C++ Programming
Lecture 1: C++ Basics

Nguyen, Thien Binh
Mechatronics and Sensor Technology
Vietnamese-German University

18 March 2019

VGU
Vietnamese - German University

Outline

Outline

Course Introduction

Welcome to C++ Programming! In this course, we are going to discuss

- Basic C++ programming techniques including object-oriented programming (OOP) concepts (e.g., classes and objects, encapsulation and data hiding, abstraction and class interfaces, inheritance, polymorphism and virtual methods, templates)
- Examples and coding illustrations evolve around applications in scientific computing, in particular, numerical linear algebra.
- Throughout the lectures, we will step-by-step use C++ to develop classes and implement algorithms to handle operations with vectors and matrices, as well as commonly used solvers, e.g., the LU decomposition, for solving linear systems which arise in many scientific and engineering problems.

VGU
Vietnamese - German University

Course Introduction

Topics tentative to be covered

- Lecture 1: *C++ Basics*
- Lecture 2: *Variables, References, Pointers, and Arrays*
- Lectures 3 - 4: *Functions*
- Lecture 5: *Object-Oriented Programming: Classes*
- Lectures 6 - 7: *Inheritance and Polymorphism*
- Lecture 8: *Namespaces and Templates*
- Lecture 9: *Debugging and Exception Handling*
- Lectures 10 - 11: *A Simple PDE Approximation*

VGU
Vietnamese - German University

Course Introduction

Recommended requirements:

- Basic knowledge of programming: coding structures, data types and operators, functions, etc.
- Basic understanding of linear algebra: vector and matrix operations, common linear solvers, e.g., Gaussian elimination, LU decomposition, Thomas' algorithm, etc. (supplementary handouts will be delivered when needed).

VGU
Vietnamese - German University

Course Introduction

Course materials and references:

- Textbooks:
  1. Capper, *Introducing C++ for Scientists, Engineers, and Mathematicians*, 2nd Edition, Springer, 2001
     https://www.springer.com/gp/book/9781852334888
  2. Pitt-Francis, and Whiteley, *Guide to Scientific Computing in C++*, Springer, 2012
     https://www.springer.com/la/book/9781447127369#otherversion=9781447127352
  3. Yevick, *A First Course in Computational Physics and Object-Oriented Programming with C++*, Cambridge University, 2006
     https://www.cambridge.org/vn/academic/subjects/physics/mathematical-methods/first-course-computational-physics-and-object-oriented-prog
     format=WW&isbn=9780521827782

VGU Vietnamese - German University

Course Introduction

Course materials and references:

- Online C++ tutorials:
  1. Learn Cpp: https://www.learncpp.com/
  2. Tutorial Point: https://www.tutorialspoint.com/cplusplus/cpp_overview.htm
  3. Geeks for Geeks: https://www.geeksforgeeks.org/c-plus-plus/
  4. etc.

VGU
Vietnamese - German University

Course Introduction

Course schedule:

- Contact hours: 44h
- Period: March 18 - 29, 2019 (Mar. 18: two classes of 4 hours in the morning and afternoon, the other days: one class of 4 hours in the morning)
- Classroom: B302 (computer lab), VGU Binh Duong
- Examination: 90-minute written exam with marks

VGU
Vietnamese - German University

Course Introduction

Learning style:

- *Get your hands dirty:*
  - The more you involve in coding, the better understanding you can acquire
  - Try to code yourself all the programs in class.
- *Think before you act:*
  - Before compiling and running a piece of code, expect first what the outcomes would be.
- *Coding tasks:*
  - There will be coding tasks during which you write your programs in class.

VGU
Vietnamese - German University

1. What is C++? Why Learning C++?

What is C++?

- "C++ is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language that supports procedural (functions-based), object-oriented (classes-based), and generic programming (interfaces-based)." (*TutorialPoint*)
- C++ was developed by Bjarne Stroustrup at Bell Labs starting from 1979 as an extension to C, originally named as *"C with Classes"*, and can be considered as a superset of C since almost all C programs could be run as C++ programs.

VGU
Vietnamese - German University

What is C++?

