AI INSTITUTE #AIISC
UNIVERSITY OF SOUTH CAROLINA

UNIVERSITY OF SOUTH CAROLINA

CSCE 240: Advanced Programming Techniques
Lecture 19: Advanced Pointers, Input/ Output

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

22ND MARCH 2022

Carolinian Creed: "I will practice personal and academic integrity."

Credits: Some material reused with permission of Dr. Jeremy Lewis. Others used as cited with thanks.

Organization of Lecture 19

- Introduction Section
  - Recap of Lecture 18
  - Class Pulse Survey
  - CASY 2
  - TA and SI Updates
- Main Section
  - Concept: Pointer arrays
  - Concept: Function Pointers
  - Concept: Buffering
  - Task: Project – PA #4 ongoing – check on issues
- Concluding Section
  - About next lecture – Lecture 20
  - Ask me anything

Introduction Section

Recap of Lecture 18

- We reviewed HW 5
- We looked at pointers
  - Pointers and references
  - Pointer arrays
  - Pointer based swapping of numbers and user-defined types
- Checked on PA 4, due on Thursday (March 24, 2022)

Updates from TA, SU

- TA update: Yuxiang Sun (Cherry)

- SI update: Blake Seekings

- Codeathon
  - 8 students got 100 bonus points with confirmation received from Blake/ ACM. See marks under Bonus in blackboard.

Course Mid-Point Pulse Survey

a) Do you like the pace of the course ? - Y/N

b) Do you like the content on which the course is focusing? - Y/N

c) Should the number of HWs be reduced? - Y/N

Yes	No
6	2
8	0
4	4

d) What more topic(s) will you like to be covered? - [Open ended]

Arrays, pointers, and vectors; review inheritance; external libraries; AI and ML

e) Any other feedback? - [Open ended]

Solutions of HW being posted, little time for HW, review before quizzes

Actions on Survey

- Solutions volunteered by students posted on github
  - 2 up, 3 others pending (make repo public, put HWs in a sub-dir called homeworks)
  - Students will be given bonus points
- Material changes
  - One lecture on AI/ML
  - One lecture reviewing material before Quiz 2

Update on CASY 2.2

- Chatbots – Event on March 18, 2022
  - Collaborative Assistants for Society (CASY) – in person and virtual event on campus
  - 9:30 am – 1:00 pm; talks and student use-cases

- Details and registration info: https://casy.aiisc.ai
- Summary:
  - https://www.linkedin.com/pulse/casy-22-building-momentum-collaborative-assistants-srivastava/
  - Thanks to all who attended
  - Students attending significant time will be given bonus points

Main Section

Concept: Pointers – Advanced (Contd.)

Function Pointers

- Functions can be treated as data
  - Passed using pointers
  - Selected dynamically and iterated
- Example
  - **int** (*f_ptr)(**int**, **int**); // declaring a function variable
  - f_ptr = &add; // assigning a value, i.e., function – add here - which matches the function signature
    // i.e., arguments and return type
  - f_ptr(a, b)   // invoking the function

Function Arrays

- Group of functions can be manipulated in an array

- Example
  - **int** (*f[3])(**int**, **int**); // Declaring variable

  - f[0] = &add; // Assigning
  - f[1] = &multiply; // Assigning
  - f[2] = &subtract; // Assigning

  - f[i](a, b) // Invoking

Review: Pointers and Examples

- int *a; // a is a pointer to int
- int **a; // a is a pointer to a pointer to a
- int *a[10]; // a is an array of size 10 of pointer to integers
- int (*a)[10]; // a is a pointer to an array of size 10 to integers
- char *(*fp)(int, float *); // fp is a pointer to a function, passing an integer and a pointer to a float,
  // returning a pointer to a char

Practical Advice: http://c-faq.com/decl/spiral.anderson.html

Further Exploration

- Tutorials
  - https://www.cplusplus.com/doc/tutorial/pointers/
  - https://www.cprogramming.com/tutorial/function-pointers.html

- Books
  - The Annotated C++ manual, https://www.stroustrup.com/arm.html
  - The C++ Programming Language (4th Edition), Addison-Wesley ISBN 978-0321563842. May 2013, https://www.stroustrup.com/C++.html
  - Fundamentals of C++ Programming , by Richard L. Halterman https://archive.org/details/2018FundamentalsOfCppProgramming/page/n333/mode/2up

Concept: Adv. I/O - Buffering

Why Buffer Input or Output

- Computer has access to both memory (temporary storage) and disk (permanent storage)
- Properties
  - Faster to write data to memory than to disk.
  - Faster to write one block of N bytes to disk in a single operation than it is to write N bytes of data one byte at a time using N operations

