OntarioTech Science

Kourosh Davoudi
kourosh@uoit.ca

Week 1: Introduction and C++ Basics

CSCI 1061: Programming Workshop II

Welcome to 1061U !

In this week:

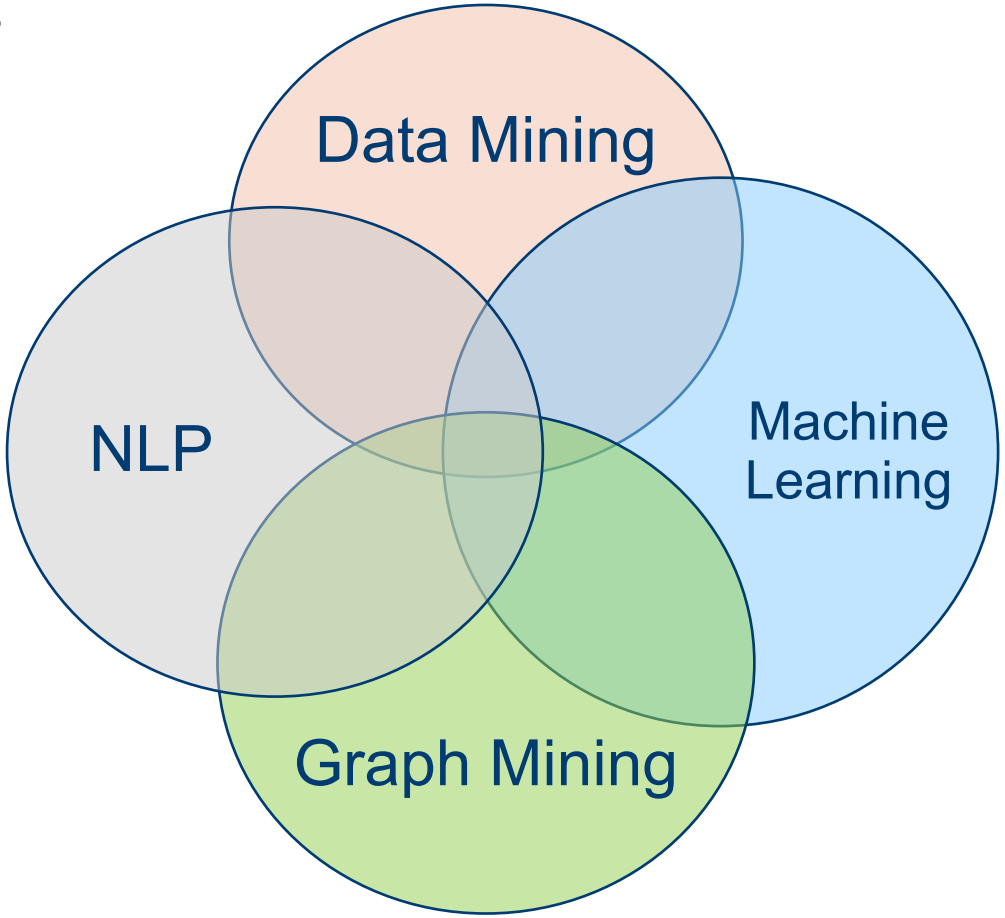
- We get to know each other
- We learn about:
  - Course Objectives
  - Course Structure
  - Learning Outcomes
- C++ Basics and overview

Kourosh Davoudi

- PhD: Computer Science (York University)
- Postdoctoral: Computer and Management Science (University of Waterloo)
- Private partners:

Kourosh Davoudi

- Areas of interest:



How about your?

- How do you like your major?
- What is your favorite course?
- Which jobs in computer science are you interested in?
- What do you expect from this course?
- Which programming languages have you work with?
- …

Course Content

- What is this course about?
  - It is about understanding the fundamental and modern programming skills through study of C++ language

- What is not this course about?
  - Designing advance algorithms and data structure

Labs

- Our TAs will run the lab sessions (6 sections)

Check MyCampus to see
which lab you should attend !

- It is highly recommended to complete lab assignments by the time of each lab session.

