C++ Under the Hood

Internal Class Mechanisms

CHRIS RYAN

2024
September 15 - 20

C++ Under the Hood

CppCon 2024 – Chris Ryan

C++ Under the Hood: (Internal Class Mechanisms)

I find it much easier to use something correctly,
if I can understand what it is doing inside and why.

We are going to talk about internal C++ class mechanisms:

- C++ Onion, inheritance and polymorphic mechanisms,
- Member Data Pointers,
- Member Function Pointers,
- * Stack Frame / Base Pointer mechanics
- * Construction/Destruction order,
- * Running code before and after main(),

C++ Under the Hood: (Internal Class Mechanisms)

About Me: Chris Ryan, Bellevue, Washington, USA
Linkedin.com/in/chrisr98008
(No blind invites! Tell me how we know each other)

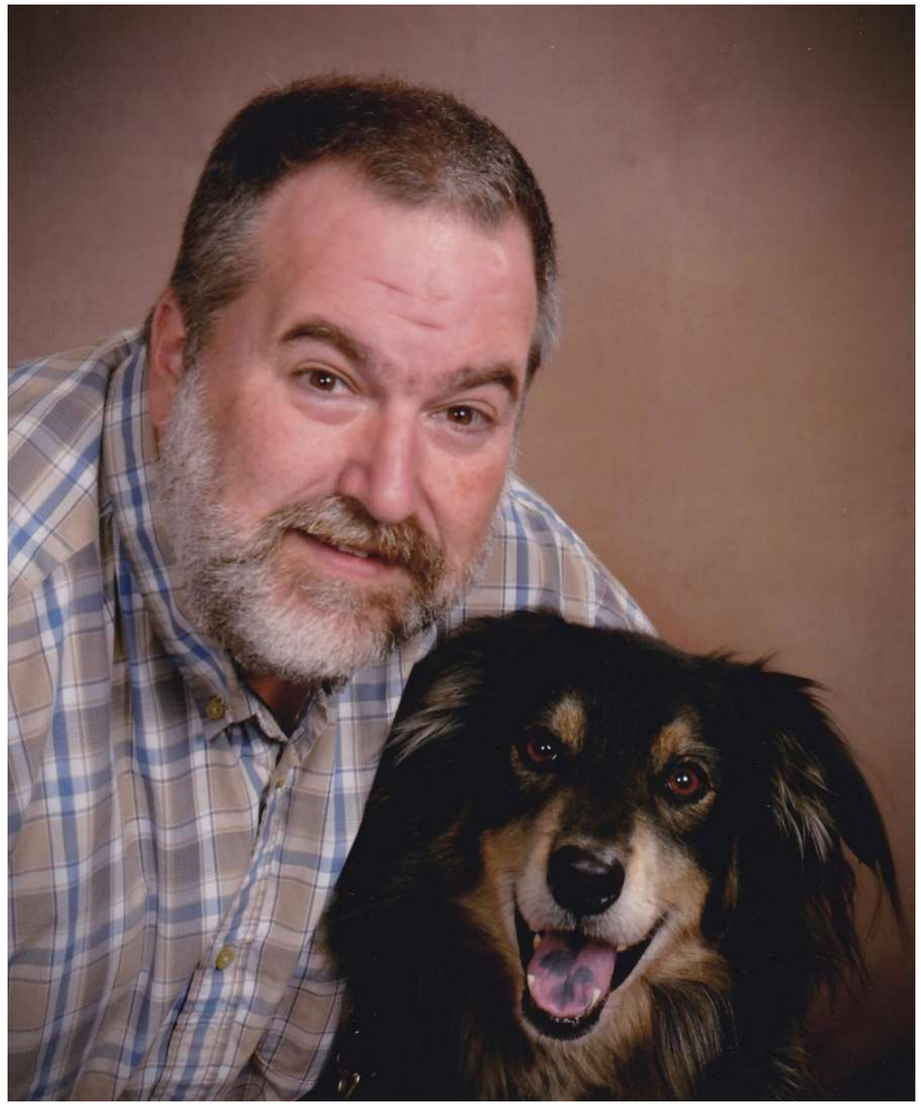
- Classically trained in Hardware and Software Engineering,
- Well experienced in Modern C++ (& Classic 'C' ***),
- ISO C++ Standards Committee member/WG21, EWG.

- Extremely Large Scale/Complex Problem Spaces,
- (on hiatus from***) Embedded/Firmware (large & small),

- Prefer Windows, because that is what I know,
- Win32/64 SDK/MFC, ***Embedded/Firmware, Bare Metal, Linux

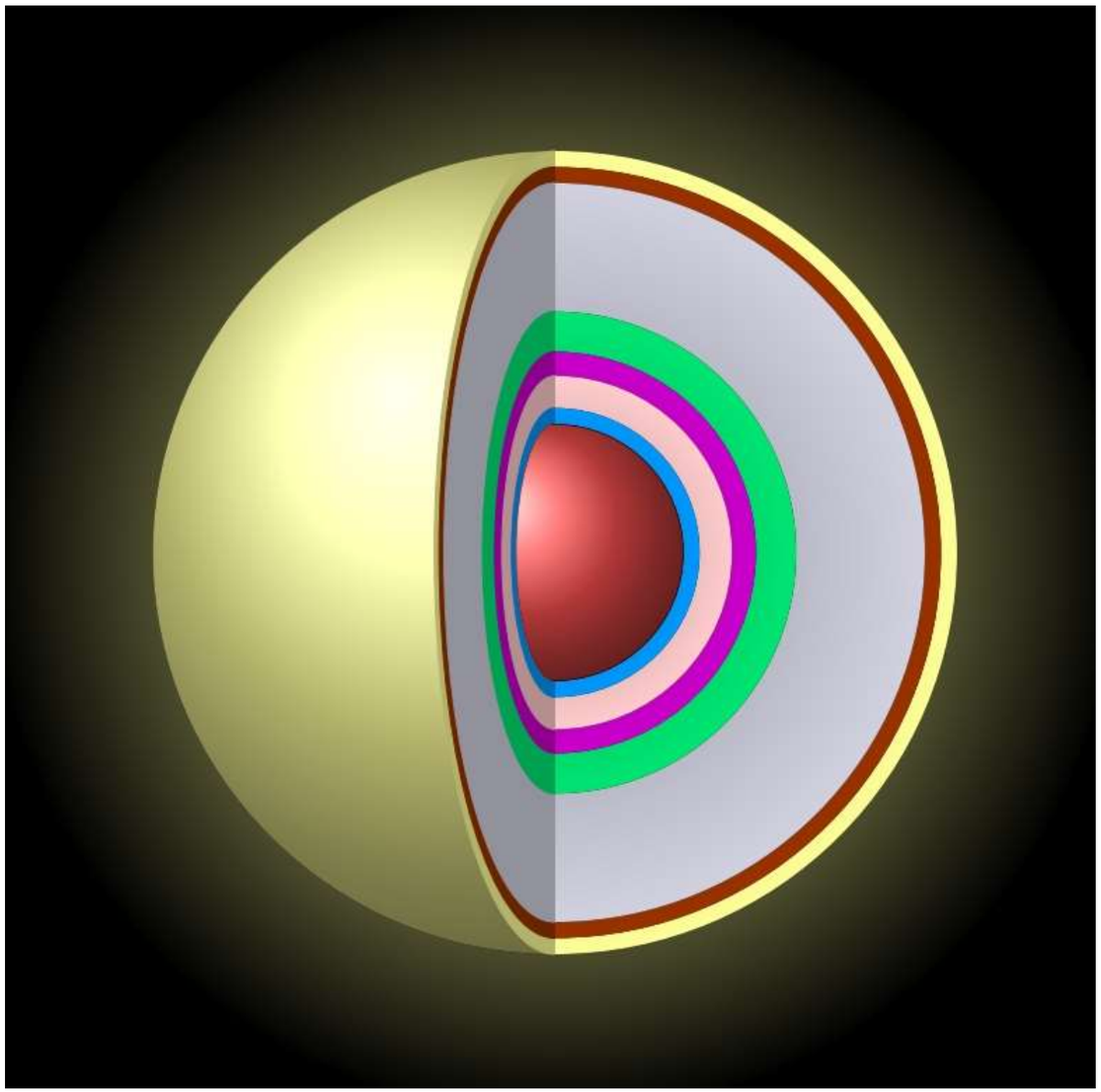
- Worked in many domains:
  - Number Crunching
  - Business / Internet
  - Networking / Protocols
  - Operating System Internals
  - Data Engines
  - Scientific / Medical
  - Engineering / Robotics

•I believe in reducing complexity through simplification



C++ Under the Hood – The Onion

C++ Under the Hood – The Onion



C++ Under the Hood – The Onion

C++ Under the Hood – The Onion (abused inheritance, low cohesion)

C++ Under the Hood – Inheritance vs. Aggregation / Composite

Inheritance

- **Is-a**

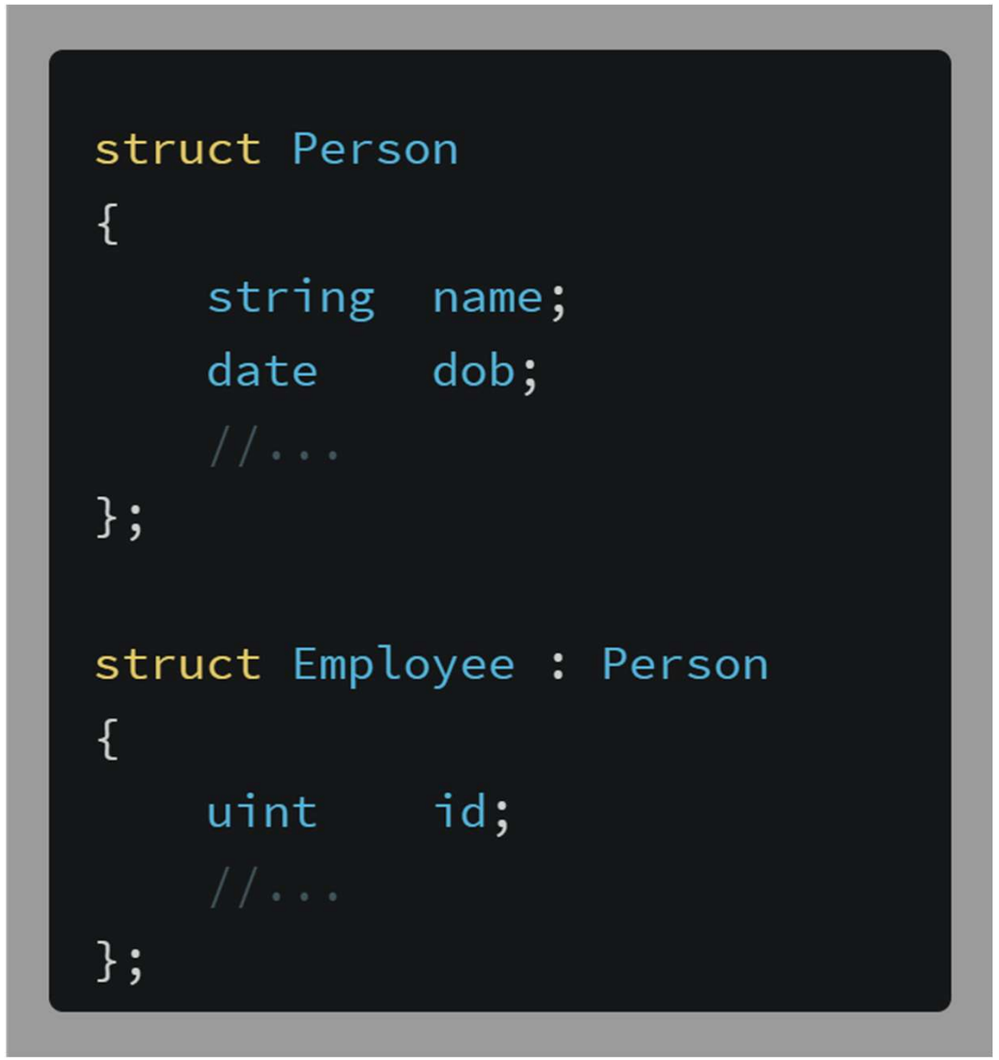
Aggregation / Composite

- **Has-a**

A Person **has a** Name

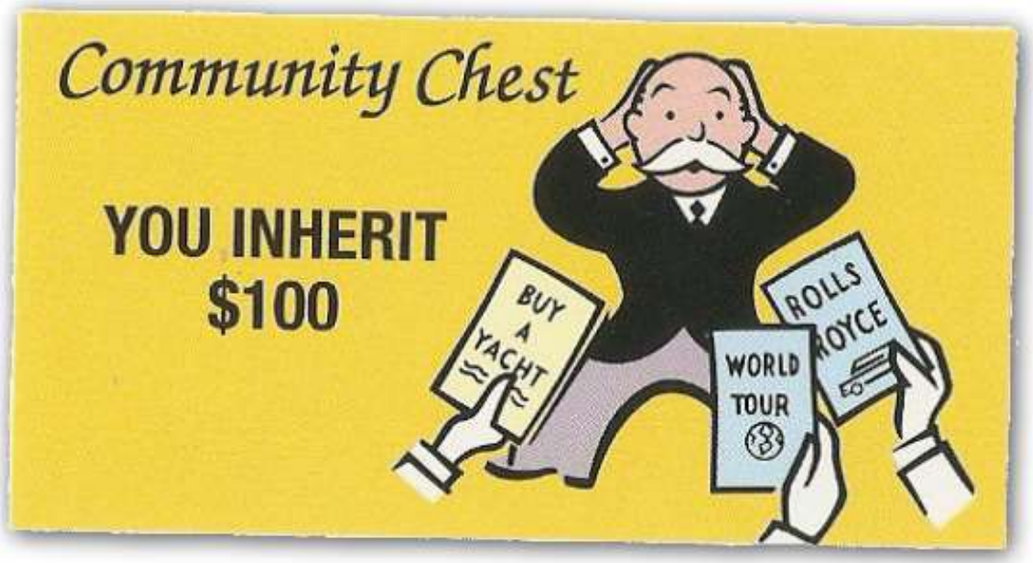
A Person is NOT a Name

An Employee **is a** Person



```
struct Person
{
    string  name;
    date    dob;
    //...
};

struct Employee : Person
{
    uint    id;
    //...
};
```

C++ Under the Hood – Inheritance & Construction



C++ Under the Hood – Inheritance Hierarchy

```
struct A {

    int a{};    string str1{};
            A() { Bar(); ... }
    virtual ~A() { Bar(); ... }
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {
    int b{};    string str2{};
            B() { Bar(); ... }
    virtual ~B() { Bar(); ... }
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {
    int c{};    string str3{};
            C() { Bar(); ... }
    virtual ~C() { Bar(); ... }
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – __v_table_ptr

```
struct A {
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }
    virtual ~A() { Bar(); ... }
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {
    int b{};    string str2{};
            B() { Bar(); ... }
    virtual ~B() { Bar(); ... }
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {
    int c{};    string str3{};
            C() { Bar(); ... }
    virtual ~C() { Bar(); ... }
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – __v_table_ptr / v_table

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }
    virtual ~A() { Bar(); ... }
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }
    virtual ~B() { Bar(); ... }
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }
    virtual ~C() { Bar(); ... }
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – C'tor (Constructor Implementation)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – D'tor (Destructor Implementation)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction (construct base members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction (construct base members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

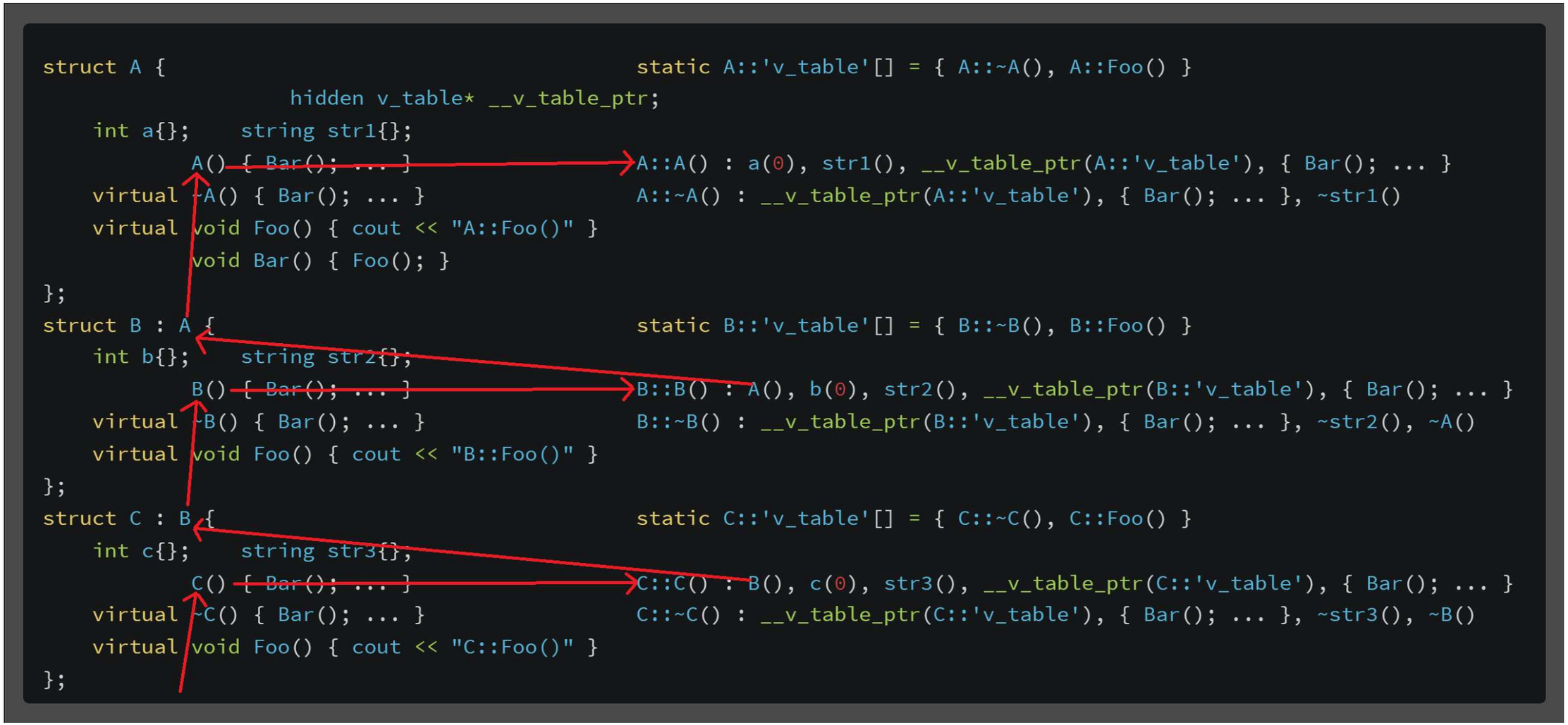
C++ Under the Hood – Construction (construct base members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction (construct base members)

```
struct A {                                          static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                      A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                     A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                      static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                      B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                     B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                      static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                      C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                     C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction



C++ Under the Hood – Construction (construct members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction (set this v_table)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Polymorphism During Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Polymorphism During Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Polymorphism During Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                  hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Polymorphism During Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction

```
struct A {                                  static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }              A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }             A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                              static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }              B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }             B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                              static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }              C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }             C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction (construct members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction (set this v_table)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Polymorphism During Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(),   v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(),   v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction (construct members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction (set this v_table)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                  hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Polymorphism During Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Construction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Fully Constructed

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction (set this v_table)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Polymorphism During Destruction

```
struct A {                                          static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                      A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                     A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                      static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                      B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                     B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                      static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                      C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                     C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction (destruct members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction (destruct base members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction (destruct base members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction

```
struct A {                                  static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }              A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }             A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                              static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }              B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }             B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                              static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }              C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }             C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction (set this v_table)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Polymorphism During Destruction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction (destruct members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction (destruct base members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction (destruct base members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction

```
struct A {                                  static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }              A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }             A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                              static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }              B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }             B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                              static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }              C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }             C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction (set this v_table)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Polymorphism During Destruction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction (destruct members)

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction

```
struct A {                                  static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }              A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }             A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                              static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }              B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }             B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                              static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }              C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }             C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction

```
struct A {                                  static A::'v_table'[] = { A::~A(), A::Foo() }
                  hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }              A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }             A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                              static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }              B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }             B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                              static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }              C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }             C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Destruction

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood – Fully Destructed

```
struct A {                                      static A::'v_table'[] = { A::~A(), A::Foo() }
                    hidden v_table* __v_table_ptr;
    int a{};    string str1{};
            A() { Bar(); ... }                  A::A() : a(0), str1(), __v_table_ptr(A::'v_table'), { Bar(); ... }
    virtual ~A() { Bar(); ... }                 A::~A() : __v_table_ptr(A::'v_table'), { Bar(); ... }, ~str1()
    virtual void Foo() { cout << "A::Foo()" }
            void Bar() { Foo(); }
};
struct B : A {                                  static B::'v_table'[] = { B::~B(), B::Foo() }
    int b{};    string str2{};
            B() { Bar(); ... }                  B::B() : A(), b(0), str2(), __v_table_ptr(B::'v_table'), { Bar(); ... }
    virtual ~B() { Bar(); ... }                 B::~B() : __v_table_ptr(B::'v_table'), { Bar(); ... }, ~str2(), ~A()
    virtual void Foo() { cout << "B::Foo()" }
};
struct C : B {                                  static C::'v_table'[] = { C::~C(), C::Foo() }
    int c{};    string str3{};
            C() { Bar(); ... }                  C::C() : B(), c(0), str3(), __v_table_ptr(C::'v_table'), { Bar(); ... }
    virtual ~C() { Bar(); ... }                 C::~C() : __v_table_ptr(C::'v_table'), { Bar(); ... }, ~str3(), ~B()
    virtual void Foo() { cout << "C::Foo()" }
};
```

C++ Under the Hood

‘C’ Pointer types

int *	**Pointer to Data**
int(*)(int)	Pointer to Function

Member Pointer types

int Foo::*	Pointer to Member Data
int(Foo::*)(int)	Pointer to Member Function

C++ Under the Hood - ‘C’ Style Pointers - Pointer to Data(aka object)

```cpp
struct Foo {
    int i{};
};

int* Bar(int* p) {
    (*p)++;
    return p;
}

int main()
{
    int i{2};
    Foo foo{3};

    int* p1 = &i;
    int* p2 = &foo.i;
    int* p3 = Bar(p1);

    return *p2 + *p3;
}
```

godbolt.org/z/rKzdTsejz

C++ Under the Hood

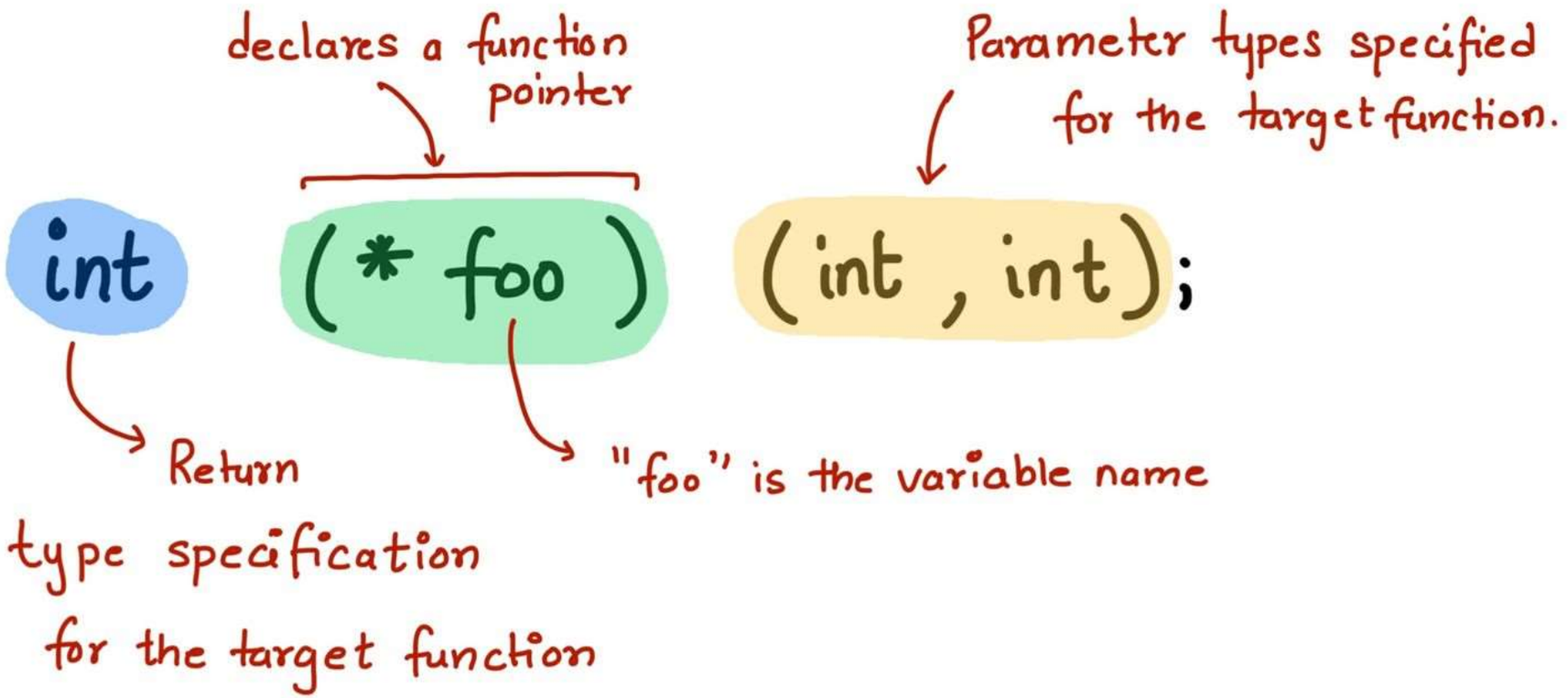
‘C’ Pointer types

int *	Pointer to Data
int(*)(int)	**Pointer to Function**

Member Pointer types

int Foo::*	Pointer to Member Data
int(Foo::*)(int)	Pointer to Member Function

C++ Under the Hood



C++ Under the Hood - ‘C’ Style Pointers - Pointer to Function (PFN)

