C and C++

I

Spring 2014

Carola Wenk

Different Languages

Python

```
sum = 0
i = 1
while (i <= n):
  sum += i
  i += 2
```

Python Interpreter

Java/C++

```
int sum = 0;
for (int i = 1; i <= n; i +=2) {
  sum += i;
}
```

Java/C++ Compiler

C

```
int sum = 0;
int i;
for (i = 1; i <= n; i +=2) {
  sum += i
}
```

C Compiler

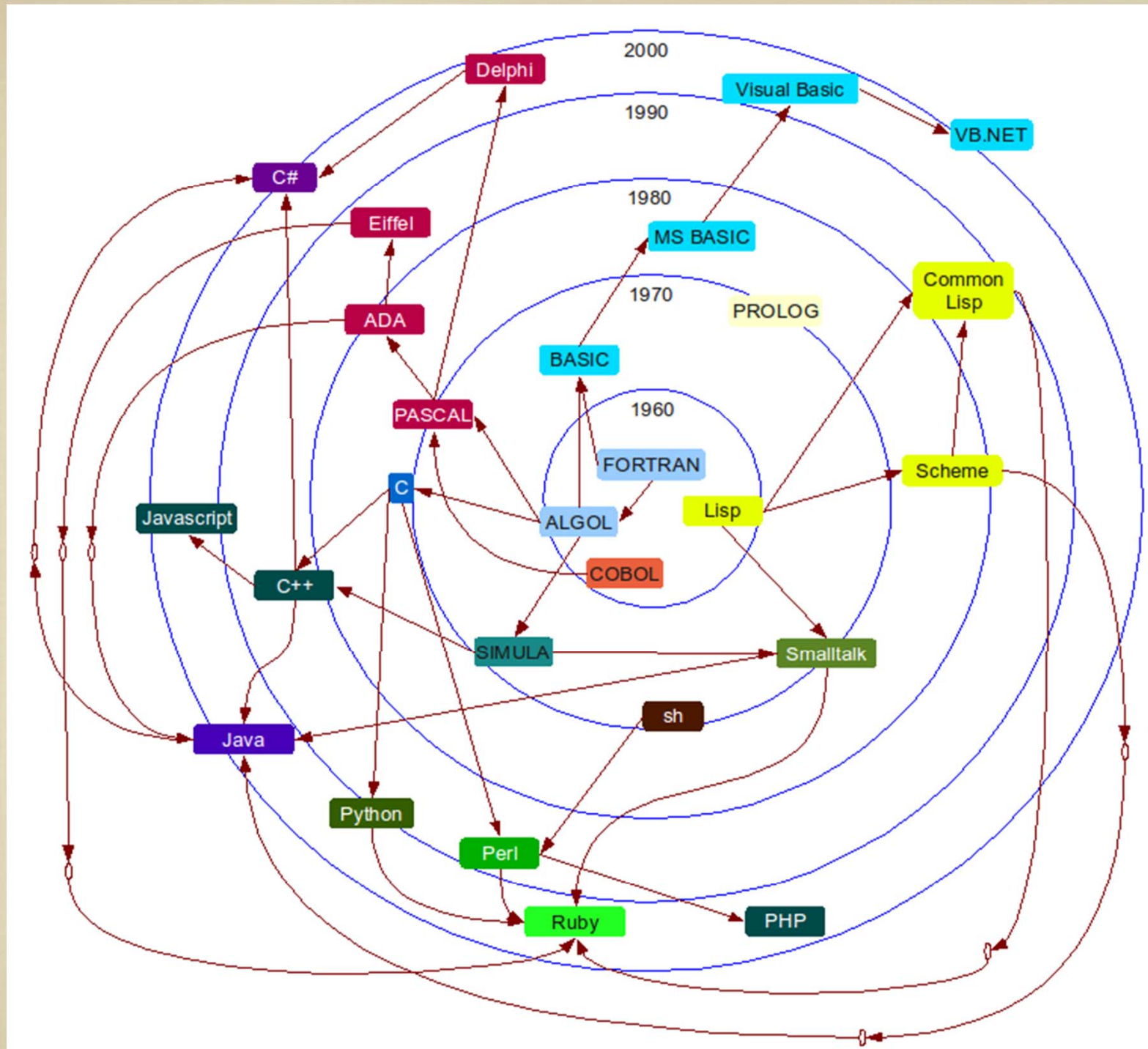
```
lw $t0, 1
lw $t1,0
lw $t2, n
loop:
beq $t0,$t2,done
add $t0, $t1, $t1
add $t0, 2
jmp loop
done:
```

