

CSCE 240: Advanced Programming Techniques

Lecture 20: Advanced Input/ Output, Operators, HW5 Given

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

21ST MARCH 2024

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 20

- Introduction Section
 - Recap of Lecture 19
- Main Section
 - Concept: Buffering continued
 - Concept: Operator overloading
 - Task: Project – PA #4 due
 - HW5 given
- Concluding Section
 - About next lecture – Lecture 21
 - Ask me anything

Introduction Section

Recap of Lecture 19

- We looked at function pointers and function arrays
- We looked at testing strategies and considerations in a small problem - FN
- Re-looked at I/O and discussed buffering
- Reviewed PA4

Announcements

- Course in Fall 2024

CSCE 581 - Trusted Artificial Intelligence (3 Credits)

AI Trust – responsible/ethical technology, fairness/ lack of bias, explanations (XAI), machine learning, reasoning, software testing, data quality and provenance, tools and projects.

Prerequisites: C or better in [CSCE 240](#) and [CSCE 350](#)

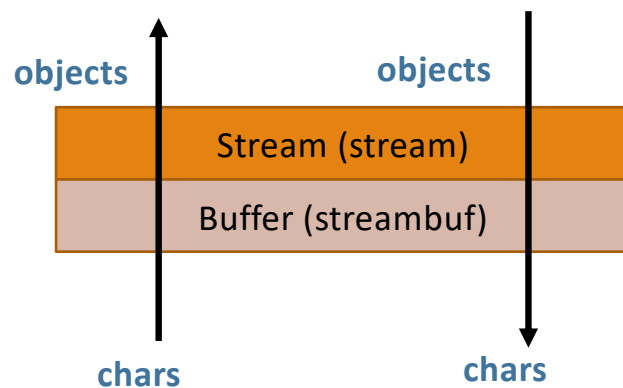
Prerequisite or Corequisite: D or better in [CSCE 330](#)

Main Section

Concept: Adv. I/O – Buffering (Continued)

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



- Developer has 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

Operations on Stream

- Position
 - **get**: position of the next character to be fetched into the sequence (extraction)
 - **put**: position of the next character to be deposited into the sequence (insertion)
- Operations
 - **seek**: move pointer with a given offset
 - **tell**: inform about the position of pointer

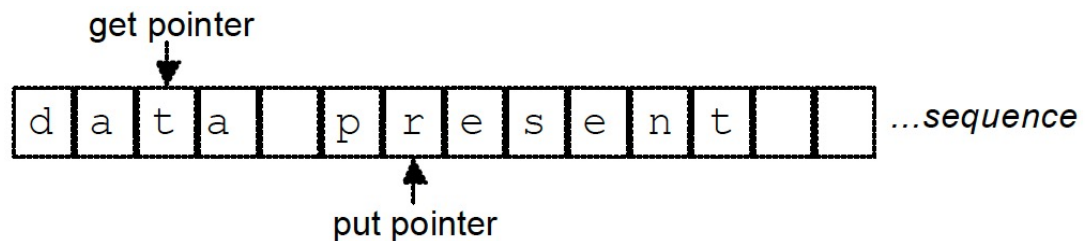


Image credit: C++ Essentials, Sharam Hekmat

Code Examples

- Steam write operations (option – 4)
- Reading and writing
 - with no buffering (option – 5)
 - with buffer size same as file length; extremely memory efficient (option – 6)

Code: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class19To22_AdvTopics/src/Class19To22_AdvTopics.cpp

Discussion on Streams and Buffers

- Streams give a very convenient interface over I/O
 - Hides details of the physical systems (disks, displays, printer, string, web-connected resource)
 - But performance can be a challenge
- Buffers give a way to manage performance
 - Relies on differential speeds of access of I/O devices
 - Design issues about size of buffers, practical issues of initialization of content, flushing content (write situation)

In-Class Programming

- Implement time and benchmark file reading for data from different companies (e.g., your two from project)
- What can you do to speed up reading?
- What about writing?

```
int main ()  
  
auto start = std::chrono::steady_clock::now(); // measures start time  
// ..... // core processing  
  
auto end = std::chrono::steady_clock::now(); // measures end time  
// prints result  
  
// prints time elapsed
```

Concept: Operator Overloading

Operator Overloading – What

- Overloading happens when we have multiple functions of the same name
 - Functions distinguished by signature, i.e., parameters and return types
 - Constructors are the common form of overloaded functions
- Operator overloading
 - When operators are overloaded
 - Examples: <<, >>, [], +, -, ...

Operator Overloading - Why

- Commonly used with user defined types / classes
- Provide convenience to user, improve usability
- Avoid meaningless / error-prone behavior, especially when operator behavior is inherited due to class hierarchy

Example 1 – Strings

- Suppose you are working with text. Can be in any language.
 - You want to refer to strings and their relationships to each other
 - **Example:** combining two strings
- String representation:
 - Array of characters
- Operation
 - +, -, ...