- C++ is an object-oriented programming (OOP) language. That is, it supports user-defined types called classes which contains not only the properties/attributes of those data types but also their associated functions/methods used as the interface to interact with the outside program. An object is a class realization.
- Example: a table is an object with
  - Properties: material (wood, steel, glass, etc.), size (length, height, width, etc.), design (round, rectangular, L-shaped, etc.)
  - Functionality: used to place things, to work on, to have meals, to decorate a living room, etc.
  - In C++, a user can define a class named *Table* which includes both the properties and functionality aforementioned. Furthermore, s/he can realize particular objects of that class, e.g., *my table, your table*, etc.

VGU
Vietnamese - German University

What is C++?

- Key concepts of OOP:
  - Encapsulation: a mechanism that
    - ★ binds together the data/attributes and functions/methods that manipulate the data/attributes into user-defined classes
    - ★ keeps these implementations hidden away from users in order to prevent data misuse or interception.
  - Abstraction: users work with objects at their abstract level, i.e.,
    - ★ their important characteristics represented through their interfaces
    - ★ users don't need to worry about the underlying details of how the objects are functioning.

  Example: a laptop computer is a multi-purpose object which comprises of various and complex components served for different functionalities. Yet, a user need not to master how each component of a laptop works in order to use the laptop efficiently. All s/he needs to know is how to use it (*abstraction*), and all the other detail is unrelated and kept hidden from the user (*encapsulation*).

VGU Vietnamese - German University

What is C++?

- Key concepts of OOP:
  - Inheritance: represents an *"is-a"* relationship of objects.
    - ★ Inheritance allows the definition of derived classes which inherit the properties and functionality of a base class, i.e., a more general class.
    - ★ Example: (inherited classes: from generality to specifics) shape → polygons → rectangles → squares.
  - Polymorphism: the ability to process objects differently depending on their data types or classes.
    - ★ Example: the same method *area* computes the area of an object differently depending on what class the object belongs to.

VGU
Vietnamese - German University

Why Learning C++?

- Benefits from OOP:
  - Modularity:
    - ★ Thanks to encapsulation, all defined objects in a program are self-contained and interact with each other through their interfaces only.
    - ★ This makes code maintenance and debugging more feasible since errors could be narrowed down and traced back easier.
    - ★ Furthermore, modularity enables code fixing or upgrading to be done simultaneously since all classes are independent from one another.
    - ★ It also makes possible code packaging to provide users pre-compiled shared libraries.
  - Code re-usability:
    - ★ Thanks to inheritance and templates, the same properties and methods in derived classes could be directly inherited from a base class.
    - ★ This reduces code redundancy (the same piece of code copied and pasted to many different functions/procedures).
    - ★ This also relieves code fixing and modifications, as well as speeds up implementation time (whatever modified in a base class automatically takes effects in it derived classes).

VGU Vietnamese - German University

Why Learning C++?

- Benefits from OOP:
  - Flexibility: thanks to polymorphism, one needs not to define things like *areaPolygon*, *areaRectangle*, *areaSquare* to compute the area of different types of polygons. Rather, one just needs one `virtual` *area* for all polygons. This reduces unnecessary complexity and enhances code readability.
- Efficiency: C++ grants programmers access to low-level hardware resources which makes C++ one of the fastest programming languages.
- Popularity: C++ is consistently ranked as one of the most popular programming languages (ranked #4 (04/2019), https://www.tiobe.com/tiobe-index/)

VGU
Vietnamese - German University

Why Learning C++?

- C++ has a vast ecosystem of applications, shared libraries, and tools (https://en.cppreference.com/w/cpp/links/libs, https://www.mentofacturing.com/vincent/implementations.html)
  - OS: MS Windows, Apple MacOS, Apple iOS, etc.
  - Applications: MS Office (Windows-based), LibreOffice (Linux-based), Adobe systems, etc.
  - Web browsers: MS Edge, Mozilla Firefox, Google Chrome, Safari, Opera, etc.
  - Websites: Facebook, Amazon, Paypal, etc.
  - Game engine architecture: EntityX, EntityPlus, Box2D, etc.
  - 3D graphics: VTK (visualization toolkit), OpenGL, etc.
  - Maths: Eigen (vector-matrix library), Blitz++ (high-performance vector-matrix library), LAPACK++ (wrappers for LAPACK and BLAS), DEAL.II (package for FEM approximations of partial differential equations), etc.

