CSCE 240: Advanced Programming Techniques
Lecture 17: Advanced Pointed, HW 5 (review)

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

17TH 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 18

- Introduction Section
  - Recap of Lecture 17
  - TA and SI Updates
- Main Section
  - Task: HW 5 – review
  - Review: Pointers and References
  - Concept: Pointer arrays
  - Concept: Function Pointers
  - Task: Project – PA #4 ongoing – check on issues
- Concluding Section
  - About next lecture – Lecture 19
  - Ask me anything

Introduction Section

Recap of Lecture 17

- We looked at common testing types
- Considered an example and different pitfalls
- Gave HW5, due today
- Gave PA 4, due on Thursday (March 24, 2022)

Updates from TA, SU

- TA update: Yuxiang Sun (Cherry)

- SI update: Blake Seekings

Main Section

Home Work 5 (Peer Review)

Due Thursday, March 7, 2022

Home Work (#5) – C++ - Background

- A *factorial* is a function that multiplies a number by every number below it. For a number N, it is denoted N!
  - Example: 4! = 4 x 3 x 2 x 1 = 24
- Factorial notation is used in many problems dealing with permutations and combinations
- Note:
  - 0! = 1
  - 1! = 1

- *Combination*: Number of ways **r** items can be selected from a set of size **n** where the order of picking does not matter
  - Example: Handshakes between 6 people = C^6_2
  - = (6!) / (2! * 4!) = (6 * 5 * 4!) / (2! * 4!) = 15
- Note:
  - r is smaller than n

$$_nC_r = \frac{n!}{r!(n-r)!}$$

Credit: https://en.wikipedia.org/wiki/Combination

Home Work (#5) – C++ - Requirement

- So, write a program named: ***FactorialFun***
- It will support inputs/ arguments in two formats:
  - N: number // to find factorial of N
  - N: number, r: number // to find C^N_r
- Output:
  - Value // computed value
  - Time taken // time for processing

Example invocation

```
>  FactorialFun 4
24
Time for processing: 0.023 seconds


>  FactorialFun 6 2
15
Time for processing: 0.0034 seconds
```

Home Work (#5) – C++ - Code Design

- Create test cases, i.e., input/ output pairs, to test for boundary conditions
- Use exception to handle likely errors – user may given any input

Peer Review: Homework Assignment #5

1. Go to spread sheet and on ”Homework Assignments - Peer Review” tab. Go for today’s date
2. Go to the row with your name
3. Peer review (10 mins)
   1. Enter serial number of person on your **LEFT** under “ID of code reviewer”
   2. Share code for the reviewer to see
   3. Reviewer: enter review (1-5)
   4. **Note**: negotiate – review code of neighbor or get own’s code reviewed
4. Peer test (10 mins)
   1. Enter serial number of person on your **RIGHT** under “ID of code tester”
   2. Share command line for the tester to see
   3. Tester: enter review (1-5)
   4. **Note**: negotiate – test code of neighbor or get own’s code tested

Peer Reviewing Guideline (10 mins)

- Look out for
  - Can you understand what the code is doing ?
  - Can you explain the code to someone else (non-coder) ?
  - Can you spot possible issues without running it?
    - Are the variables initialized ?
    - Are files closed?
    - Is their unnecessary code bloat ?
- What not to judge
  - Usage of language features, unless they are inappropriate

Assign rating

1: code not available
2: code with major issues
3: code with minor issues
4: -
5: no issues

Peer Testing Guideline (10 mins)

- Look out for
  - Does the program run as the coder wanted it to be (specification) ?
  - Does the program run as the instructor wanted it to be (requirement - customer) ?
  - Does the program terminate abruptly ?
  - Any special feature?
- What not to judge
  - Person writing the code

Assign rating

1: code not available
2: code runs with major issues (abnormal termination, incomplete features)
3: code runs with minor issues
4: -
5: No issues

Discussion on HW

- Peer Code Reviewing
- Peer Testing

Concept: Pointers - Advanced

Recap - Concept: Pointers

- Pointers refer to accessing and manipulating location of variables
  - a = 12 // variable is a, value is 12
  - b = &a // b has the address of a, i.e., 0 here. It is called a pointer
  - c = a // c has the value of a, i.e., 12
  - d = *b // will refer to a. That is, d will be equal to value pointed by b, i.e., 12