Scheme Interpreter

Scheme

```
(define (sum n)
(if (= n 0) 0
                (+ n (sum n-1))))
```

Languages are translated to machine code by either by a compiler or interpreter.


2000
1990
1980
1970
1960
Delphi
Visual Basic
VB.NET
C#
Eiffel
MS BASIC
Common Lisp
PROLOG
ADA
BASIC
PASCAL
FORTRAN
Scheme
C
Javascript
ALGOL
Lisp
COBOL
C++
SIMULA
Smalltalk
sh
Java
Python
Perl
Ruby
PHP

Categories and Uses

- Imperative
  - Python: Interpreted, Easy to prototype ideas
  - Java : Interpreted/Compiled, Platform-independent
  - C/C++: Compiled, General purpose
  - PHP: Interpreted/Compiled, Web scripting

- Functional
  - LISP/Scheme: Interpreted, no differentiation between data/instructions

Languages are translated to machine code by either by a compiler or interpreter.

C and C++

- The C language was originally developed in the 1970s to assist in the implementation of the UNIX operating system. It was designed to be one step above machine language.

- C++ is a superset of C introduced in the early 1980s to add objected-oriented features to C.

Hello World in C:

```c
#include <stdio.h>

int main() {
    printf("Hello World!!");
    return 0;
}
```

Hello World in C++:

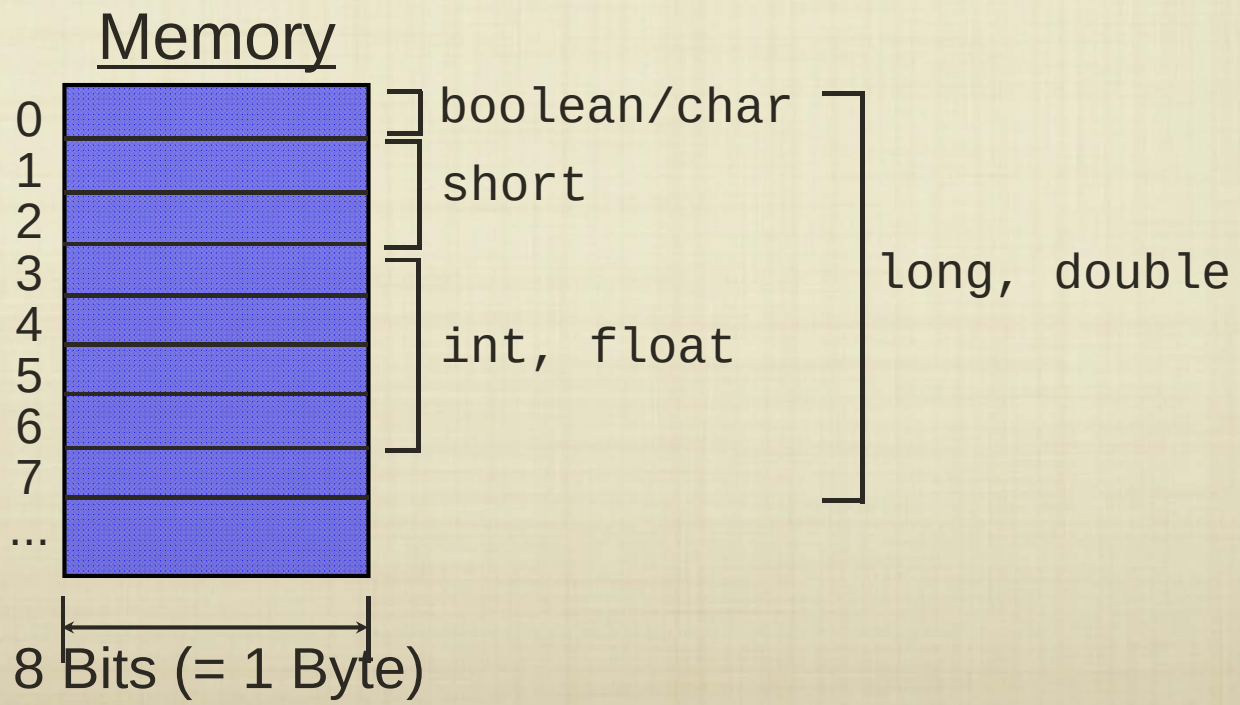
```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl;
    return 0;
}
```