```
int Foo(int i)
{
    return i++;
}

int Bar(int(*pfn)(int), int i)
{
    return pfn(i);
}

int main()
{
    return Bar(&Foo, 1);
}
```

godbolt.org/z/hvYM1no1b

C++ Under the Hood – 'C' Style Pointers - Pointer to Function (PFN)

```
int Bar(int(*pFN)(int))
{
    return pFN(10);
}

int Foo(int i)
{
    return i * (i+1) / 2;
}

int main()
{
    int(*pFN)(int) = &Foo;

    return Bar(pFN);
}
```

godbolt.org/z/Y1c7eWjb6

C++ Under the Hood – 'C' Style Pointers - Pointer to Function (PFN)

```
typedef int(*PFN)(int);

int Bar(PFN pFN)
{
    return pFN(10);
}

int Foo(int i)
{
    return i * (i+1) / 2;
}

int main()
{
    PFN pFN = &Foo;

    return Bar(pFN);
}
```

godbolt.org/z/s9cvWaEq8

C++ Under the Hood – 'C' Style Pointers - Pointer to Function (PFN)

```
typedef int(PFN)(int);

int Bar(PFN pFN)
{
    return pFN(10);
}

int Foo(int i)
{
    return i * (i+1) / 2;
}

int main()
{
    PFN *pFN = &Foo;

    return Bar(pFN);
}
```

godbolt.org/z/s9cvWaEq8

C++ Under the Hood - ‘C’ Style Pointers - Pointer to Function (PFN)

```
typedef int(PFN)(int);

int Bar(PFN pFN)
{
    return pFN(10);
}

int Foo(int i)
{
    return i * (i+1) / 2;
}

int main()
{
    PFN *pFN = Foo;

    return Bar(pFN);
}
```

godbolt.org/z/Pr1ovqvon

C++ Under the Hood - ‘C’ Style Pointers – qsort(…) Pointer to Function (PFN)

```c
int compare(const void* a, const void* b) {
   return (*(int*)a - *(int*)b);
}

int main() {
   int arr[] = {10, 5, 4, 6, 9};
   int count = sizeof(arr) / sizeof(arr[0]);

   qsort(arr, count, sizeof(arr[0]), compare);

   printf("Sorted array: ");
   for (int i = 0; i < count; ++i) {
      printf("%d ", arr[i]);
   }
   printf("\n");
   return 0;
}
```

godbolt.org/z/vc8PzYcs3

C++ Under the Hood

‘C’ Pointer types

int *	Pointer to Data
int(*)(int)	Pointer to Function

Member Pointer types

int Foo::*	Pointer to Member Data
int(Foo::*)(int)	**Pointer to Member Function**

C++ Under the Hood - ‘C’ Style Pointers - Pointer to Member Function

```cpp
typedef int(*PFN)(int);

struct Foo
{
    int i;
    int Bar(int j) { return i+j; }
};

int main()
{
    PFN pFN = &Foo::Bar;

    Foo obj{10};

    return (obj.*pMF)(20);
}
```

godbolt.org/z/jfr9dMj7G

C++ Under the Hood - ~~'C' Style~~ Pointers - Pointer to ~~Member Function~~

```cpp
typedef int(*PFN)(int);

struct Foo
{
    int i;
    int Bar(int j) { return i+j; }
};

int main()
{
    PFN pFN = &Foo::Bar;

    Foo obj{10};

    return (obj.*pMF)(20);
}
```

godbolt.org/z/jfr9dMj7G

C++ Under the Hood – C++ Pointer to Member Function

```
struct Foo;
typedef int(Foo::*PMF)(int);

struct Foo
{
    int i;
    int Bar(int j) { return i+j; }
};

int main()
{
    PMF pMF = &Foo::Bar;

    Foo obj{10};

    return (obj.*pMF)(20);
}
```

godbolt.org/z/eY1n7a97x

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```cpp
struct Foo;
typedef int(Foo::*PMF)(int);
struct Foo
{
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
};
struct Foo2 : Foo
{
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    PMF pMF = &Foo::Bar;
    Foo2 obj{10};
    return (obj.*pMF)(20);
}
```

godbolt.org/z/qWhE5ErKz

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```cpp
struct Foo;
typedef int(Foo::*PMF)(int);
struct Foo
{
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
};
struct Foo2 : Foo
{
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    PMF pMF = &Foo::Bar;
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/vTfM143d7

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```cpp
struct Foo;
using PMF = int(Foo::*)(int);
struct Foo
{
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
};
struct Foo2 : Foo
{
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    PMF pMF = &Foo::Bar;
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/188fcronz

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```cpp
struct Foo
{
    using PMF = int(Foo::*)(int);

    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
};
struct Foo2 : Foo
{
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    Foo::PMF pMF = &Foo::Bar;
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/hE3nMhvET

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```
struct Foo
{
    using PMF = int(Foo::*)(int);


    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }

};
struct Foo2 : Foo
{

    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    Foo::PMF pMF = &Foo::Bar;
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/jTqzxGeeo

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                                        V_table* pV_table;
                                        static V_table Foo::v_table[] = { &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }

};
struct Foo2 : Foo
{
                                        static V_table Foo2::v_table[] = { &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    Foo::PMF pMF = &Foo::Bar;
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/jTqzxGeeo

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                                        V_table* pV_table;
                                        static V_table Foo::v_table[] = { &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
                                        int Foo::Bar_Thunk(int i)       { return (this->pV_table[0])(int i); }
};
struct Foo2 : Foo
{
                                        static V_table Foo2::v_table[] = { &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    Foo::PMF pMF = &Foo::Bar; // = &Foo::Bar_Thunk
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/jTqzxGeeo

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                                        V_table* pV_table;
                                        static V_table Foo::v_table[] = { &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
                                        int Foo::Bar_Thunk(int i)       { return (this->pV_table[0])(int i); }
};
struct Foo2 : Foo
{
                                        static V_table Foo2::v_table[] = { &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    Foo::PMF pMF = &Foo::Bar; // = &Foo::Bar_Thunk
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/jTqzxGeeo

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                                        V_table* pV_table;
                                        static V_table Foo::v_table[] = { &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
                                        int Foo::Bar_Thunk(int i)       { return (this->pV_table[0])(int i); }
};
struct Foo2 : Foo
{
                                        static V_table Foo2::v_table[] = { &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    Foo::PMF pMF = &Foo::Bar; // = &Foo::Bar_Thunk
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/jTqzxGeeo

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                                        V_table* pV_table;
                                        static V_table Foo::v_table[] = { &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
                                        int Foo::Bar_Thunk(int i)       { return (this->pV_table[0])(int i); }
};
struct Foo2 : Foo
{
                                        static V_table Foo2::v_table[] = { &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    Foo::PMF pMF = &Foo::Bar; // = &Foo::Bar_Thunk
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/jTqzxGeeo

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                                        V_table* pV_table;
                                        static V_table Foo::v_table[] = { &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
                                        int Foo::Bar_Thunk(int i)       { return (this->pV_table[0])(int i); }
};
struct Foo2 : Foo
{
                                        static V_table Foo2::v_table[] = { &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    Foo::PMF pMF = &Foo::Bar; // = &Foo::Bar_Thunk
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/jTqzxGeeo

C++ Under the Hood – C++ Pointer to (virtual) Member Function

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                                        V_table* pV_table;
                                        static V_table Foo::v_table[] = { &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
                                        int Foo::Bar_Thunk(int i)       { return (this->pV_table[0])(int i); }
};
struct Foo2 : Foo
{
                                        static V_table Foo2::v_table[] = { &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    Foo::PMF pMF = &Foo::Bar; // = &Foo::Bar_Thunk
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/jTqzxGeeo

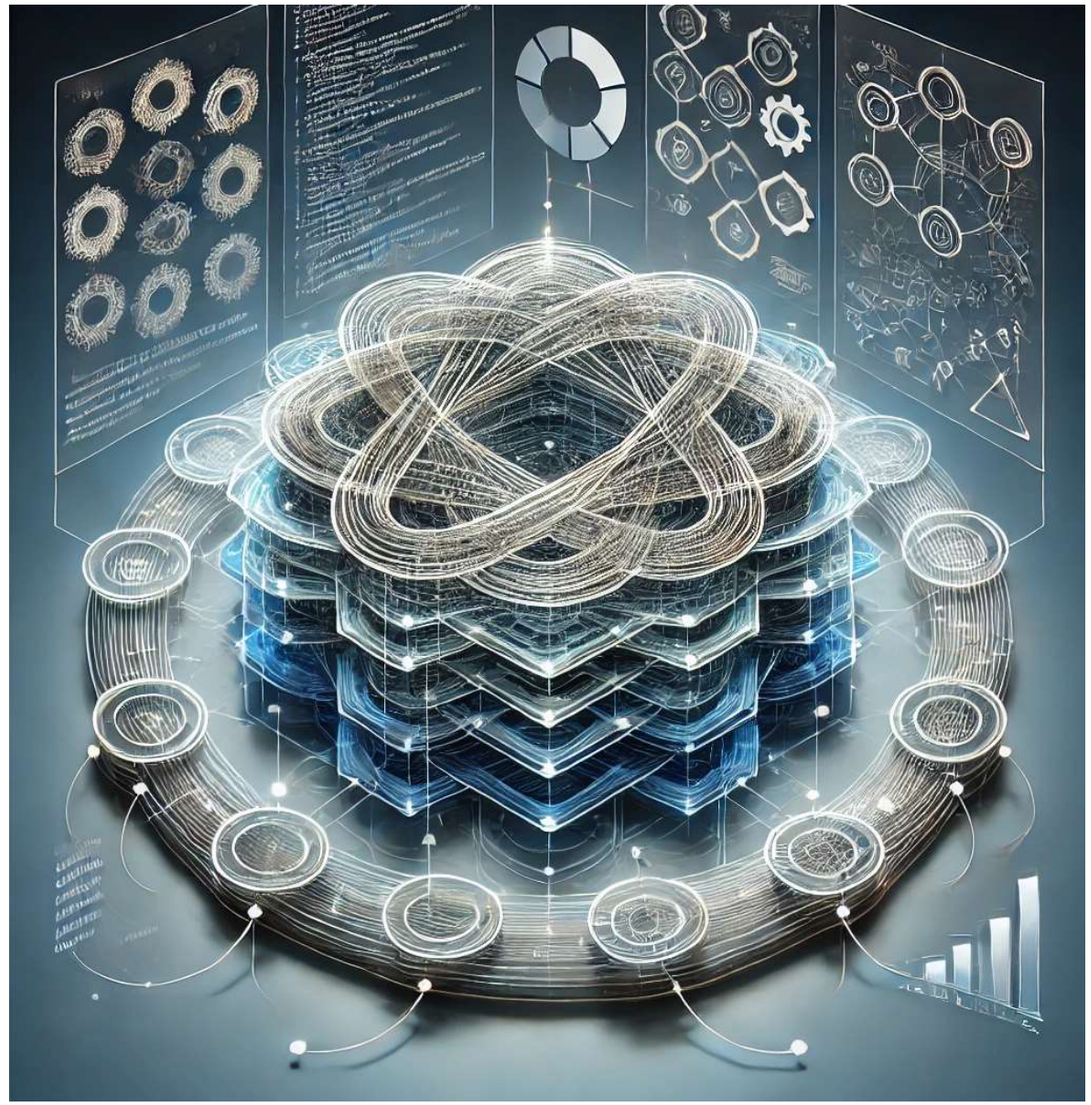
C++ Under the Hood – C++ Pointer to (virtual) Member Function

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                                        V_table* pV_table;
                                        static V_table Foo::v_table[] = { &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i+j; }
                                        int Foo::Bar_Thunk(int i)       { return (this->pV_table[0])(int i); }
};
struct Foo2 : Foo
{
                                        static V_table Foo2::v_table[] = { &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i*j; }
};
int main()
{
    Foo::PMF pMF = &Foo::Bar; // = &Foo::Bar_Thunk
    Foo2 obj{10};
    Foo *pFoo = &obj;
    return (pFoo->*pMF)(20);
}
```

godbolt.org/z/jTqzxGeeo

Fundamental Theorem of Software Engineering:

Any problem can be solved by adding a layer of indirection.



C++ Under the Hood

‘C’ Pointer types

int *	Pointer to Data
int(*)(int)	Pointer to Function

Member Pointer types

int Foo::*	**Pointer to Member Data**
int(Foo::*)(int)	Pointer to Member Function

C++ Under the Hood – C++ Pointer to Member Data

```
int main()
{
    struct Foo { int a, b, c, d; };

    Foo foo1{ 2,  3,  5,  7};
    Foo foo2{11, 13, 17, 19};

    Foo* pFoo1 = &foo1;
    Foo* pFoo2 = &foo2;


    cout << "foo1.a  = "   << foo1.a    << '\n';                    //foo1.a  = 2
    cout << "foo2.a  = "   << foo2.a    << '\n';                    //foo2.a  = 11


    cout << "pFoo1->c  = " << pFoo1->c  << '\n';                    //foo1.c  = 5
    cout << "pFoo2->c  = " << pFoo2->c  << '\n';                    //foo2.c  = 17
}
```

godbolt.org/z/K5nf9jYPv

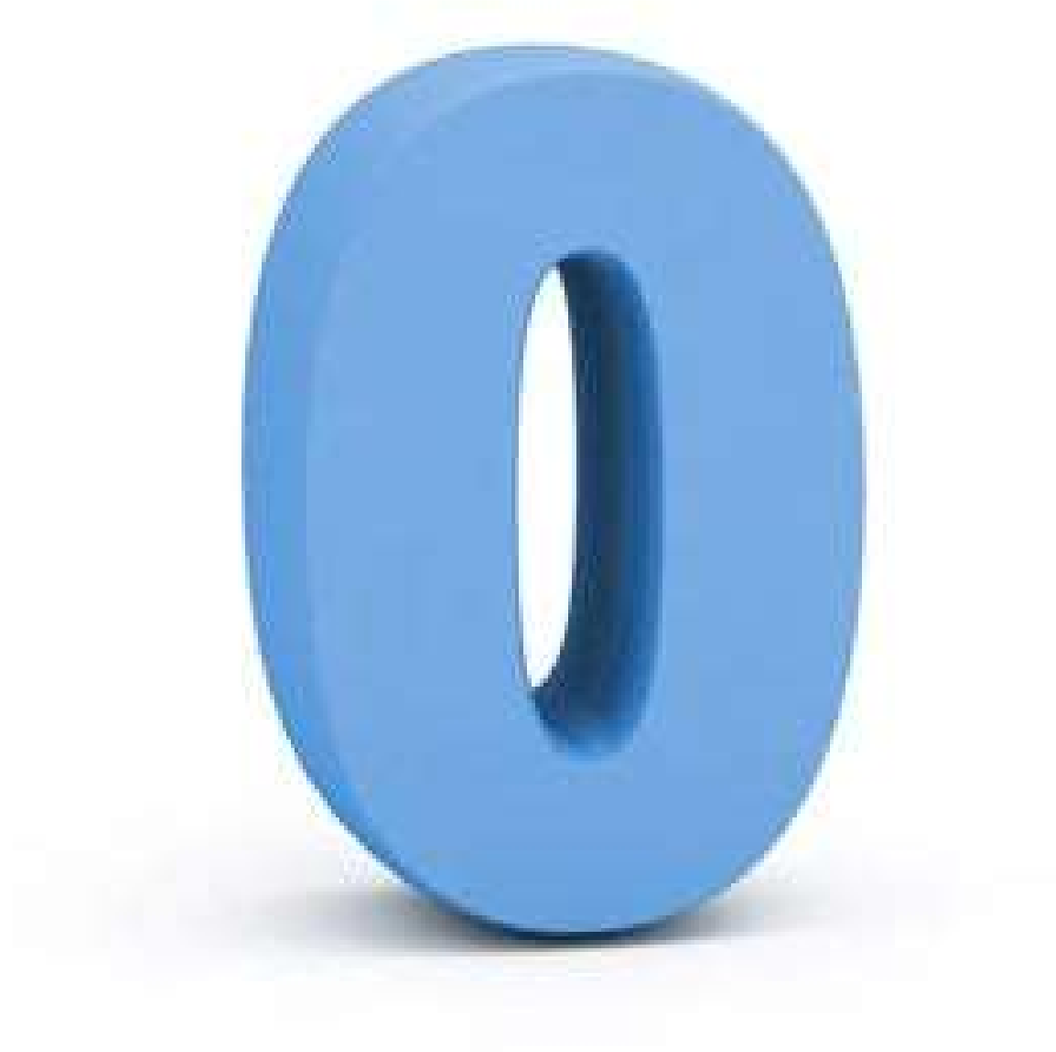
C++ Under the Hood – C++ Pointer to Member Data

```cpp
int main()
{
    struct Foo { int a, b, c, d; };

    Foo foo1{ 2,  3,  5,  7};
    Foo foo2{11, 13, 17, 19};

    Foo* pFoo1 = &foo1;
    Foo* pFoo2 = &foo2;

    int Foo::* p = &Foo::a;
    cout << "foo1.*p = "   << foo1.*p   << '\n';                    //foo1.*p = 2 (aka foo1.a)
    cout << "foo2.*p = "   << foo2.*p   << '\n';                    //foo2.*p = 11 (aka foo2.a)

    p = &Foo::c;
    cout << "pFoo1->*p = " << pFoo1->*p << '\n';                    //pFoo1->*p = 5 (aka foo1.c)
    cout << "pFoo1->*p = " << pFoo2->*p << '\n';                    //pFoo2->*p2 = 17 (aka foo2.c)
}
```

godbolt.org/z/K5nf9jYPv

NULL

C++ Under the Hood



nullptr

C++ Under the Hood

C++ Under the Hood – C++ Pointer to Member Data

```
int main()
{
    struct Foo { int a, b, c, d; };
    int Foo::* p;

    p = nullptr;    cout << "(p) = " << p << '\n';          //0 (zero, aka false) unassigned
    p = &Foo::a;    cout << "(p) = " << p << '\n';          //1 (one, aka true) assigned
    p = &Foo::b;    cout << "(p) = " << p << "\n\n";        //1 (one, aka true) assigned









}
```

godbolt.org/z/K5nf9jYPv

C++ Under the Hood – C++ Pointer to Member Data

```
int main()
{
    struct Foo { int a, b, c, d; };
    int Foo::* p;

    p = nullptr;    cout << "(p) = " << p << '\n';                  //0 (zero, aka false) unassigned
    p = &Foo::a;    cout << "(p) = " << p << '\n';                  //1 (one, aka true) assigned
    p = &Foo::b;    cout << "(p) = " << p << "\n\n";                //1 (one, aka true) assigned

    p = &Foo::a;    cout << "*(int*)&p = " << *(int*)&p <<'\n';     //p = 0
    p = &Foo::b;    cout << "*(int*)&p = " << *(int*)&p <<'\n';     //p = 4
    p = &Foo::c;    cout << "*(int*)&p = " << *(int*)&p <<'\n';     //p = 8
    p = &Foo::d;    cout << "*(int*)&p = " << *(int*)&p <<"\n\n";   //p = 12




}
```

godbolt.org/z/K5nf9jYPv

C++ Under the Hood – C++ Pointer to Member Data

```
int main()
{
    struct Foo { int a, b, c, d; };
    int Foo::* p;

    p = nullptr;    cout << "(p) = " << p << '\n';                  //0 (zero, aka false) unassigned
    p = &Foo::a;    cout << "(p) = " << p << '\n';                  //1 (one, aka true) assigned
    p = &Foo::b;    cout << "(p) = " << p << "\n\n";                //1 (one, aka true) assigned

    p = &Foo::a;    cout << "*(int*)&p = " << *(int*)&p <<'\n';     //p = 0
    p = &Foo::b;    cout << "*(int*)&p = " << *(int*)&p <<'\n';     //p = 4
    p = &Foo::c;    cout << "*(int*)&p = " << *(int*)&p <<'\n';     //p = 8
    p = &Foo::d;    cout << "*(int*)&p = " << *(int*)&p <<"\n\n";   //p = 12


    p = nullptr;    cout << "(p) = " << p << '\n';                  //0 (zero, aka false) unassigned
                    cout << "*(int*)&p = " << *(int*)&p <<'\n';     //p = -1
}
```

godbolt.org/z/K5nf9jYPv

C++ Under the Hood – C++ Pointer to Member Data (pointer to object + offset)

```cpp
int main()
{
    struct Foo { int a, b, c, d; };

    Foo foo1{ 2,  3,  5,  7};
    Foo foo2{11, 13, 17, 19};

    Foo* pFoo1 = &foo1;
    Foo* pFoo2 = &foo2;

    int Foo::* p = &Foo::a;
    cout << "foo1.*p = "   << foo1.*p   << '\n';                    //foo1.*p = 2 (aka foo1.a)
    cout << "foo2.*p = "   << foo2.*p   << '\n';                    //foo2.*p = 11 (aka foo2.a)

    p = &Foo::c;
    cout << "pFoo1->*p = " << pFoo1->*p << '\n';                    //pFoo1->*p = 5 (aka foo1.c)
    cout << "pFoo1->*p = " << pFoo2->*p << '\n';                    //pFoo2->*p2 = 17 (aka foo2.c)
}
```

godbolt.org/z/K5nf9jYPv

C++ Under the Hood - ‘C’ Style Pointers – qsort(…) Pointer to Function (PFN)

```c
int compare(const void* a, const void* b) {
   return (*(int*)a - *(int*)b);
}

int main() {
   int arr[] = {10, 5, 4, 6, 9};
   int count = sizeof(arr) / sizeof(arr[0]);

   qsort(arr, count, sizeof(arr[0]), compare);

   printf("Sorted array: ");
   for (int i = 0; i < count; ++i) {
      printf("%d ", arr[i]);
   }
   printf("\n");
   return 0;
}
```

godbolt.org/z/vc8PzYcs3

C++ Under the Hood – C++ style Sort(...) with lambda [](l&, r&){ l < r}

```cpp
struct Foo { int a, b, c, d; };

void Sort(std::vector<Foo>& data) {
    std::sort(std::begin(data), std::end(data),
        [](auto& lhs, auto& rhs){ return lhs.a < rhs.a; });
}

int main()
{
    auto data = GetData();

    Sort(data);
    Output(data);
}
```

godbolt.org/z/8vvr9brEo

C++ Under the Hood – C++ style Sort(...) with lambda +[](l&, r&){ l < r}

```
struct Foo { int a, b, c, d; };

void Sort(std::vector<Foo>& data, bool(*compare)(Foo&,Foo&)) {
    std::sort(std::begin(data), std::end(data), compare);
}

int main()
{
    auto data = GetData();

    Sort(data, +[](Foo& lhs, Foo& rhs)->bool { return lhs.a < rhs.a; });
    Output(data);

    Sort(data, +[](Foo& lhs, Foo& rhs)->bool { return lhs.b < rhs.b; });
    Output(data);

    Sort(data, +[](Foo& lhs, Foo& rhs)->bool { return lhs.c < rhs.c; });
    Output(data);

    Sort(data, +[](Foo& lhs, Foo& rhs)->bool { return lhs.d < rhs.d; });
    Output(data);
}
```

godbolt.org/z/48aTYosKr

C++ Under the Hood – C++ Pointer to Member Data

```cpp
struct Foo { int a, b, c, d; };
//...
template<typename T>
void Sort(auto& data, int T::* key) {
    std::sort(std::begin(data), std::end(data),
        [key](auto& lhs, auto& rhs){ return lhs.*key < rhs.*key; });
}
int main()
{
    auto data = GetData();

    Sort(data, &Foo::a);    std::cout<<"a:\n";      Output(data);
    Sort(data, &Foo::b);    std::cout<<"b:\n";      Output(data);
    Sort(data, &Foo::c);    std::cout<<"c:\n";      Output(data);
    Sort(data, &Foo::d);    std::cout<<"d:\n";      Output(data);
}
```

godbolt.org/z/1G3ooPqr7

C++ Under the Hood – C++ Pointer to Member Data

