C++

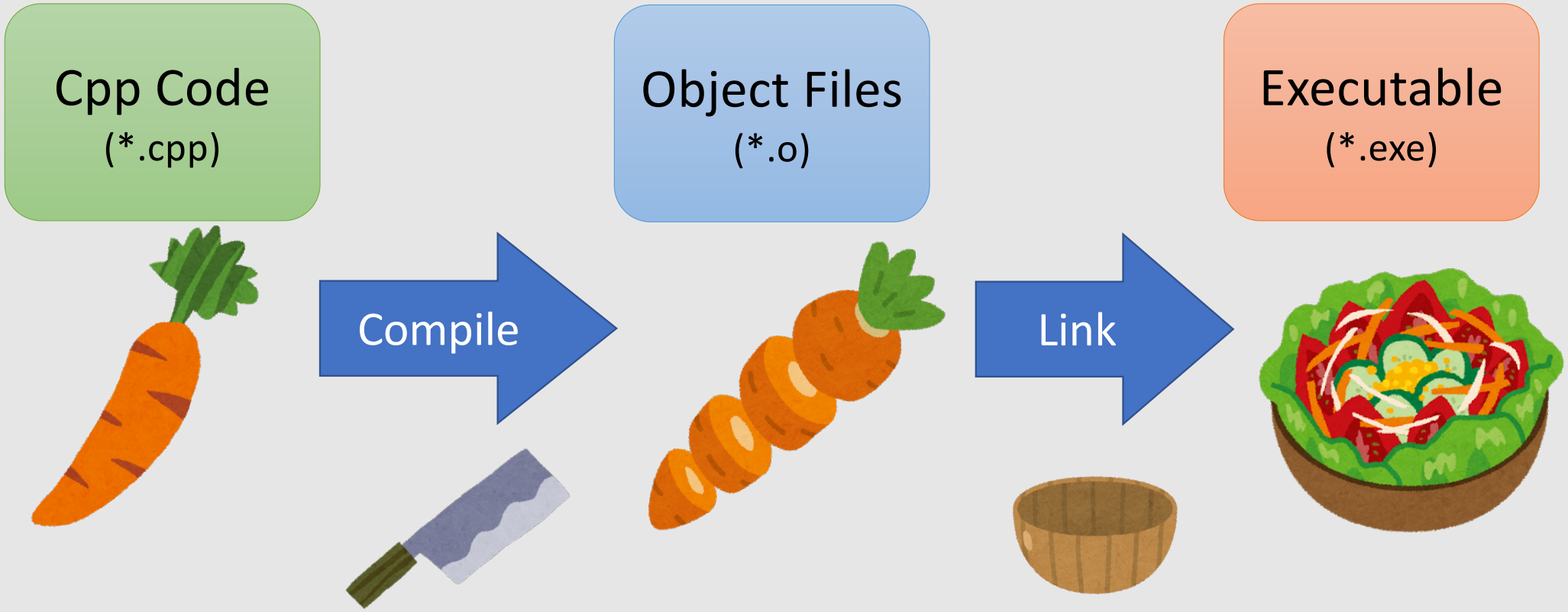
Why C++?

- C++ is the standard language for computer graphics industry
  - ☺ Very fast
  - ☺ Close to hardware (many low-level APIs)
  - ☹ Too much freedom, too many pitfalls, too powerful