VGU
Vietnamese - German University

Getting Started

- In order to write and compile a C++ code, one needs the following ingredients
  - A editor: on which programmers write their source codes. Basically, any text editor can serve for this purpose.
  - A C++ compiler: to compile the source codes into object files, and link them to make an executable.
  - A C++ debugger: tools to find and fix coding errors.
  - A console to run the executable: e.g., Windows Power Shell, Linux terminal
- Integrated Developer Environments (IDEs): contains all the things one needs to write, compile, link, run, and debug a code
- Makefile: a manual way to compile source files and link object files $\Rightarrow$ More freedom and compiling options, no needed time to load heavy IDEs, favorable for Linux-based programming

VGU
Vietnamese - German University

Getting Started

1. Using an IDE:
   - Visual Studio Community 2017: for Windows and MacOS https://visualstudio.microsoft.com/vs/
   - Code::Blocks: cross-platform (Windows, MacOS, Linux) http://www.codeblocks.org/downloads (recommended in this course)
2. Using a Makefile: (recommended in this course)
   - Editor: Codelite (https://codelite.org/), Notepad++ (https://notepad-plus-plus.org/)
   - C++ compiler and debugger: g++ for Windows Power Shell through MinGW (http://www.codebind.com/cprogramming/install-mingw-windows-10-gcc/)
   - Make for makefile: (http://gnuwin32.sourceforge.net/packages/make.htm)

VGU Vietnamese - German University

Outline

The First C++ Program

```cpp
/* Block comment:
 * The first C++ code
 * Coded by Thien Binh Nguyen, 28 jan 2019 */
#include <iostream>
using namespace std;
int main()
{
   // line comment: print "Hello World" to the screen
   cout << "Hello World!" << endl;
   return 0;
}
```

Listing 1: HelloWorld.cpp

The First C++ Program

Source file structure:

- *Lines 1 - 3 and 8:* multiple-line (in between `/*...*/`) and single-line (after `//`) comment lines
- *Line 4:* a preprocessor directive, to include the contents of the header file `iostream`, which is a C++ standard library for input/output streaming, e.g., `cout <<` (print the data to a standard console) and `cin >>` (read data from a keyboard).
- *Line 5:* to use namespace `std` where function `cout` and `endl` belongs to. If `std` is not specified, `std::cout` and `std::endl` which indicates their scope must be used.

The First C++ Program

Program structure:

- *Lines 9 - 10*: *statements* which contain instructions to the computer. All statements must be ended with a semi-colon (`;`), otherwise C++ will complain with a syntax error message. Here, `cout` is to print the string `Hello World` to the screen, `endl` is to break and enter a new line, `return 0` is a returned type of function `main`, which indicates that the program is run successfully.
- *Lines 6 - 7, 11*: the special `main` function in C++, which always runs first. Every C++ program must have a `main` function, otherwise it will fail to compile. The content in-between the curly bracket { } is called the function *body*.

The First C++ Program

Common C++ header files:

- `iostream` standard stream objects, e.g., `cout` cout and `cin`
- `fstream` if you want to read and write files
- `cstdlib` for general operations, e.g., dynamical memory allocation, random numbers, etc.
- `cassert` for assertion commands
- `cmath` common mathematics functions
- Others: `https://en.cppreference.com/w/cpp/header`

VGU
Vietnamese - German University

The First C++ Program

How to compile a C++ source file?

- 2 steps from C++ source files to an executable:
  - First, each C++ `.cpp` source file is compiled to a `.o` object file, which is the process of translating a high-level programming to machine language, so that the computer CPU can understand it. Here, the compiler will check all errors relating to C++ syntax correctness, and aborts the compilation if there is any.
    ```
    g++ -Wall -g -c HelloWorld.cpp
    ```
  - Second, the compiler will link all object files and included libraries to create an executable (`.exe`) or (`.out`). Linking errors will be reported at this phase.
    ```
    g++ -Wall -g -o aaa.exe HelloWorld.o
    ```

VGU
Vietnamese - German University

The First C++ Program

How to compile a C++ source file?