```cpp
template<typename T>
bool compare(auto& lhs, auto& rhs, auto T::* key, auto...keys) {
    if (lhs.*key < rhs.*key) return true;
    if (lhs.*key > rhs.*key) return false;
    if constexpr (sizeof...(keys)>0)
        return compare(lhs, rhs, keys...);
    return false;
}
void Sort(auto& data, auto ...keys) {
    std::sort(std::begin(data), std::end(data),
            [keys...](auto& lhs, auto& rhs)
                { return compare(lhs, rhs, keys...); });
}

int main()
{
    std::vector<Data> data = GetData();

    std::cout << "Sort based on b\n";
    Shuffle(data);          Sort(data, &Data::b, &Data::a);              Output(data);

    std::cout << "Sort based on c,d,e\n";
    Shuffle(data);          Sort(data, &Data::c, &Data::d, &Data::e);    Output(data);
}
```

godbolt.org/z/e9r9fbvvz

C++ Under the Hood – Inheritance vs. Aggregation

Inheritance

- **Is-a**

Aggregation

- **Has-a**

A Person is NOT a Name

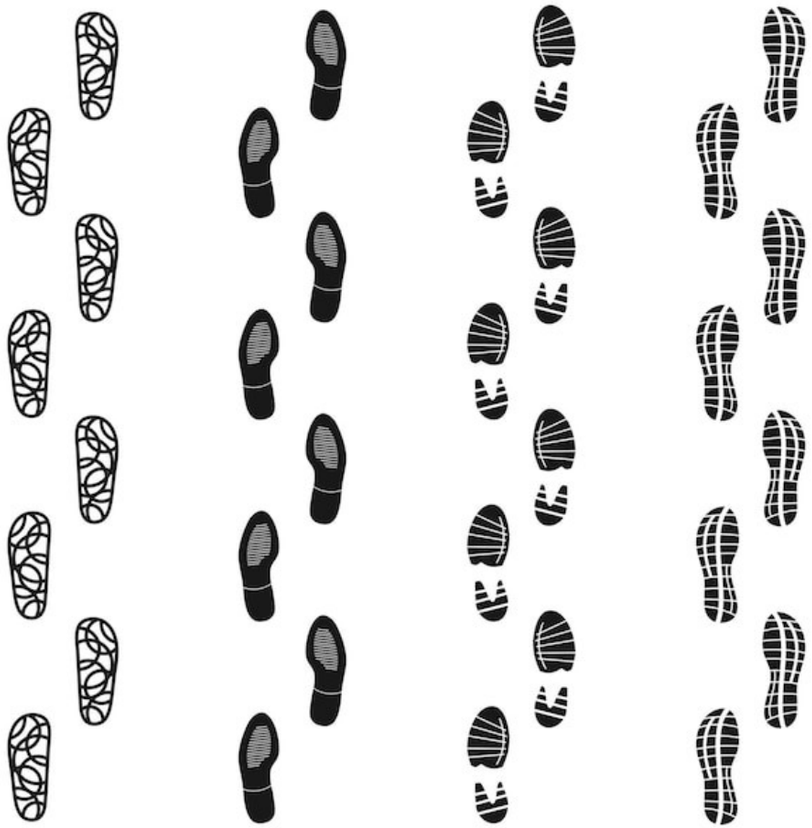
A Person **has a** Name

An Employee **is a** Person

```
struct Person
{
    string  name;
    date    dob;
    //...
};

struct Employee : Person
{
    uint    id;
    //...
};
```

C++ Under the Hood

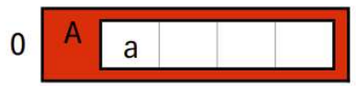
Object Memory Footprint



v-table Mechanism

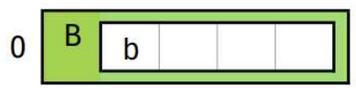
C++ Under the Hood

```
struct A {
  int a;
  int Foo(...) {...};
};
```



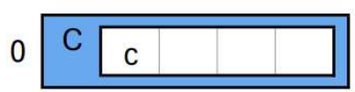
A::Foo(...) {...}

```
struct B {
  int b;
  int Foo(...) {...};
};
```



B::Foo(...) {...}

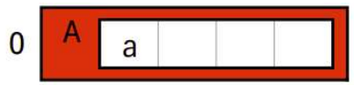
```
struct C {
  int c;
  int Foo(...) {...};
};
```



C::Foo(...) {...}

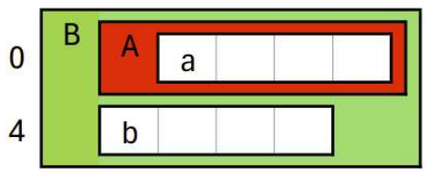
C++ Under the Hood

```
struct A {
  int a;
  int Foo(...) {...};
};
```



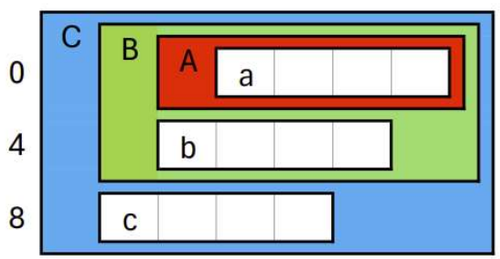
A::Foo(...) {...}

```
struct B : A {
  int b;
  int Foo(...) {...};
};
```



B::Foo(...) {...}

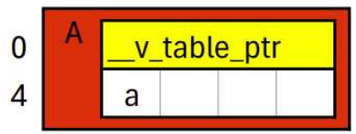
```
struct C : B {
  int c;
  int Foo(...) {...};
};
```



C::Foo(...) {...}

C++ Under the Hood

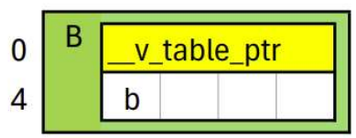
```
struct A {
  int a;
        A(...) {...};
  virtual int Bar(...) {...};
        int Foo(...) {...};
};
```



A::'v_table'= {A::Bar};

A::A(...) {...}
A::Bar(...) {...}
A::Foo(...) {...}

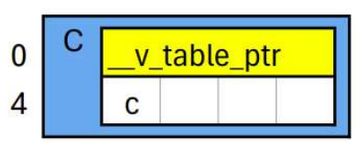
```
struct B {
  int b;
        B(...) {...};
  virtual int Bar(...) {...};
        int Foo(...) {...};
};
```



B::'v_table'= {B::Bar};

B::B(...) {...}
B::Bar(...) {...}
B::Foo(...) {...}

```
struct C {
  int c;
        C(...) {...};
  virtual int Bar(...) {...};
        int Foo(...) {...};
};
```

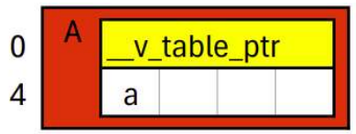


C::'v_table'= {C::Bar};

C::C(...) {...}
C::Bar(...) {...}
C::Foo(...) {...}

C++ Under the Hood

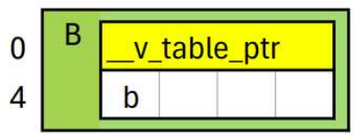
```
struct A {
  int a;
          A(...) {...};
  virtual ~A() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



A::'v_table'= {A::~A, A::Bar};

A::A(...) {...}
A::~A() {...}
A::Bar(...) {...}
A::Foo(...) {...}

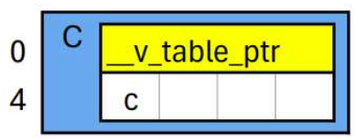
```
struct B {
  int b;
          B(...) {...};
  virtual ~B() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



B::'v_table'= {B::~B, B::Bar};

B::B(...) {...}
B::~B() {...}
B::Bar(...) {...}
B::Foo(...) {...}

```
struct C {
  int c;
          C(...) {...};
  virtual ~C() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```

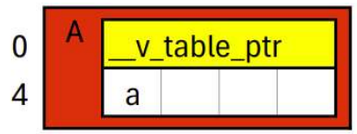


C::'v_table'= {C::~C, C::Bar};

C::C(...) {...}
C::~C() {...}
C::Bar(...) {...}
C::Foo(...) {...}

C++ Under the Hood

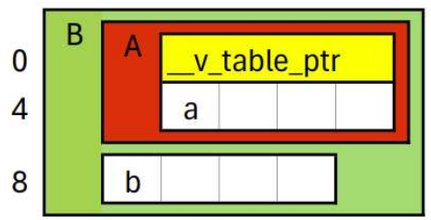
```
struct A {
  int a;
          A(...) {...};
  virtual ~A() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



A::'v_table'= {A::~A, A::Bar};

A::A(...) {...}
A::~A() {...}
A::Bar(...) {...}
A::Foo(...) {...}

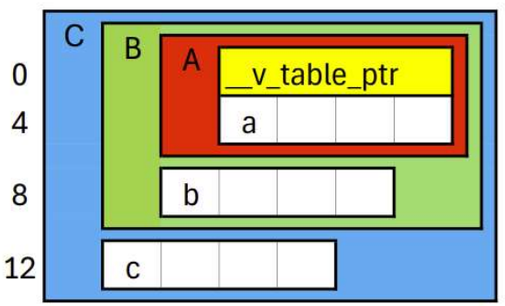
```
struct B : A {
  int b;
          B(...) {...};
  virtual ~B() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



B::'v_table'= {B::~B, B::Bar};

B::B(...) {...}
B::~B() {...}
B::Bar(...) {...}
B::Foo(...) {...}

```
struct C : B {
  int c;
          C(...) {...};
  virtual ~C() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```

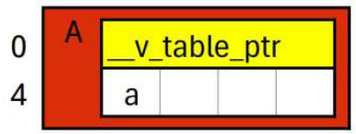


C::'v_table'= {C::~C, C::Bar};

C::C(...) {...}
C::~C() {...}
C::Bar(...) {...}
C::Foo(...) {...}

C++ Under the Hood

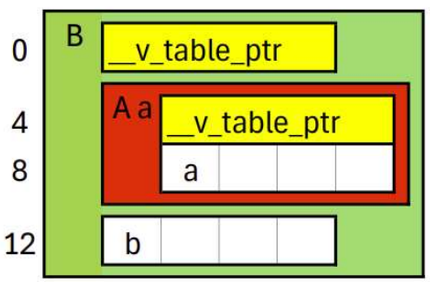
```
struct A {
  int a;
          A(...) {...};
  virtual ~A() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



A::'v_table'= {A::~A, A::Bar};

A::A(...) {...}
A::~A() {...}
A::Bar(...) {...}
A::Foo(...) {...}

```
struct B {
  A  a;
  int b;
          B(...) {...};
  virtual ~B() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```

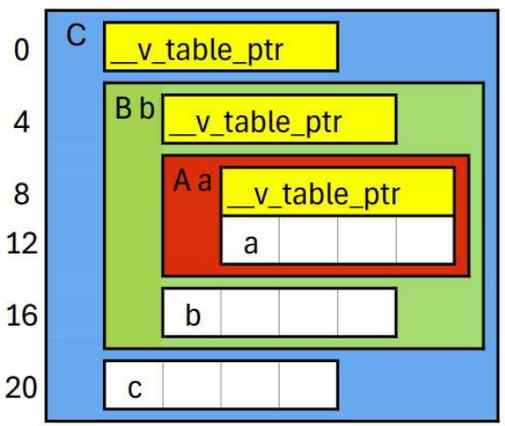


B::'v_table'= {B::~B, B::Bar};

A::'v_table'= {A::~A, A::Bar};

B::B(...) {...}
B::~B() {...}
B::Bar(...) {...}
B::Foo(...) {...}

```
struct C {
  B  b;
  int c;
          C(...) {...};
  virtual ~C() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



C::'v_table'= {C::~C, C::Bar};

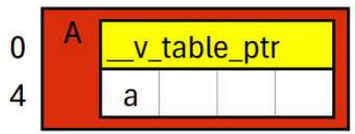
B::'v_table'= {B::~B, B::Bar};

A::'v_table'= {A::~A, A::Bar};

C::C(...) {...}
C::~C() {...}
C::Bar(...) {...}
C::Foo(...) {...}

C++ Under the Hood

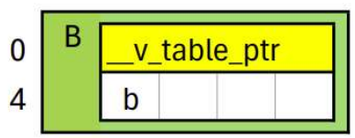
```
struct A {
  int a;
            A(...) {...};
  virtual   ~A() {...};
  virtual int Bar(...) {...};
};
```



A::'v_table'= {A::~A, A::Bar};

A::A(...) {...}
A::~A() {...}
A::Bar(...) {...}

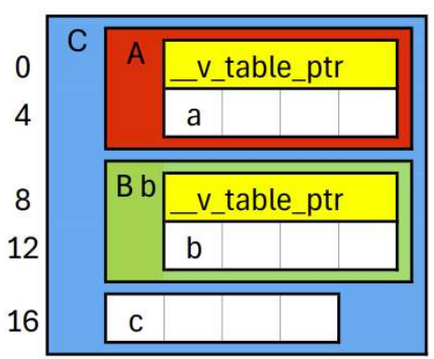
```
struct B {
  int b;
            B(...) {...};
  virtual   ~B() {...};
  virtual int Bar(...) {...};
            int Foo(...) {...};
};
```



B::'v_table'= {B::~B, B::Bar};

B::B(...) {...}
B::~B() {...}
B::Bar(...) {...}
B::Foo(...) {...}

```
struct C : A {
  B   b;
  int c;
            C(...) {...};
  virtual   ~C() {...};
  virtual int Bar(...) {...};
};
```



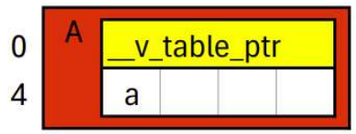
C::A_'v_table'= {C::~C, C::Bar};

B::B_'v_table'= {B::~B, B::Bar};

C::C(...) {...}
C::~C() {...}
C::Bar(...) {...}

C++ Under the Hood

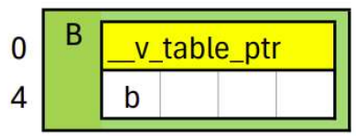
```
struct A {
  int a;
        A(...) {...};
  virtual ~A() {...};
  virtual int Bar(...) {...};
};
```



A::'v_table'= {A::~A, A::Bar};

A::A(...) {...}
A::~A() {...}
A::Bar(...) {...}

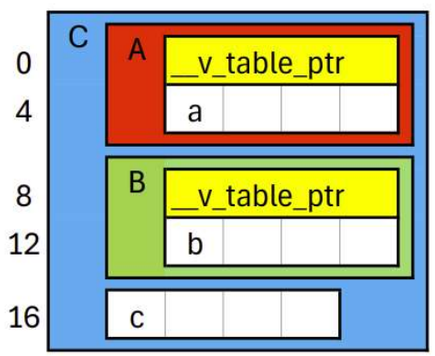
```
struct B {
  int b;
        B(...) {...};
  virtual ~B() {...};
  virtual int Bar(...) {...};
        int Foo(...) {...};
};
```



B::'v_table'= {B::~B, B::Bar};

B::B(...) {...}
B::~B() {...}
B::Bar(...) {...}
B::Foo(...) {...}

```
struct C : A, B {
  int c;
        C(...) {...};
  virtual ~C() {...};
  virtual int Bar(...) {...};
};
```



C::A_'v_table'= {C::~C, C::Bar};

C::B_'v_table'= {C::~C_adj8, C::Bar_adj8};

C::C(...) {...}
C::~C() {...}
C::Bar(...) {...}
C::~C_adj8() {...}
C::Bar_adj8() {...}

C++ Under the Hood – Multiple this pointers

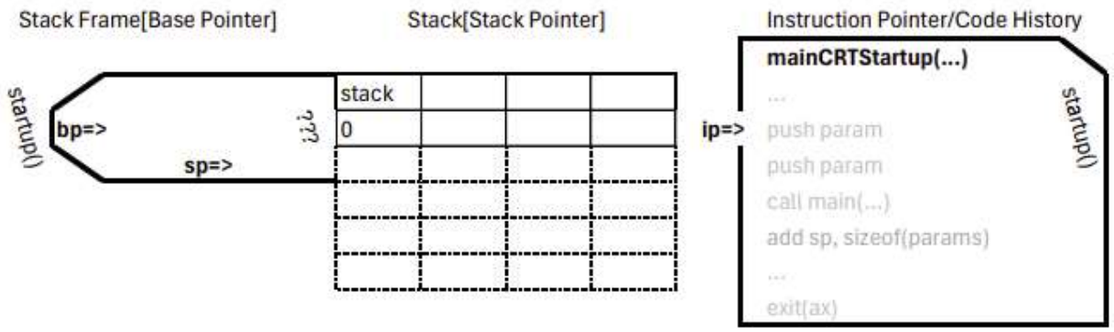
```
struct A {
    int a{};
    A()                { cout << "A::A()   this=" << this << "\n"; }
    virtual ~A()       { cout << "A::~A()  this=" << this << "\n"; }
    virtual void Bar() { cout << "A::Bar() this=" << this << "\n"; }
};
struct B {
    int b{};
    B()                { cout << "B::B()   this=" << this << "\n"; }
    virtual ~B()       { cout << "B::~B()  this=" << this << "\n"; }
    void Foo()         { cout << "B::Foo() this=" << this << "\n"; }
};
struct C : A ,B  {
    int c{};
    C()                { cout << "C::C()   this=" << this << "\n"; }
    virtual ~C()       { cout << "C::~C()  this=" << this << "\n"; }
    virtual void Bar() { cout << "C::Bar() this=" << this << "\n"; Foo(); }
};
int main() {
    cout << "\nC* pC = new C;\n";                                  C* pC = new C;
    cout << "\npA=" << pA << " pA->Bar();\n";       A* pA = pC;    pA->Bar();
    cout << "\npC=" << pC << " pC->Foo(); \n";                     pC->Foo();
    cout << "\npB=" << pB << " delete pB;\n";       B* pB = pC;    delete pB;
}
```

godbolt.org/z/77WGcq4er

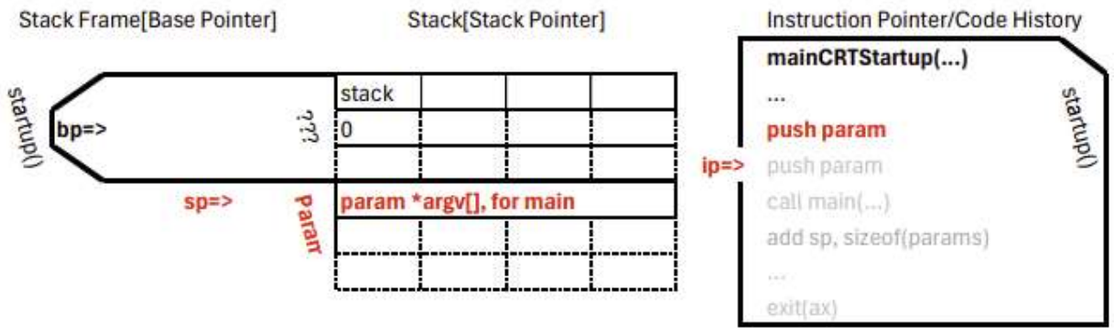
```
C* pC = new C;
A::A()   this=009E85C8
B::B()   this=009E85D0
C::C()   this=009E85C8

pA=009E85C8 pA->Bar();
C::Bar() this=009E85C8
B::Foo() this=009E85D0

pC=009E85C8 pC->Foo();
B::Foo() this=009E85D0

pB=009E85D0 delete pB;
C::~C()  this=009E85C8
B::~B()  this=009E85D0
A::~A()  this=009E85C8
```

C++ Under the Hood – Stacks of Frames

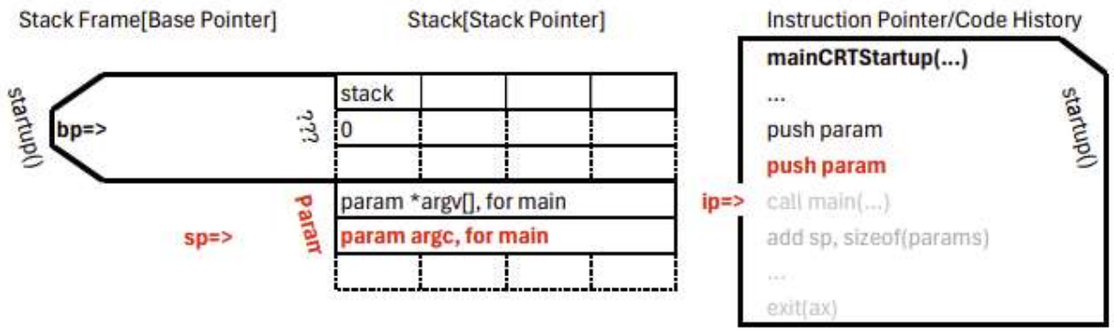
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



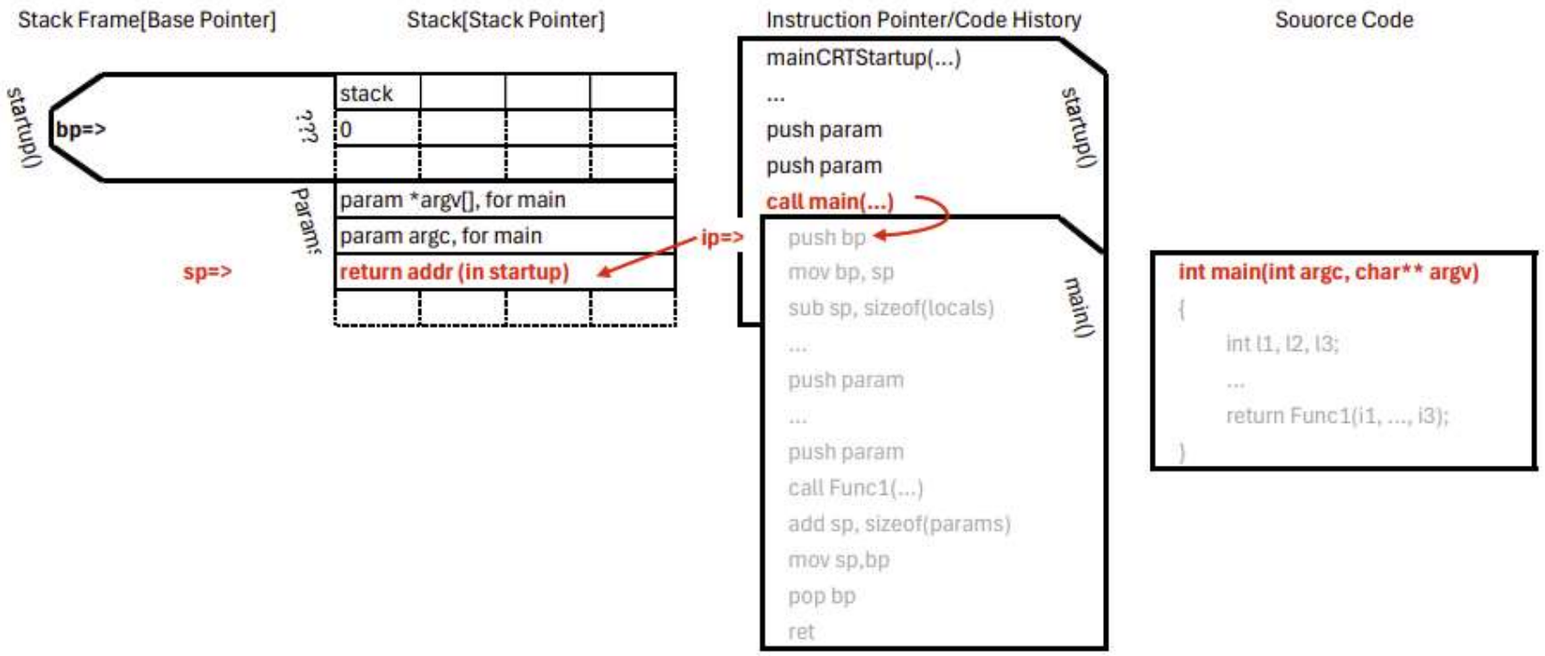
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



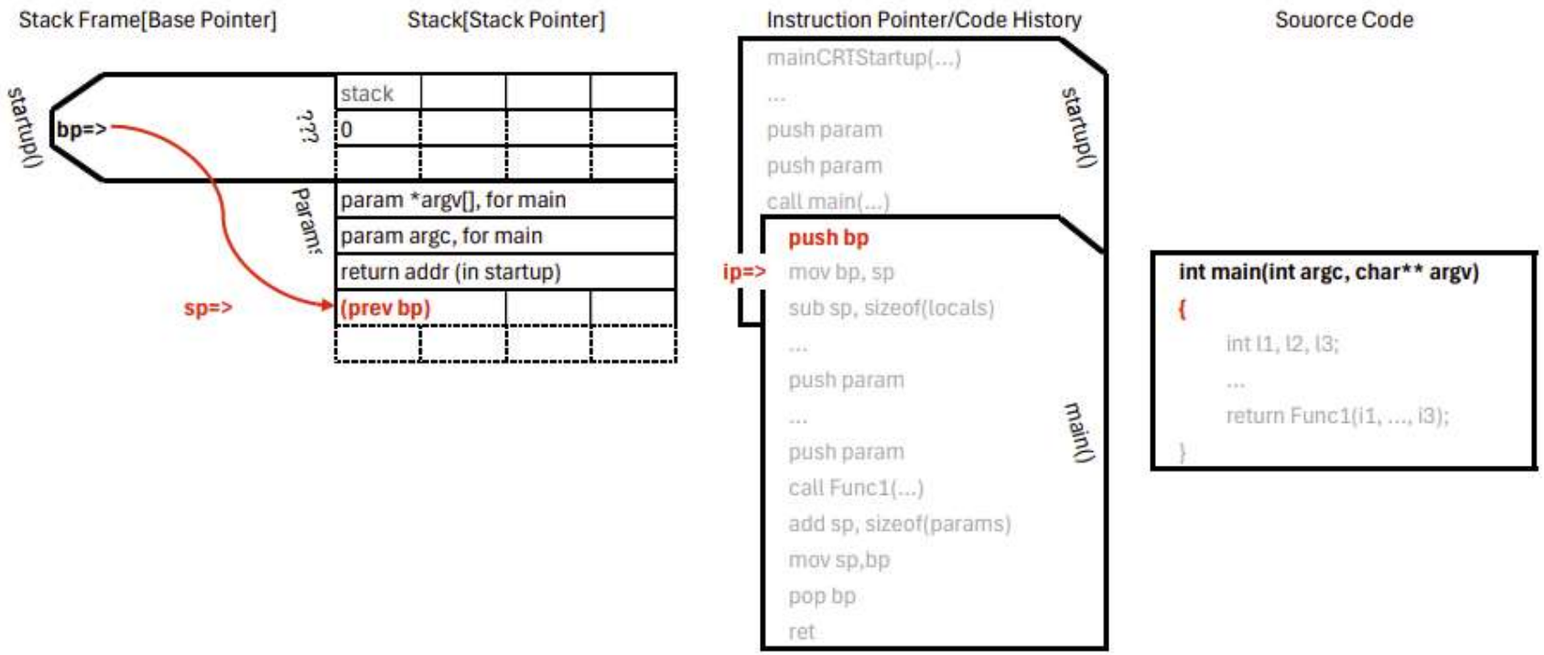
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