Example 2 – Point and Operations

- Suppose you are working in Geometry. Can be in any dimension.
 - You want to refer to points and their relationships with each other
 - **Example:** a point that is twice away from another point, with respect to a reference
- Point representation: 2-D: Cartesian Geometry
 - (x, y)
 - (angle, distance)
- Operation
 - +, -

Code: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class19To22_AdvTopics/src/Class19To22_AdvTopics.cpp

Argument: 7

Class Exercise – 10 Mins

- Implement operators
 - * with a Point argument: multiplies x and y of two points (self and argument) respectively, respectively
 - ^ with an int argument: raises x to-the-power of the passed point argument, i.e., y

Home Work 5

Due Tuesday, March 26, 2024

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: $4! = 4 \times 3 \times 2 \times 1 = 24$
- 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_2^6
 - $= (6!) / (2! * 4!) = (6 * 5 * 4!) / (2! * 4!) = 15$
- 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 three formats:
 - N: number // to find factorial of N
 - N: number, r: number // to find C_r^N
 - N: number, r: number, “compare” // to find $N!$ and C_r^N and tell which computation is faster.
- Output:
 - Value // computed value
 - Time taken // time for processingOR
 - Comparison report in the format on right
- **Variants**
 - Have numeric (int) arguments
 - Stretch
 - Have string arguments
 - Have a combination

Example invocation

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

> FactorialFun 6 2
15
Time for processing: 0.0034 seconds

> FactorialFun 6 2 compare
6! is 240, took : 0.0034 seconds
C^6_2 is 15, took : 0.0043 seconds
Time for processing C^6_2 is more.
```

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 give any input

Discussion: Course Project

Course Project – Knowing About Companies

- **Project:** Develop collaborative assistants (chatbots) that offer useful information about companies
- Specifically, use the EDGAR dataset on companies at:
<https://www.sec.gov/edgar/searchedgar/companysearch>.
 - For Apple, it is: <https://www.sec.gov/edgar/browse/?CIK=320193&owner=exclude>
- **Each student will choose two companies (from thousand available).**
- Programming assignment programs will: (1) extract data about two companies from 10-k, (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
 - Understand query and company of interest
 - Match to the closest supported query
 - **Intents**: [Parts and Items] + **3 more**
 - Also, add a confidence estimate
 - If confidence greater than a threshold and if the company is supported
 - 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
 - "Tell me about **IBM**" or "What are the risk factor for **IBM**?" => (Part 1), or (Part 1: Item 2), accordingly
 - "What markets does **IBM** operate in?", "Are there any disclosures from **IBM**?" => (Part 2)
 - "who are the directors? " => (Part 3: Item ..) // assume company, or tell of all companies, or ask ...
 - "Tell me about **IBM's** statements" => (Part 4)
 - ...
 - **"Tell me everything"** => **Give all information extracted**

Intents: [Parts and intents] +
tell everything, chitchat and quit

Content Reference: Queries for (Answers) Data We Have

- What does the (company) do? // Answers in Part 1
 - What is the (company's) business?
 - What are (company's) risk factors?
 - What does (company) own?
 - ...
- Where does (company) operate? // Answers in Part 2
 - What has (company) disclosed?
- How is (company) structured? // Answers in Part 3
 - Who is (company's) CEO?
 - How much does (person) earn?
 - ...
- What was in (company) statements? // Answers in Part 4
 - ...

Concepts: 10-K, Parts, Items

Parts

- Part 1: Business Background and Risks
 - Item 1: Business
 - Item 2: Risk factors
 - Item 3: Properties
 - Item 4: Legal Proceedings
- Part 2: Operations and Disclosures
 - .. Market
 - .. Disclosures
- Part 3: Company Structure
 - Directors
 - Compensation
- Part 4: Financial Statements
 - Statements

Hint: Programming Assignment # 4

- Goal: **make an utterance to intent query mapper** [Name: **prog4-userintent2querymapper**]
- Program **may** do the following – pseudo-code
 - 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 intents in Q **// 3 + financial doc info type**
 - Split u into words
 - For each information type – supported query q - in Q
 - Split q into words - w
 - Check how many words of u and w match **// one can also consider partial match**
 - Compute a percentage of match
 - q_i: let this be the query with the highest match percentage
 - If q_i > 0.7 **// 0.7: parameter**
 - Consider it to be the query. Inform user and execute; give information (result)
 - Else
 - Tell user cannot understand u. Example: 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
- Use concepts learned in class
 - Classes
 - Exceptions
 - UML Diagrams

Concluding Section

Lecture 20: Concluding Comments

- We looked at buffering for inputs and outputs
- We looked at operator overloading
- Both useful across OO programming languages
- PA4 due

About Next Lecture – Lecture 21

Lecture 21: Advanced: Memory Mgmt

- Fixed memory
 - Vectors
 - Arrays
- Dynamic memory
 - List
 - User defined types
- Freeing memory
- PA 5 starts

19	Mar 19 (Tu)	Advanced: Pointers, I/O	
20	Mar 21 (Th)	Advanced: Operator overloading	Prog 4 – end
21	Mar 26 (Tu)	Advanced: Memory Management	Prog 5 – start HW 5 due
22	Mar 28 (Th)	Advanced: Code efficiency	
23	Apr 2 (Tu)	Advanced: Templates	
24	Apr 4 (Th)	AI / ML and Programming	Prog 5 – end
25	Apr 9 (Tu)	Project code summary – student presentation for reuse Review material for Quiz 2	HW 6 due Prog 6 – assembling start
26	Apr 11 (Th)	In class test	Quiz 2 – In class
27	Apr 16 (Tu)	Project presentation	Prog 6 - due
28	Apr 18 (Th)	Project presentation	Last day of class (April 22 per bulletin)
	Apr 23 (Tu)		Reading Day
29	Apr 25 (Tu)	9am – Exam or Final Overview	Examination