

CSCE 240: Advanced Programming Techniques

Lecture 18: Advanced Pointers, HW 4 (review), Advanced Testing

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

14TH 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 18

- Introduction Section
 - Recap of Lecture 17
- Main Section
 - Task: HW 5 – review
 - Review: Pointers and References
 - Concept: Pointer arrays
 - Concept: Function Pointers
 - Concept: Advanced Testing
 - 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 the c++ standard library
 - Many types of functionality
 - String, I/O, Mathematical libraries most commonly used
- Remember that many implementations of C++ standard library, usually based on different OS or hardware
 - Implements changing specs
- Be ready to implement one's own (rather than reuse), if necessary, for performance
- Testing Background
- Gave HW4, due today
- Gave PA 4, due on Thursday (March 21, 2024)

Course Mid-Point Pulse Survey – Sample Qs

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

Main Section

Home Work 4 (Peer Review)

Due Thursday, March 14, 2024

Home Work (#4) – C++ - Background

- Email programs parse Email headers and show content. The headers have **parts** (e.g., CC, To, From) that are part of a standard and also proprietary extensions.
- Defined with IETF RFC - <https://datatracker.ietf.org/doc/html/rfc5322>
 - description <https://www.tutorialspoint.com/rfc-5322-internet-message-format>
 - Examples for Microsoft Outlook and Gmail are shown.
- Let us assume that parts which are common to both are the standard and those unique are proprietary. So, “CC” is common and “X-MS-Has-Attach” is unique.
- **Write a program, *EmailInformationExtractor***, which, when given a message header from either of the two programs, and a part name, will read the value of the message part.

Microsoft Outlook Header

- Received: from DS7PR19MB5853.namprd19.prod.outlook.com ...
- Authentication-Results: dkim=none (message not signed)
- Received: from ...
- Content-Type: application/ms-tnef; name="winmail.dat"
- Content-Transfer-Encoding: binary
- From: "Sri Naga Sushmitha, Satti" <SATTI@cse.sc.edu>
- To: "Srivastava, Biplav" <BIPLAV.S@sc.edu>
- CC: "Baldwin, Randi" <baldwin@cse.sc.edu>
- Subject: Re: Possible need for ... 240
- Thread-Topic: Possible need for printout for .. 240
- Thread-Index: ... +AAAIRpoAAAp/ggAAAJH0=
- Date: Tue, 15 Feb 2022 13:52:33 +0000
- Message-ID: <...>
- References: ...
- In-Reply-To: <...>
- Accept-Language: en-US
- Content-Language: en-US
- X-MS-Has-Attach:
- X-MS-Exchange-Organization-SCL: -1

Home Work (#4) – C++ - Requirement

- So, program name:
EmailInformationExtractor
- Inputs:
 - message header
 - Part name
- Output:
 - Value
- Hint
 - Use regex
 - Use standard libraries

Gmail Header

- Delivered-To: biplav.srivastava@gmail.com
- Received: by 2002:a05:7000:1f97:0:0:0:0 with SMTP ...
- X-Google-Smtp-Source: ABdhPJz/...
- Received: from m08b.cvent-planner.com ...
- From: Reply-To:To:Message-ID:Subject:MIME-Version:
- Content-Type: List-Unsubscribe; /Tvkd8/15SWIBA=; ...
- Date: Thu, 17 Feb 2022 23:56:12 +0000
- From: AAAI Staff <aaai22@aaai.org>
- Reply-To: <aaai22@aaai.org>
- To: Biplav Srivastava <biplav.srivastava@gmail.com>
- Message-ID: <..>
- Subject: AAAI-22 General Information
- MIME-Version: 1.0
- Content-Type: multipart/alternative; ..
- Content-Type: text/plain; charset=UTF-8
- Content-Transfer-Encoding: quoted-printable

Home Work (#4) – C++ - Code Design

- Create 3 classes:
 - Base class with common parts: BaseEmailHeaderType
 - Children classes with custom parts: GmailHeaderType, OutlookHeaderType
- Use exception to handle likely errors

