CSI33 Data Structures

Department of Mathematics and Computer Science
Bronx Community College

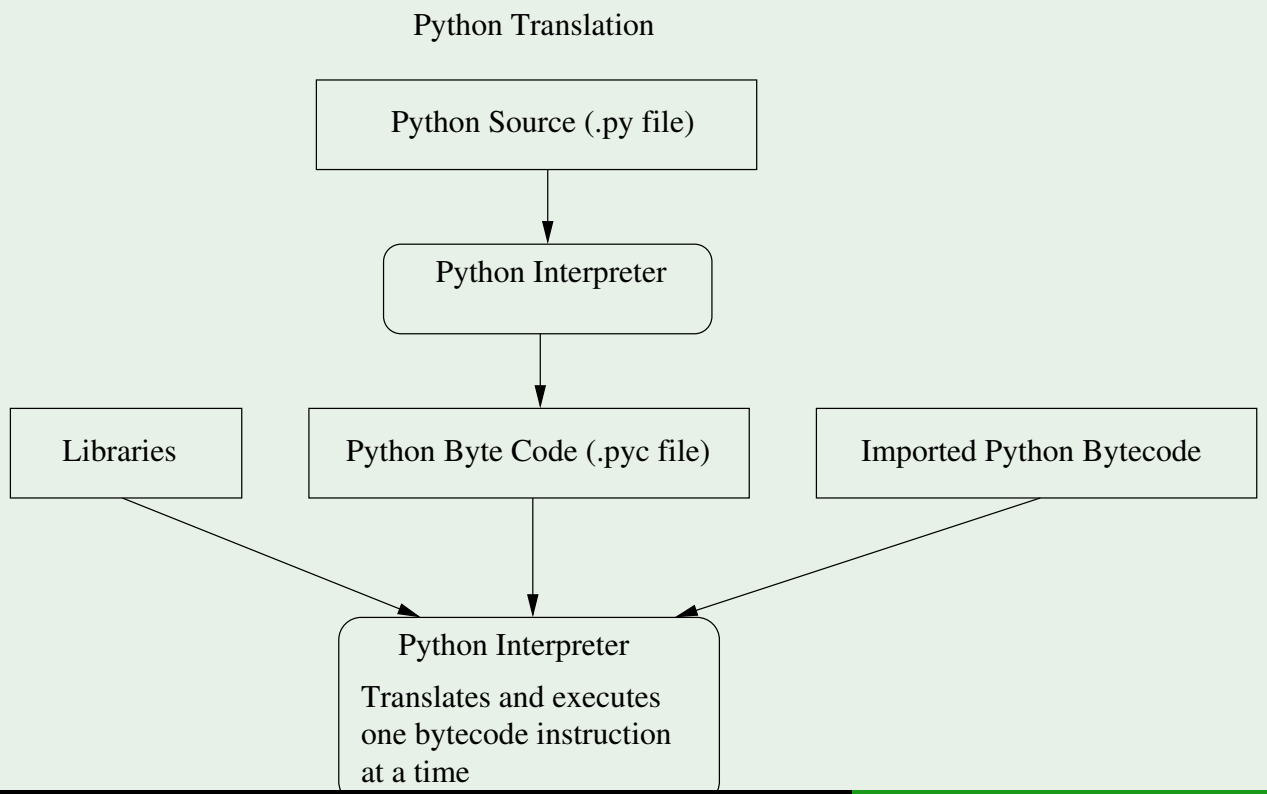
Outline

History

Read about the C++ history and background in the textbook and see some other resources as well.