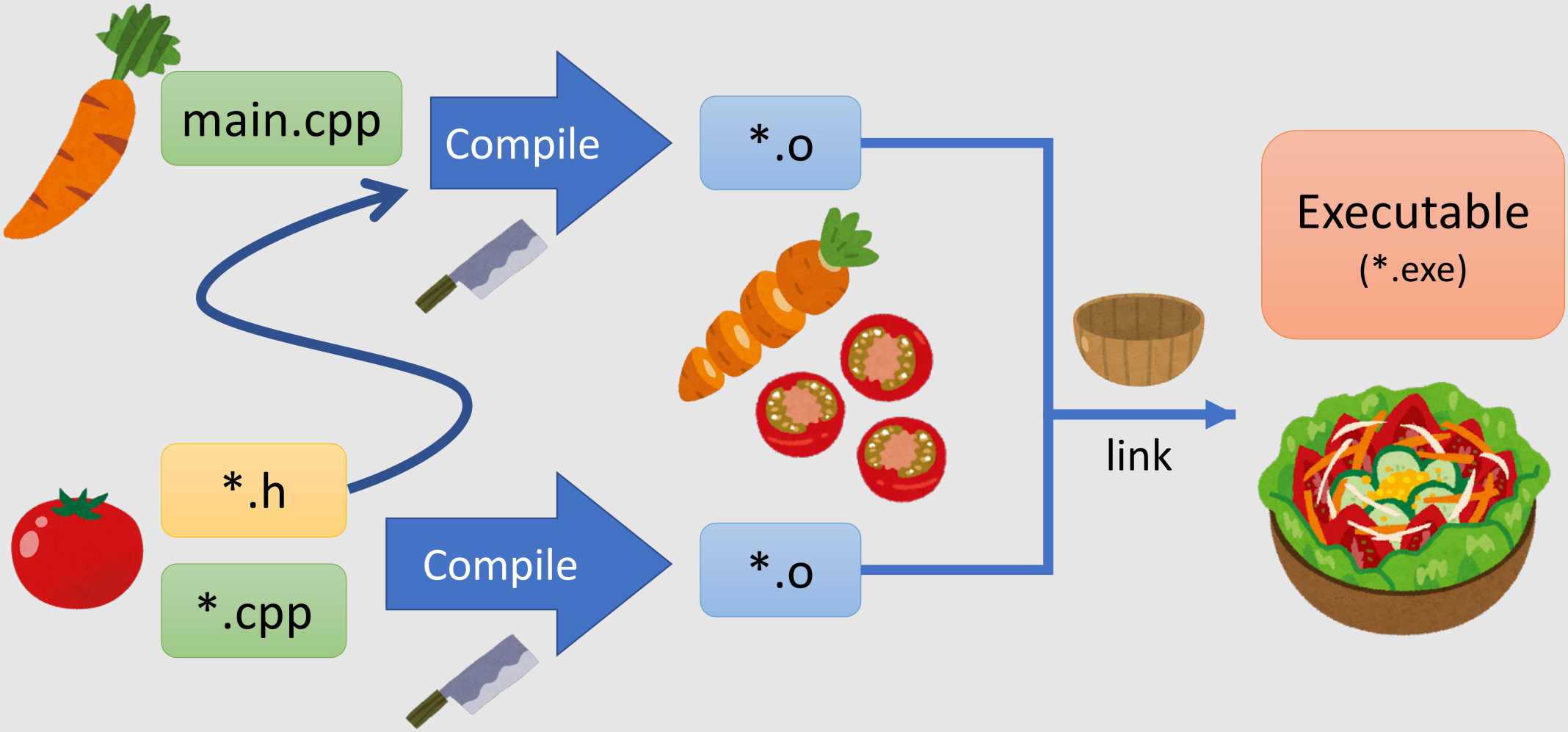
"There are only two kinds of languages: the ones people complain about and the ones nobody uses."
— Bjarne Stroustrup, Father of C++

(wikipedia)

