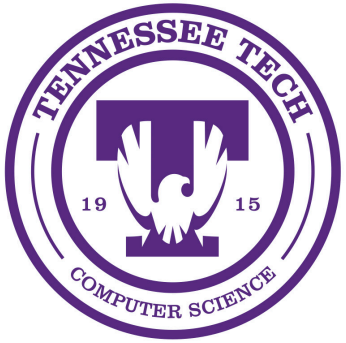
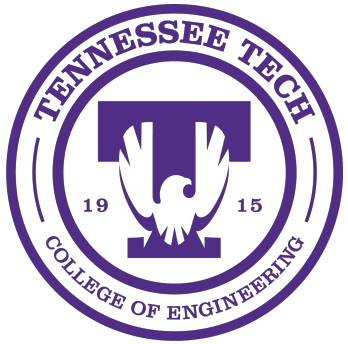


Tennessee
TECH



Introduction to C++

Ahsan Ayub

Ph.D. Student, Department of Computer Science

Graduate Research Assistant, CEROC

CSC 1300: Introduction to Programming

Wednesday, August 25, 2021

The features of C++ language

- Open ISO-standard
- Compiled
- Strongly-typed
- Supports both dynamic and static type checking
- Offers procedural and object-oriented programming paradigms
- Portable
- Upwards compatible with *C language*
- Over 3,000 library support

Source: https://www.cplusplus.com/info/description/

Compiled Language vs Interpreted Language

- Compiled directly into the machine code that the processor can execute.
- Faster and more efficient
- Allows the developer to have more control over the hardware (e.g., memory management).
- Example: C, C++, Go, etc.

- *Interpreters* run through the program line-by-line and execute each command.
- Becoming faster every year
- Popular in the Command Line Interface (CLI) tools and the web programming.
- Example: PHP, Ruby, Python, JavaScript, etc.

Source: https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/

History of C++ (1/2)

- Bjarne Stroustrup designed and implemented the C++ language.
  - He began to work on ”C with Classes” for object oriented programming functionality shortly after **1979**.
- In **1983**, the name of the language was changed to C++.
- In **1985**, Stroustrup's reference to the language entitled *The C++ Programming Language* was published.
- In **1998**, the C++ standards committee published the first international standard for C++ ISO/IEC 14882:1998, which is also known as **C++98**.

Source: https://www.cplusplus.com/info/history/

History of C++ (2/2)

- The following are the versions of the C++ programming language –
  - C++98 (ISO/IEC 14882:1998)
  - C++03 (ISO/IEC 14882:2003)
  - **C++11** (the second major version of C++)
  - C++14
  - **C++17** (the third major version of C++)
  - **C++20** (the fourth major version of C++)

My First C++ Program: Hello World (1/7)

- Code – textual representation of a program
- Line – a row of text
- Main – where the program starts (a function)
- Braces – enclose a block of statements
- Header Files – Used for the input/output features (declared in the standard library)
- Statement – a program instruction (all end in a semicolon ";")
- Return statement – where the program ends. The zero in return 0; tells the operating system that the program is ending without an error.

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

int main()
{
	cout << "Hello class!\n";
	return 0;
}
```

My First C++ Program: Hello World (2/7)

- Code – textual representation of a program
- Line – a row of text
- Main – where the program starts (a function)
- Braces – enclose a block of statements
- Header Files – Used for the input/output features (declared in the standard library)
- Statement – a program instruction (all end in a semicolon ";")
- Return statement – where the program ends. The zero in return 0; tells the operating system that the program is ending without an error.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6          cout << "Hello class!\n";
7          return 0;
8  }
```

My First C++ Program: Hello World (3/7)

- Code – textual representation of a program
- Line – a row of text
- Main – where the program starts (a function)
- Braces – enclose a block of statements
- Header Files – Used for the input/output features (declared in the standard library)
- Statement – a program instruction (all end in a semicolon ";")
- Return statement – where the program ends. The zero in return 0; tells the operating system that the program is ending without an error.

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

int main()
{
	cout << "Hello class!\n";
	return 0;
}
```

My First C++ Program: Hello World (4/7)

- Code – textual representation of a program
- Line – a row of text
- Main – where the program starts (a function)
- Braces – enclose a block of statements
- Header Files – Used for the input/output features (declared in the standard library)
- Statement – a program instruction (all end in a semicolon ";")
- Return statement – where the program ends. The zero in return 0; tells the operating system that the program is ending without an error.

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

int main()
{
	cout << "Hello class!\n";
	return 0;
}
```

My First C++ Program: Hello World (4/7)

- Code – textual representation of a program
- Line – a row of text
- Main – where the program starts (a function)
- Braces – enclose a block of statements
- Header Files – Used for the input/output features (declared in the standard library)
- Statement – a program instruction (all end in a semicolon ";")
- Return statement – where the program ends. The zero in return 0; tells the operating system that the program is ending without an error.