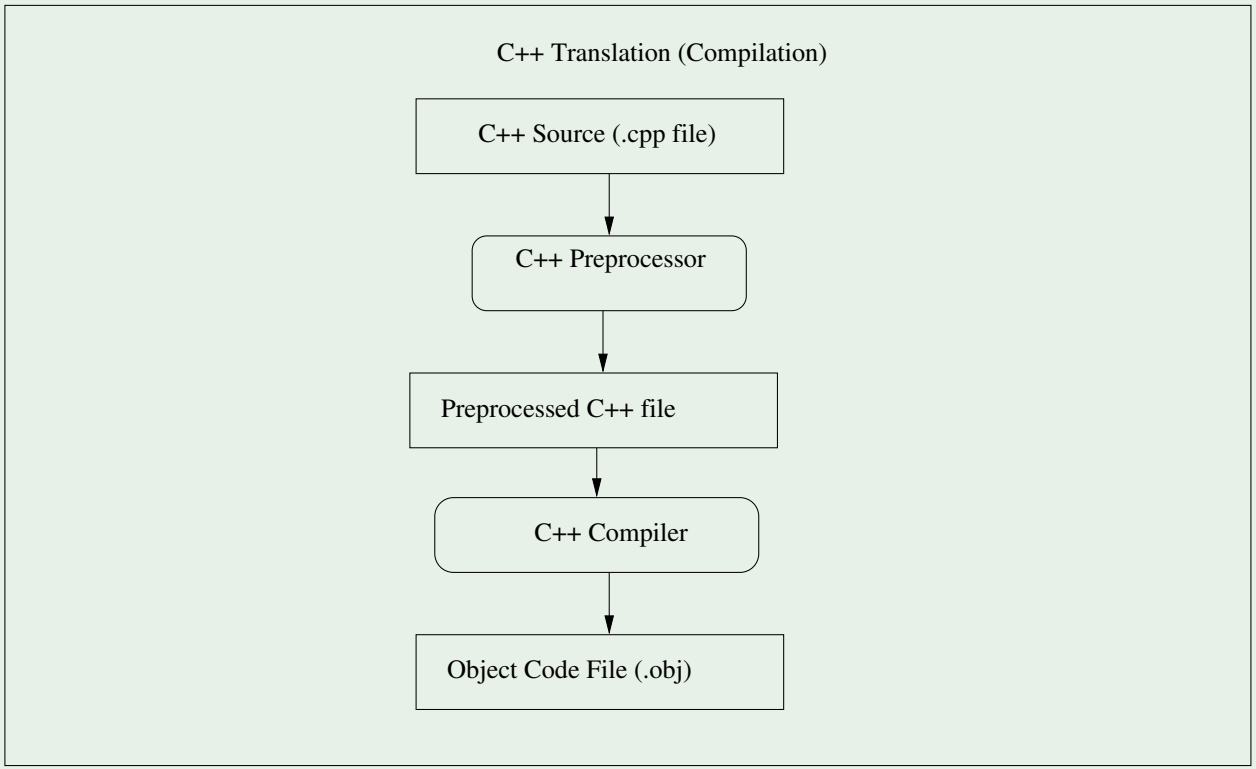
Python Translation Process

Interpretation



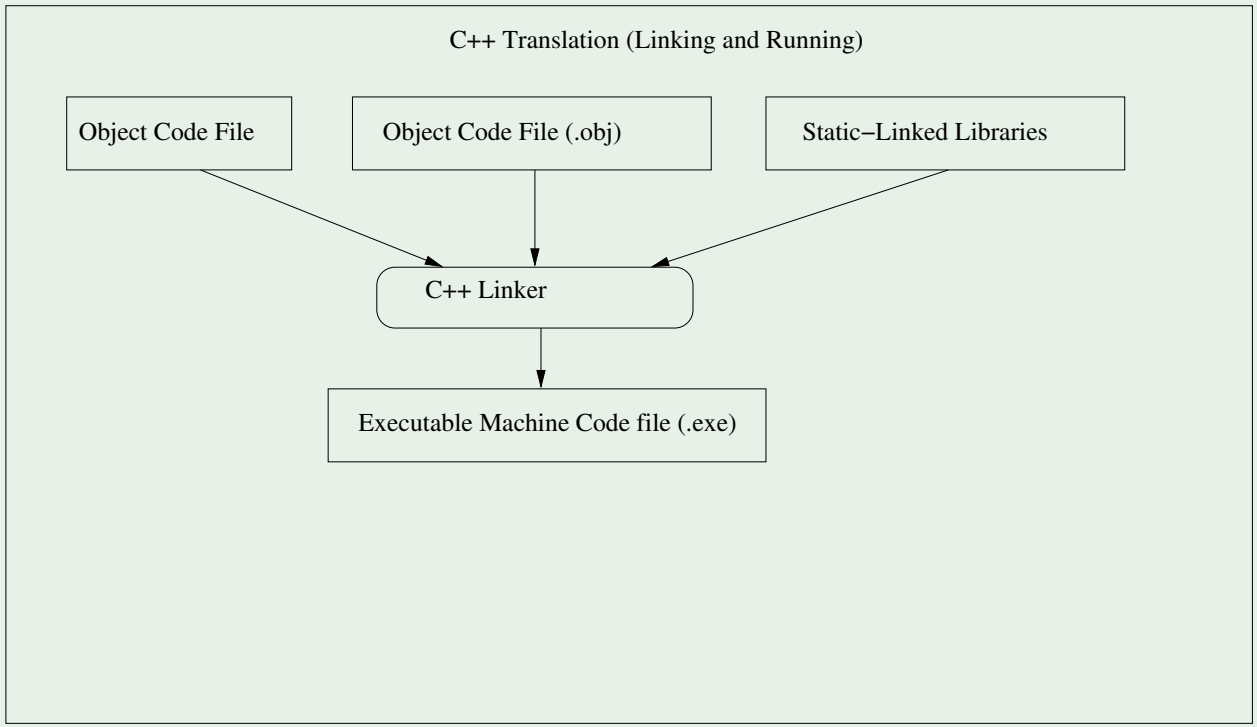
