

# CSCE 240: Advanced Programming Techniques

## Lecture 14: Constructors and Destructors

---

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

23<sup>RD</sup> FEBRUARY 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 14

---

- Introduction Section
  - Recap of Lecture 13
- Main Section
  - Concept: Constructors
  - Concept: Destructors
  - Home work #4
  - Discussion: Project, Programming Assignment #3
- Concluding Section
  - About next lecture – Lecture 15
  - Ask me anything

# Introduction Section

---

# Recap of Lecture 13

---

- Looked at Errors
- Looked at Exception Handling
- Examples of Exceptions
  - In C++, Java, Python
  - Creating new exception handlers in C++

# Main Section

---

# Concept: Constructors

---

# Constructor - What is It?

---

- Special function in every class
  - Always has the same name as the class itself
  - Does not have an explicit return type
  - Multiple constructors possible per class
- **Purpose:** Used to initialize objects of that class

## Specification

```
class PersonName {  
    string firstName;  
    string lastName;  
  
public:  
    PersonName();  
    PersonName(string);  
    PersonName(string, string);  
  
    ...  
}
```

<https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class7and8%20C%2B%2B%20OO/src/headers/PersonName.h>

# Observation: Constructor

---

- Declaration is usually public
- What happens if a constructor is **private** ?
  - Could declare, but no object can be declared
- A program cannot explicitly call constructors like other member functions
  - Implicitly called by instantiating a class

```
PersonName p1;
```



# Constructor - What is It?

- Special function in every class
  - Always has the same name as the class itself
  - Does not have an explicit return typeMultiple constructors possible per class
- **Purpose:** Used to initialize objects of that class

## Usage

```
PersonName p1;  
PersonName p2("Joginder");  
PersonName p3("Joginder", "Singh")
```

## Implementation

```
PersonName::PersonName() {  
    firstName = "default-Maria";  
    lastName = "default-Wang";  
}  
  
PersonName::PersonName(string first) {  
    firstName = first;  
    lastName = "default-Wang";  
}  
  
PersonName::PersonName(string first,  
                        string last) {  
    firstName = first;  
    lastName = last;  
}
```

[https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class7and8\\_C%2B%2B\\_OO/src/implement/PersonName.cpp](https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class7and8_C%2B%2B_OO/src/implement/PersonName.cpp)

# Order of Calling Constructors in Hierarchy

---

- Parent then Child or Child before Parent ?

- **Parent first**

```
*** DEMO of Grand Child Class ***
```

```
Testing: data member -
```

```
    DEMO of Constructor - Parent Class ***
```

```
        DEMO of Constructor - Another Child Class ***
```

```
            DEMO of Constructor - GrandChild Class ***
```

# Discussion: Using Constructor in Project

---

- Disease class
  - Initialization / customization of disease object
    - Websites url for CDC content
    - Website url WebMD content
    - Initializing parsing rules
  - Allocating memory
  - Customizing content response
  - Reusing common services – logging, error handling
    - Initializing log file
    - Customizing error messages

# Concept: Destructors

---

# Destructors - What is It?

---

- Special function in every class
  - Always has the same name as the class itself but prefixed with ~
  - Does not have an explicit return type
  - **Does not take an argument**
  - **Maximum one destructor per class**
- **Purpose:** Used to cleanup before removing objects of that class
  - Common usage: freeing memory allocated by the object's data members before the object is destroyed
  - Common usage: Close files, streams

## Implementation

```
PersonName::~~PersonName() {  
  
}
```

<https://github.com/biplav-s/course-adv-proglang/blob/main/sample-code/CandC%2B%2B/Class7and8 C%2B%2B OO/src/implement/PersonName.cpp>

# Order of Calling Destructors in Hierarchy

---

- Parent then Child or Child before Parent ?
- **Child first !**

```
*** DEMO of Grand Child Class ***  
  
...  
The grandchild's location is: AnotherChild:default-location  
    Demo of Destructor – GrandChild Class ***  
    DEMO of Destructor – Another Child Class ***  
    DEMO of Destructor – GrandChild Class ***
```

# Full Example

---

**Program:** Class9and10\_C++\_OOAdv.cpp    **Argument:** 3

```
*** DEMO of Grand Child Class ***  
  
Testing: data member -  
  
    DEMO of Constructor - Parent Class ***  
  
        DEMO of Constructor - Another Child Class ***  
  
            DEMO of Constructor - GrandChild Class ***  
  
The grandchild's name is: Parent:default-name  
The grandchild's location is: AnotherChild:default-location  
  
            Demo of Destructor - GrandChild Class ***  
  
        DEMO of Destructor - Another Child Class ***  
  
    DEMO of Destructor - GrandChild Class ***
```

# Discussion: Using Constructors / Destructors Effectively

---

- Remember: Create automatically if none provided by developer
- Constructor: initialization of data members
- Destructor: clean-up
- Remember the order, use it productively but do not overly depend on it.