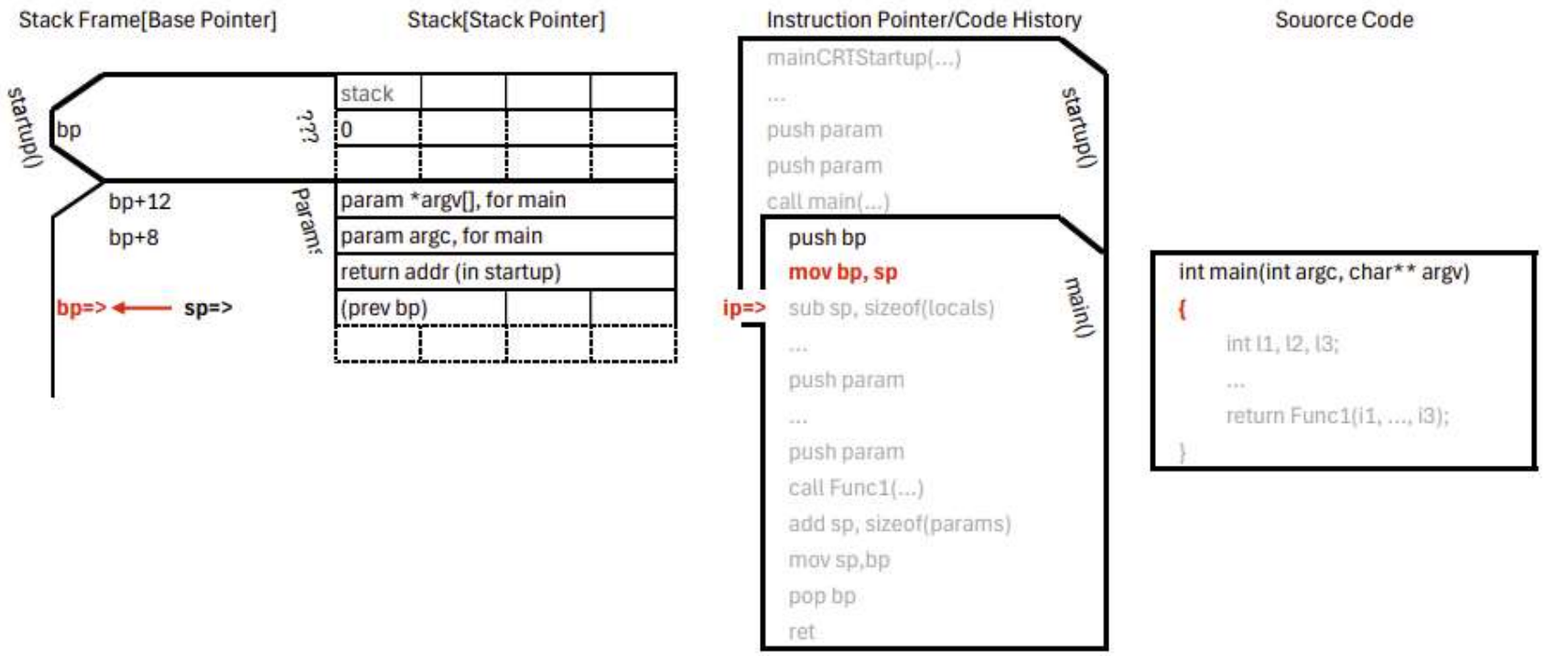
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



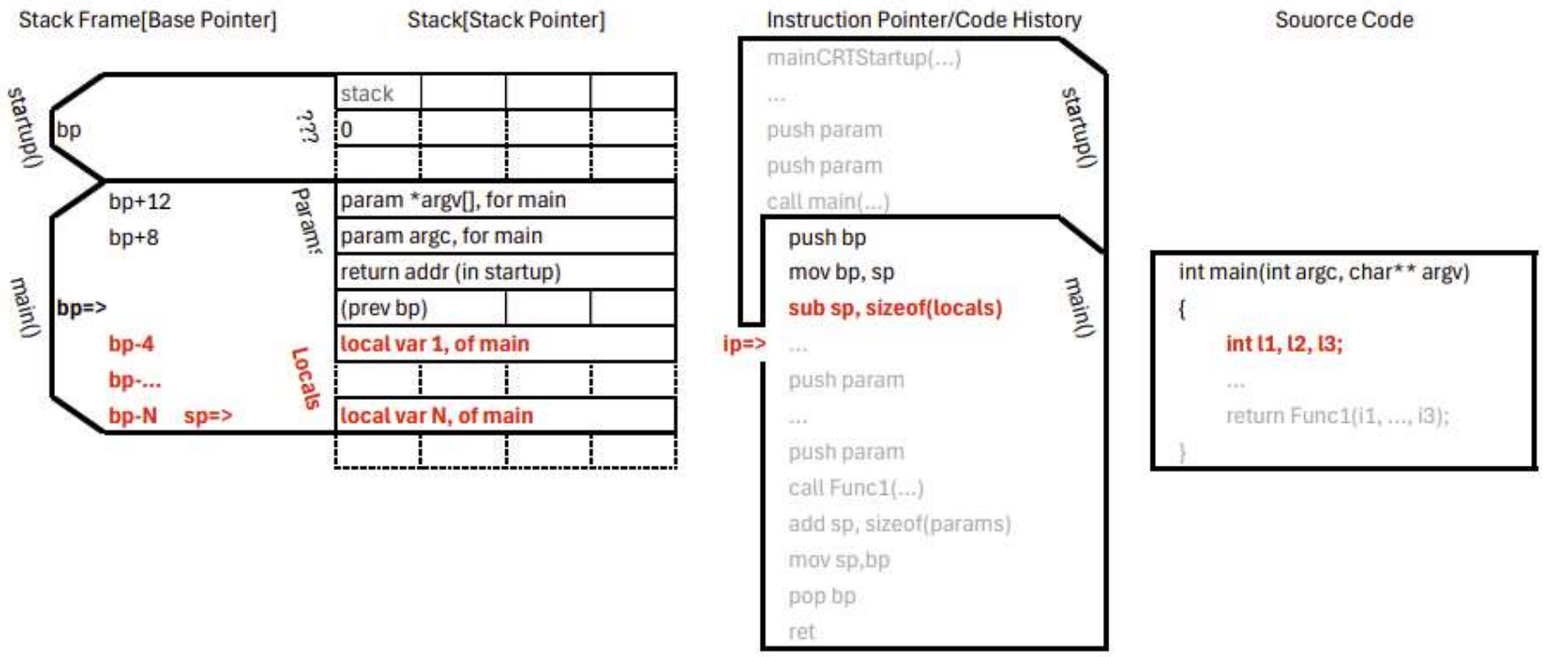
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



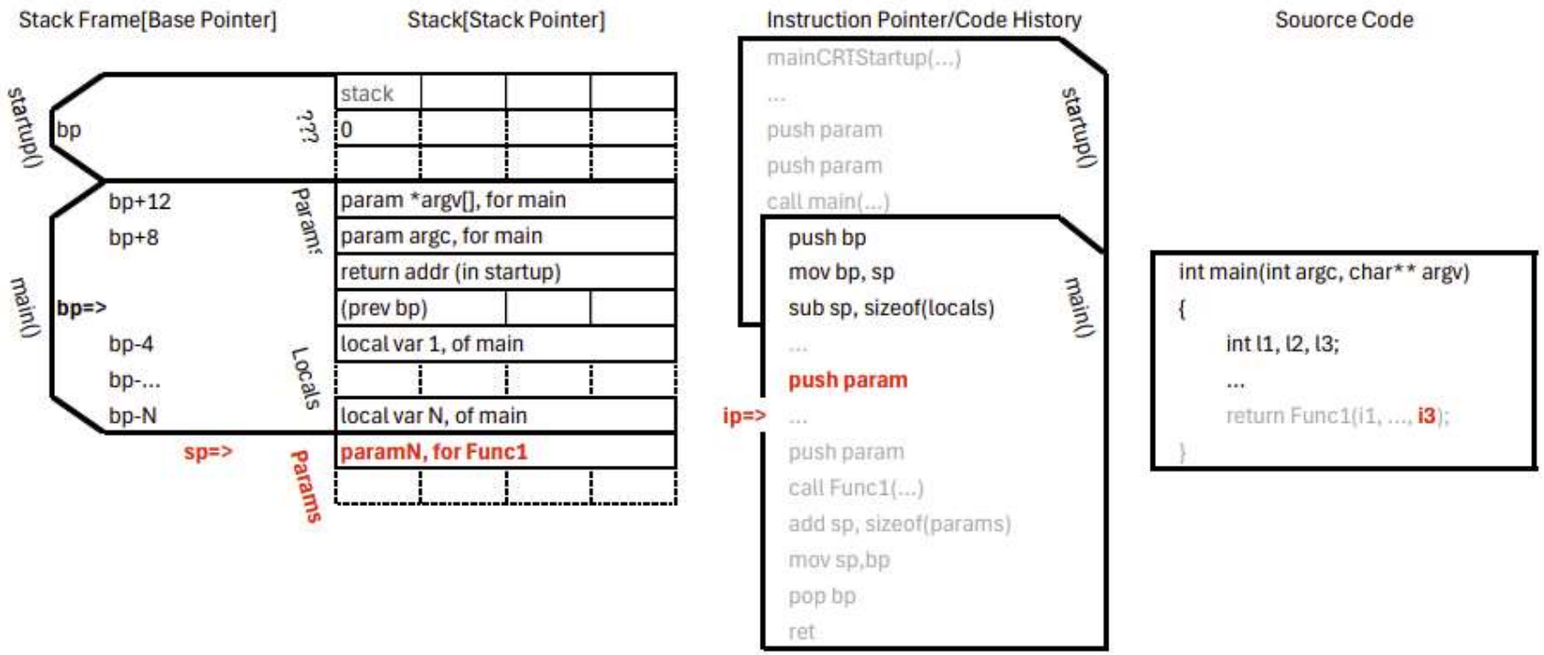
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



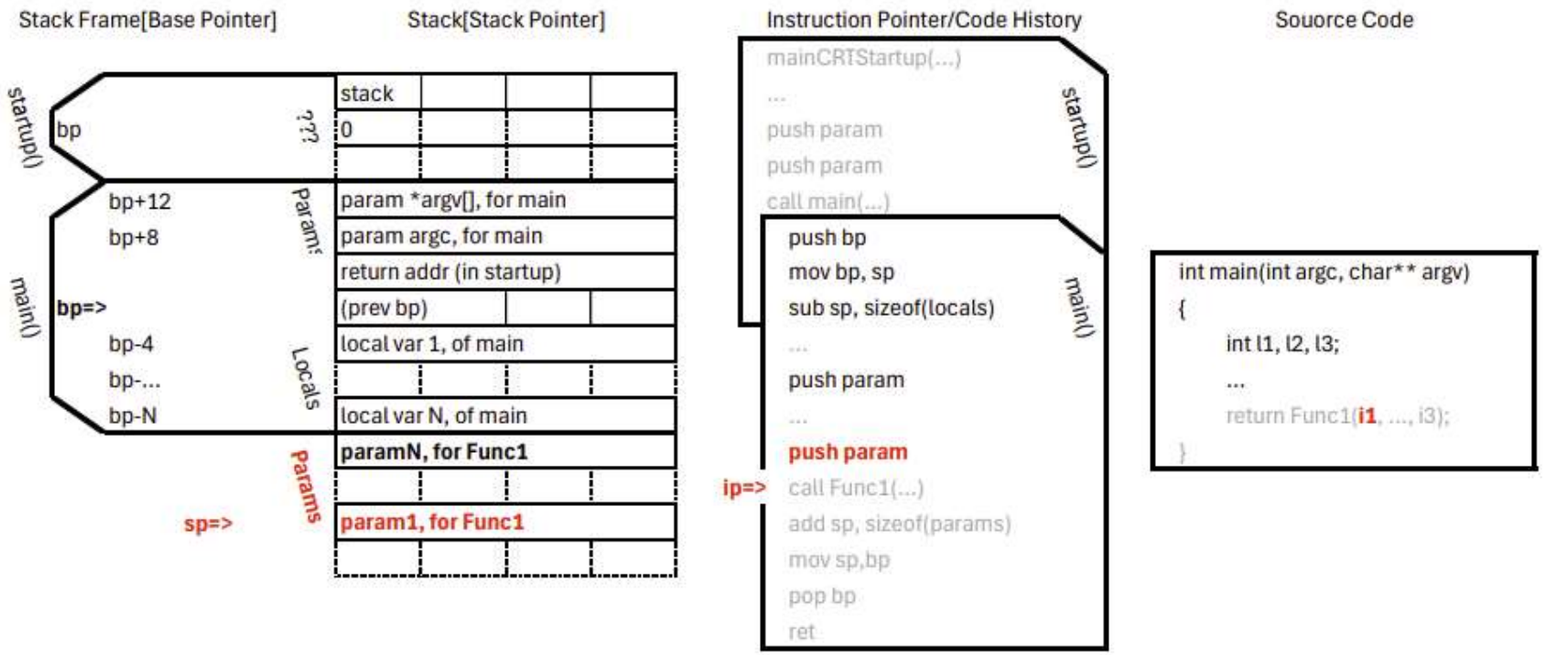
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



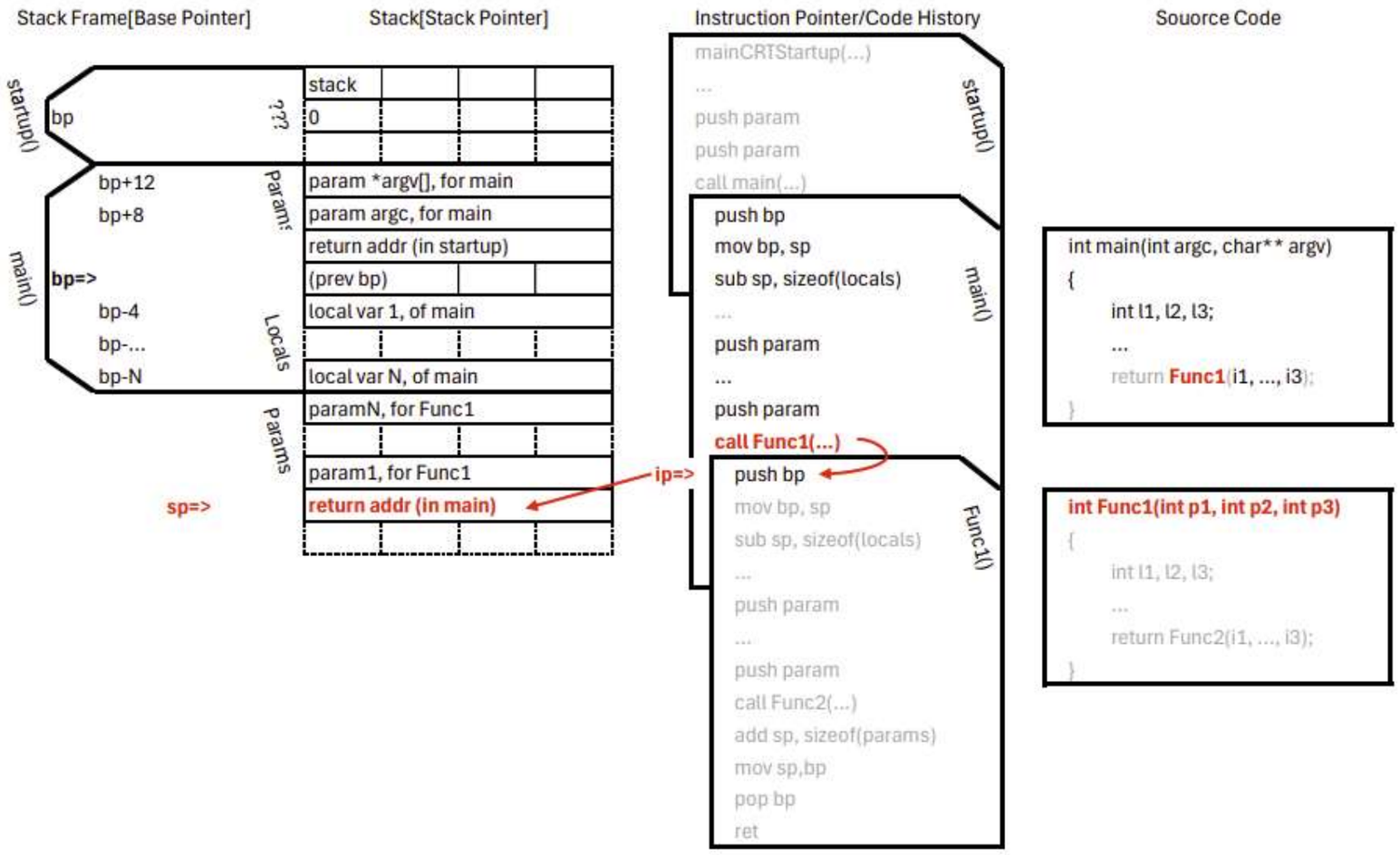
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



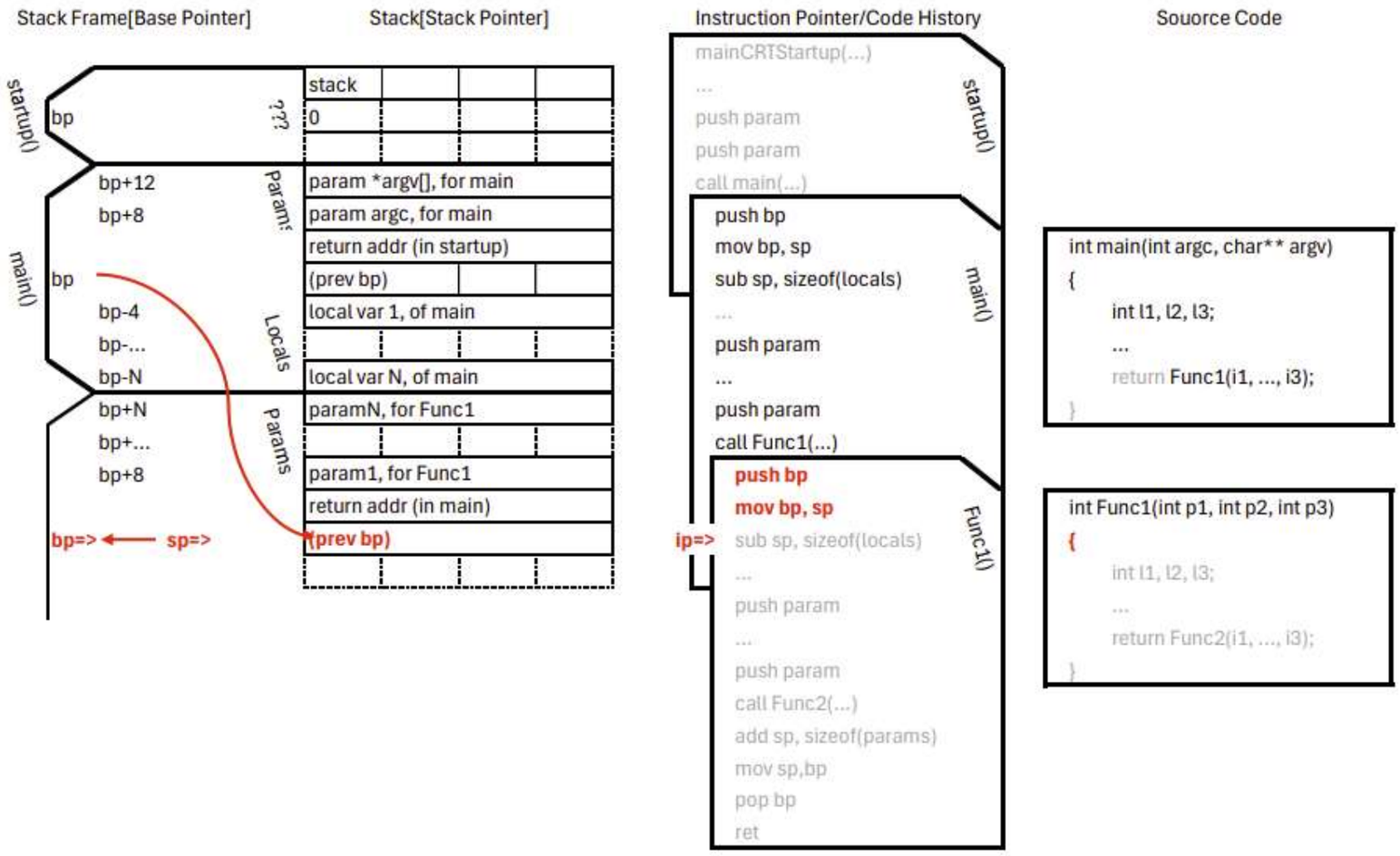
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



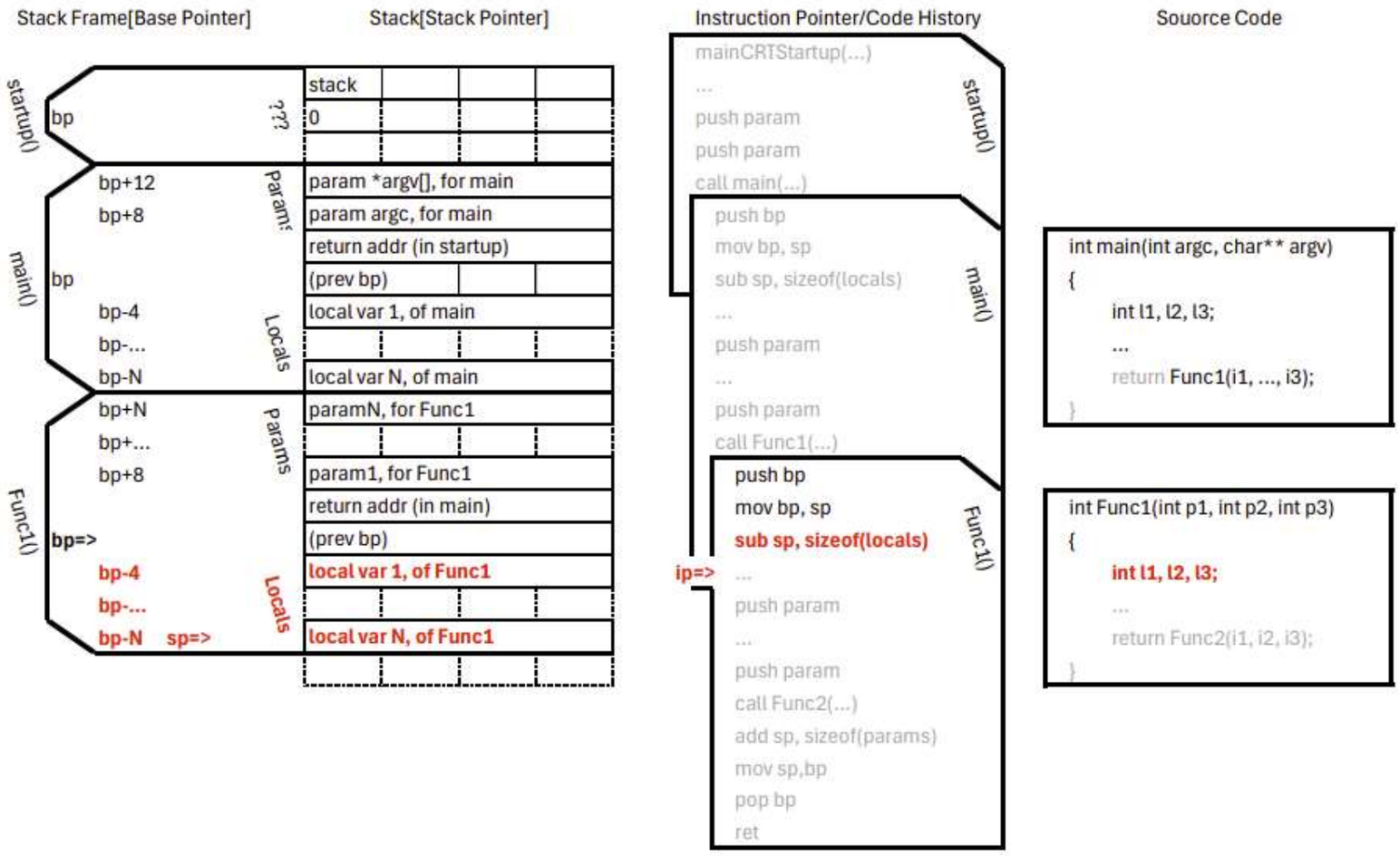
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