C++ Translation Process - "Building" a Program

Compilation



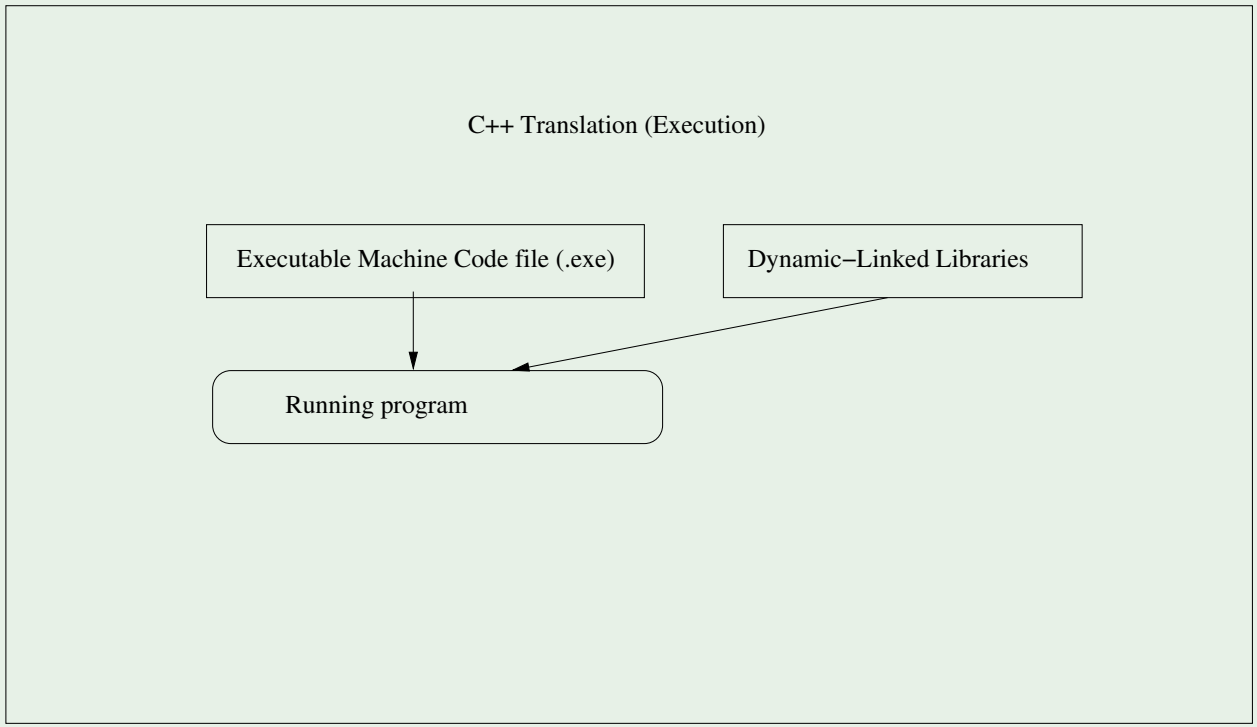
C++ Translation Process - "Building" a Program