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
 - Memory efficient
- Doing explicit memory management
- Polymorphism

Swapping Values of a Built-in Type

Illustration for integer switching
using references

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

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

temp

Code: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class17and18_TestingAdvPointers/src/Class17and18_TestingAdvPointers.cpp

Argument: 1

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

Swapping Values of a Built-in Type

Illustration for integer switching
using pointers

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

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

temp

Code: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class17and18_TestingAdvPointers/src/Class17and18_TestingAdvPointers.cpp

Argument: 2

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;
}
```

Code: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class17and18_TestingAdvPointers/src/Class17and18_TestingAdvPointers.cpp

Argument: 3

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

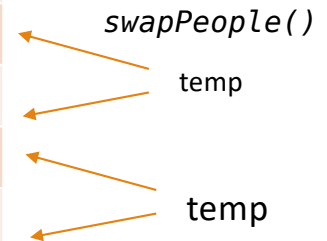
```
// Demonstrate user defined swap of values using pointers
void swapPeople
(PersonName *a, PersonName *b) { // Passes pointer by value
    PersonName *temp = a;
    a = b;
    b = temp;}

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

Code: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class17and18_TestingAdvPointers/src/Class17and18_TestingAdvPointers.cpp

Argument: 4

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)



Function Pointers

- Functions can be treated as data
 - Passed using pointers
 - Selected dynamically and iterated

```
int (*f_ptr)(int, int);  
f_ptr = &add;
```

- Group of functions can be manipulated in an array

```
int (*f[3])(int, int);  
f[0] = &add;  
f[1] = &multiply;  
f[2] = &subtract;
```

Code: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class17and18_TestingAdvPointers/src/Class17and18_TestingAdvPointers.cpp

Argument: 5

Code: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class17and18_TestingAdvPointers/src/Class17and18_TestingAdvPointers.cpp

Argument: 6

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`

Code: https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class17and18_TestingAdvPointers/src/Class17and18_TestingAdvPointers.cpp

Arguments: 1 through 6

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

Tip for Deciphering Pointer Statements

There are three simple steps to follow:

Starting with the unknown element, move in a spiral/clockwise direction; when encountering the following elements replace them with the corresponding english statements:

1. [X] or [] => Array X size of... or Array undefined size of... (type1, type2) => function passing type1 and type2 returning... * => pointer(s) to...
2. Keep doing this in a spiral/clockwise direction until all tokens have been covered.
3. Always resolve anything in parenthesis first!

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

Example #1: Simple declaration

```
char *str[10];
```

"str is an array 10 of pointers to char"

Example #2: Pointer to Function declaration

```
char *(*fp)( int, float *);
```

"fp is a pointer to a function passing an int and a pointer to float returning a pointer to a char"

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 Testing Strategies

Important Types of Testing

- Unit testing
 - Purpose: Check a basic functionality is working. Example, a function or programming assignment in course project
 - Developer does on their own
- Integration testing
 - Purpose: Ensure different components of project work together. Example, complete course project
 - Developer or dedicated tester performs
- Functional testing
 - Purpose: business requirement is met. Checks output, not intermediate results
 - Tester performs
- Acceptance testing
 - Purpose: business requirement is met both functionally and non-functionally like performance, throughput. Checks output, not intermediate results
 - Tester performs; customer performs
- Regression testing
 - Purpose: ensure existing functionality is preserved; especially after a code change
 - Tester performs

We are mostly doing unit and integration testing in the course

Example – Calculating Fibonacci Number

- Concept in mathematics:
 - Fibonacci number of a number is the sum of F numbers of its two predecessors
 - Credit: https://en.wikipedia.org/wiki/Fibonacci_number
- Popularized by Fibonacci around 1200 AD, known before in India as early as 450 BC

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181	6765

Implementing and Testing in C++ (V1)

```
int fibonacci(int n)
{
    return fibonacci(n-1) + fibonacci(n-2);
}
```

What can be wrong ?