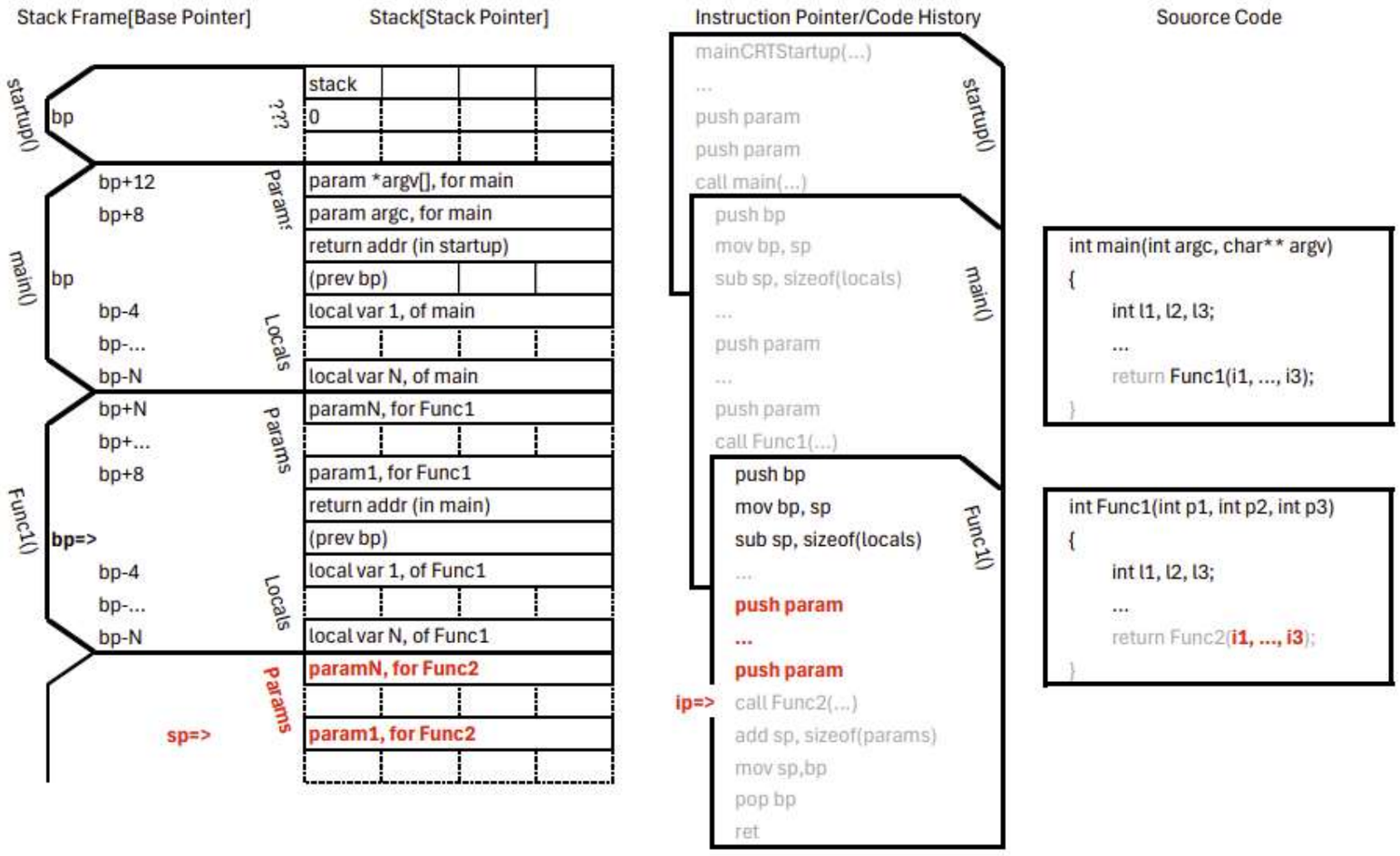
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



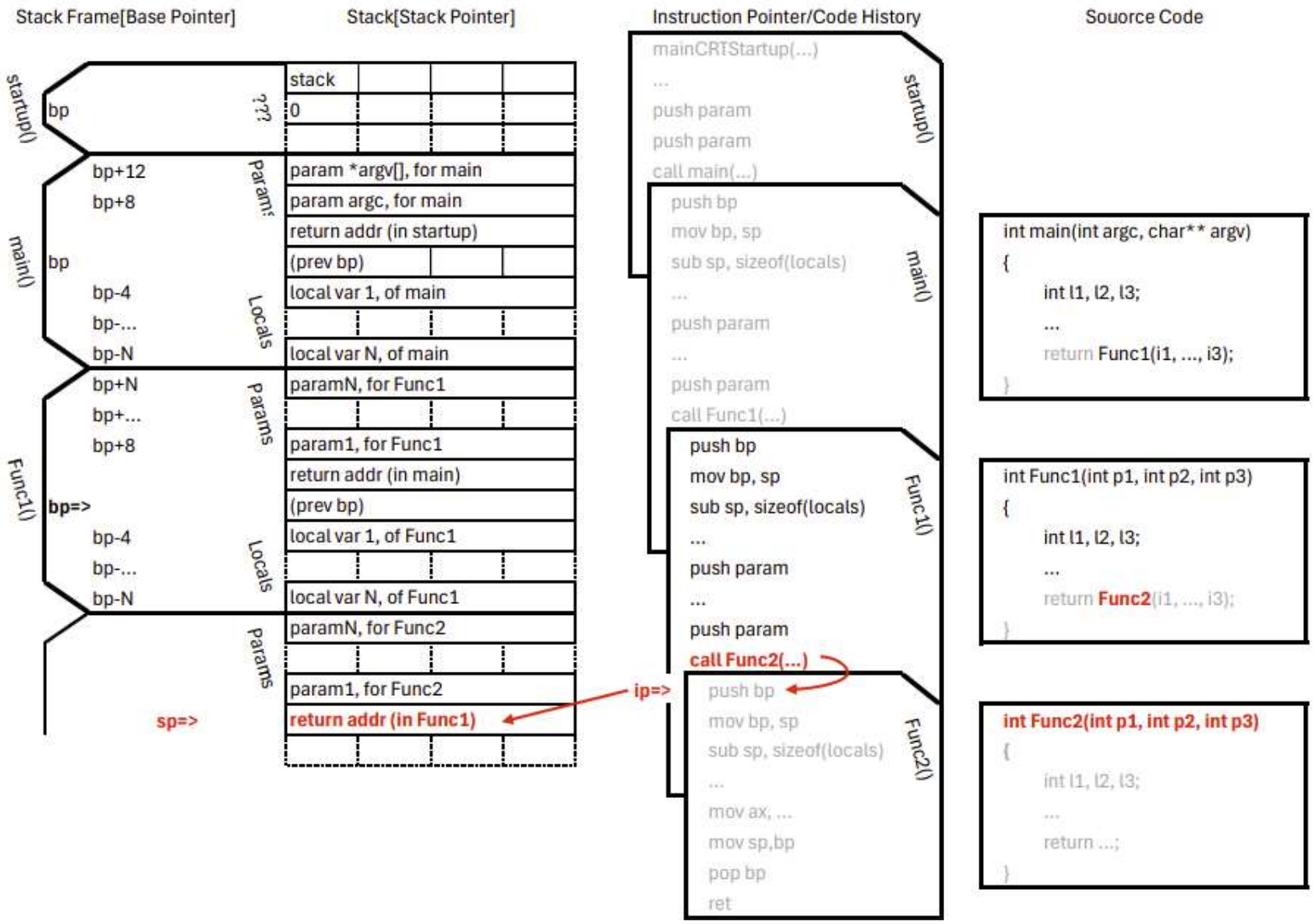
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



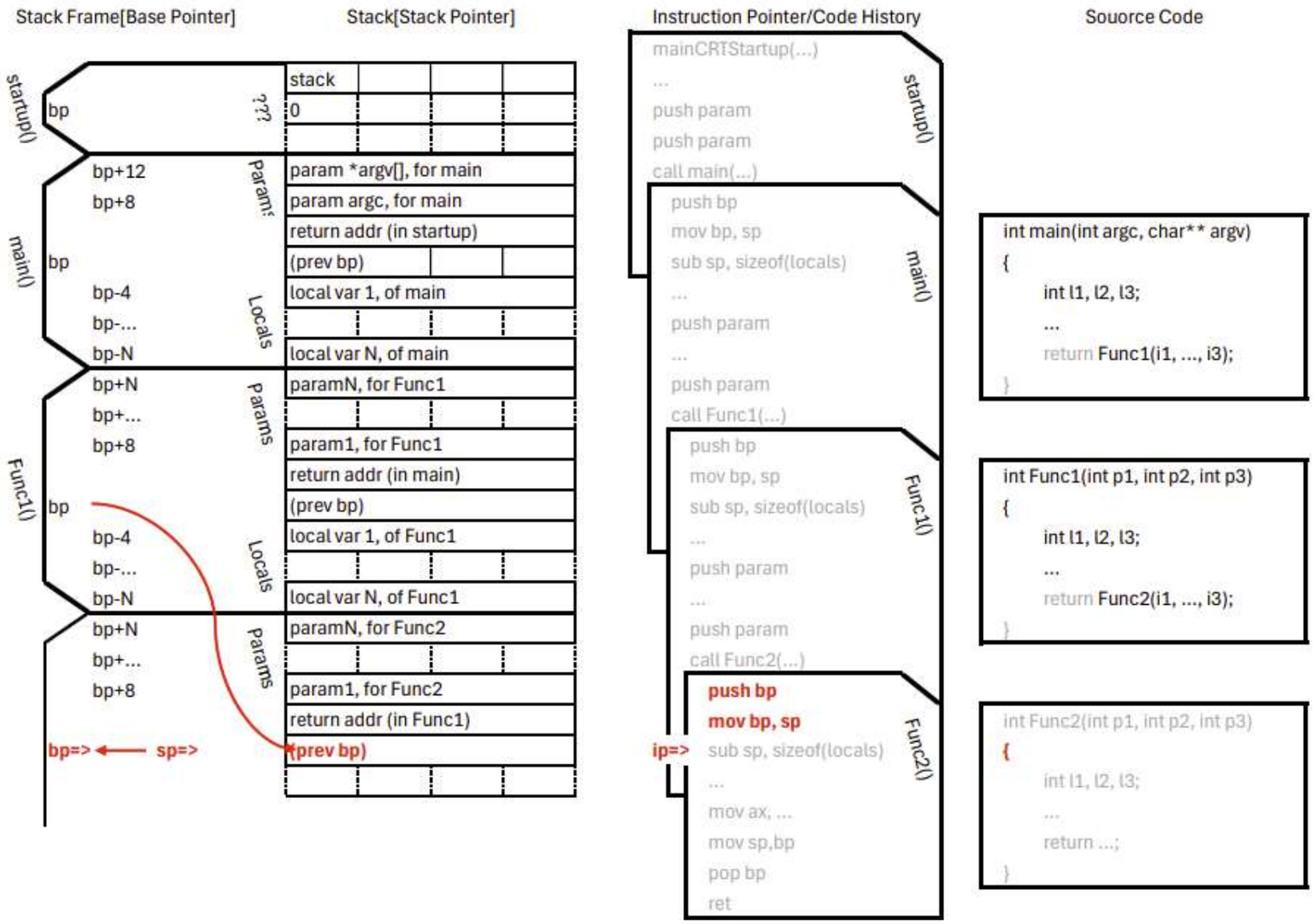
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



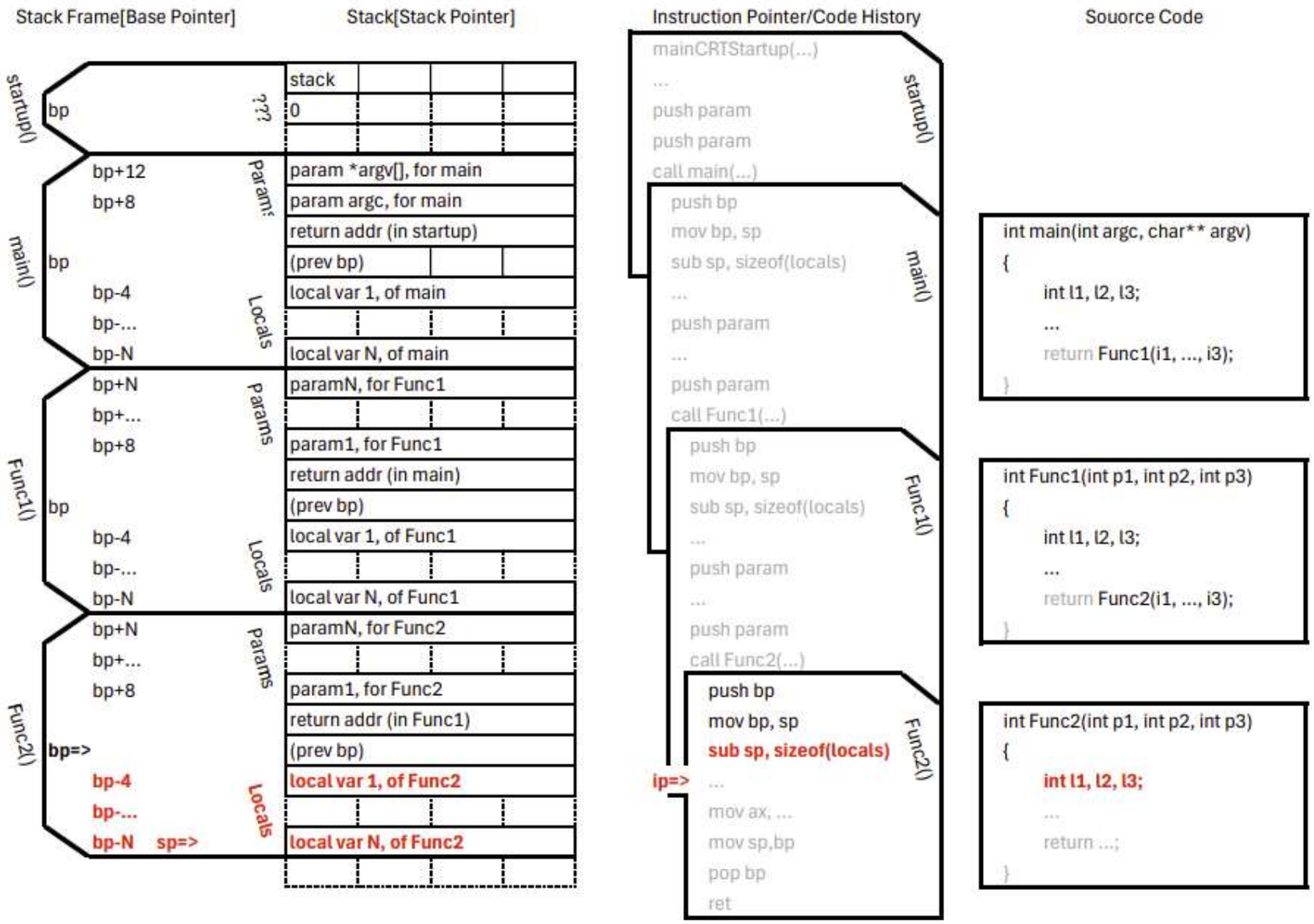
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



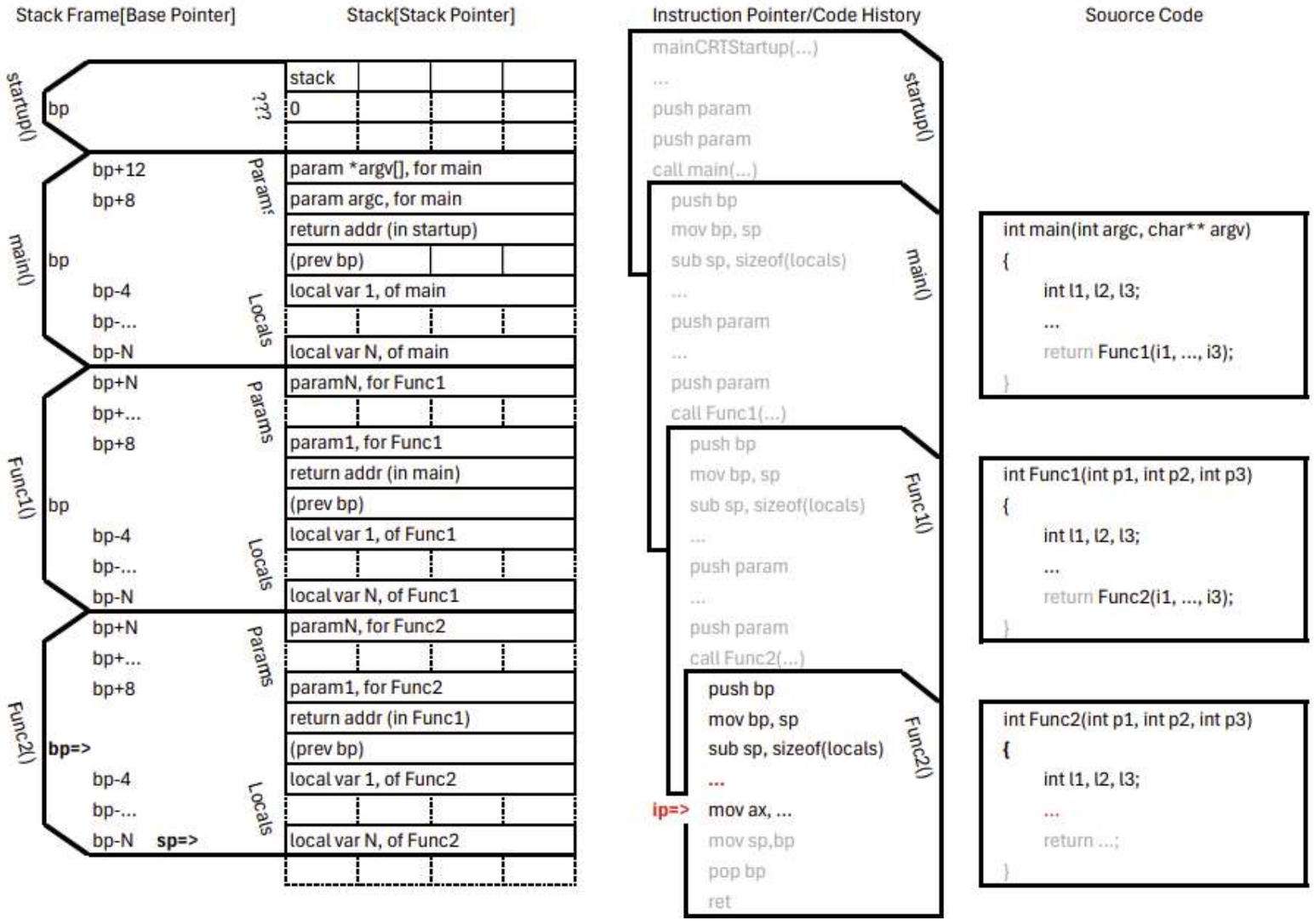
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



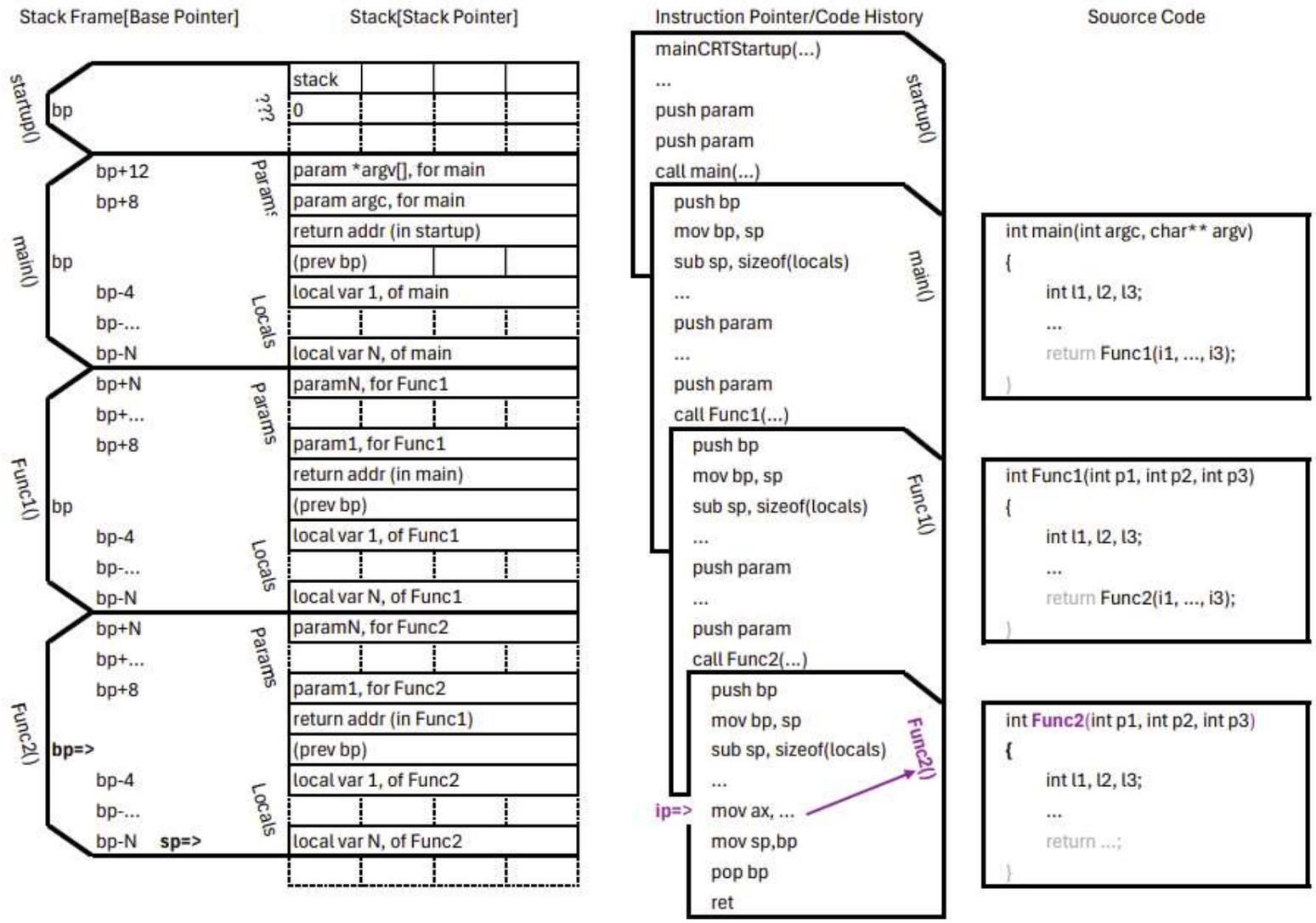
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



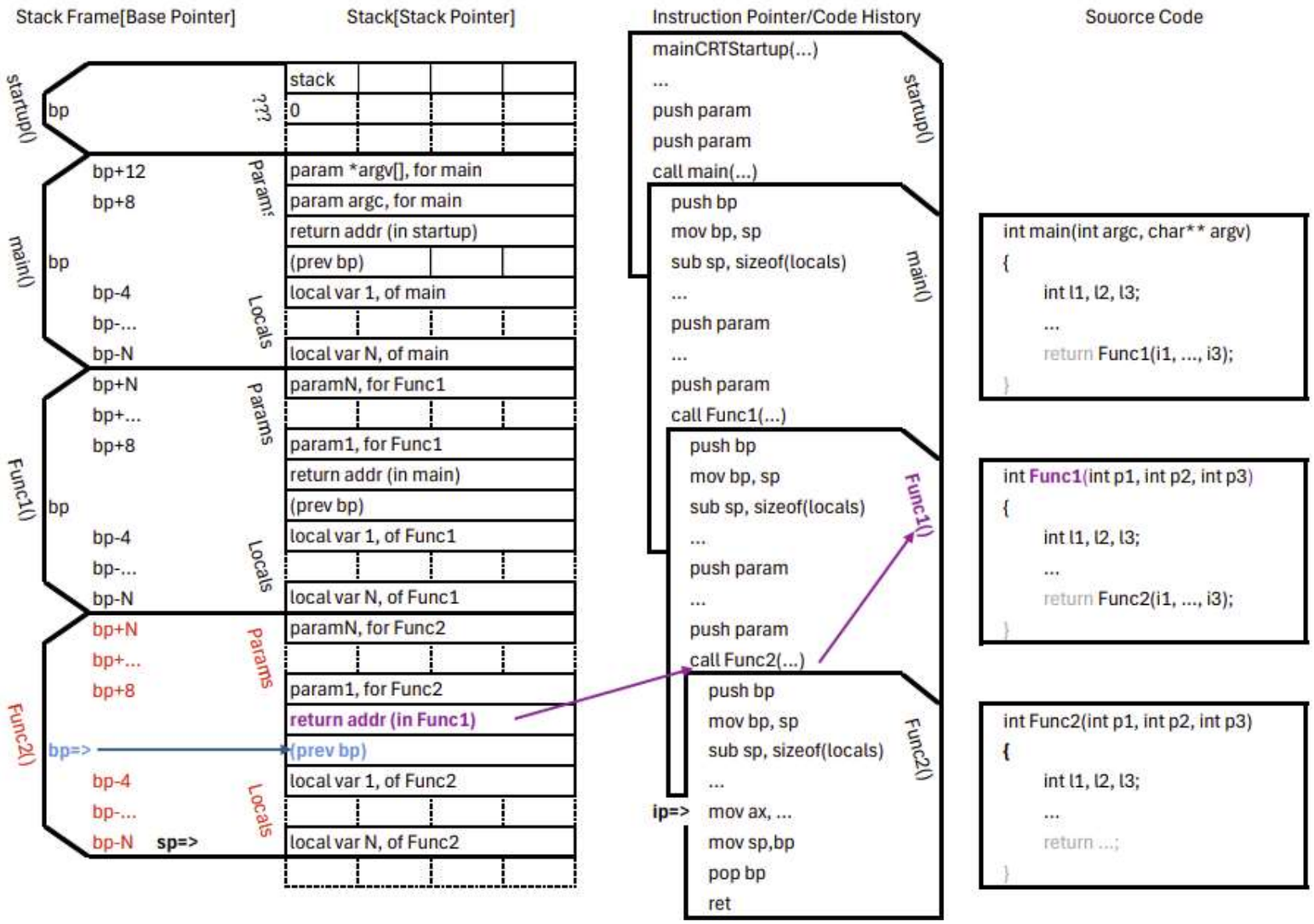
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