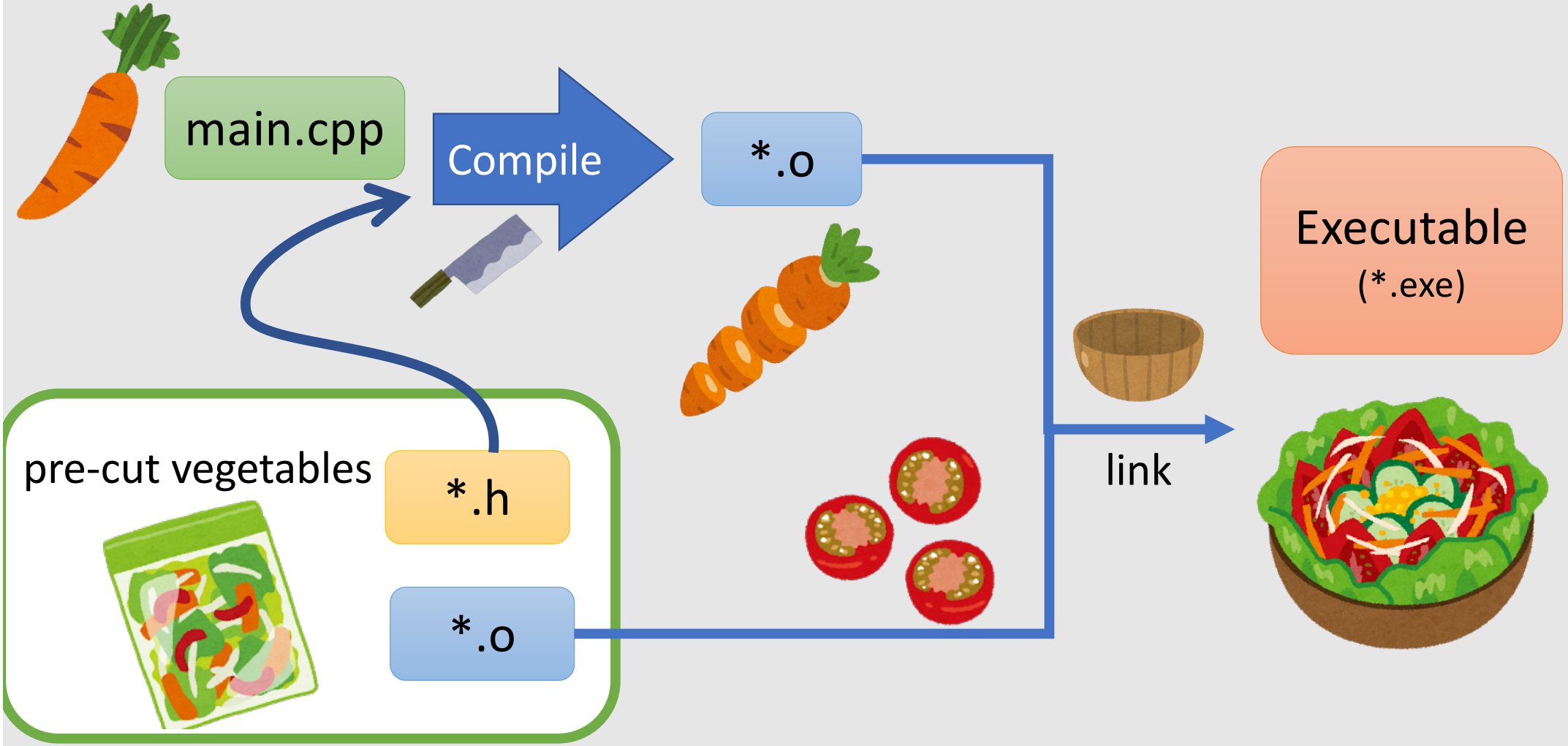
C++ is a Compiled Language



Compilation of Multiple Files



Library: Pre-Compiled Source Codes



CMake: Platform Independent Cooking

Let’s find out the tools in our kitchen!

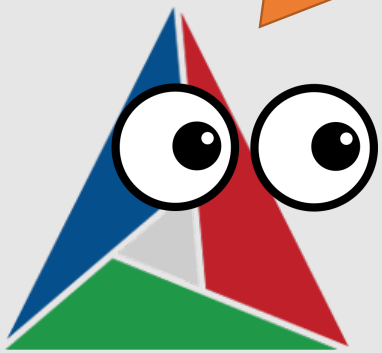
Windows kitchen

Ubuntu kitchen

MacOS kitchen

platform independent

CMakeLists.txt



platform dependent

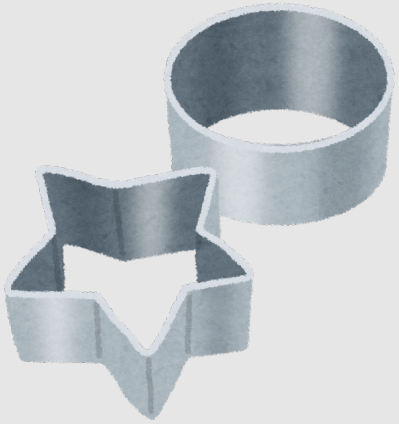
MakeFile

Project Files

(*.xcode,*.vs,*.proj)

Executable

Template (Compile-time Generics)



- Programming with general types
- Template function/class automatically generate codes for different types upon compile

```
template <typename T>
void Hoge(T v0){
  // do something
}

Hoge<float>(3.f)
Hoge<int>(1)
```

Compile

generated invisible code

```
void Hoge(float v0){
// do something
}

void Hoge(int v0){
// do something
}

Hoge<float>(3.f);
Hoge<int>(1);
```

*.o

Example of Template Function

without template

```
void Print(int value){
  std::cout << value << std::endl;
}

int main(){
  Print(5);
  Print("hello");
  Print(3.f);
}
```

with template

```
template <typename T>
void Print(T value){
  std::cout << value << std::endl;
}

int main(){
  Print(5);
  Print("hello");
  Print(3.f);
}
```

This code doesn't compile..

This code compiles !

Non-type Template Function

- Values inside a function is determined at the compiling time

```
template <int N>
class CArray{
public:
  int m_array[N];
  int size() const { return N; }
};

CArray<5> array;
```

Non-type template parameter (compile time argument)

This code runs fast !

Standard Template Library (STL)

- Default utility template function/classes in C++

Array of value that can change size

```
#include <vector>
std::vector<int> aInt;
```

Sorted set of values

```
#include <set>
std::set<int> aSet;
```

Function to sort array

```
#include <algorithm>
std::vector<int> aInt = {3,2,1};
std::sort(aInt.begin(),aInt.end());
```

Looping Through a std::set<int> Container

```
// for ancient compiler where the declaration inside "for" is not allowed
std::set<int>::iterator itr;
for(itr=stack.begin();itr!=stack.end();++itr){}

// old fashion, too complicated :(
for(std::set<unsigned int>::iterator itr=stack.begin();itr!=stack.end();++itr){}


// modern, the reference suggests ”ic” can be changed, is it safe to change ic?
for(auto& ic: stack){}

// modern, const reference is not bad, but “ic” is just an integer, it’s too much
for(const auto& ic: stack){}

// modern, very good!
for(auto ic : stack){}
```

Bad Practice in C++

- Global variables
- GOTO statement
- STL library without “std::” namespace
- Reference with long scope
- Raw pointer
- New/Delete

Integrated Development Environment (IDE)

- Code Editor with Lint, suggestion, jumps
- Static program analysis
- Debugger
- …

CLion

Visual Studio Code

Eigen Matrix Library (https://eigen.tuxfamily.org)

- C++ template matrix library
- Highly optimize-able for may environment

```
#include <Eigen/Dense>

int main()
{
  Eigen::MatrixXd m(2,2);
  m(0,0) = 3;
  Eigen::VectorXd v(2);
  v(0) = 4;
  Eigen::VectorXd mv = m*v;
}
```