# Home Work 4

---

Due Tuesday, March 2, 2022

# 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.
- 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

## 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

# 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.

# Course Project – Building and Assembling of Prog. Assignments in Health

---

- **Project:** Develop collaborative assistants (chatbots) that offer useful information about diseases
- Specifically, use the CDC dataset on diseases at: <https://wwwnc.cdc.gov/travel/diseases>
  - For polio, it is: <https://wwwnc.cdc.gov/travel/diseases/poliomyelitis>
  - Each student will choose two diseases (from 47 available).
  - Each student will also use data about the disease from WebMD. Example for polio - <https://www.webmd.com/children/what-is-polio>
- Programming assignment programs will: (1) extract data about a disease from two sites, (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 and
- Prog 5: report statistics on interaction of a session, across session



# Programming Assignment # 2

- Goal: **process extracted text based on questions**
  - Language of choice: Any from the three (C++, Java, Python)
- Program should do the following:
  - Take input from a local file with whose content is obtained from Prog#1 (when **disease** name given as input)
  - Given an information type as input, the program will return its content
    - Examples: what is disease (I1), who is at risk (I2), disease vaccine (I12)
    - Input type can be given as command line argument.  
Examples:
      - prog2processor -t "what is **malaria**?" // Tell about disease
      - prog2processor -t "more information" // Get more info
  - For demonstrating that your program works, have a file called "test\_output.txt" showing the set of supported commandline options and output in the doc folder.
- Code organization
  - Create a folder in your GitHub called "prog2-processor"
  - Have sub-folders: src (or code), data, doc, test
  - Write a 1-page report in ./doc sub-folder
  - Send a confirmation that code is done to instructor and TA, and update Google sheet

S1: <https://www.cdc.gov/travel/diseases/malaria>

- What is malaria? [I1]
- Who is at risk? [I2]
- **What can travelers do to prevent malaria?** [I3]
- After Travel [I4]
- More Information [I5]

S2: <https://www.webmd.com/a-to-z-guides/malaria-symptoms>

- **What Is Malaria?** [I1]
- [Malaria Causes and Risk Factors](#) [I2]
- [Types of Malaria](#) [I6]
- [Symptoms](#) [I7]
- [When to Call a Doctor About Malaria](#) [I8]
- [Malaria Diagnosis](#) [I9]
- [Malaria Treatment](#) [I10]
- [Malaria Complications](#) [I11]
- **Malaria Vaccine** [I12]

# Programming Assignment # 3

- **Goal:** make content available in a command-line interface  
[Name: prog3-ui]

- Program should do the following:
  - Run in an infinite loop until the user wants to quit
  - Handle any user response
    - 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 types
    - “Tell me about the disease”, “What is *malaria*?” => (Type-I1)
    - “What can I do after travel?” => (Type-I4)
    - “what is the treatment? ” => (Type-I10)
    - “Tell me about *malaria* vaccine” => (Type-I2)
    - ...
    - “**Tell me everything**” => *Give all information extracted (I1-I12)*

S1: <https://www.cdc.gov/travel/diseases/malaria>

- What is malaria? [I1]
- Who is at risk? [I2]
- What can travelers do to prevent malaria? [I3]
- After Travel [I4]
- More Information [I5]

S2: <https://www.webmd.com/a-to-z-guides/malaria-symptoms>

- [What Is Malaria?](#) [I1]
- [Malaria Causes and Risk Factors](#) [I2]
- [Types of Malaria](#) [I6]
- [Symptoms](#) [I7]
- [When to Call a Doctor About Malaria](#) [I8]
- [Malaria Diagnosis](#) [I9]
- [Malaria Treatment](#) [I10]
- [Malaria Complications](#) [I11]
- [Malaria Vaccine](#) [I12]

# Notes on PA#3

---

- Handle all 12 information types
  - Multiple ways to ask for same information type
  - Variant assumes disease name from context or is specified
- Handle special query: *Tell me everything*
- Handle others
  - Chit-chat
  - Give controlled response under all condition

# Programming Assignment # 3

---

- Code organization
  - Create a folder in your GitHub called “prog3-ui”
  - Have sub-folders: src (or code), data, doc, test
  - Write a 1-page report in ./doc sub-folder
  - Send a confirmation that code is done by updating Google sheet; optionally, send email to instructor and TA
- Use concepts learned in class
  - Classes
  - Exceptions
  - UML Diagrams

# Concluding Section

---

# Lecture 14: Concluding Comments

---

- We looked at the concept constructor
- We looked at the concept of destructor
- Home Work #4 – due Tuesday, Feb 28, 2023
- Programming Assignment #3 starts, due Thursday, March 2, 2023

# About Next Lecture – Lecture 15

---

# Lecture 15: Operators and Overloading

---

- C++ operators
- Overloading operators

13	Feb 21 (Tu)	Exceptions	Prog 2 - end
14	Feb 23 (Th)	OO – Constructor, Destructor	Prog 3 - start
15	Feb 28 (Tu)	OO – operators, access control	HW 4 due
16	Mar 2 (Th)	C++ standard library	Prog 3 - end Semester - Midpoint
	Mar 7 (Tu)		Spring break – No class