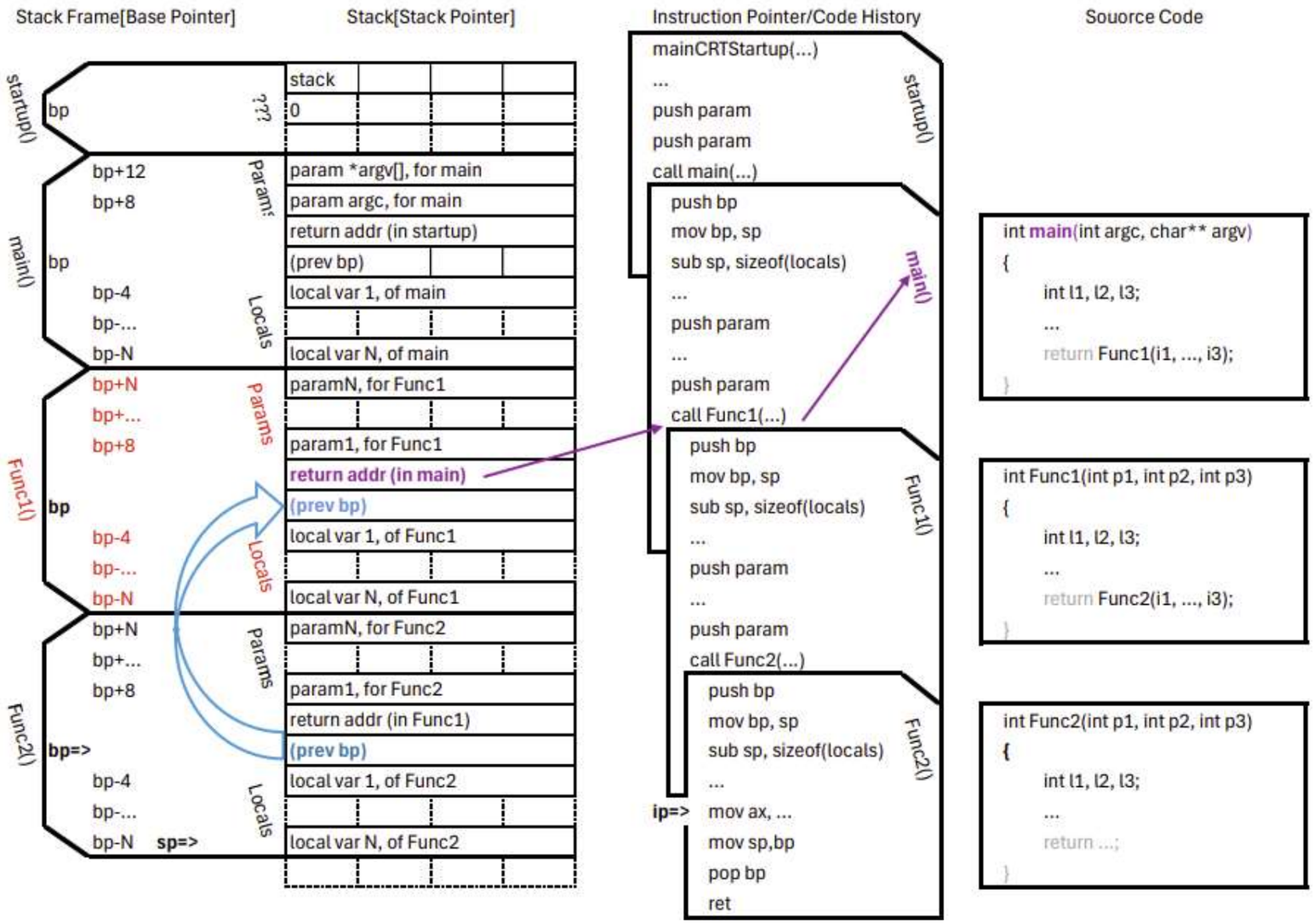
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



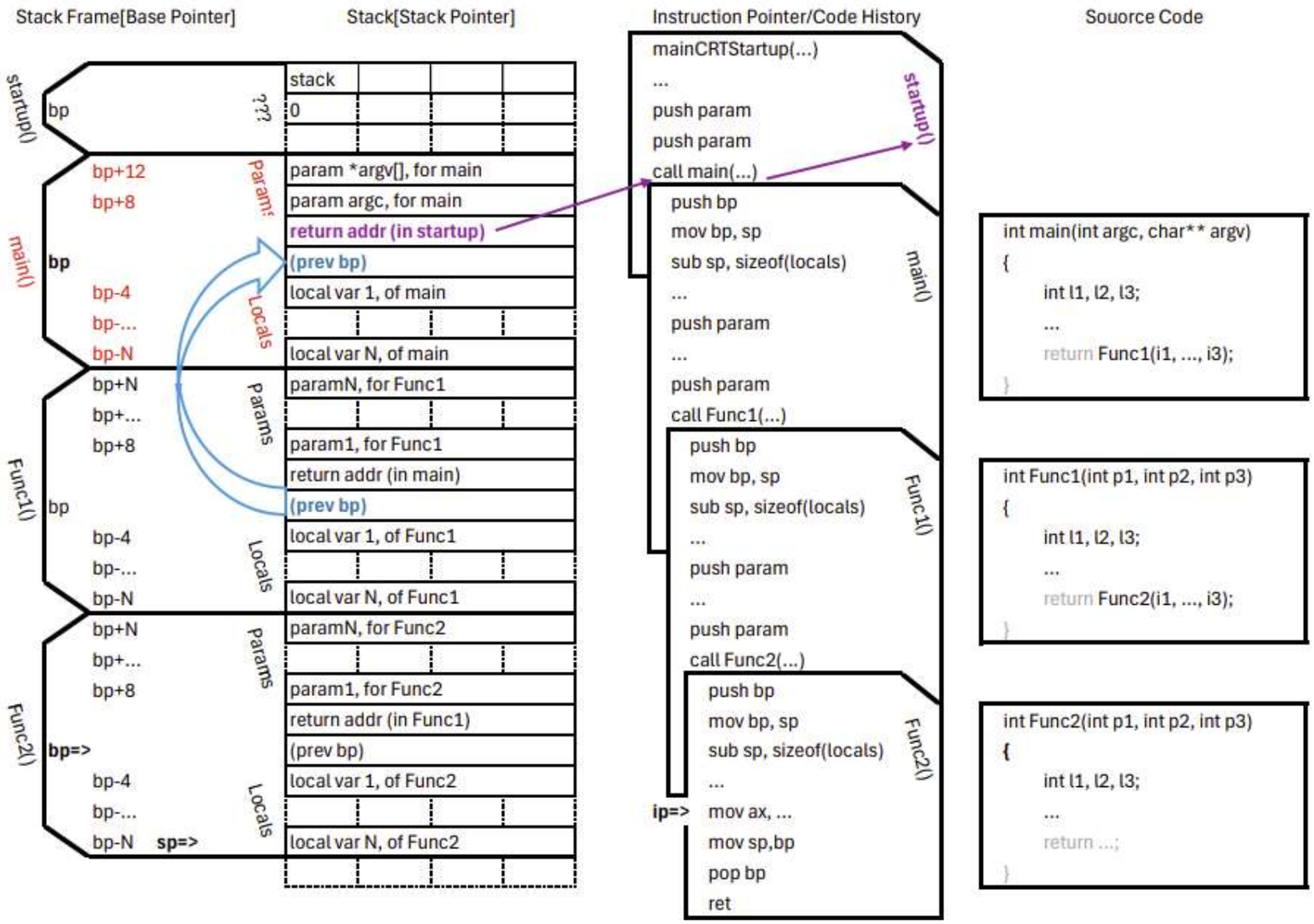
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



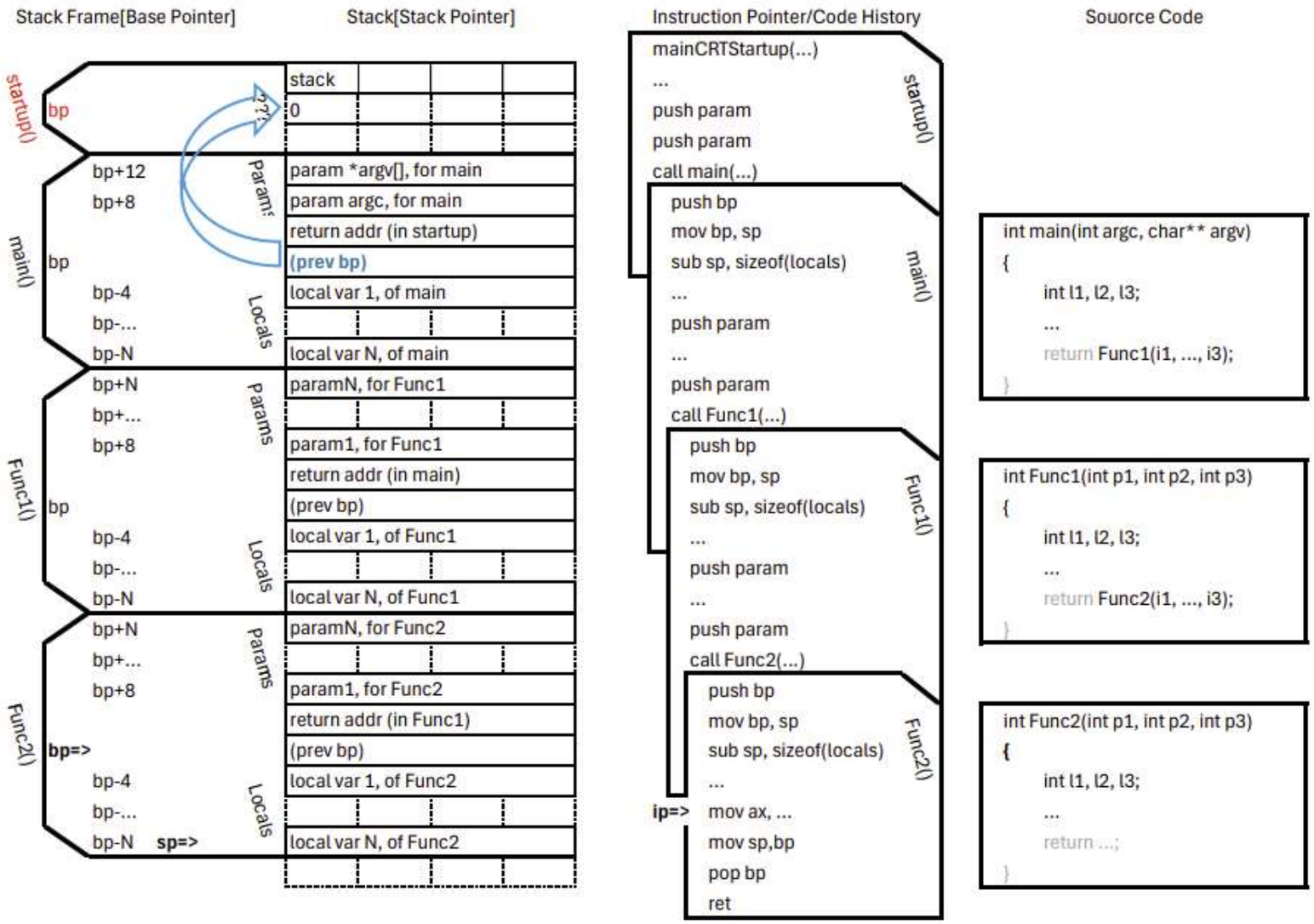
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



C++ Under the Hood – Stack Frame / Base Pointer Mechanics

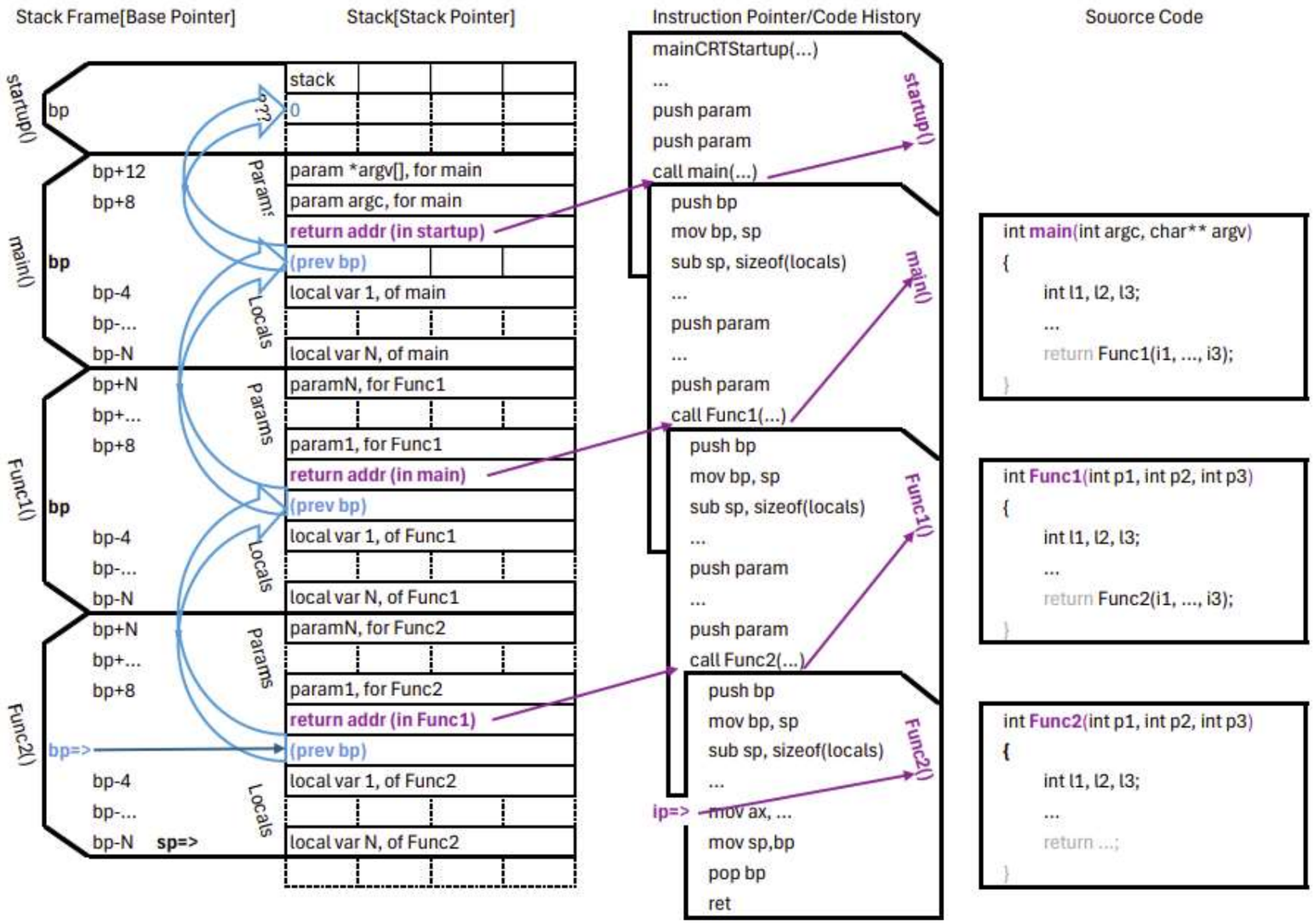


C++ Under the Hood – Stack Frame / Base Pointer Mechanics

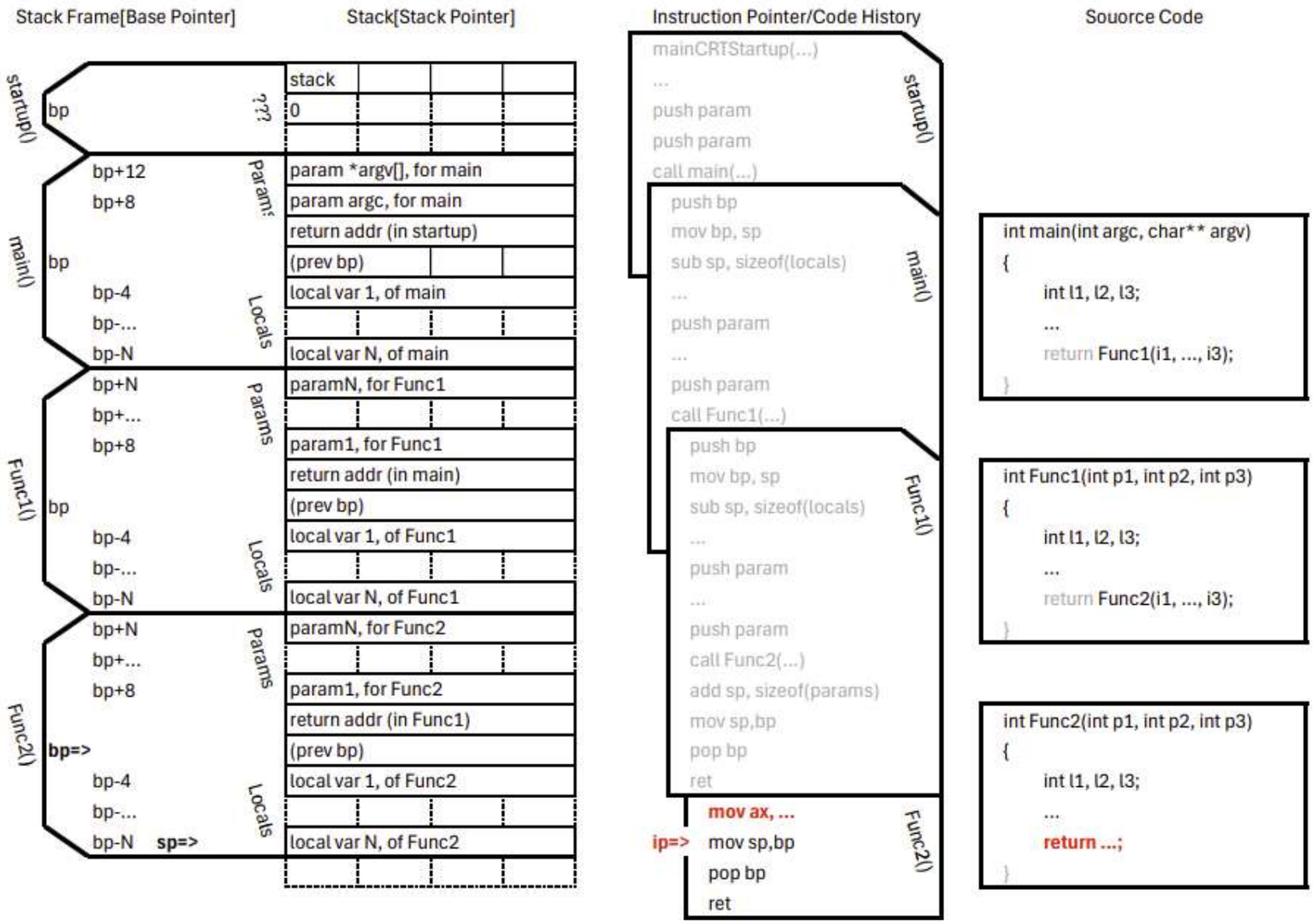


142

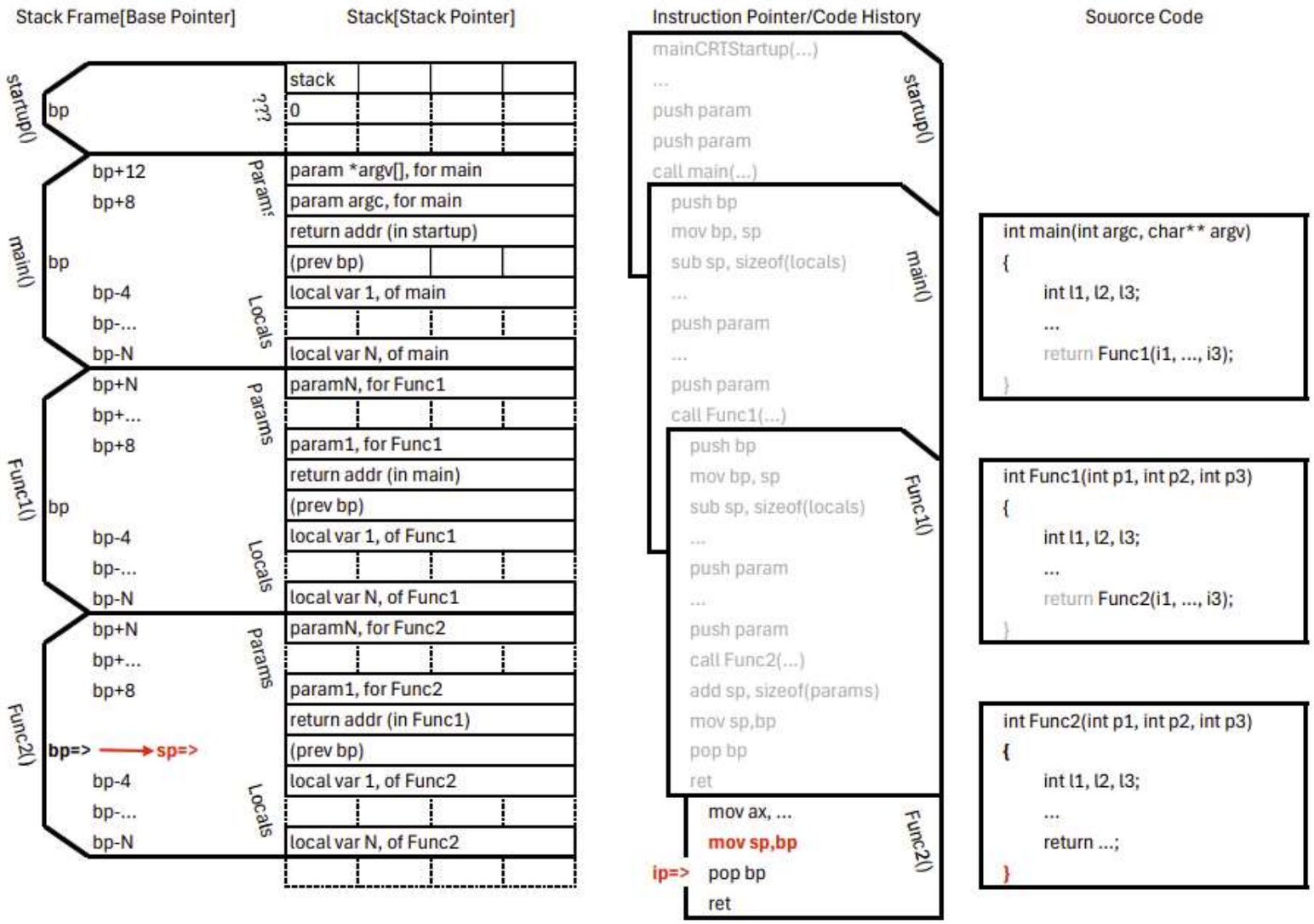
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



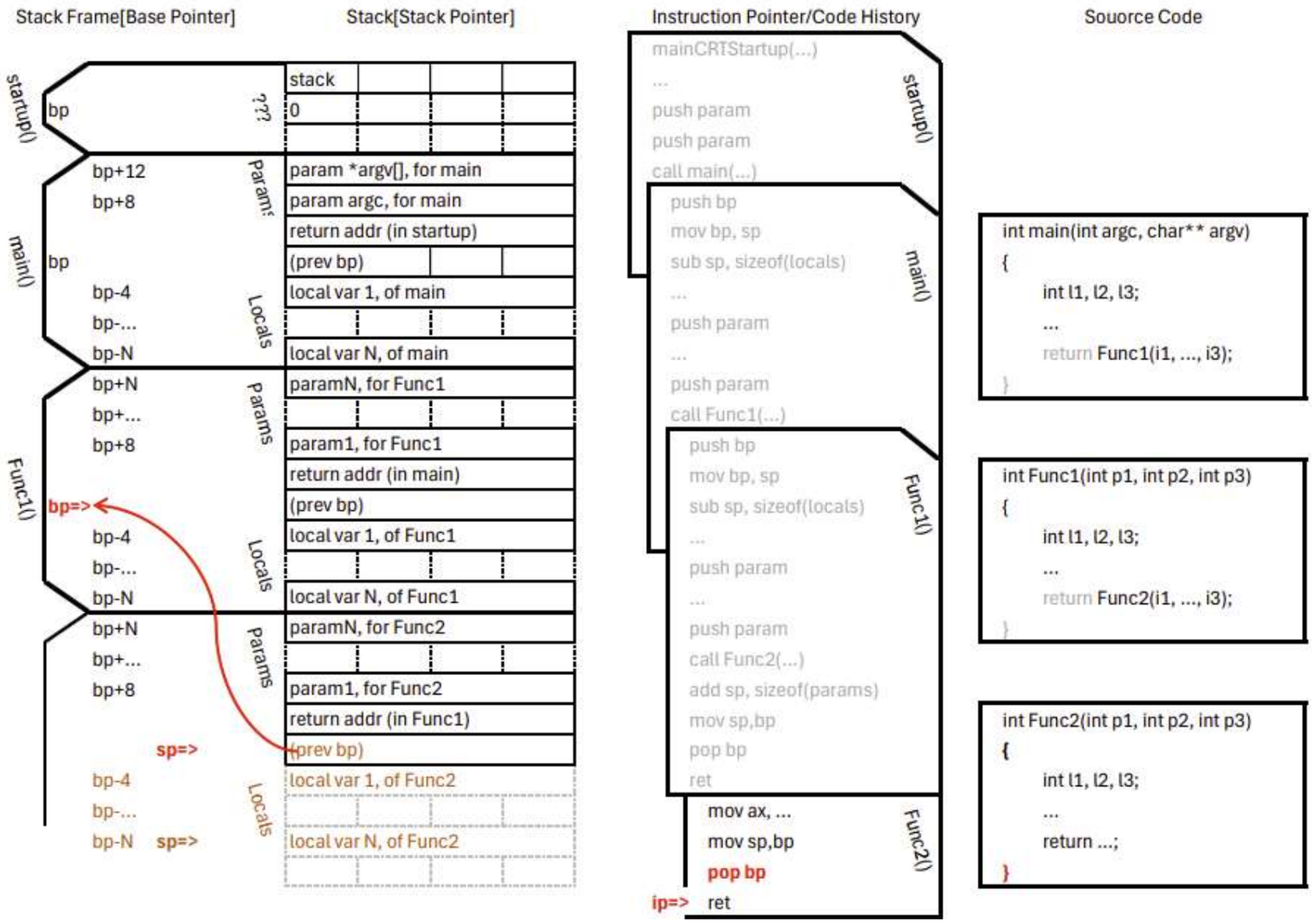
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



