Math 4997-3

Lecture 1: Introduction and Getting started

https://www.cct.lsu.edu/~pdiehl/teaching/2019/4977/

Administration/Organization

Getting started

Working with strings

Looping and counting

Summary

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: 15.10 during the lecture
- Final: 12.10 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.

Submitting home work

Theory exercises

At the beginning of the lecture in printed form

Programming exercises

- Github Classroom[1] for submission of the programming exercises and the course project.
- Juypter Server[2] to work in your browser on the exercises and course project[3].

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/

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++ 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

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 = "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`

Looping and counting

Using loops and counting

```
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;
}
```

Output

```
**********************
*                    *
* Hi, M4997-3!       *
*                    *
**********************
```

More functionality of strings

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

Concatenation

`+` operator combines string

Defining constants

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

```
const std::string::size_type cols = greeting.size()
```

Getting the size

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

The while statement[4]

```
size_t pad = 1;
const std::string::size_type cols
        = greeting.size() + pad * 2 + 2;

while (c != cols ) {
// do formatting and printing
}
```

Condition

- `r != rows` the statement within the curly braces will be repeated while the condition is true
- `!=` is the inequality operator and once `r` is equal to `rows` the loop stops

[4] https://en.cppreference.com/w/cpp/language/while

The while statement[4]

```
size_t pad = 1;
const std::string::size_type cols
        = greeting.size() + pad * 2 + 2;

while (c != cols ) {
// do formatting and printing
}
```

Storing sizes

`size_t` is the type of any `sizeof` expression and as is guaranteed to be able to express the maximum size of any object in C++

[4] https://en.cppreference.com/w/cpp/language/while

The loop statement[5]

```
const size_t rows = pad * 2 + 3;

for(size_t r = 0; r != rows; r++){

//do formatting and printing

}
```

Condition

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

[5] https://en.cppreference.com/w/cpp/language/for

Conditionals[6]

```
if ( r == pad + 1 && c == pad + 1){
        std::cout << greeting;
    c += greeting.size();
} else
{
// do something
}
```

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

- `&&` Logical and operator

[6] https://en.cppreference.com/w/cpp/language/if

Operators[7]

Logical operators

- `&&` Logical and
- `||` Logial 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

[7] https://en.cppreference.com/w/cpp/language/operator_precedence

Built-in types[8]

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

[8] `https://en.cppreference.com/w/cpp/language/types`

Summary

Summary

After this lecture, you should know

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