Variables and Types

- Although Python doesn't care about types, they exist: numbers, strings, and lists.

- C and C++ have the same primitive types as Java: `int/short/long, float/double, boolean, char`.

- A variable name is simply a placeholder for a memory address.



Conditionals

Python

```
if <condition>:
    <block of statements>
elif <condition>:
    <block of statements>
else:
    <block of statements>
```

Java/C/C++

```
if (<condition>) {
<block of statements>
}
else if (<condition>){
<block of statements>
}
else {
<block of statements>
}
```

For conditional statements, the only real difference in syntax between Python and Java/C/C++ has to do with scope declaration.

Java/C/C++ use braces to delimit blocks of statements, instead of indentation.

Also, in Java/C/C++ the condition has to be enclosed in parentheses.

Looping

Python

```
for i in <list>:
    <block of statements>


while (<condition>):
    <block of statements>
```

Java/C/C++

```
for (<init>; <condition>; <increment>) {
<block of statements>
}

while (<condition>) {
<block of statments>
}

do {
<block of statements>
} while (<condition>);
```

Again, looping constructs are fairly similar, except for how scope is defined.

Java/C/C++ also have a “do-while” construct that can be convenient at times.

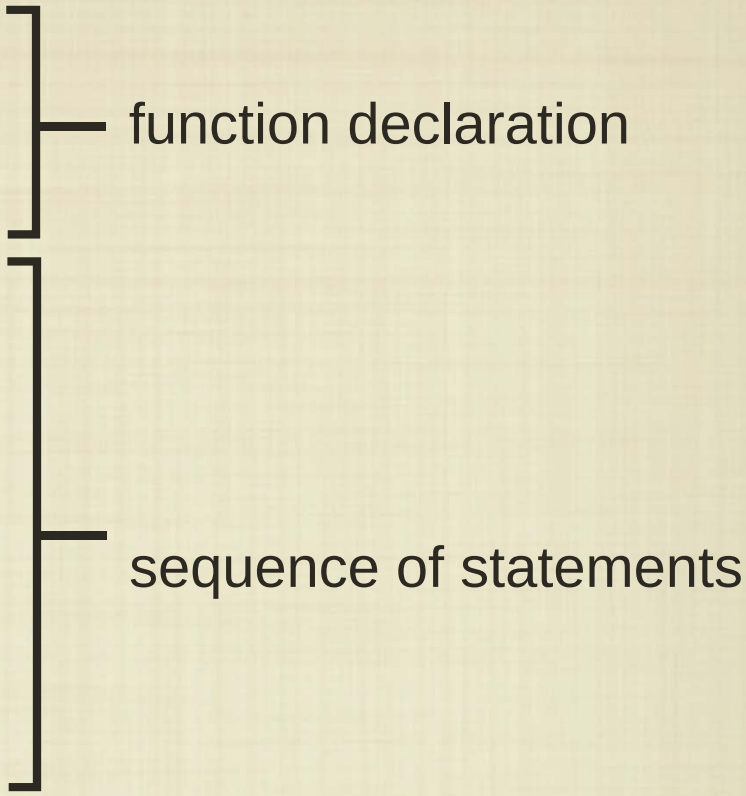
C++

```
#include <iostream>

using namespace std;

int main() {
  cout << “Hello World!” << endl;
  return 0;
}
```

One key difference between Java and C/C++ is that **not** everything is (necessarily) object. So C/C++ program flow is Python-like.