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181	6765

Implementing and Testing in C++ (V2)

```
long fibonacci(unsigned int n)
{
    if (n < 2) return n;
    return fibonacci(n-1) + fibonacci(n-2);
}
```

Fixed for handling

- Negative numbers
- Larger return type

But may take too long

Implementing and Testing in C++ (V3) With Measuring Time

```
long fibonacci(unsigned int n)
{
    if (n < 2) return n;
    return fibonacci(n-1) + fibonacci(n-2);
}

int main ()
{
    auto start = std::chrono::steady_clock::now(); // measures start time
    long result = fibonacci(n); // calls function
    cout << "f(" << n << ") = " << result << '\n'; // prints result
    auto end = std::chrono::steady_clock::now(); // measures end time

    // prints time elapsed
}
```

Fixed for handling

- Negative numbers
- Larger return type

Reports time

* But time includes printing time

Code sample:

https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class17and18_TestingAdvPointers/src/Class17and18_TestingAdvPointers.cpp

Implementing and Testing in C++ (V4) With Measuring Time

```
long fibonacci(unsigned int n)
{
    if (n < 2) return n;
    return fibonacci(n-1) + fibonacci(n-2);
}

int main ()
{
    auto start = std::chrono::steady_clock::now(); // measures start time
    long result = fibonacci(n); // calls function
    auto end = std::chrono::steady_clock::now(); // measures end time

    cout << "f(" << n << ") = " << result << '\n'; // prints result

    // prints time elapsed
}
```

Fixed for handling

- Negative numbers
- Larger return type

Reports time

Testing Frameworks

- Java
 - JUnit
 - Code sample: https://github.com/biplav-s/course-adv-proglang-s23/tree/main/sample-code/Java/L17_Testing/src/autotest
- C++
 - Boost.Test
 - Google Test

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

Review of Assignments PA1,PA2, PA3 - Feedback

- Do not put *.class, a.out or .exe in git; it is a binary
- Put a Readme.md or Readme.txt in your assignment's main directory so that the reviewer knows what is the main file, where is the data, how is your program invoked, etc
- Avoid hardcoding in code
 - Paths an absolute no-no
 - Data based string extraction
 - [Students have hardcoded line number, character offset, or simply written values in code (manual extraction). Regex hardcoding most common.]
 - Will make code hard to generalize; no one else will be able to reuse
 - Regex makes extraction easy to understand and simpler
 - Loading extraction logic (regex, string indexes) from a config file makes code easy to generalize

Suggestion: Externalizing Extraction Logic From Code

Loading extraction logic (regex, string indexes) from a config file makes code easy to generalize

Configuration file (Data)

Format: entity name, regex pattern

Format: entity name, line, start index, end index
IBM, (l|i)bm

Code

1. Read configuration file
2. Read data stream
3. For each pattern
 - extract entity value from data stream
4. Close files
5. *# Do rest of the processing*

Now, to extract a new pattern or change extraction rule, we just have to modify the configuration file!

Concluding Section

Lecture 18: Concluding Comments

- We looked pointers and references
- Pointers are useful for dynamic behavior - memory management, function invocation, ...
- Reviewed HW4
- Checked on PA4, due on Thursday (March 21, 2024)

About Next Lecture – Lecture 19

Lecture 19: Advanced Input / Output

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

12	Feb 15 (Th)	OO – Constructor, Destructor	Prog 2 – end
13	Feb 20 (Tu)	Review: inheritance, Polymorphism	Prog 3 - start
14	Feb 22 (Th)	In class test	Quiz 1 – In class
15	Feb 27 (Tu)	In class Project Review: PA1 and PA2	
16	Feb 29 (Th)	OO – operators, access control	Prog 3 - end Semester - Midpoint
	Mar 5 (Tu)		Spring break – No class
	Mar 7 (Th)		Spring break – No class
17	Mar 12 (Tu)	C++ standard library, Testing strategies	Prog 4 - start
18	Mar 14 (Th)	Advanced: Pointers	HW 4 due
19	Mar 19 (Tu)	Advanced: Pointers, I/O	
20	Mar 21 (Th)	Advanced: Operator overloading	Prog 4 – end (March 26, 2023)