Linking



C++ Translation Process - "Building" a Program

Execution



Comments

Single-Line Or Multiline

- A single-line comment begins with //
- A multi-line comment begins with /* and ends with */

Whitespace and Blocks of Code

Whitespace

- Whitespace consists of spaces, tabs, and newline characters.
- C++ uses whitespace as a separator between keywords, identifiers, and operation symbols.
- There is no other special meaning for whitespace.
- Indentation is done for readability only–it is not used for the body of a function, a loop, or an if/else clause.
- Each C++ statement must be terminated with a semicolon.

Blocks Of Code

Function Bodies

In C++, instead of indentation, function bodies and **if** clauses are specified by enclosing them in curly brace symbols { and }. The code inside a pair of braces is called a Code Block.

```
int main()
{
   int i, j;
   double x;

   x = (i+j)/5;     return x;

}
```

Identifiers And Keywords

Keywords

Words which have special meaning for C++ cannot be used as identifiers for variables, function names, or class names.

See page 263, Figure 8.3 for a list of reserved keywords.

Identifiers And Keywords

Identifiers

An identifier can be any sequence of letters (uppercase or lowercase), decimal digits (0-9) or the underscore character "_", as long as the first character of the identifier is not a digit.

Data Types

Variables Hold Values (Not References)

In Python, every variable name uses a reference (a four-byte address), which can point to any type of data (int, str, or any object).

The type of a variable can be changed by an assignment statement in Python.

Python is said to use dynamic typing.

Data Types

In C++, Declarations Are Required, Static Typing

In C++, every variable uses its actual value, so the compiler needs to know the type of data a variable will take, since different types occupy different amounts of memory. The compiler reserves as many bytes of memory as required for each declared variable, based on the data type specified in the declaration.

- int = 4 bytes: the declaration `int a;` allocates 4 bytes for a
- char = 1 byte: the declaration `char c;` allocates 1 byte for c
- double = 8 bytes: the declaration `double d;` allocates 8 bytes for d
- bool = 1 byte: the declaration `bool b;` allocates 1 byte for b
  See page 265, Figure 8.4 for more details.

Include Statements

Include Statements in C++ Are Like Import Statements in Python

In C++, an `include` statement is used to copy the contents from another file into the file being translated.

It is similar to Python's import statement.

This is useful, for example, when the same variable is used in different C++ program files since its declaration can be written once and then included whenever that variable is used.

```
# include<iostream>
```

Namespaces

Namespaces Are Like Modules in Python

In Python, if a variable in one module has the same name as one in another module, the two variables are kept separate by Python, and can have different values.

In C++, declaring and naming a **namespace** (surrounded with braces) will cause every variable declared in the **namespace** to exist separately from any variable with the same name declared outside the **namespace**.

using namespace `std;`

It is similar to Python's from math import *

Input/Output

cin and cout

`cin` is an object in the `istream` class in the namespace `std`. Using the >> operator allows user input, from the keyboard, to become a variable's value:

```
cin >> a;
```

`cout` is an object in the `ostream` class in the namespace `std`. Using the << operator allows a variable's values to become output on the display console:

```
cout << a;
```

Input/Output

''Hello, world''

```
// hello.cpp
#include <iostream>

using namespace std;

int main()
{
   cout << "hello world\n";
   return 0;
}
```

Input/Output

''Hello, world'' without namespace used

```
// hello.cpp
#include <iostream>

int main()
{
   std::cout << "hello world\n";
   return 0;
}
```

Build

The make Utility

- A script (short program) which tells the computer to perform the steps of building an executable file:
- Preprocessing and compiling a `.cpp` file into an object(`.o`) file.
- Linking the object file with other machine code files to produce an executable (`.exe`) file.
- Only perform operations if the output file is older than the input file.