Credit: Fundamentals of Programming C++, Richard L. Halterman

Illustrating Regular and Buffered I/O

- Have to be aware of
  - buffer size // impacts I/O performance or memory usage
  - Initial and last values // In case last chunk is less than buffer size
  - Clearing off of the buffer // Affects what is read/ written at the end; flush the values

- Buffered reading/ writing supported in most languages

Code Examples

- Buffering in C style
- Buffering in C++, with streams

Discussion: Course Project

Course Project – Assembling of Prog. Assignments

- **Project**: Develop collaborative assistants (chatbots) that offer innovative and ethical solutions to real-world problems ! *(Based on competition - https://sites.google.com/view/casy-2-0-track1/contest)*
- Specifically, **the project will be building a chatbot that can answer questions about a South Carolina member of state legislature from**: https://www.scstatehouse.gov/member.php?chamber=H
  - Each student will choose a district (from 122 available).
  - Programming assignment programs will: (1) extract data from the district, (2) process it, (3) make content available in a command-line interface, (4) handle any user query and (5) report on interaction statistics.

Core Programs Needed for Project

- Prog 1: extract data from the district **[prog1-extractor]**
- Prog 2: process it (extracted data) based on questions **[prog2processor]**
- Prog 3: make content available in a command-line interface **[prog3-ui]**
- **Prog 4: handle any user query [prog4-userintent2querymapper]**
- Prog 5: report statistics on interaction of a session, across session

Objective in Programming Assignment # 4: *Remove Requirement on User to Know Supported Queries!*

- Until now, use needed to know what the program supports.
- **Can the system adapt rather than ask the user to adapt ?**

- **Approach Suggested**
  - Take user's utterance
  - Match to the closest supported query (six) and a confidence estimate
  - If confidence greater than a threshold
    - Run the query,
  - Otherwise
    - Ask user to re-phrase and ask again

- Program should do the following:
  - Run in an infinite loop until the user wants to quit
  - Handle any user response
    - **[#1]** User can quit by typing "Quit" or "quit" or just "q"
    - User can enter any other text and the program has to handle it. The program should write back what the user entered and say – "I do not know this information".
  - Handle known user query
    - **[#2]**"Tell me about the representative", "Tell me about the rep" => Personal Information (Type-I2)
    - **[#3]**"Where does the rep live" => Contact Information (Type-I1): Home Address
    - **[#4]**"How do I contact my rep " => Contact Information (Type-I1)
    - **[#5]**"What committees is my repo on" => Committee Assignments (Type-I3)
    - **[#6]**"Tell me everything" => *Give all information extracted*

Programming Assignment # 4

- Goal: **make an utterance to query [**Name: **prog4-userintent2querymapper**]
- Program may do the following:
  - Run in an infinite loop until the user wants to quit
  - Get a user utterance. We will call it u
  - See if u matches to supported queries in Q   // 6 until now
    - Split u into words
    - For each query q in Q
      - Split q into words
      - Check how many words of u and w match
      - Compute a percentage of match
    - q_i: let this be the query with the highest match percentage
    - If q_i > 0.7 (a parameter),
      - Consider it to be the query. Inform user and execute; give information (result)
    - Else
      - Tell user cannot understand u. Rephrase and try again.

Programming Assignment # 4

- Code organization
  - Create a folder in your GitHub called "**prog4-userintent2querymapper**"
  - Have sub-folders: src (or code), data, doc, test
  - Write a 1-page report in ./doc sub-folder
  - Put a log of system interacting in ./test
  - Send a confirmation that code is done by updating Google sheet; optionally, send email to instructor and TA

- Use concepts learned in class
  - Exceptions

Concluding Section

Lecture 19: Concluding Comments

- We looked at class survey results and made some changes
- We looked at function pointers and function arrays
- Re-looked at I/O and discussed buffering
- Checked on PA4, due on Thursday (March 24, 2022)

About Next Lecture – Lecture 20

Lecture 20: Advanced: Operator Overloading

- Adv I/O
  - Buffered writing
- Adv: operator overloading
- Prog 4 ends

17	Mar 15 (Tu)	Testing strategies	Prog 4 - start
18	Mar 17 (Th)	Advanced: Pointers	HW 5 due
19	Mar 22 (Tu)	Advanced: I/O	
20	Mar 24 (Th)	Advanced: Operator overloading	Prog 4 - end
21	Mar 29 (Tu)	Advanced: Memory Management	Prog 5 - start
22	Mar 31 (Th)	Advanced: Code efficiency	