Programs

CSE 232 - Dr. Josh Nahum

Reading:

Preface, Section 1.1, and Section 1.2

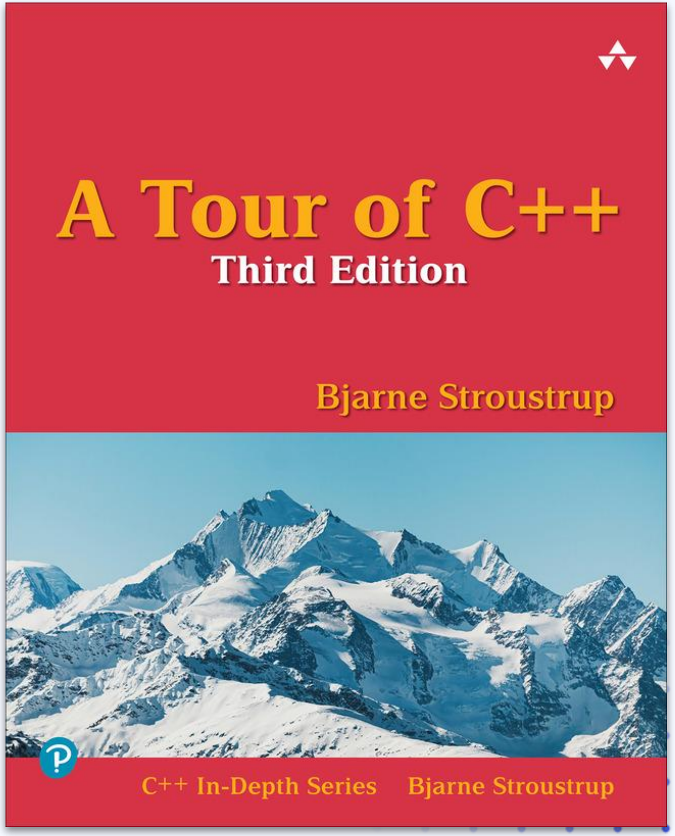

Table of contents

00
Preface

01
Programs

02
Hello, World!

03
Print Square

00

Preface

Textbook

Required Textbook

Tour of C++, 3rd Ed.
Physical Copy

"Tour"

It is a brief overview, not a tutorial, nor a exhaustive reference

Other Textbooks

The supplemental (and optional) textbooks can fill in gaps, if you prefer learning from text.

Readings

Only the required textbook will ever have assigned reading or be on assessments

Role of Lectures

Supplement to the Readings

These lectures build on the material in the assigned readings, they are **not a substitute**. Do the reading first.

Active Learning

After learning a concept, apply it. Write code that uses it, find other sources that cover it. Discuss it with peers. **Make sure you REALLY understand it.** Don't just passively "absorb" the material

Why C++?

Every programmer needs to know two classes of language

- Script-y language for everyday / simple kinds of things
  - Ex: Python, javascript
- System-y kind of language that provides speed, efficiency, power to do harder, more computational stuff
  - Ex: C++, Rust

01

Programs

C++ Versus Python

C++	**Compiler** Code is turned into a separate executable program	**Static Typing** Every variable as an explicit type that can't ever change
Python	**Interpreter** Code is run line by line by another program	**Dynamic Typing** A variable's type can change and is often only implicitly stated

import directive

Examples

```
import std;
import string;
```

Alternative, use include directive

```
#include <iostream>
#include <string>
```

Not Implemented (Yet)

Many topics in Tour are cutting edge and not supported by all compilers. We won't be using the import directive.

2 Kinds of Comments

Single Line Comment

```
code ... // C++ Style comment
```

Multiline Comment

```
code ... /* First line of comment.
Second line of comment.
Last line of comment. */
```

Comment Policy

You should comment your code to make it clear why the code was written that way. **Add comments for clarity.**

02

Hello, World!

hello.cpp

```cpp
#include <iostream>

int main()
{
    std::cout << "Hello, World!\n";
}
```

03

Print Square

print_square.cpp

```cpp
#include <iostream> // Using include, not import

double square(double x) { // Different brace style
  return x*x; // Indentation with 2 spaces
}

void print_square(double x) {
  std::cout << "the square of " << x << " is "
            << square(x) << "\n"; // Broke statement into 2
}

int main() {
  print_square(1.234);
}
```

Formatting

Curly braces {} denote blocks of code and are mandatory in many situations.

C++ code is largely free-form, whitespace is a matter of style, and usually has no effect on the functionality of the program.

```cpp
void print_square(double x) {
  std::cout << "the square of " << x << " is "
            << square(x) << "\n";
}

void print_square(double x){std::cout <<
"the square of "<<x<<" is "<<square(x)<<"\n";}

void print_square(
  double x
){std::cout <<
"the square of "<<x       <<" is "
              <<square(    x)<<"\n";}
```

Attribution

Please ask questions via Piazza

Dr. Joshua Nahum
www.nahum.us
EB 3504