Build

Eclipse Console View

```
**** Build of configuration Debug for project HelloWorld ****
**** Internal Builder is used for build ****

g++ -O0 -g3 -Wall -c -fmessage-length=0 -ohello.o ..\hello.cpp
g++ -oHelloWorld.exe hello.o
Build complete for project HelloWorld
Time consumed:  2266 ms.
```

USING FORMULA

Take a look at the program CtoF.cpp that performs conversion of temperature reading in Celsius degrees to Fahrenheit degrees.

Take a look as the program swap.cpp that swaps values of variables.

Local Variables

- Must be declared giving their datatypes, so values are guaranteed to fit in the memory locations reserved by the compiler.
- May optionally be initialized when declared.
- Local variables declared in a function (not formal parameters) are called automatic variables. They are given memory locations without putting values in those locations. So automatic variables must be initialized before use (to get predictable value).
- The memory for a variable holds an actual value, not a reference (as in Python).
- Exception: References (pointer variables) are specially denoted by *, the dereference operation.

Local Variables

- Must be declared giving their datatypes, so values are guaranteed to fit in the memory locations reserved by the compiler.
- May optionally be initialized when declared.
- Local variables declared in a function (not formal parameters) are called automatic variables. They are given memory locations without putting values in those locations. So automatic variables must be initialized before use (to get predictable value).
- The memory for a variable holds an actual value, not a reference (as in Python).
- Exception: References (pointer variables) are specially denoted by *, the dereference operation.

Local Variables

- Must be declared giving their datatypes, so values are guaranteed to fit in the memory locations reserved by the compiler.
- May optionally be initialized when declared.
- Local variables declared in a function (not formal parameters) are called automatic variables. They are given memory locations without putting values in those locations. So automatic variables must be initialized before use (to get predictable value).
- The memory for a variable holds an actual value, not a reference (as in Python).
- Exception: References (pointer variables) are specially denoted by *, the dereference operation.

Local Variables

- Must be declared giving their datatypes, so values are guaranteed to fit in the memory locations reserved by the compiler.
- May optionally be initialized when declared.
- Local variables declared in a function (not formal parameters) are called automatic variables. They are given memory locations without putting values in those locations. So automatic variables must be initialized before use (to get predictable value).
- The memory for a variable holds an actual value, not a reference (as in Python).
- Exception: References (pointer variables) are specially denoted by *, the dereference operation.

Local Variables

- Must be declared giving their datatypes, so values are guaranteed to fit in the memory locations reserved by the compiler.
- May optionally be initialized when declared.
- Local variables declared in a function (not formal parameters) are called automatic variables. They are given memory locations without putting values in those locations. So automatic variables must be initialized before use (to get predictable value).
- The memory for a variable holds an actual value, not a reference (as in Python).
- Exception: References (pointer variables) are specially denoted by *, the dereference operation.

Expressions

Expressions

- Just as in any programming language, expressions have values.
- The simplest expressions are constants or variable names.
- Just as in any programming language, expressions are constructed by connecting smaller sub-expressions with operator symbols. These values are calculated by performing the operations, with the highest precedence operation first.
- Expressions can also be formed by making function calls which return values.

see program `uninit.cpp`

Expressions

Operators Are Similar To Those In Python

Exceptions:

- && for and.
- || for or.
- increment operators: `x++` or `++x` for `x += 1`
- decrement operators: `x--` or `--x` for `x -= 1`

see program `increment.cpp`

Expressions

Operator Precedence

The order of operations in C++ follows the same standard rules as Python:

(a+b*c) means multiply b times c, then add a.

In-class work

1. Write a program in C++ that converts Fahrenheit degrees to Celsius degrees.
2. Write a program that takes in three floating point values and outputs their sum and product.