A brisk introduction

Michael Nowak

Texas A&M University

Overview

Overview

The writing process

- When writing in the English language, we can use
  - a word processing program providing utilities that check our spelling and grammar

  or a text editor supporting the bare-bone necessities for composing a text document
- When writing in a programming language, we can use
  - an integrated development environment (IDE) providing elaborate capabilities, with many bells and whistles

  or a text editor supporting the bare-bone necessities for composing a source document
- In this class, we will write code using a text editor to create and modify our source files
- For the C++ language, we will save our source files using the `.cpp` extension
  - ex. `helloworld.cpp`

The writing process cont.

- When writing code, we provide the computer with a sequence of instructions that are executed to perform a computation or solve a problem
- Think:
  - step-wise instructions to put a new piece of furniture together

  or a detailed recipe for a cook to follow during the preparation of a meal
- Not:
  - "put it together"

  or "cook me a meal"
- It is important that the sequence of instructions are precisely and unambiguously specified; the computer cannot infer your intentions

Overview

Syntax and semantics

- A language is composed of a set of valid sentences
- A valid sentence is one that is syntactically correct and semantically sound (sensible)
  - `Syntax` is to structure
  - `Semantics` is to meaning
- Programming languages, like the English language, have `grammar`s that dictate which sentences are syntactically correct.

Syntax: Tokens

- The smallest piece of a programming language that has meaning is called a `token`
- In English, a token is like a word or punctuation mark
- If you change a token in C++, you change its meaning
  - This is similar to breaking up a word
  - can result in something that is no longer a word
    - often without any meaning at all
- Many tokens in C++ are words; others are symbols like punctuation

Syntax: Expressions

- In English, phrases are built from words
- In C++, the equivalent of a phrase is an `expression`
- An `expression` is a group of `token`s that yield a result when evaluated

Syntax: Expressions cont.

- In C++, some `tokens` are interpreted as `operands` in an `expression`
- Other `tokens` comprise `operators`
- The simplest form of an `expression` is composed using one or more `operands` that yield a result when evaluated
- More complicated expressions are formed by incorporating an `operator` and one or more `operands`

Syntax: Statements

- In English, putting phrases together builds sentences
  - A sentence is a grouping that stands on its own in written English
- The equivalence of a sentence in C++ is a `statement`
  - A `statement` is a complete and meaningful command that can be given to a computer
- In C++, a `semicolon` denotes the end of a `statement`
  - In English, we end sentences with a period or some other punctuation mark

Semantics

- In English, a syntactically correct sentence does not imply that it is semantically sound (sensible)
  - Our black cat is yellowish brown.
- In C++, a statement can be composed with correct syntax, but may not be semantically sound (sensible)
  - This means that the statement may not do what the programmer intended
  - It may cause the program to
    - crash
    - produce a wrong value
    - perform a behavior incorrectly

Overview

Hello, World!

```cpp
#include <iostream>

int main()
{
    // print "Hello, World!" to standard output
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

▶

Hello, World!

```
#include <iostream>

int main()
{
    // print "Hello, World!" to standard output
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Hello, World!

```cpp
#include <iostream>

int main()
{
    // print "Hello, World!" to standard output
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Hello, World!

```cpp
#include <iostream>

int main()
{
    // print "Hello, World!" to standard output
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Hello, World!

```cpp
#include <iostream>

int main()
{
    // print "Hello, World!" to standard output
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Hello, World!

```cpp
#include <iostream>

int main()
{
    // print "Hello, World!" to standard output
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Overview

Overview

Compilation

- When writing a program, you read and write human-readable `source code`
- The code that a computer is able to run is called `object code` or `machine code`.
- Your source code must be translated to a machine-readable `executable` in order to run your program
- This translation is done through the C++ `compilation` pipeline

Preprocessor

- Prepares the source file for the compiler
- The output of the preprocessor is fed into the compiler

Compiler

- Reads your source code one file at a time
- Checks to see if it is grammatically correct, if every token has meaning, and for any inconsistencies that it considers obviously wrong
- Has no common sense and is very picky about details; won't try to guess what you meant under any scenario
- The compiler outputs assembly code that is fed into the assembler

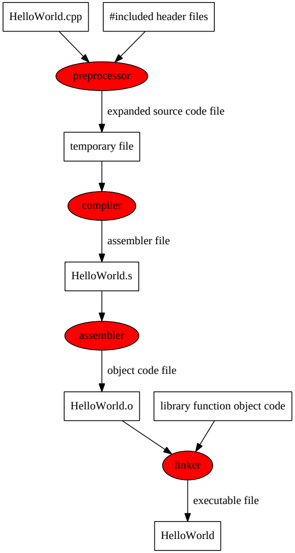
Assembler

- The output of the compiler (assembly code) is feed into the assembler
- Assembler is responsible for converting that assembly code to object code

Linker

- Source code for our programs can be written across many source files; the compiler outputs one object code file for each source file submitted to it
- These object code files must be “linked together”
- The choreography of this linking is handled by the `linker`
- The output of the linker is an executable (runnable) file

Summary of the C++ compilation processes



Overview

Compiling and executing your own programs

- In this course, you will compile your C++ programs in a console window
- If your source file `HelloWorld.cpp` is in `/path/to/dir`, you first need to change to that directory in your console window:

  ```
  cd /path/to/dir
  ```
- Thereafter, you can compile `HelloWorld.cpp` by issuing the following command:

  ```
  g++ -std=c++14 HelloWorld.cpp
  ```
- This command takes the source file `HelloWorld.cpp` – and as long as its contents contain a `main` function – compiles it into an executable named `a.out`
- You can execute your compiled program by issuing the following command:

  ```
  ./a.out
  ```

Overview

References

- Sebesta, R. W. (2016). *Concepts of programming languages* (11th ed.). Pearson Education.
- Stroustrup, B. (2014). *Programming: principles and practice using C++* (2nd ed.). Pearson Education.