- `g++` is the GNU C++ compiler. To check if g++ is successfully installed, type the following command line in Windows Power Shell
  `g++ --version`

```
PS D:\work\teaching_vgu\MSST_Cpp_Mar18to30\Lectures_sides\Beamer\Lec1\codes> g++ --version
g++.exe (tdm64-1) 5.1.0
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

- `-Wall -c -o` are the compiler flags. Common compiler flags include
  - `-Wall`: to turn on all compilation warning messages
  - `-c`: to compile a `.cpp` source file to a `.o` object file
  - `-o`: to link the object files to make an executable, e.g., `aaa.exe`
  - `-g`: to turn on the debugger
  - `-Werror`: to convert all warnings into error messages (strict!)
  - `-O`: to compile with optimization, i.e., compilation takes longer time, but the execution time is much reduced.

VGU
Vietnamese - German University

The First C++ Program

How to compile a C++ source file?

- Compiling with a `makefile`:
  - Convenient when compiling and linking many source files
  - User-defined compilation process ⇒ More options and freedom
  - For `makefile`, indentation is critical
  - To compile, simply type `make -f [makefile_name]`

```
1  CC       = g++
2  CFLAGS   = -g -Wall
3  LDFLAGS  =
4  OBJS     = HelloWorld.o
5  TARGET   = aaa
6
7  all: $(TARGET)
8
9  $(TARGET): $(OBJS)
10   $(CC) $(CFLAGS) -o $(TARGET) $(OBJS) $(LDFLAGS)
11
12 HelloWorld.o: HelloWorld.cpp
13   $(CC) $(CFLAGS) -c -o HelloWorld.o HelloWorld.cpp $(LDFLAGS)
14
15 clean:
16   rm -f $(OBJS)
```

Outline

Data Types

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5    int      i, j;          // variables of type integer
6    double     x, y, z;         // variables of type double
7    i = 10; j = 20;
8    x = 5.3; y = 1.0e+2; z = x * y;
9
10   cout.precision(16);       // double precision
11   cout << "i = " << i << ", j = " << j << endl;
12   cout << "fixed:" << endl;
13   cout << fixed;
14   cout << "x = " << x << ", y = " << y << ", z = " << z << endl;
15   cout << "scientific:" << endl;
16   cout << scientific;
17   cout << "x = " << x << ", y = " << y << ", z = " << z << endl;
18
19   return 0;
20 }
```

Listing 3: DataType.cpp

Data Types

- `i`, `i`, `j`, `y`, `z`: variables (to be discussed later)
- `int`, `double`: data types $\Rightarrow$ to indicate how much memory is allocated to store a variable of a data type
- Generally, a memory allocation of size `n` bits can store to 2^n values. For example, a memory of 1 byte (8 bits) can store up to $2^8 = 256$ values.

VGU
Vietnamese - German University

Data Types

- Data types:
  - `char`: stores characters
  - `int`: stores integer numbers
  - `float`: stores single precision (i.e., 7 decimal digits of precision) floating-point numbers
  - `double`: stores double precision (i.e., 15 decimal digits of precision) floating-point numbers
  - `bool`: stores boolen values, requires 1 byte of memory. In C++, `(bool) TRUE = (int) 1, (bool) FALSE = (int) 0`
- Data modifiers: used before basic data types to modify the their range
  - `signed`: signed values
  - `unsigned`: non-negative values
  - `short`
  - `long`

VGU
Vietnamese - German University

Data Types

Question: How to check the range of a data type??? ⇒ `sizeof(data type)`

```
1  #include <iostream>
2  #include <climits>          // for int, char macros
3  #include <cfloat>           // for float, double macros
4  #include <cstddef>          // for size_t type
5  using namespace std;
6  int main()
7  {
8    cout << "int range min = " << INT_MIN << ", max = " << INT_MAX << endl;
9    cout << "size(int) = " << sizeof(int) << " byte(s)" << endl;
10   cout << "short int range: min = " << SHRT_MIN << ", max = " << SHRT_MAX << endl;
11   cout << "size(short int) = " << sizeof(short int) << " byte(s)" << endl;
12   cout << "long int range: min = " << LONG_MIN << ", max = " << LONG_MAX << endl;
13   cout << "size(long int) = " << sizeof(long int) << " byte(s)" << endl;
14   cout << "unsigned int range: min = " << 0 << ", max = " << UINT_MAX << endl;
15   cout << "size(unsigned int) = " << sizeof(unsigned int) << " byte(s)" << endl;
16
17   return 0;
18 }
```

Listing 4: DataTypes.cpp

VGU Vietnamese - German University

Data Types

Exercise: Check the range and data size of the following types.