Labs

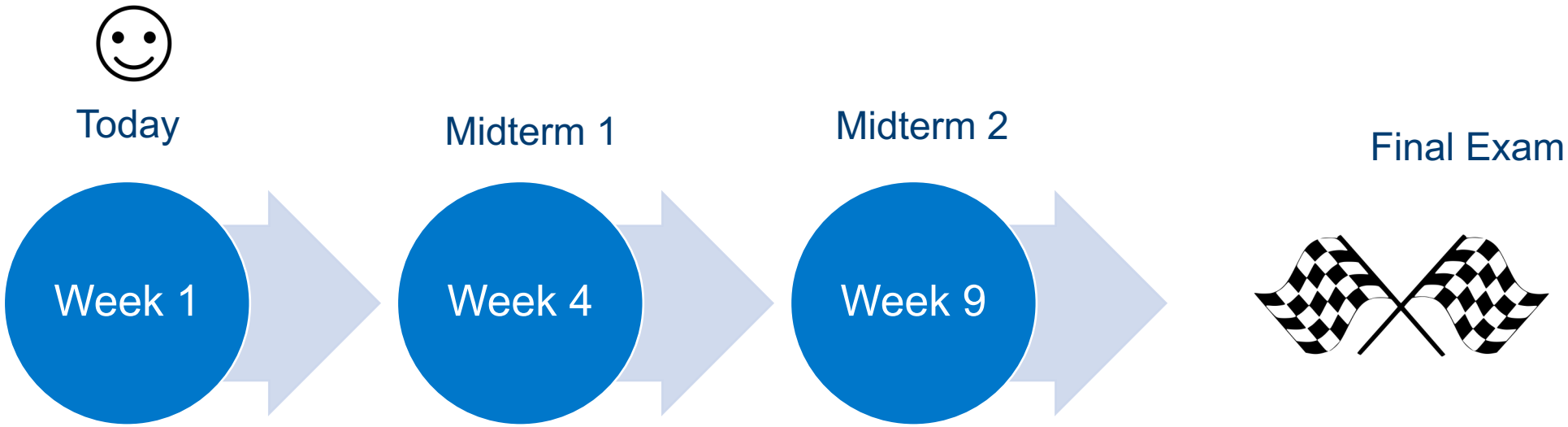
- Our TAs will run the lab sessions

Teaching Assistant Name	Email
Vincent Galloro	vincent.galloro@uoit.net
Huynh Minh	huynh.minh@ontariotechu.net
John Nemec	john.nemec@ontariotechu.net
Jeremy Friesen	Jeremy.friesen@ontariotechu.net
Nandor Gallo	nandor.gallo@ontariotechu.net
Luke Tran	luke.tran@ontariotechu.net

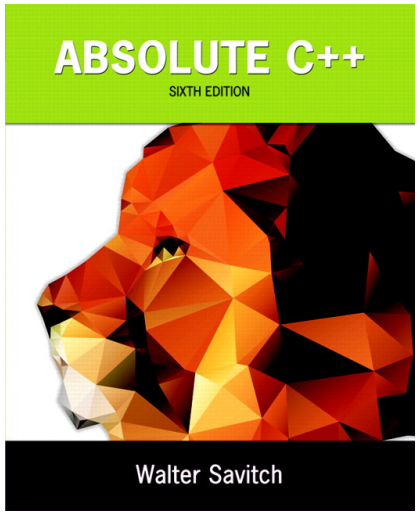
Evaluation

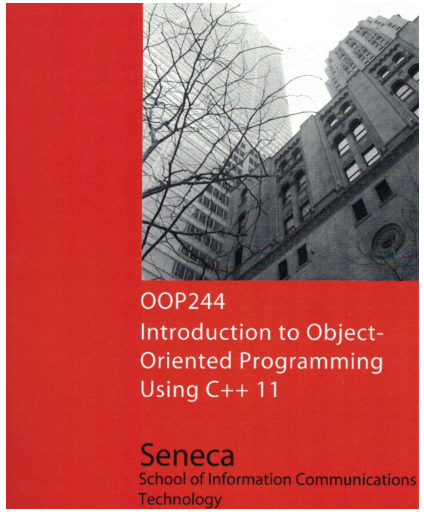
Component	Date	Weight
Midterm 1	January 28, 2020 (in-class)	15 %
Midterm 2	March 10, 2020 (in-class)	15 %
Final	TBA	30 %
Lab + Participation	Due next Friday 11:59 PM[1]	40 %