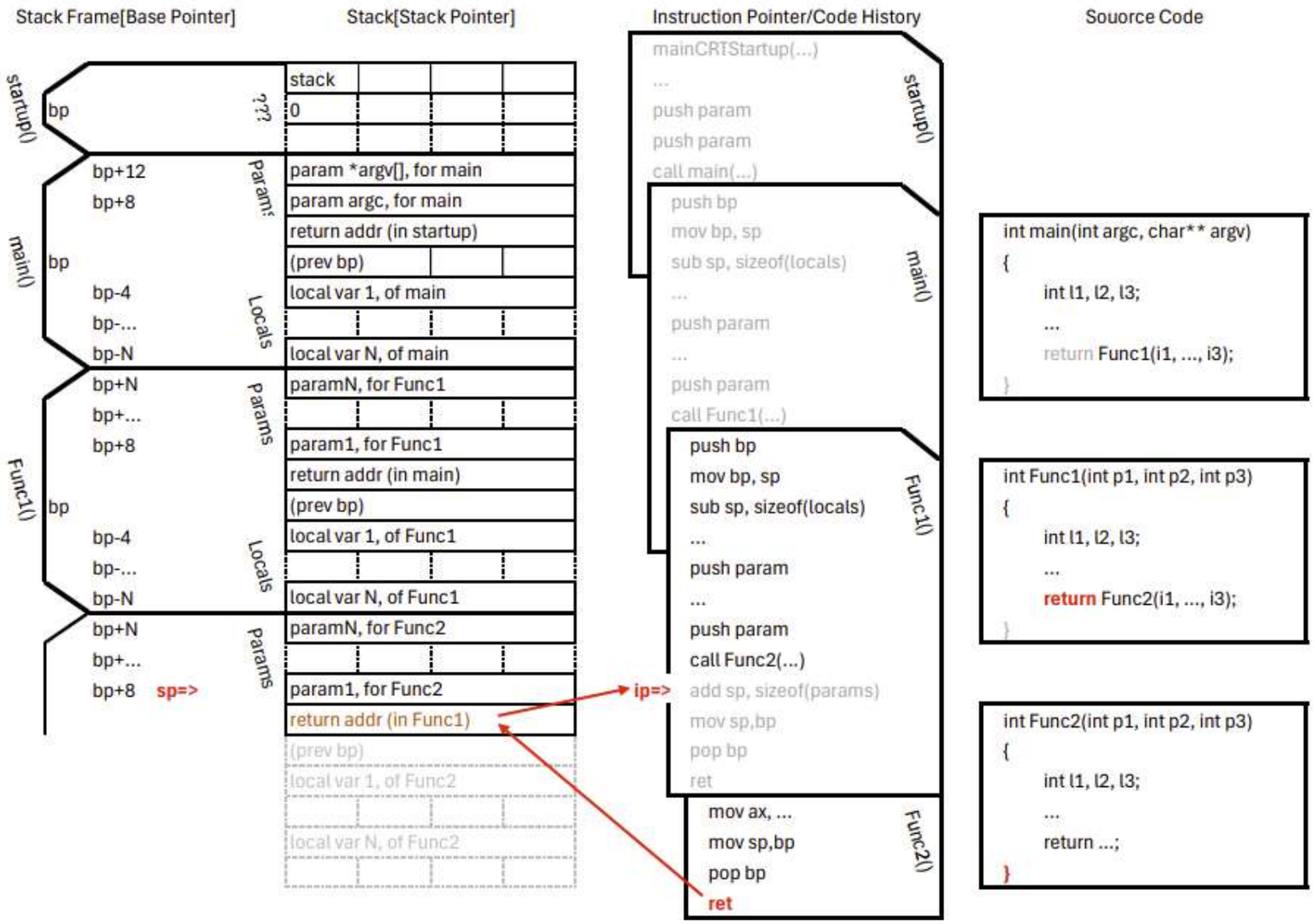
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



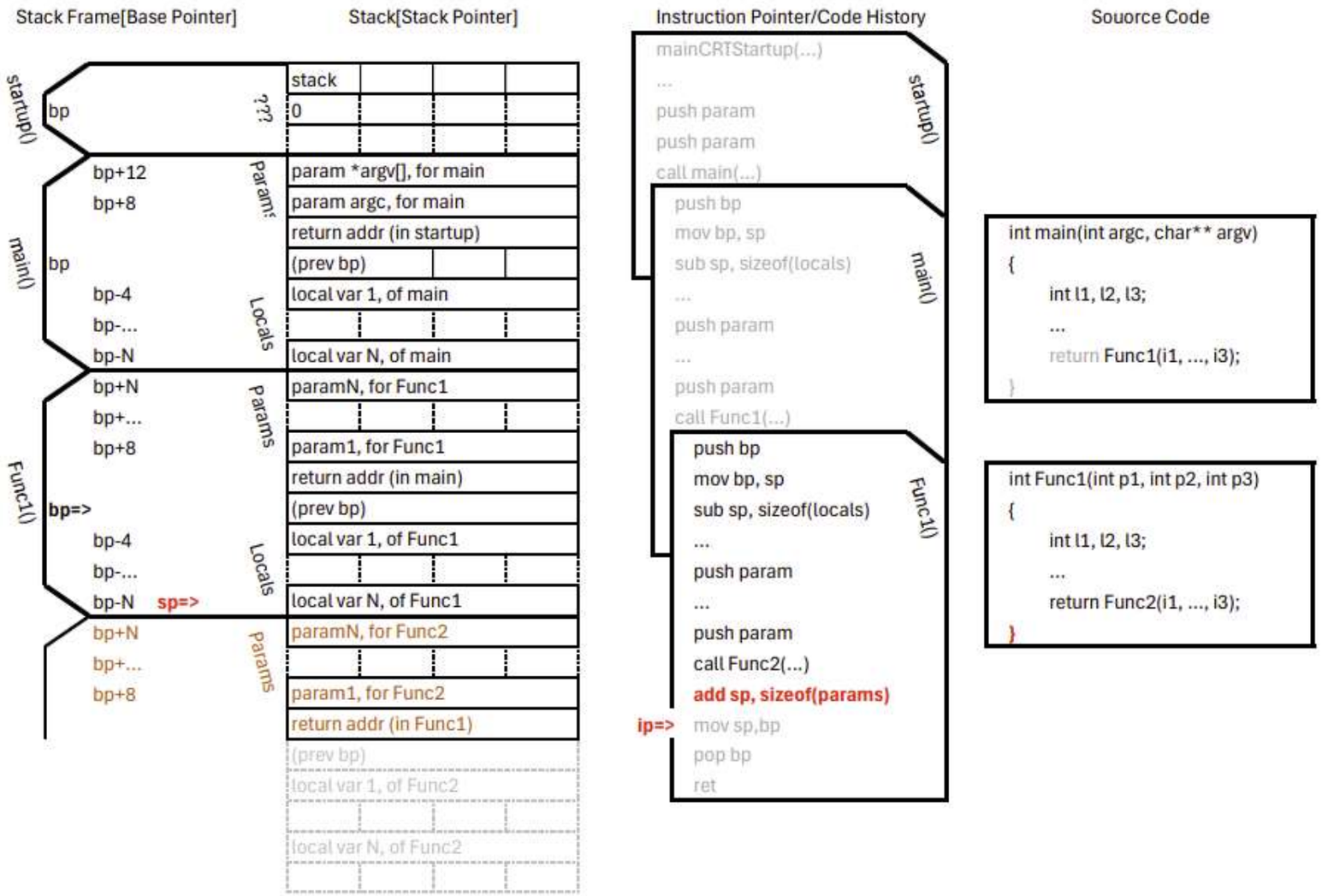
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



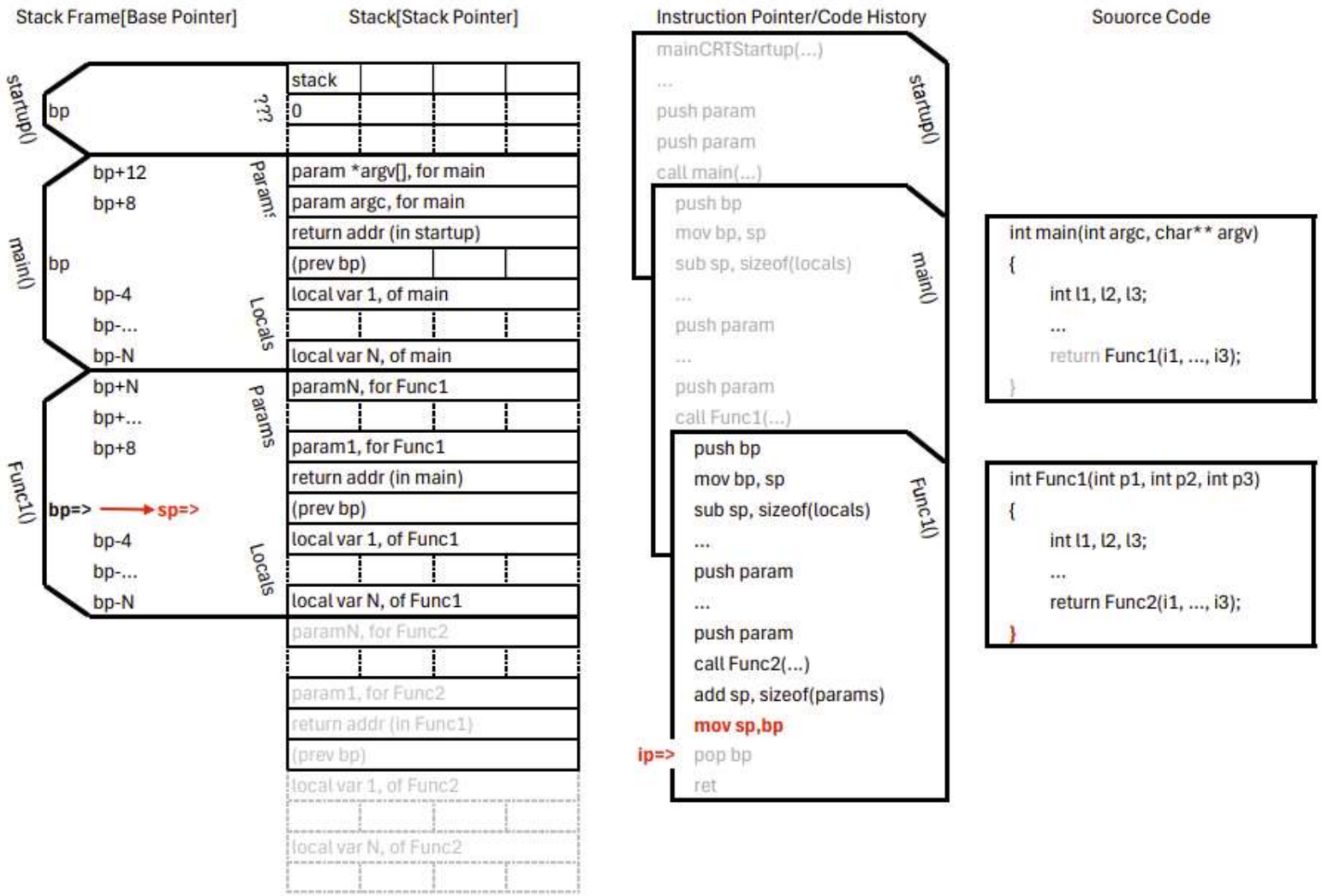
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



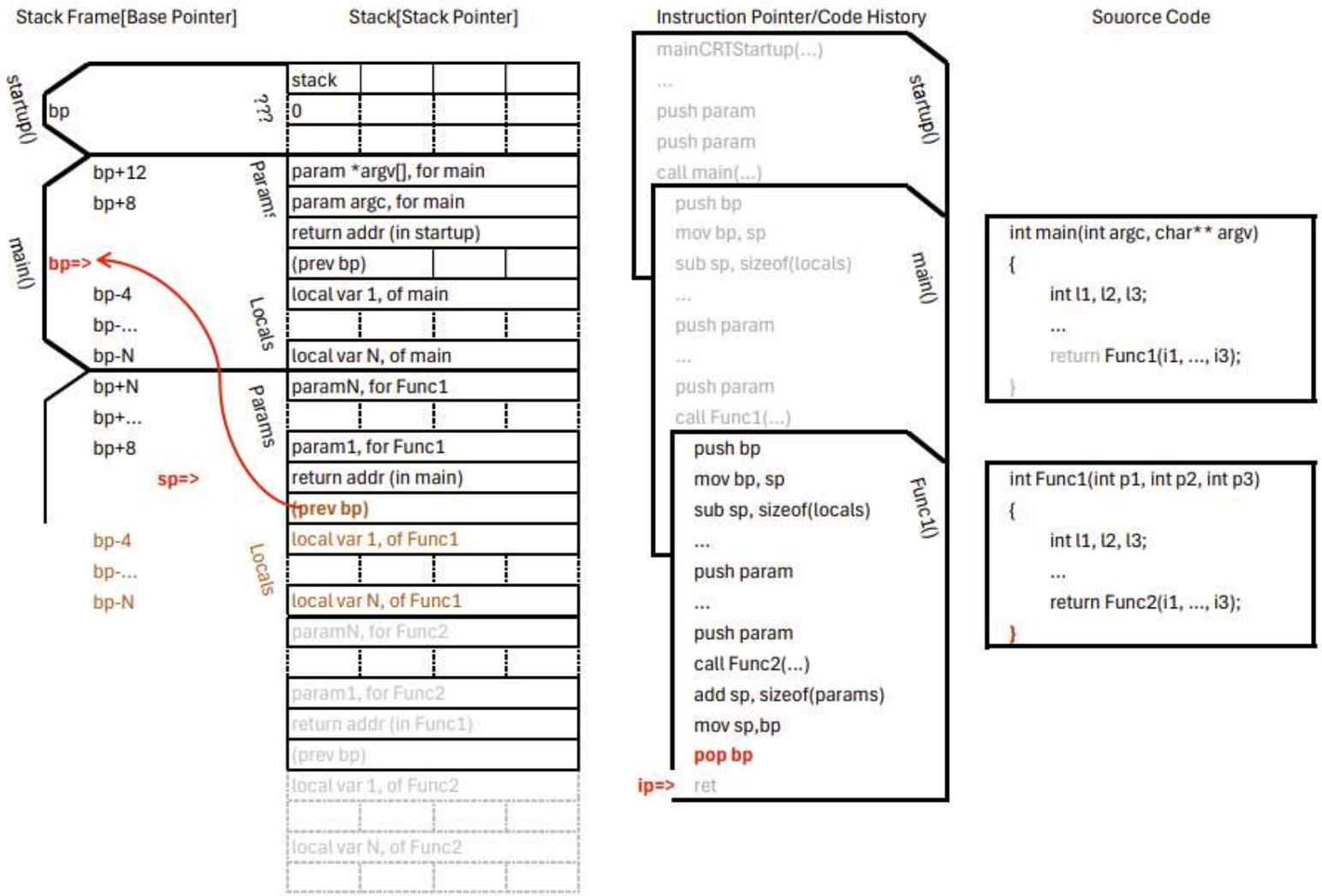
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



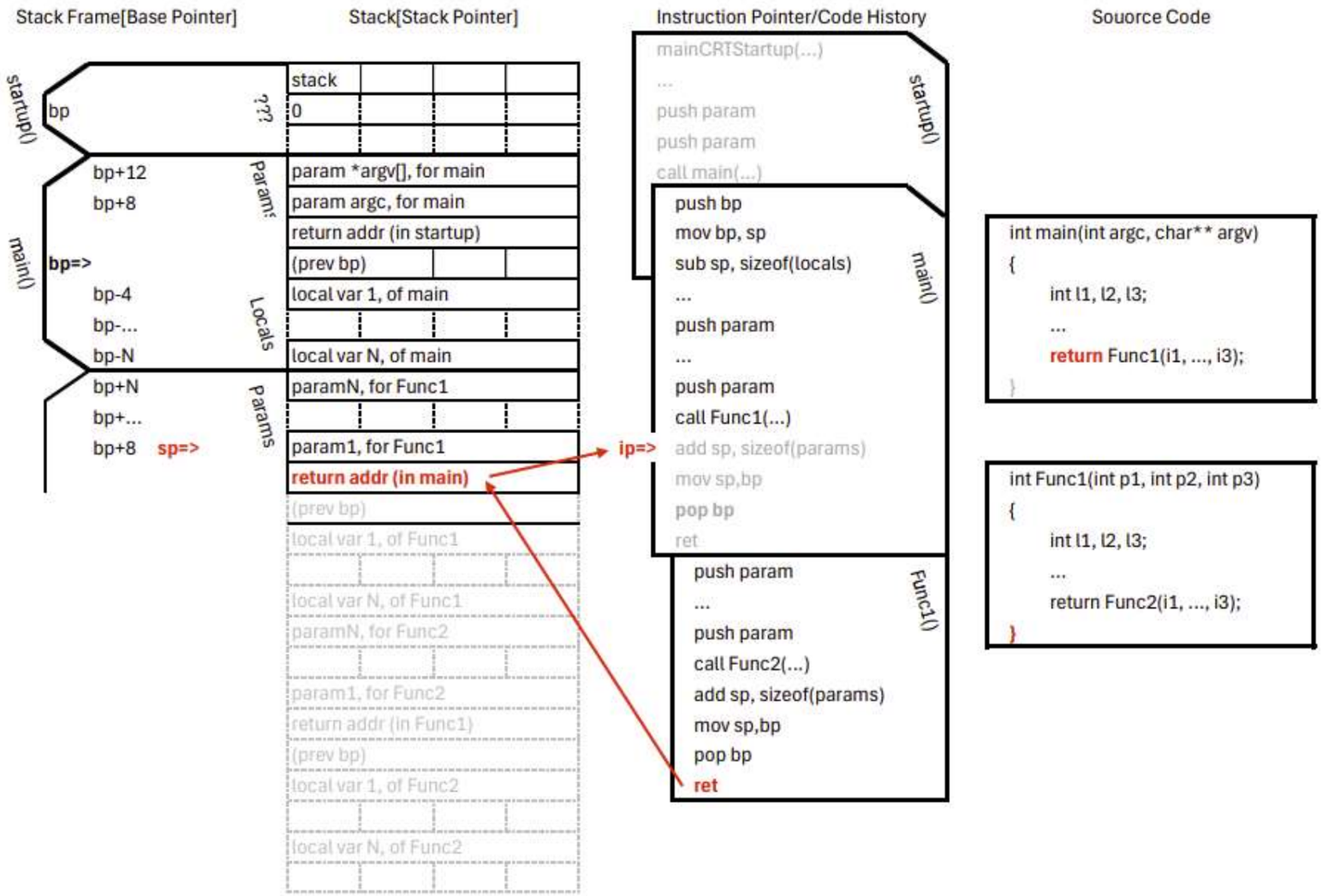
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



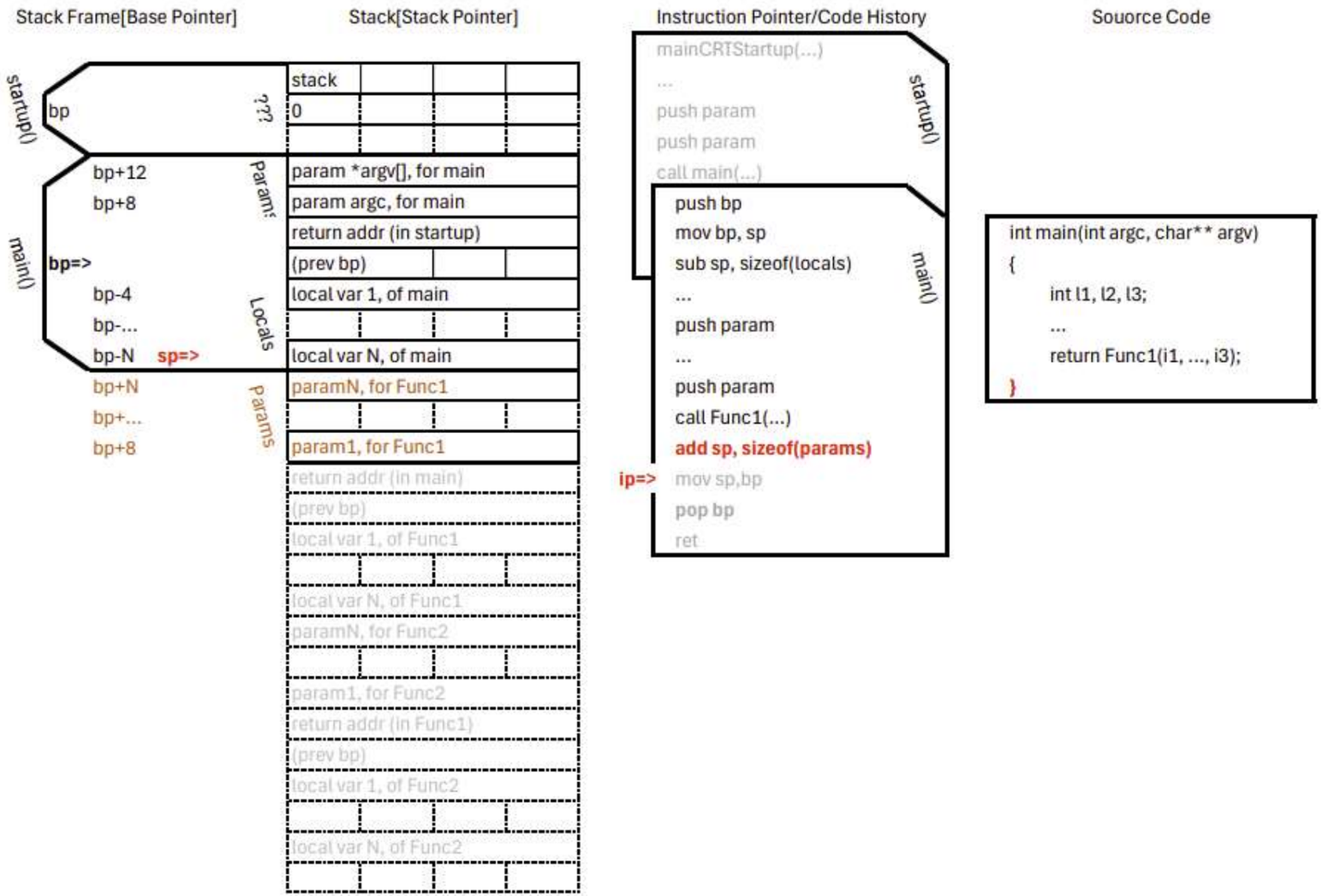
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