1. `bool`
2. `signed char`, `unsigned char`
3. `float`, `short float`, `long float`
4. `double`, `short double`, `long double`

Refs:

- `https://docs.microsoft.com/en-us/cpp/c-language/cpp-integer-limits?view=vs-2017`
- `https://docs.microsoft.com/en-us/cpp/cpp/floating-limits?view=vs-2017`

VGU Vietnamese - German University

typedef

- `typedef`: to define aliases for a type.
  `typedef type ALIAS 1, ALIAS 2;`

```
1    typedef double DISTANCE, VELOCITY, ACCELERATION;
2    DISTANCE x;         // = double x;
3    VELOCITY v;           // = double v;
4    ACCELERATION a;     // = double a;
```

Listing 5: DataTypes.cpp

Operators

Commonly used operators

- Unary: `++a (a=a+1), a++ (a=a+1), --a (a=a-1), a-- (a=a-1), -a, &a (address of a)`
- Binary: `a+b, a-b, a*b, a/b, i%j (modulus of i/j where i,j are integers)`
- Assignment: `a=b, a+=b (a=a+b), a-=b (a=a-b), a*=b (a=a*b), a/=b (a=a/b), i%=j (i=i%j)`
- Relational: `a>=b a<=b, a>b, a<b`
- Logical: `a==b, a!=b, a||b (a OR b), a&&b (a AND b)`

VGU Vietnamese - German University

Operators

Exercise: What are the results of the following operations?

```cpp
#include <iostream>
using namespace std;

int main()
{
  int      i(4), j(2);
  double     x(6.0), y(2.0);

  cout << "++x␣=␣" << ++x << endl;     // prefix operator
  cout << "x␣=␣" << x << endl;
  cout << "x++␣=␣" << x++ << endl;     // postfix operator
  cout << "x␣=␣" << x << endl;

  cout << "i␣%␣j␣=␣" << i%j << ",␣i/j␣=␣" << i/j << endl;
  cout << "x/y␣=␣" << x/y << ",␣x/i␣=␣" << x/i << ",␣i/x␣=␣" << i/x << endl;
  cout << "&x␣=␣" << &x << endl;
  x += y;
  cout << "x␣+=␣y␣=␣" << x << endl;

  return 0;
}
```

Listing 6: Operator.cpp

VGU Vietnamese - German University

Type Conversion

- Type promotion and conversion:

```
1     double x;
2     float y;
3     int i;
4     long j;
5     x = 1 + 3.7;      // int promoted to double type
6     y = 3.7;         // float promoted to double type
7     j =  i;          // int promoted to long int type
```

- Type demotion:

```
1     int i, j, k;
2     i = 3.142;           // double truncated to into
3     j = -3.142;          // double truncated to into
4     k =  2.997e40;       // Undefined.
5     cout << "i = " << i << ", j = " << j << endl;
6     cout << "k = " << k << endl;
```

VGU Vietnamese - German University

Type Conversion

- Casts: explicit type conversion

```
1  double x;
2  int i = 4;
3  x = static_cast<double>(i);
```

- Exercise: $x = y$?

```
1  double x, y;
2  x = (1.0 + 1) / 2;
3  y = 0.5 + 1 / 2;
4  cout << "x = " << x << ", y = " << y << endl;
```

VGU
Vietnamese - German University

Maths functions

Common maths functions with `<cmath>`:

- Power: `pow`, `sqrt`, ...
- Exponential: `exp`, `exp2`, `log`, `log10`, ...
- Trigonometric: `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, ...
- Rounding: `ceil`, `floor`, `round`, ...
- Error functions: `erf`, `erfc`, `tgamma`, `lgamma`, ...

Ref: `http://www.cplusplus.com/reference/cmath/`

Control Structures

- `if` statement: bi-branching

```
if (condition)
  {statements;}
else
  {statements;}
```

- Condition expression operator

```
(condition) ? result 1 : result 2;
```

- `switch` statement: multiple branching

```
switch (condition) {
  case 1: statements; break;
  case 2: statements; break;
      :
  case n: statements; break;
  default: statements; break;
}
```

VGU Vietnamese - German University

Control Structures

- `for` loop: finite iterations

```
for (initializing; condition; stepping)
  {statements;}
```

- `while` loop: infinite iterations, `condition` is checked before `statements` are executed

```
while (condition)
  {statements;}
```

- `do` loop: infinite iterations, `statements` are executed before `condition` is checked

```
do
  {statements;}
while (condition);
```

VGU
Vietnamese - German University

Control Structures

Example:

$$I = \sum_{k=0}^{100} k^2 = 338350$$

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int i, sum;

  // for loop
  sum = 0;
  for (i = 0; i < 101; ++i) sum += pow(i, 2);

  // while loop
  i = 0; sum = 0;
  while (i < 101) { sum += pow(i, 2); ++i; }

  // do loop
  i = 0; sum = 0;
  do { sum += pow(i, 2); ++i; } while (i<101);

  cout << "sum = " << sum << endl;
  return 0;
}
```

Listing 7: Sum.cpp

Control Structures

Exercise:

1. What is the result of `sum`?