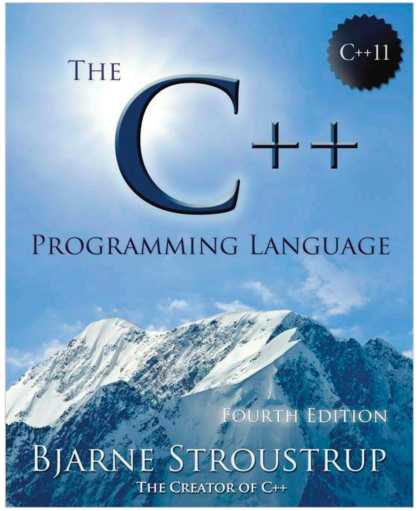
[1]unless otherwise mentioned



References





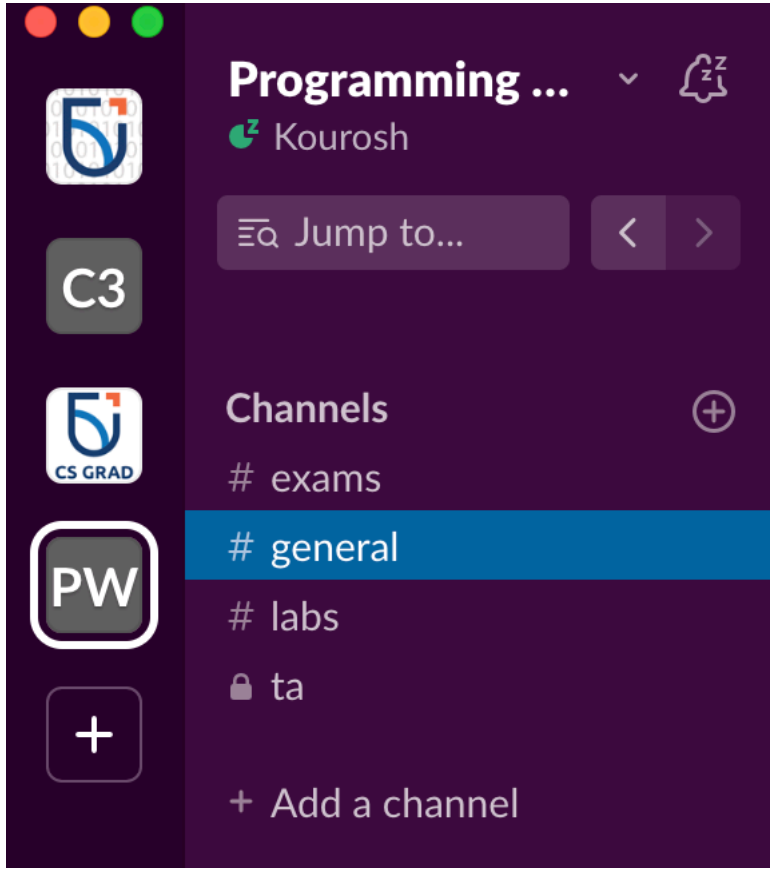


- https://www.tutorialspoint.com/cplusplus/
- http://www.cplusplus.com

Communication

Slack

- Please note that questions about lectures/labs/exams should be posted to the appropriate slack channels.



csci1061u.slack.com

Office Hours and Contacts

- Course Instructor:
- Dr. Kourosh Davoudi
  - **Email:** kourosh@uoit.ca (For official matters. Email subject should be: 1061U)
  - **Office Location:** UA 2015
  - **Office Hours:**

    Tuesday, 1:30 PM – 2:30 PM, UA 2015

    Friday, 10:00 AM – 11:00 AM, UA 2015

    OR by appointment
  - **Phone:** (905) 721-8668 x 2779
  - **Webpage:** http://dmalab.science.uoit.ca/hdavoudi/

Suggestions !

- Attend every single session
- Take note
- Participate in in-class activities
- Attend all labs and complete assignments
- User office hours to discuss
- Give me feedback
- Practice, practice, practice !

Laptops in Class !

- Appropriate uses:
- Note taking
- Researching course topics
- Participation in class activities

Inappropriate uses:

Games

Social networking

Listening to music

Watching YouTube

In-class gamming = Immediate ask to leave !

Learning Outcomes

- C++ introduction
- Variables
- Name space
- Control flow
- Functions
- Console I/O
- I/O redirection

Why Object Oriented Programming?

- Real-world software applications are typically complex and dynamic.
  - For example, the number of statements may be quite large.
- The object-oriented paradigm offers a solution to deal with this complexity.
- How?
  - OOP paradigm divides a big, complex software into highly cohesive/loosely coupled components.

Object

- Has a boundary
- Has a private part
- Has a public part
- …

Why C++?

- C++ provides:
  - Comprehensive set of features needed for Object Oriented Programming
  - High performance object code (based on C language)
    - C plus OOP features = C++

Namespace

- In large-scale applications there is chance that different developers use the same identifiers for different variables.
- We avoid such conflicts by identifying each part of an application with its own unique namespace.
- A *namespace* is a scope for the entities that it encloses.
- Syntax:
  - Definition

```
namespace ns1{
        // members
}
```

  - Access:

```
ns1::member
```

Namespace Example:

```cpp
#include <iostream>

// first name space
namespace first_space{
    int x = 1;
}

// second name space
namespace second_space{
    int x = 2;
}

int main () {

   printf("%d\n",first_space::x); // 1

   printf("%d\n",second_space::x); // 2

   return 0;
}
```

Scope Resolution Operator

Namespace Example:

```cpp
#include <iostream>

// first name space
namespace first_space{
    int x = 1;
}

// second name space
namespace second_space{
    int x = 2;
}

int main () {

   printf("%d\n",first_space::x); // 1

   printf("%d\n",second_space::x); // 2

   return 0;
}
```