Variable	Location	Value
a	0	12
b	4	0
c	8	

Reference: https://www.cplusplus.com/doc/tutorial/pointers/

From 2nd Lecture

Pointer Management

Knowing what a pointer refers to at all times is critical for a (C++) program's stability

- Initialization
- Updates to values, due to
  - Operation
  - Memory allocation
  - Memory de-allocation

Pointers and References in Languages

- C++: fully supported
  - "A pointer is a variable that stores a memory address, for the purpose of acting as an alias to what is stored at that address."
  - Pointer arithmetic
  - Arguments of functions can be passed by value or references
  - Pointers are first class data types; they can also be passed by value and reference
- Java, Python: references
  - "A reference is a variable that refers to something else and can be used as an alias for that something else."
  - When a variables is initialized to another variable, references are passed.
  - No pointer arithmetic by programmer

Reference:
- https://nickmccullum.com/python-pointers/#why-dont-pointers-exist-in-python
- https://www.geeksforgeeks.org/is-there-any-concept-of-pointers-in-java/

Pointer v/s References

- One cannot have NULL reference. One must always be able to assume that a reference is connected to a legitimate piece of storage.
- Once a reference is initialized to an object, it cannot be changed to refer to another object. Pointers can be pointed to another object at any time.
- A reference must be initialized when it is created. Pointers can be initialized at any time.

Credit: https://www.tutorialspoint.com/cplusplus/cpp_references.htm

Usage of Pointers

- Can be used to implement passing values to a function by reference
  - In contrast to passing value by copy
- Doing explicit memory management
- Polymorphism

Swapping Values of a Built-in Type

Illustration for integer switching using references

Variable	Location	Value
a	0	10
b	4	20
pa	8	0
pb	12	4
ppa	16	8
ppb	20	12

temp

```
void swapNumbersReference(
     int &a, int &b)
{
   int temp = a;
   a = b;
   b = temp;
}
```

Credit: Fundamentals of Programming C++, Richard L. Halterman, Page 275

Swapping Values of a Built-in Type

Illustration for integer switching using pointers

Variable	Location	Value
a	0	10
b	4	20
pa	8	0 (4)
pb	12	4 (0)
ppa	16	8
ppb	20	12

temp

```
// Demonstrate swapping of numbers
void swapNumbers(int *a, int *b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}
```

Credit: Fundamentals of Programming C++, Richard L. Halterman, Page 275

Pointers and Arrays

- Aggregate memory allocations can be referred by pointers
- Example arrays
  - int anArray[10]; // an array of 10 ints
  - int *apointer; // a pointer to int

  - apointer = anArray;   // will give address of anArray to apointer
- Equivalent statements
  - anArray[5] = 0;   *// a [offset of 5] is assigned 0*
    *(apointer+5) = 0;   // a pointer + offset of 5 is assigned 0

Credits: https://www.cplusplus.com/doc/tutorial/pointers/

Swapping Values of a Struct

Using references

```
// Demonstrate user defined swap of values using references
void swapPeopleReference(PersonName &a, PersonName &b)
{
   PersonName temp = a;
    a = b;
    b = temp;
}
```

Variable	Location	Value
a	0	{John, First}
b	4	{Jane, Second}
pa	8	0
pb	12	4
ppa	16	8 (12)
ppb	20	12 (8)

Swapping Values of a Struct

Using pointers

Variable	Location	Value
a	0	{John, First}
b	4	{Jane, Second}
pa	8	0
pb	12	4
ppa	16	8 (12)
ppb	20	12 (8)

swapPeople()

temp

temp

```
// Demonstrate user defined swap of values using pointers
void swapPeople
   (PersonName *a, PersonName *b) {  // Passes pointer by value
           PersonName *temp = a;
           a = b;
           b = temp;}

void swapPeopleCorrect
   (PersonName **a, PersonName **b){
           PersonName *temp = *a;
           *a = *b;
           *b = temp; }
```

Function Pointers

- Functions can be treated as data
  - Passed using pointers
  - Selected dynamically and iterated
- Group of functions can be manipulated in an array

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

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

Announcements

- 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
- Looking for a panelist from class

Concluding Section

Lecture 18: Concluding Comments

- We looked pointers and references
- Pointers are useful for dynamic behavior - memory management, function invocation, …
- Reviewed HW5
- Checked on PA4, due on Thursday (March 24, 2022)

About Next Lecture – Lecture 19

Lecture 19: Advanced Input / Output

- Pointers (remaining topics)
- Adv I/O
  - Buffering
  - Seek/ going to specific data items

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	