```
1 sum = 0;
2 for (int i = 0; i < 10; ++i);
3   sum += 100;
4 cout << "sum = " << sum << endl;
```

2. What is the result of `x`?

```
1 x = 10;
2 if (x = 5)
3   x = 50;
4 cout << "x = " << x << endl;
```

VGU Vietnamese - German University

Coding 1 I

Coding 1:

1. Investigate how to use the `cmath` library by writing a code to compute the following expressions. Take $x = 2$ and $x = -3$.

$$\sqrt{\cos(\pi x/2) + sin^3(\pi x/3)} \tag{1}$$

$$\sqrt[3]{2\exp(2x) - |x|^3} \tag{2}$$

2. `std::cout <<` is used to print a result to the console. `std::cin >>`, on the other hand, is used to read a input parameter in from the keyboard. Write code that asks a user to enter two integers from the keyboard and then writes the product of these integers to the screen.
3. Use the following code to complete the tasks.

VGU Vietnamese - German University

Coding 1 II

```cpp
#include <iostream>
int main()
{
   double p, q, x, y;
   int j;
   return 0;
}
```

1. Set the variable x to the value 5 if either p is greater than or equal to q, or the variable j is not equal to 10.
2. Set the variable x to the value 5 if both y is greater than or equal to q, and the variable j is equal to 20. If this compound condition is not met, set x to take the same value as p.
3. Set the variable x according to the following rule.

$$x = \begin{cases} 0, & p > q, \\ 1, & p \leq q, \text{ and } j = 10, . \\ 2, \text{otherwise} \end{cases}$$

VGU Vietnamese - German University

Coding 1 III

4. What is wrong with the following `while` loop?

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6     double positive_numbers[4] = {1.0, 5.65, 42.0, 0.01};
7     double max = 0.0;
8     int count = 0;
9     while (count < 4)
10      if (positive_numbers[count] > max)
11        max = positive_numbers[count];
12    return 0;
13  }
```

Coding 1 IV

5. π can be calculated by the following infinite series

$$\sum_{k=0}^{\infty} \frac{1}{k^4} = \frac{\pi^4}{90}. \tag{3}$$

Write a code to approximate the π number with 5 terms, 10 terms, and 100 terms. Check how much the accuracy is improved.

6. Numerical integration: the trapezoidal rule

$$\int_{x_0}^{x_m} f(x)dx \approx h\left[\frac{1}{2}f_0 + f_1 + \ldots + f_{m-1} + \frac{1}{2}f_m\right], \tag{4}$$

where $h = (x_m - x)/m$.

VGU Vietnamese - German University

Coding 1 V

Write code to apply the trapezoidal rule with $m = 20$ to approximate

$$\int_0^{\pi} \sin(x)dx, \tag{5}$$

and compare your obtained estimate with the exact solution. Try with different values of m.

7. The Newton's method for approximating nonlinear equations $f(x) = 0$:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}, \quad x = 1, 2, 3, \ldots, \tag{6}$$

where the iteration is terminated if the approximation sufficiently approaches to the fixed point solution by the estimate

$$|x_i - x_{i-1}| < \varepsilon, \tag{7}$$

VGU Vietnamese - German University

Coding 1 VI

with ε is called the tolerance.
Write a code using a `while` to approximate equation

$$f(x) = e^x + x^3 - 5 = 0 \tag{8}$$

with $\varepsilon = 1.0E - 5$ and $x_0 = 0$ as an initial guess. Print out the convergent solution at each iteration.
Hint:

- You don't need to store all x_i for each iteration. Just use x_n and x_p for x_i and x_{i-1}.
- Use `assert` to check if the denominator in Eq. (6) is zero or not.

Reading

1. Capper, *Introducing C++ for Scientists, Engineers, and Mathematicians*, **Chapters 1 - 4**
2. Pitt-Francis, and Whiteley, *Guide to Scientific Computing in C++*, **Chapters 1 - 2**