Scope Resolution Operator

```cpp
#include <iostream>

// first name space
namespace first_space{
    int x = 1;
}

// second name space
namespace second_space{
    int x = 2;
}

using namespace second_space;

int main () {

   printf("%d\n",first_space::x); // 1

   printf("%d\n",x); // 2

   return 0;
}
```

C++ Variables

- C++ Identifiers
  - Keywords/reserved words vs. Identifiers
  - Case-sensitivity and validity of identifiers
  - Meaningful names!
- Variables
  - A memory location to store data for a program
  - Must declare all data before use in program

Data Types

TYPE NAME	MEMORY USED	SIZE RANGE	PRECISION
`short` (also called `short int`)	2 bytes	−32,768 to 32,767	Not applicable
`int`	4 bytes	−2,147,483,648 to 2,147,483,647	Not applicable
`long` (also called `long int`)	4 bytes	−2,147,483,648 to 2,147,483,647	Not applicable
`float`	4 bytes	approximately 10^{-38} to 10^{38}	7 digits
`double`	8 bytes	approximately 10^{-308} to 10^{308}	15 digits
`long double`	10 bytes	approximately 10^{-4932} to 10^{4932}	19 digits
`char`	1 byte	All ASCII characters (Can also be used as an integer type, although we do not recommend doing so.)	Not applicable
`bool`	1 byte	`true`, `false`	Not applicable

Variables

- Local
- Global

The difference between them are their scopes and permanency

Control Flow

- Branching Mechanisms
  - if-else
  - switch
  - Nesting if-else
- Loops
  - while,
  - do-while
  - for

if-else Example

What expression is considered as true?

```
if(-6){
    cout << "TRUE" << endl;
}
else{
    cout << "FALSE" << endl;
}
```

Every non zero expression is TRUE in C++

What is the output?

```
int x = 5, y =2;
if (x <= y)
    if(x > 1)
        cout << 1;
else
    cout << 2;
```

Each **else** belong to the closest **if** in the same block

switch Example

What is the output?

```
switch(x){
    case 1:
        cout << "One" << endl;
    case 2:
        cout << "Two" << endl;
    case 3:
        cout << "Three" << endl;
    default:
        cout << "None" << endl;

}
```

What is the output?

```
switch(x){
    case 1:
        cout << "One" << endl;
    case 2:
        cout << "Two" << endl;
        break;
    case 3:
        cout << "Three" << endl;
    default:
        cout << "None" << endl;
```

while Loop Example

```
int count = 0;                          // Initialization
while (count < 3)                  // Loop Condition
{
        cout << "Hi ";                  // Loop Body
        count++;                        // Update expression
}
```

- Loop body executes how many times?

do-while Loop Example

```
int count = 0;                  // Initialization
do
{
        cout << "Hi ";          // Loop Body
        count++;                // Update expression
} while (count < 3);            // Loop Condition
```

- Loop body executes how many times?
- do-while loops always execute body at least once!

for Loop Example

- ```
  for (int count = 0; count<3 ; count++)
  {
        cout << "Hi ";          // Loop Body
  }
  ```
- How many times does loop body execute?
- Initialization, loop condition and update all "built into" the for-loop structure!
- A natural "counting" loop


  ```

Function Basic Concepts

- Function Declaration (Prototype)
- Function Definition
- Function Call
- Default parameters
- Call by reference vs. Call by value
- Function overloading

Bitwise Operators

&	bitwise AND
|	bitwise OR
^	bitwise XOR
~	1’s compliment
<<	Shift left
>>	Shift right

Console Output

- What can be outputted?
  - Any data can be outputted to display screen
    - Variables
    - Constants
    - Expressions (which can include all of above)

```
cout << numberOfGames << " games played.";
```

  - Cascading: multiple values in one cout

Error Output

- Output with cerr
  - cerr works same as cout
  - Provides mechanism for distinguishing between regular output and error output
- Re-direct output streams
  - Most systems allow cout and cerr to be "redirected" to other devices
    - e.g., line printer, output file, error console, etc.

Input Using cin

- cin for input, cout for output
- Differences:
  - ">>" (extraction operator) points opposite
    - Think of it as "pointing toward where the data goes"
  - Object name "cin" used instead of "cout"
  - No literals allowed for cin
    - Must input "to a variable"

```
cin >> num;
```

  - Waits on-screen for keyboard entry
  - Value entered at keyboard is "assigned" to num

I/O Redirection

- Redirect the stderr to *errfile*:

```
./console 2 > errfile
```

- Redirect the stdout to *outfile*

```
./console 1 > outfile
```

- Assign stdin to *infile*

```
./console 0 < infile
```