


Math 4997-1

Lecture 1: Introduction and Getting started

Patrick Diehl 

<https://www.cct.lsu.edu/~pdiehl/teaching/2020/4997/>

This work is licensed under a Creative Commons "Attribution-NonCommercial-NoDerivatives 4.0 International" license.



Outline

Administration/Organization

Getting started

Looping and counting

Working with strings

Summary

References

Administration/Organization

Important dates

Lectures

Tuesday and Thursday, 09:00 to 10:20, 130 LCKT

Grading

- ▶ Homework 30%
- ▶ Project 20%
- ▶ Midterm exam 20%
- ▶ Final exam 30%

Exams

- ▶ Midterm exam: 13.10 during lecture
- ▶ Final exams: 10.12 from 12:30 to 2:30

More: Syllabus and Timeline.

Reading

Course's books

- ▶ Andrew, Koenig. Accelerated C++: practical programming by example. Pearson Education India, 2000.
- ▶ Stroustrup, Bjarne. Programming: principles and practice using C++. Pearson Education, 2014.

Assistance C++ basics

- ▶ Stroustrup, Bjarne. A Tour of C++. Addison-Wesley Professional, 2018.
- ▶ O'Dwyer, Arthur. Mastering the C++17 STL. Packt Publishing Ltd; 2017.

Submitting home work

Theory exercises

At the beginning of the lecture in printed form

Programming exercises

- ▶ Github Classroom¹ for submission of the programming exercises and the course project.
- ▶ Jupyter Server² to work in your browser on the exercises and course project³.

Note that we use these tools the first time for this course. We anticipate to do a short survey at the end of the semester.

¹ <https://www.diehlpk.de/blog/githubclassroom/>

² <https://tutorial.cct.lsu.edu/hpx>

³ <https://www.diehlpk.de/blog/jupyter-notebooks/>

Communication-Intensive (C-I) course

Mode I: Written

- ▶ Learn how to write C++ standard conform code
- ▶ Learn how to write proper documentation
- ▶ Use the pieces of the assignments to code the course project

Mode II: Technological

- ▶ Use GitHub for remote collaborative software development
- ▶ Translate mathematical and algorithms into C++ code

Getting started

A small C++ program

```
// a small C++ program
#include <iostream>

int main()
{
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Compile

```
g++ lecture1-1.cpp -o lecture1-1
```

Run

```
./lecture1-1
```

Structure of a C++ program

```
// a small C++ program  
#include <iostream>  
  
int main()  
{  
    std::cout << "Hello, world!" << std::endl;  
    return 0;  
}
```

Comments [?]

- ▶ A one line comment starts with `//`
- ▶ A comment over multiple lines starts with `/*` and ends with `*/`
- ▶ Comments are important to understand the program, especially if the code is shared

Structure of a C++ program

```
// a small C++ program
#include <iostream>

int main()
{
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Include directives

- ▶ Is needed to include functionality of the C++ standard library, e.g. IO, which is not part of the core language
- ▶ To include functionality of external libraries or structure your own code

Structure of a C++ program

```
// a small C++ program  
#include <iostream>  
  
int main()  
{  
    std::cout << "Hello, world!" << std::endl;  
    return 0;  
}
```

Main function

- ▶ Every C++ program needs a function called main returning an integer value
- ▶ Return zero means success and any other value indicates failure
- ▶ When we execute any C++ program the main function is invoked and all instructions are executed

Structure of a C++ program

```
// a small C++ program  
#include <iostream>  
  
int main()  
{  
    std::cout << "Hello, world!" << std::endl;  
    return 0;  
}
```

return statement

- ▶ The value of the return statement is passed to the program, which called the function
- ▶ One function can have multiple return statements

Built-in types⁴

Integer types

- ▶ `bool` Representation of truth values: `true` or `false`
- ▶ `unsigned` Integral type for non-negative values only
- ▶ `short` Integral type that must hold at least 32 bits
- ▶ `long` Integral type that must hold at least 64 bits
- ▶ `size_t` Unsigned Integral type

Floating points

- ▶ `float` Single precision floating point type
- ▶ `double` Double precision floating point type
- ▶ `long double` Extended precision floating point type