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

int main()
{
	cout << "Hello class!\n";
	return 0;
}
```

My First C++ Program: Hello World (6/7)

- Code – textual representation of a program
- Line – a row of text
- Main – where the program starts (a function)
- Braces – enclose a block of statements
- Header Files – Used for the input/output features (declared in the standard library)
- Statement – a program instruction (all end in a semicolon ";")
- Return statement – where the program ends. The zero in return 0; tells the operating system that the program is ending without an error.

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

int main()
{
	cout << "Hello class!\n";
	return 0;
}
```

My First C++ Program: Hello World (7/7)

- Code – textual representation of a program
- Line – a row of text
- Main – where the program starts (a function)
- Braces – enclose a block of statements
- Header Files – Used for the input/output features (declared in the standard library)
- Statement – a program instruction (all end in a semicolon ";")
- Return statement – where the program ends. The zero in return 0; tells the operating system that the program is ending without an error.

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

int main()
{
	cout << "Hello class!\n";
	return 0;
}
```

Variables (1/2)

- A container (storage area) to hold data
- Can only hold one thing at a time
- The contents of the container (variable) may change or vary
- Must be defined with a statement (called a variable definition)

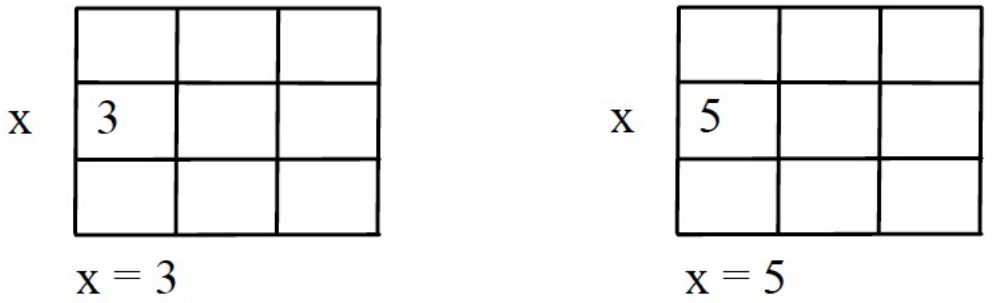


Image Source: “Let Us C” by Yashwant Kanetkar

Variables (2/2)

- Data Types
- Variable Name
- Value

```
int LaptopPrice = 1099;
```

Integer Data Types

- Short (`short`)
  - Size: 2 bytes or 16 bits
  - *Signed* range: -32,768 to 32,767
  - *Unsigned* range: 0 to 65,535
- Integer (`int`)
  - Size: 4 bytes or 32 bits
  - *Signed* range: -2,147,483,648 to 2,147,483,647
  - *Unsigned* range: 0 to 4,294,967,295
- Long long (`long long`)
  - Size: 8 bytes or 64 bits

```
int LaptopPrice = 1099;
```

Floating Point Data Types

- Float (`float`)
  - Size: 4 bytes or 32 bits
  - Range: 3.4E +/- 38 (7 digits)
- Double (`double`)
  - Size: 8 bytes or 64 bits
  - Range: 1.7E +/- 308 (15 digits)

```
float LaptopPrice = 1208.90;
```

Boolean Data Type

- Boolean (`bool`)
  - Size: 1 byte or 8 bits
  - Range: true or false

```
bool IsLaptopPurchased = true;
```

Further Reading (Optional): https://docs.microsoft.com/en-us/cpp/cpp/fundamental-types-cpp?view=msvc-160

Naming a Variable

- Can contain letters (A-Z, a-z), digits (0-9) and underscores (_)
- Must begin with a letter or an underscore (_)
- Case sensitive: `myVar` and `myvar` are *different variables*
- Cannot contain whitespaces or special characters like !, #, %, etc.
- Suggestions:
  - Make the habit of assigning the proper variable name
  - Multiple name identifier
    - Camel Case – `laptopPrice`
    - Pascal Case – `LaptopPrice`
    - Snake Case – `laptop_price`

Source: https://www.w3schools.com/cpp/cpp_variables_identifiers.asp

Basic Output

- `cout` (stands for console output) sends formatted output to standard output devices (e.g., the screen of your monitor)
- `<<` Operator is used to display the output on screen

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

int main()
{
	cout << "Hello class!\n";
	return 0;
}
```

```
Hello class!
```

Output Tracing

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

int main()
{
        cout << ”My name is Ahsan.\n”;
        return 0;
}
```

??

Output Tracing

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

int main()
{
        cout << ”My name is Ahsan.\n”;
        return 0;
}
```

```
My name is Ahsan.
```