Python Program Structure

```
def f(a, b):
  print “this is function f”
  return a+b;

x = 1; y = 2; evens = 0; odds = []
print f(1, 2)
print f('z', f('a', 'b'))for i in
range(1,10):
    if (i % 2 == 0):
      evens += i
    else:
      odds.append(i)
print evens; print sum(odds)
```

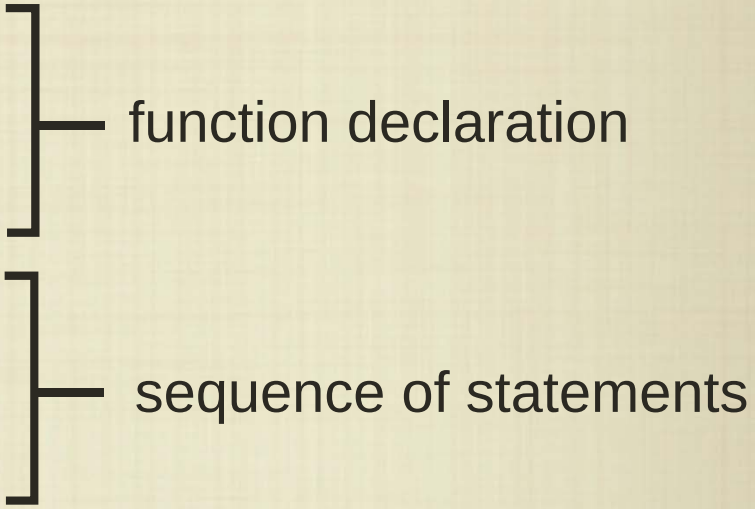


A Python script is a sequence of function declarations followed by a sequence of statements.

A function is just a way to reuse useful blocks of statements.

C++ Program Structure

```
#include <iostream>

using namespace std;

void foo(int x) {
  cout << “x is “ << x << endl;
}

int main() {
  cout << “Hello World!” << endl;
  return 0;
}
```

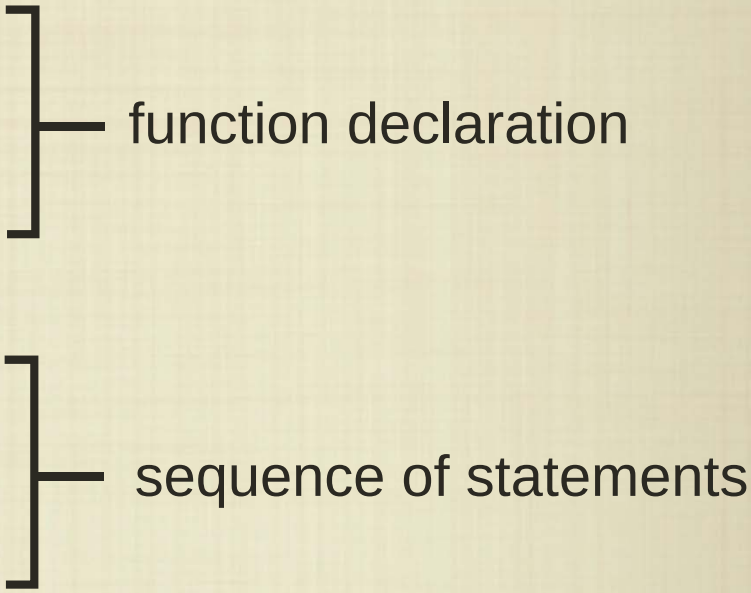


Syntax in C/C++ is very similar to Java, for historical reasons.

However, **not** everything is an object, and programs are initiated from a `main` function.

