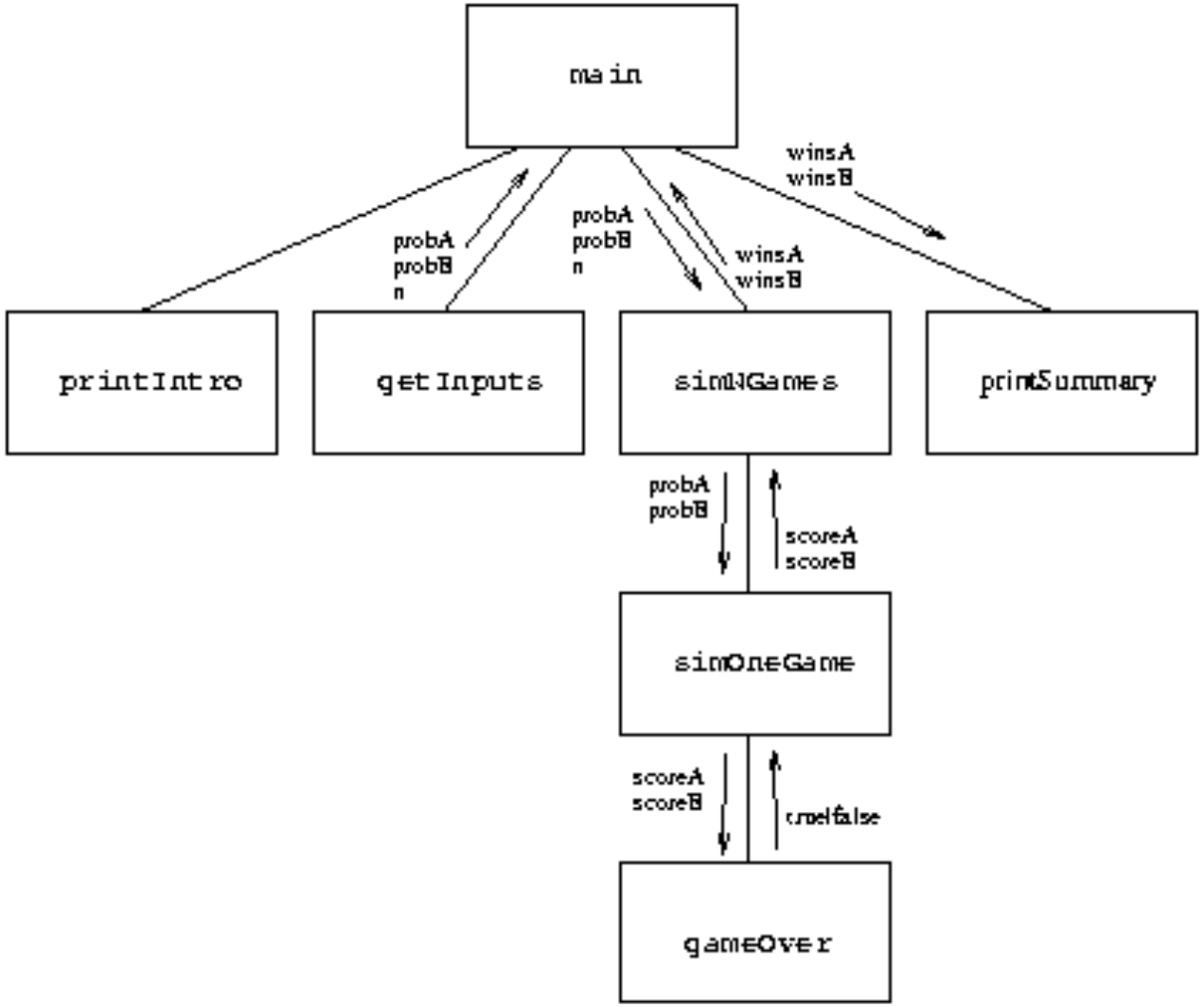
Separation of Concerns



Designing simNGames



Third-Level Design



Summary of the Design Process

- We start at the highest level of our structure chart and work our way down.
- At each level, we begin with a general algorithm and refine it into precise code.
- This process is sometimes referred to as *step-wise refinement*.

Summary of the Design Process

1. Express the algorithm as a series of smaller problems.
2. Develop an interface for each of the small problems.
3. Detail the algorithm by expressing it in terms of its interfaces with the smaller problems.
4. Repeat the process for each smaller problem.

Other Design Techniques

- Top-down design is not the only way to create a program!

Prototyping and Spiral Development

- Another approach to program development is to start with a simple version of a program, and then gradually add features until it meets the full specification.
- This initial stripped-down version is called a *prototype*.

Prototyping and Spiral Development

- Prototyping often leads to a *spiral* development process.
- Rather than taking the entire problem and proceeding through specification, design, implementation, and testing, we first design, implement, and test a prototype. We take many mini-cycles through the development process as the prototype is incrementally expanded into the final program.

Prototyping and Spiral Development

- How could the tennis simulation been done using spiral development?
  - Write a prototype where you assume there's a 50-50 chance of winning any given point, playing 30 serves.
  - Add on to the prototype in stages, including awarding of points, change of service, differing probabilities, etc.

Prototyping and Spiral Development

Examples:

- tennis_v1.cpp
- tennis_v2.cpp
- tennis_v3.cpp
- tennis_v4.cpp
- tennis_v5.cpp
- tennis_v6.cpp