Basic Output – Escape Sequence

- `\n` – newline – causes output to go down to next line
  - `endl` – endline – causes output to go down to next line
- `\b` – backspace – backup one space
- `\r` – return – like pressing home key – goes to beginning of line
- `\\` - backslash – prints one backslash to the screen
- `\’` – single quote – prints one ‘ to the screen
- `\”` – double quote – prints one “ to the screen

Output Tracing

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

int main()
{
        cout << ”My name is Ahsan.” << endl;
        return 0;
}
```

```
My name is Ahsan.
```

Output Tracing

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

int main()
{
	cout << ”*\n**\n***\n****\n*****\n”;
	return 0;
}
```

```
??
```

Output Tracing

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

int main()
{
	cout << ”*\n**\n***\n****\n*****\n”;
	return 0;
}
```

```
*
**
***
****
*****
```

Programming Challenge

Write a C++ program that will display the following message?

```
*****
****
***
**
*
```

Programming Solution

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

int main()
{
	cout << ”*****\n****\n***\n**\n*\n”;
	return 0;
}
```

```
*****
****
***
**
*
```

Output Tracing

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

int main()
{
	int LaptopPrice = 1099;
	cout << LaptopPrice << endl;
	return 0;
}
```

```
??
```

Output Tracing

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

int main()
{
	int LaptopPrice = 1099;
	cout << LaptopPrice << endl;
	return 0;
}
```

```
1099
```

Output Tracing

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

int main()
{
	float LaptopPrice = 1208.90;
	cout << “Laptop’s Price with the Sales Tax: ” << LaptopPrice << endl;
	return 0;
}
```

```
Laptop’s Price with the Sales Tax: 1208.9
```

Basic Input

- `cin` (stands for console input) takes formatted input from the standard input devices (e.g., the keyboard of your computer)
- `>>` Operator is used to take the input

Basic Input – A Sample Program (1/3)

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

int main()
{
        int UserAge;

        cout << ”Please enter your age: ”;
        cin >> UserAge;

        cout << ”Your age is: ” << UserAge << endl;
        return 0;
}
```

```
Please enter your age: _
```

Basic Input – A Sample Program (2/3)

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

int main()
{
        int UserAge;

        cout << ”Please enter your age: ”;
        cin >> UserAge;

        cout << ”Your age is: ” << UserAge << endl;
        return 0;
}
```

```
Please enter your age: 52
```

Basic Input – A Sample Program (3/3)

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

int main()
{
        int UserAge;

        cout << ”Please enter your age: ”;
        cin >> UserAge;

        cout << ”Your age is: ” << UserAge << endl;
        return 0;
}
```

```
Please enter your age: 52
Your age is: 52
```

Taking Multiple Inputs – A Sample Program

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

int main()
{
        int UserAge;
        double UserSalary;

        cout << ”Please enter your age: ”;
        cin >> UserAge;

        cout << ”Please enter your salary: ”;
        cin >> UserSalary;

        cout << “Your age is ” << UserAge << “ and your salary is “ << UserSalary << endl;
        return 0;
}
```

Comments

- Makes the program easier to read and understand
- C++ compilers ignore the comments during the execution.
- There are two ways to add comments:
  - Single line comment (`// …`)

```
// Declaring a variable to store the laptop price
int LaptopPrice;
LaptopPrice = 1099; // Initializing the laptop price variable as $1,099
```

  - Multi-line comment (`/* … */`)

```
/*
        Initializing a variable to store the laptop price as $1,099
*/
int LaptopPrice = 1099;
```

Source: https://www.programiz.com/cpp-programming/comments

Some Thoughts on Comments

- Self reference for the future usage of the codebase
- Make lives a lot easier for other developers to assess, test, and work on
- Stick to 80 character per lines
- Different ways to add comments will be covered throughout the class
  - Use the following multi-line comments to express your ownership of the code –

```
/*
 * Filename: myProgram.cpp
 * Author: Ahsan Ayub
 * Date: August 4, 2021
 * Purpose: This program computes the area of a triangle.
 */
```

Whitespaces (1/3)

- Whitespace refers to blank spaces between items within a statement, and to blank lines between statements.
- A compiler ignores most whitespace. Below is four examples of identical code that is spaced differently. All four examples will work!

`cout << "How old is your cat?\n";`
 	
`cin >> age;`
	`cout << "How old is your cat?\n";`
	
	`cin >> age;`

`cout<<"How old is your cat?\n";` `cin>>age;`	`cout<<"How old is your cat?\n";cin>>age;`

Whitespaces (2/3)

- Good practice is to deliberately and consistently use whitespace to make a program more readable.
- Each statement usually appears on its own line.

```
x = 25;
y = x + 1;
if (x == 5)
     y = 14;
```

```
x = 25; y = x+1;
if(x==5) y=14;
```

Whitespaces (3/3)

- Most items are separated by one space (and not less or more). No space precedes an ending semicolon.