C Program Structure

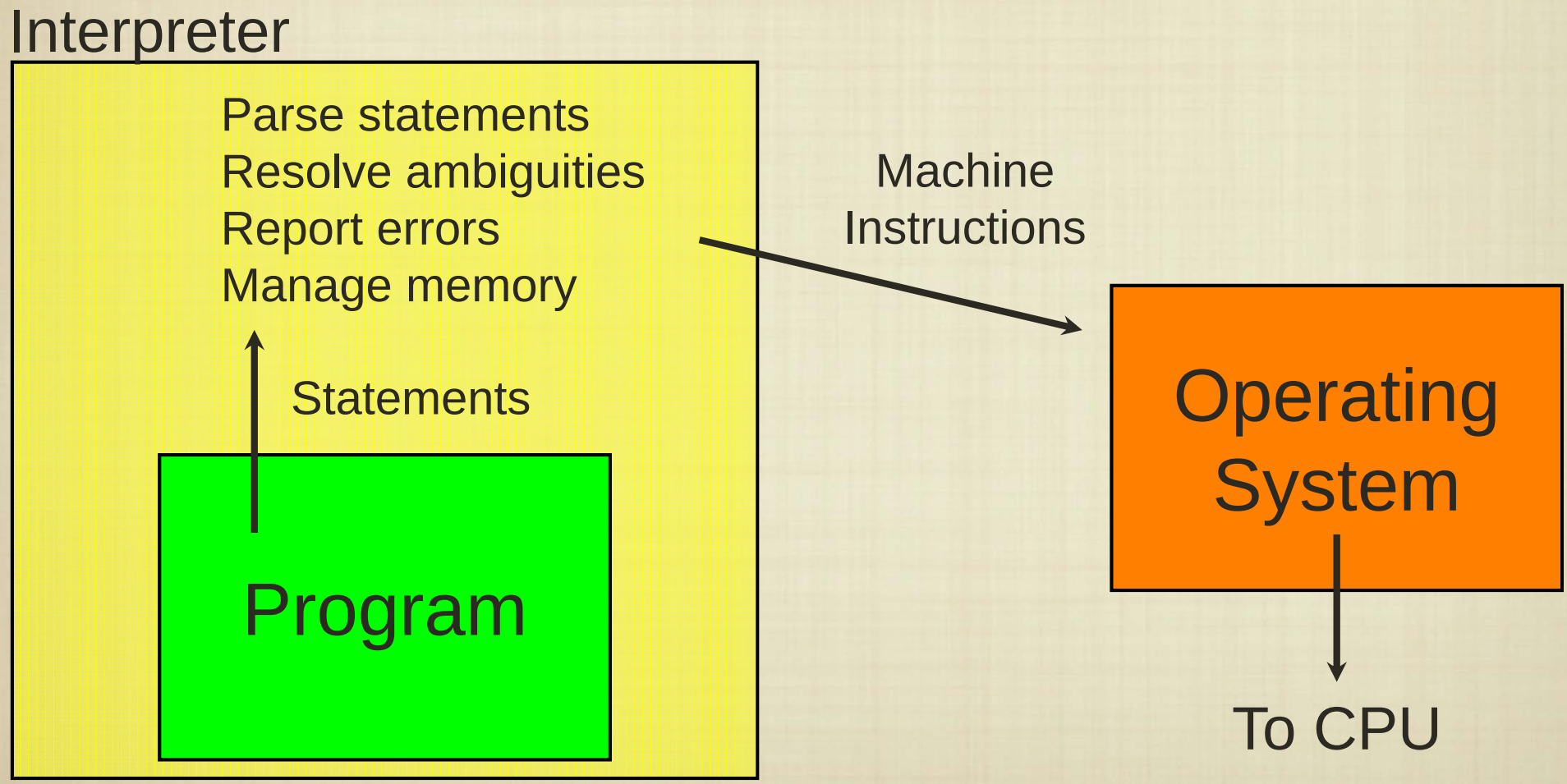
```
#include <stdio.h>


void foo(int x) {
  printf(“x is %d\n”, x);
}


int main() {
  printf(“Hello World!\n”);
  return 0;
}
```



C is older than C++, and is somewhat more low-level, with different input/output syntax, and no facility to define classes.

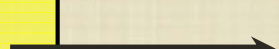
Python

Interpreted languages operate in an environment that provides some language features “under the hood”.