⁴<https://en.cppreference.com/w/cpp/language/types>

Looping and counting

Using loops and counting

Compute the sum of $1, \dots, n$

$$result = \sum_{i=1}^n i$$

Using the loop statement⁵

```
size_t result = 0;
for(size_t i = 1; i != 5; i++){
    result = result + i;
}
```

⁵<https://en.cppreference.com/w/cpp/language/for>

Using loops and counting

Using the loop statement⁵

```
size_t result = 0;
for(size_t i = 1; i != 5; i++){
    result = result + i;
}
```

Condition

- ▶ The variable `i` is only available inside the loop's body
- ▶ The loop will execute the statements in the curly braces until `i` is equal to 5
- ▶ The value of `i` is incremented after all statements are executed
- ▶ `i++` is equivalent to `i = i+1`

⁵ <https://en.cppreference.com/w/cpp/language/for>

The while statement⁶

```
size_t result = 0;
size_t i = 1;
while (i != 5) {
    result += i;
    i++;
}
```

Condition

- ▶ `i != 5` the statement within the curly braces will be repeated five times
- ▶ `!=` is the inequality operator and once `i` is equal to `5` the loop stops

⁶ <https://en.cppreference.com/w/cpp/language/while>

Conditionals⁷

Compute the sum of $f(i)$ for $i = 1, \dots, n$

$$result = \sum_{i=1}^n f(i) \text{ with } f(i) = \begin{cases} i, & \text{if } i \text{ is even} \\ i^2, & \text{else} \end{cases}$$

```
size_t result = 0;
for(size_t i = 1; i != 5; i++){
    if(i % 2 == 0)
        result = result + i;
    else
        result = result + i * i;
}
```

⁷<https://en.cppreference.com/w/cpp/language/if>

Conditionals⁷

```
size_t result = 0;
for(size_t i = 1; i != 5; i++){
    if(i % 1 == 0)
        result = result + i;
    else
        result = result + i * i;
}
```

if statement

- ▶ If the condition is `true` the statements in the `if` branch are executed
- ▶ If the condition is `false` the statements in the `else` branch are executed

Logical operator

- ▶ `%` Modulo operator for integers

⁷<https://en.cppreference.com/w/cpp/language/if>

Operators⁸

Logical operators

- ▶ `&&` Logical and
- ▶ `||` Logical or
- ▶ `!x` Logical negation

Comparison operators

- ▶ `==` Compares to equal
- ▶ `!=` Compares to unequal
- ▶ `<` Compares to be less
- ▶ `>` Compares to be higher
- ▶ `<=` Compares to be less or equal
- ▶ `>=` Compares to be higher or equal

⁸https://en.cppreference.com/w/cpp/language/operator_precedence

Working with strings

Reading strings

```
// Read person's name and greet the person
#include <iostream>
#include <string>

int main()
{
    std::cout << "Please enter your name: ";
    // Read the name
    std::string name;
    std::cin >> name;
    // Writing the name
    std::cout << "Hi, " << name << "!" << std::endl;
    return 0;
}
```

Reading strings

```
#include <string>
```

```
std::string name;
```

Variables: Definition

- ▶ Variables have a name (name) and a type
(std::string)
- ▶ We need to include the string type, since it is not in the core language
- ▶ We just defined the variable and currently it is a empty or null string

Reading strings

```
std::cin >> name;
```

Variables: Initialization

- ▶ Now we initialize the string by reading from `std::cin` and assigning the value to it
- ▶ The `<<` operator writes a string to `std::cout`
- ▶ The `>>` operator reads a string to `std::cin`

Variables can be defined in three different ways:

- ▶ `std::string name = "Peter Pan";`
- ▶ `std::string; //empty string`
- ▶ `std::string stars(3, '*') // string of three stars`

More details: https://en.cppreference.com/w/cpp/string/basic_string

More functionality of strings

```
const std::string greetings = "Hi, " + name + "!";
```

Concatenation

+ operator combines string

Defining constants

const operator to make the promise that we will not change the value later

```
const size_t length = greetings.size();
```

Getting the size

.size() operator to get the string's size

Summary

Summary

After this lecture, you should know

- ▶ Structure of a C++ program
- ▶ Handling strings
- ▶ Loops and counting
- ▶ Conditionals
- ▶ Operators
- ▶ Built-in types

References

References I