```
tempC = 25;
tempF = ((9 * tempC) / 5) + 32;
tempF = tempF / 2;
```

```
tempC=25;
tempF=((9*tempC)/5)+32;
tempF=tempF/2;
```

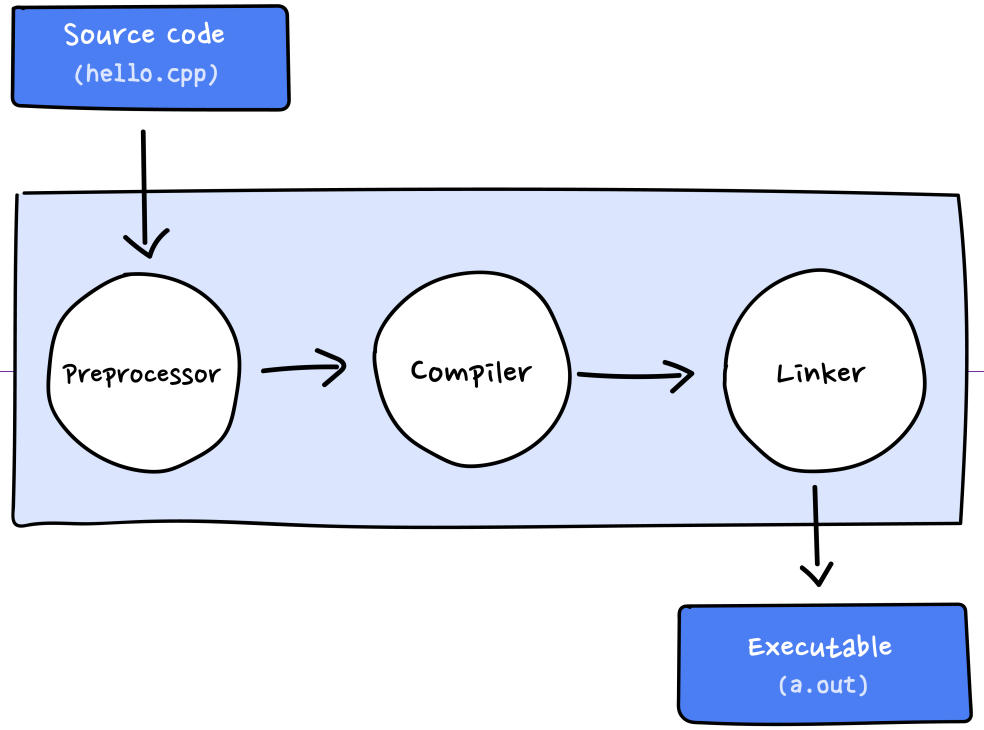
Indentions

- Indentions can be 3 to 4 spaces, or a tab.
- The yellow highlighted parts to the right are indentions. All indentions in this image are either a single tab or two tabs.

```
1   int main()
2   {
3       cout << "This is a programming statement.\n";
4
5       if (7 > 4)
6       {
7           cout << "It is greater.\n";
8       }
9       else
10          cout << "It is less.\n";
11
12      return 0;
13  }
```

Compile and Execute: C++ Program

- We read and write code in human-like language (source code).
- A C++ compile (e.g., g++) translates the source code into machine language code / object code (stored on disk).
- A linker links the object code with standard library routines and creates an executable image.
- When executed, the executable is loaded from the disk to memory and the computer's CPU (Central Processing Unit) executes the program one instruction at a time.



Source: https://www.codecademy.com/articles/cpp-compile-execute-locally

Compile and Execute: An Example

Terminal / Command

```
> g++ hello.cpp
> ./a.out
```

```
> g++ -c hello.cpp
> g++ -o hello.o hello
> ./hello
```

```
> g++ -o hello hello.cpp
> ./hello
```

```
> g++ -std=c++17 -o hello hello.cpp
> ./hello
```

hello.cpp

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

int main()
{
        cout << “Hello World!” << endl;
        return 0;
}
```

Errors and Warnings

- Syntax errors – violates a programming language's rule on how symbols can be combined to create a program (compile-time error)

```
int LaptopPrice = 1099 // Missing a semicolon at the end of statement
```

- Logic Error – encounters error during the execution (run-time error)

```
int x = 10, y = 0;       // Initialized two variables
cout << x / y << endl;   // Divide a number with 0
```

- Warnings – indicates a possible logic error but doesn't stop the compiler from creating an executable program

```
> g++ -Wall -o hello hello.cpp (to show all warnings)
```

Remarks

- The slides are adapted from Ms. April Crockett.
- Reference Books
  - ZyBooks, TNTech CSC 1300: Introduction to Problem Solving and Computer Programming
  - Kanetkar, Yashavant P. "Let Us C."
  - Balagurusamy, E. "Object-Oriented Programming with C++."