Java Runtime System

```
import --;

class HelloWorld {
public void f(int x1, char x2, ...) {
...
}

public long g(boolean y1, float y2, ...) {
...
}

private int h(double z1, int z2, ...) {
...
}

public static void main() {
       System.out.println(“hello world!”)
       System.out.println(“goodbye world!”)
}
}
```

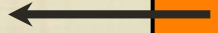


Java Compiler

Java “Byte” Code

Pretty Good Safety Net

Java Virtual Machine

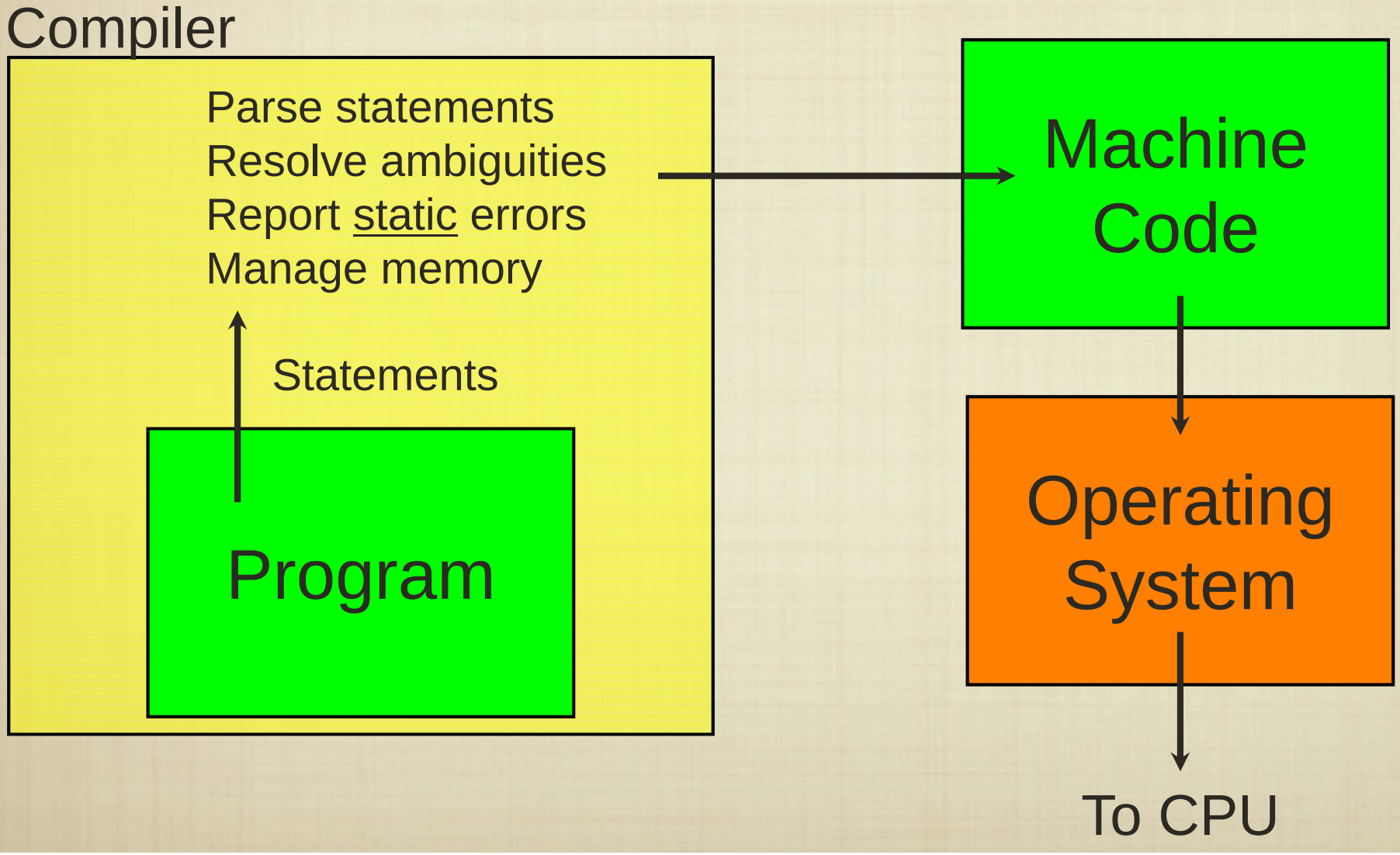


To CPU

Operating System

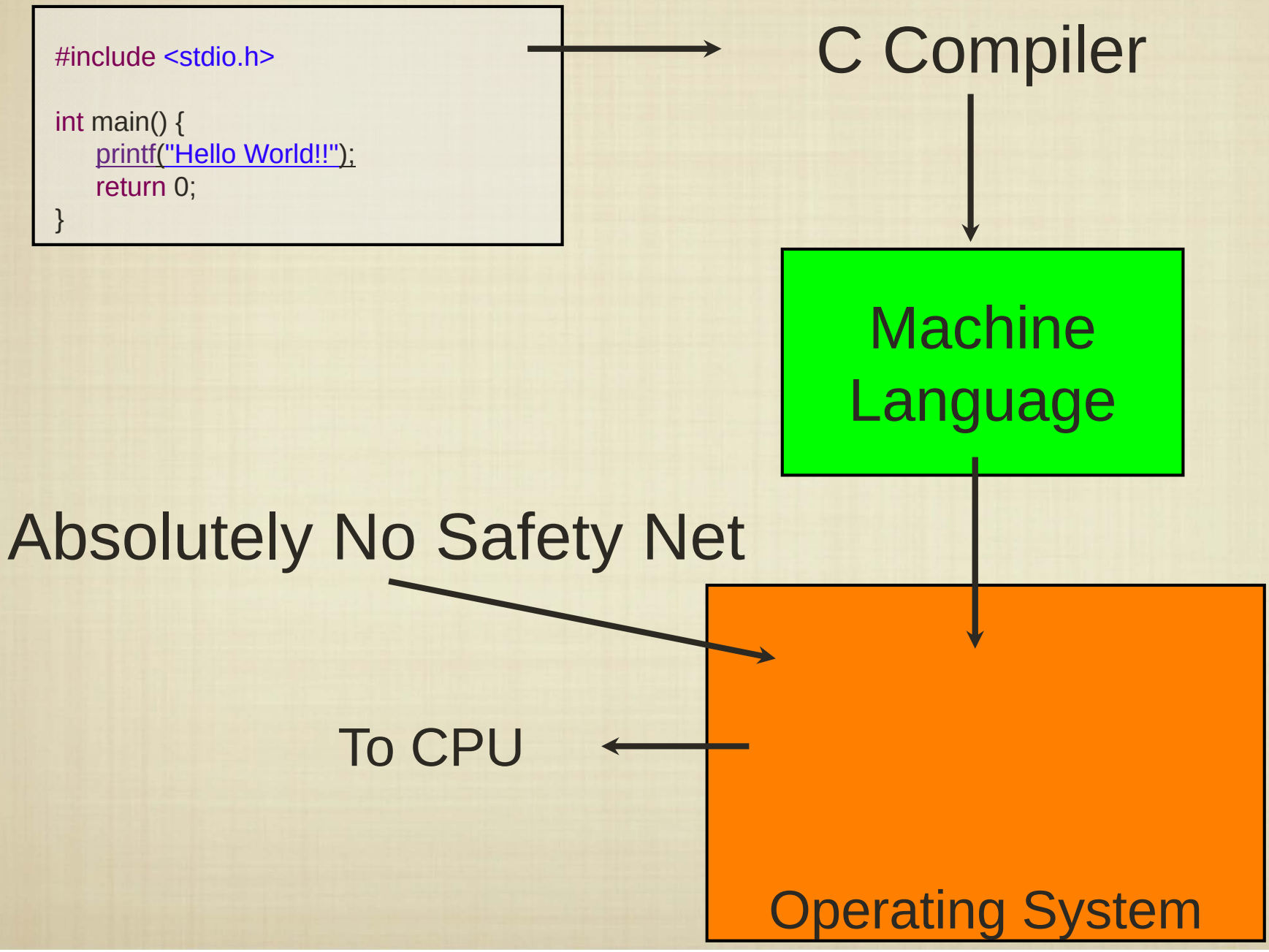
C/C++

Compiled languages operate in a self-contained environment, and generally do not have a “safety net.”



C

```
#include <stdio.h>

int main() {
    printf("Hello World!!");
    return 0;
}
```



C++

```
#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl;
    return 0;
}
```