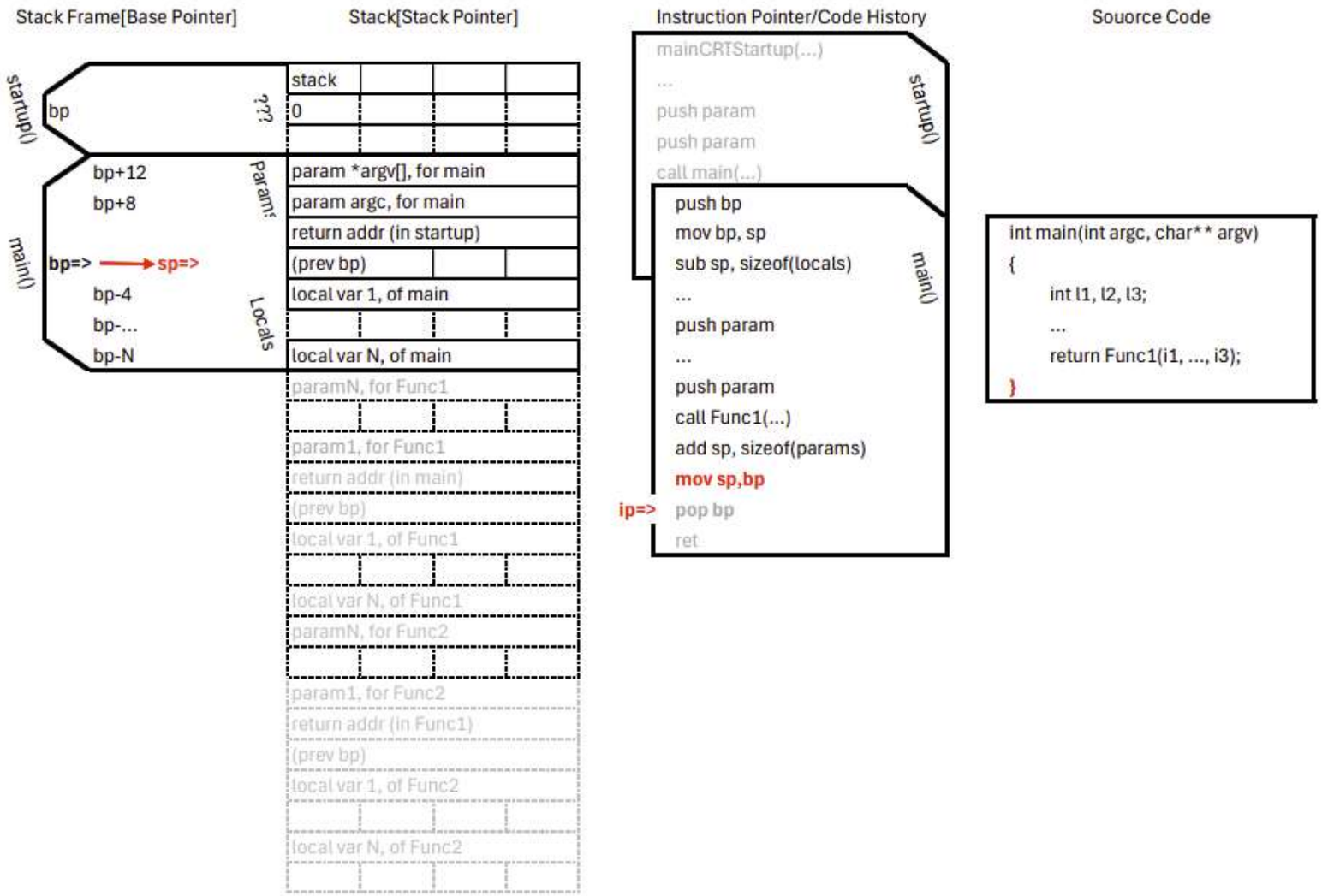
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



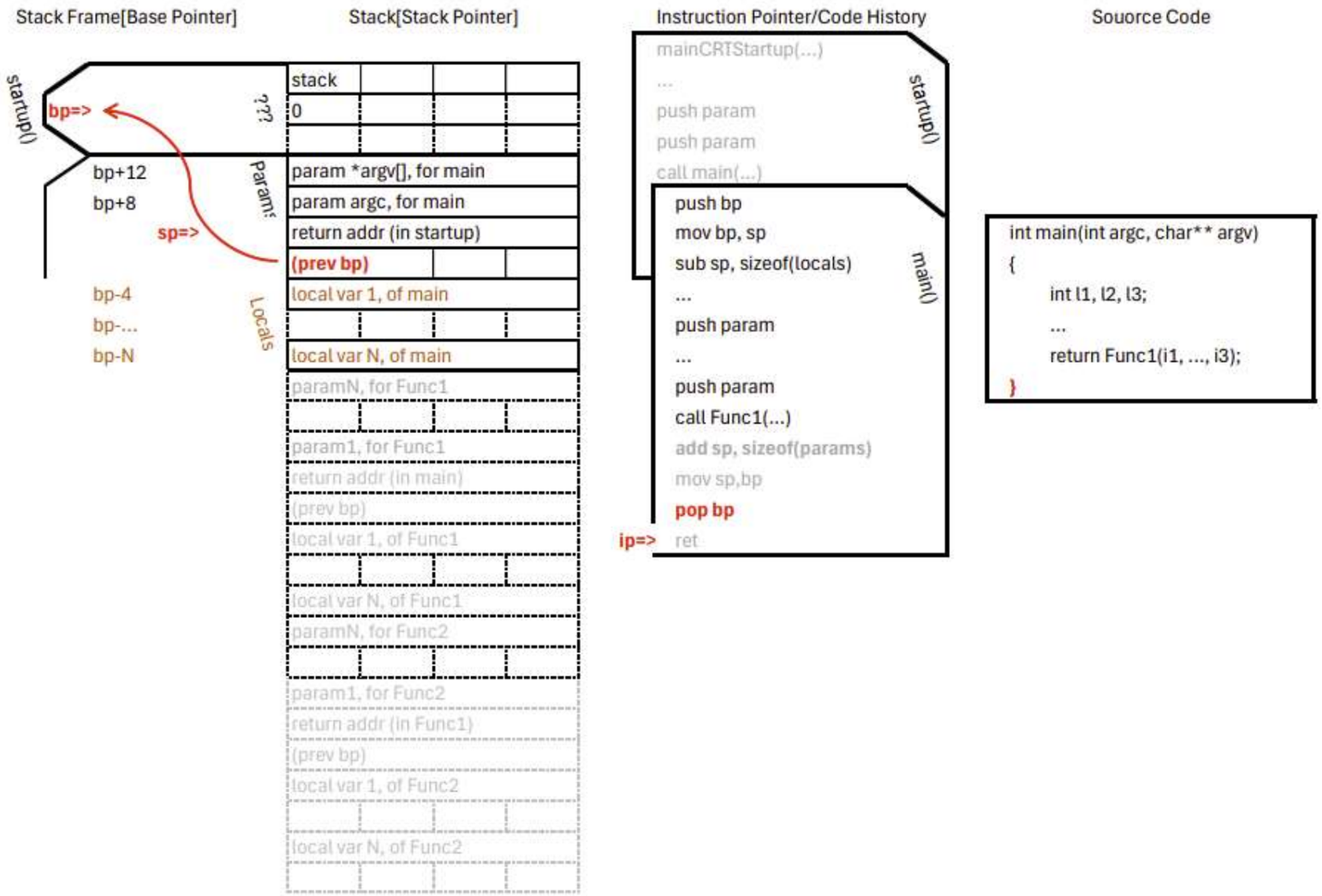
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



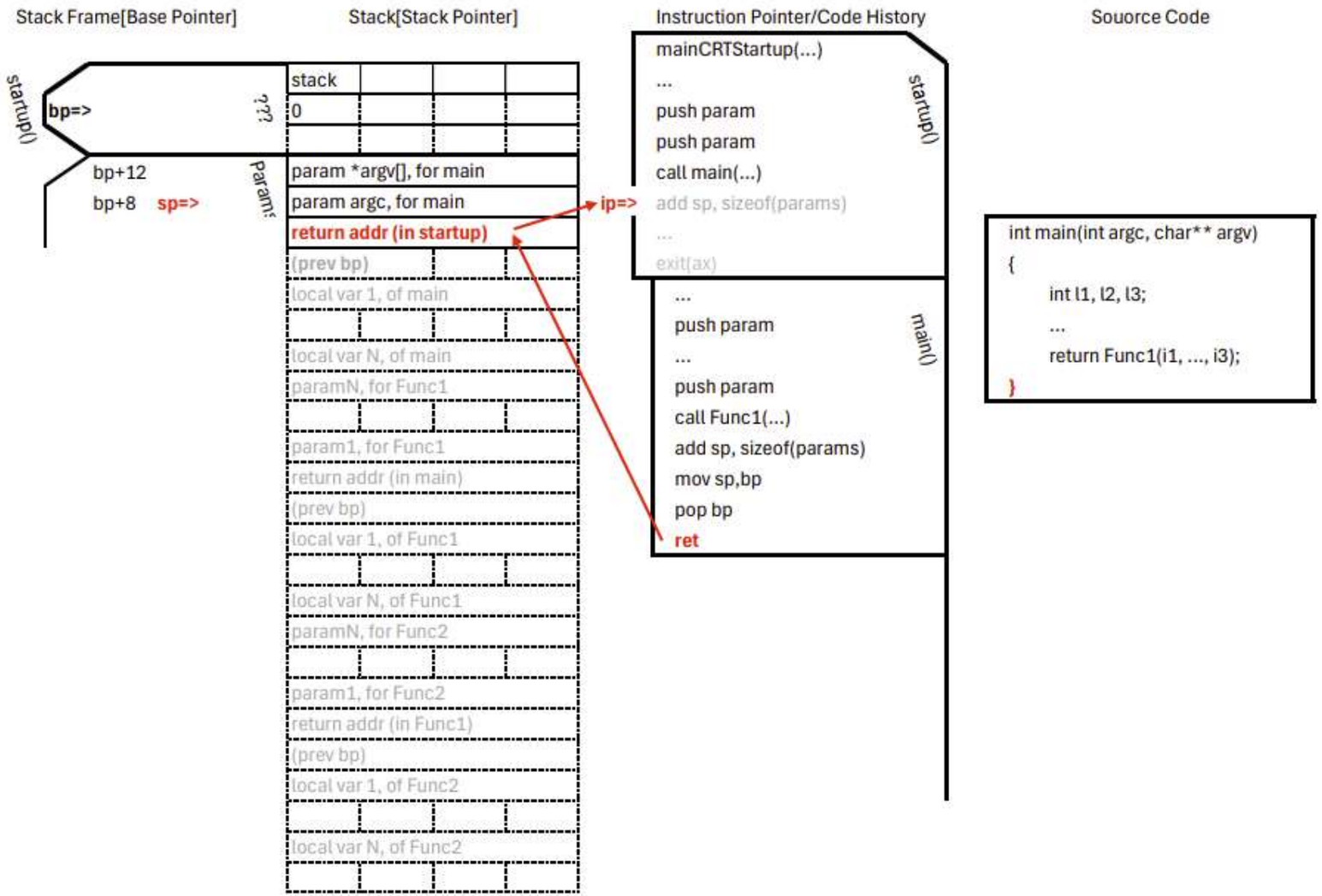
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



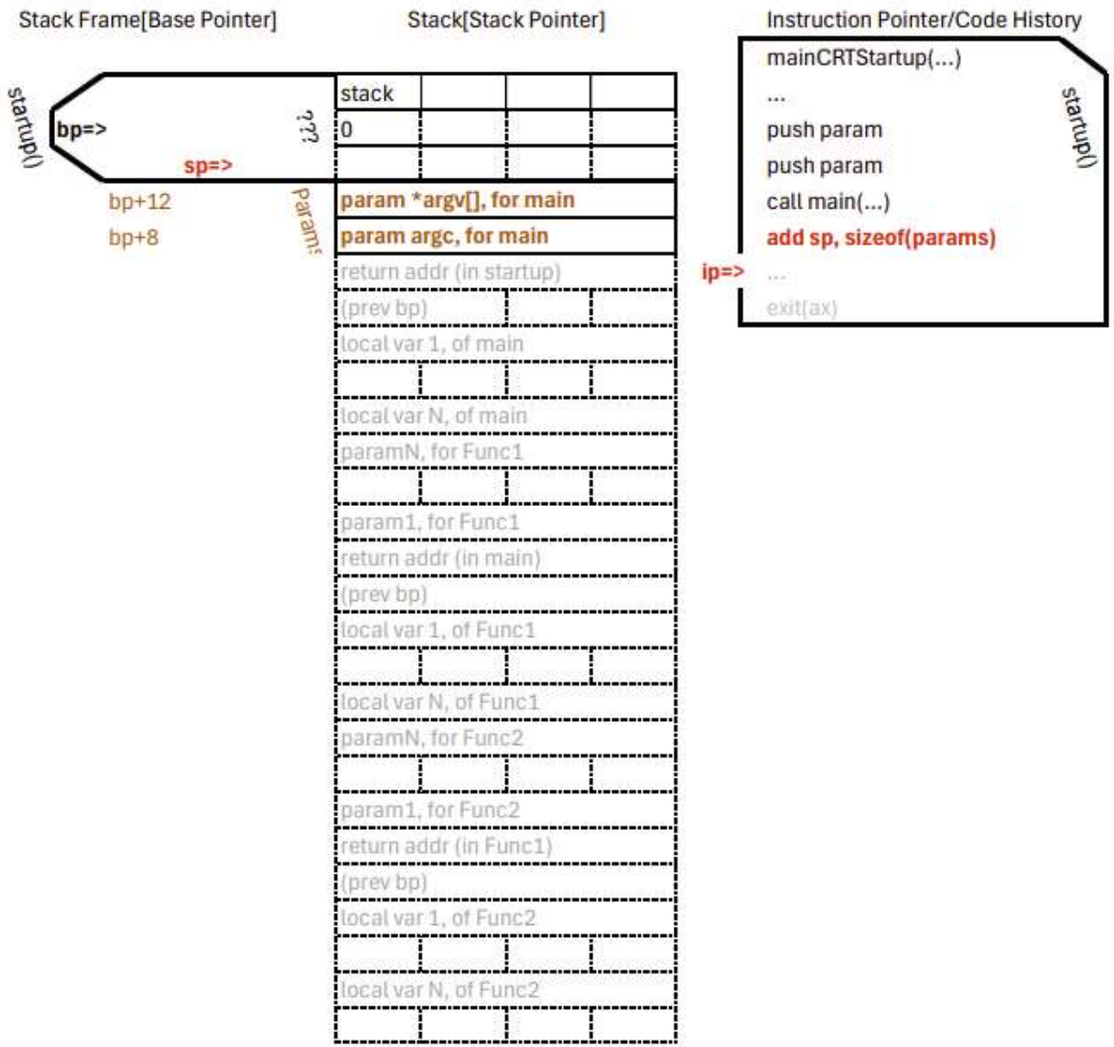
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



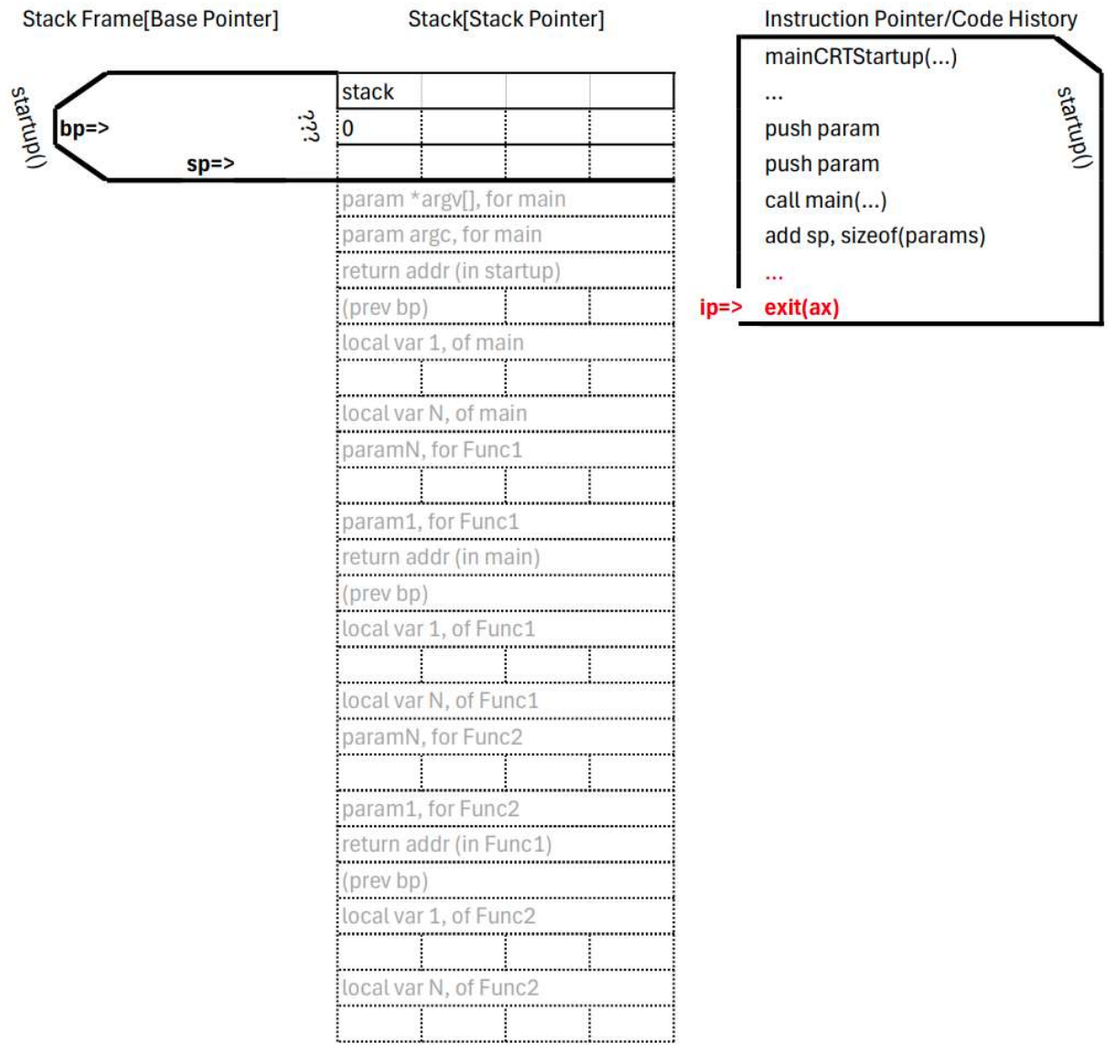
C++ Under the Hood – Stack Frame / Base Pointer Mechanics



C++ Under the Hood – Stack Frame / Base Pointer Mechanics



C++ Under the Hood – Stack Frame / Base Pointer Mechanics



C++ Under the Hood - What did we learn today ?

- Inheritance vs. Aggregation:
  - Memory layout,
  - v-table pointer placement.
- The first shall be last,
- C'tor, D'tor, Virtual Mechanism Mechanics:
  - Set up the v-table pointer on the opening curly bracket of the C'tor,
  - Set up the v-table pointer on the opening curly bracket of the D'tor,
  - Inside-Out/Outside-In.
- Pointers vs Member Pointers:
  - int * vs int Foo::*
  - int(*)(int) vs int(Foo::*)(int)
  - Magic of a nullptr (not always ~~0~~).
- Stack Frames/Base Pointer Mechanics
  - How to walk the stack

Questions?

Thank You!

Linkedin.com/in/chrisr98008
(No blind invites!
Tell me how we know each other)

Bonus slides below

C++ Under the Hood

C++ Under the Hood – Constructor/Destructor Order

```cpp
#include <iostream>
using std::cout;

struct Foo
{
    Foo() { cout << "Foo::Foo()   this=" << this << "\n"; }
    ~Foo() { cout << "~Foo::Foo()   this=" << this << "\n"; }
};

int main()
{
    cout << "new Foo[5];\n";
    Foo* pFoo = new Foo[5];
    cout << "Foo foo[5];\n";
    Foo foo[5];
    cout << "delete[] pFoo;\n";
    delete[] pFoo;
    cout << "//end of scope main()\n";
}
```

godbolt.org/z/vdPs8sfYc

C++ Under the Hood – Constructor/Destructor Order

```cpp
#include <iostream>
using std::cout;

struct Foo
{
    Foo() { cout << "Foo::Foo()   this=" << this << "\n"; }
    ~Foo() { cout << "~Foo::Foo()   this=" << this << "\n"; }
};

int main()
{
    cout << "new Foo[5];\n";
    Foo* pFoo = new Foo[5];
    cout << "Foo foo[5];\n";
    Foo foo[5];
    cout << "delete[] pFoo;\n";
    delete[] pFoo;
    cout << "//end of scope main()\n";
}
```

```
new Foo[5];
Foo::Foo()   this=0x12ca2c8
Foo::Foo()   this=0x12ca2c9
Foo::Foo()   this=0x12ca2ca
Foo::Foo()   this=0x12ca2cb
Foo::Foo()   this=0x12ca2cc
Foo foo[5];
Foo::Foo()   this=0x7ffe8c8246c3
Foo::Foo()   this=0x7ffe8c8246c4
Foo::Foo()   this=0x7ffe8c8246c5
Foo::Foo()   this=0x7ffe8c8246c6
Foo::Foo()   this=0x7ffe8c8246c7
delete[] pFoo;
~Foo::Foo()   this=0x12ca2cc
~Foo::Foo()   this=0x12ca2cb
~Foo::Foo()   this=0x12ca2ca
~Foo::Foo()   this=0x12ca2c9
~Foo::Foo()   this=0x12ca2c8
//end of scope main()
~Foo::Foo()   this=0x7ffe8c8246c7
~Foo::Foo()   this=0x7ffe8c8246c6
~Foo::Foo()   this=0x7ffe8c8246c5
~Foo::Foo()   this=0x7ffe8c8246c4
~Foo::Foo()   this=0x7ffe8c8246c3
```

godbolt.org/z/vdPs8sfYc

C++ Under the Hood – Constructor/Destructor Order

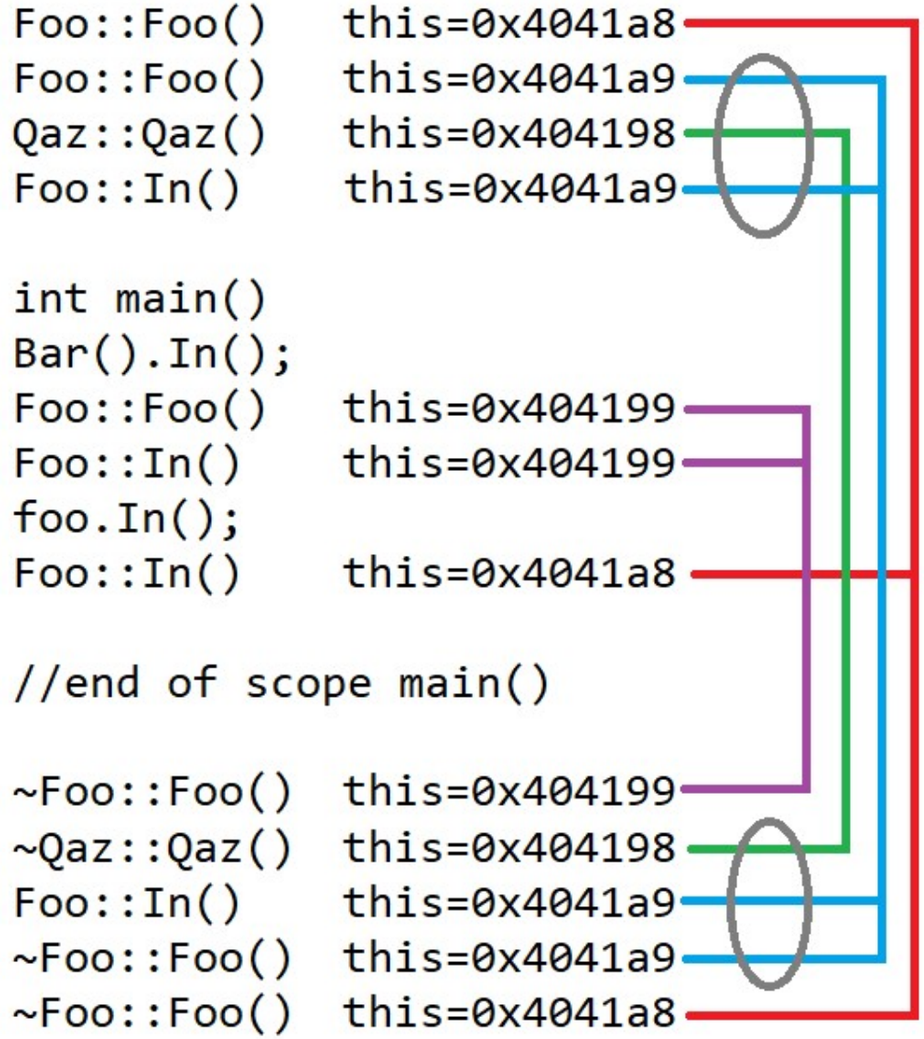
```cpp
#include <iostream>
using std::cout;

struct Foo
{
    Foo() { cout << "Foo::Foo()   this=" << this << "\n"; }
    ~Foo() { cout << "~Foo::Foo()   this=" << this << "\n"; }
};

int main()
{
    cout << "Foo foo3[3];\n";
    Foo foo3[3];
    cout << "Foo foo6[6];\n";
    Foo foo6[6];
    cout << "//end of scope main()\n";
}
```

```
Foo foo3[3];
Foo::Foo()   this=0x7fff617bb12d
Foo::Foo()   this=0x7fff617bb12e
Foo::Foo()   this=0x7fff617bb12f
Foo foo6[6];
Foo::Foo()   this=0x7fff617bb127
Foo::Foo()   this=0x7fff617bb128
Foo::Foo()   this=0x7fff617bb129
Foo::Foo()   this=0x7fff617bb12a
Foo::Foo()   this=0x7fff617bb12b
Foo::Foo()   this=0x7fff617bb12c
//end of scope main()
~Foo::Foo()   this=0x7fff617bb12c
~Foo::Foo()   this=0x7fff617bb12b
~Foo::Foo()   this=0x7fff617bb12a
~Foo::Foo()   this=0x7fff617bb129
~Foo::Foo()   this=0x7fff617bb128
~Foo::Foo()   this=0x7fff617bb127
~Foo::Foo()   this=0x7fff617bb12f
~Foo::Foo()   this=0x7fff617bb12e
~Foo::Foo()   this=0x7fff617bb12d
```

godbolt.org/z/vdPs8sfYc

C++ Under the Hood – Constructor/Destructor Order

```
struct Foo {
    Foo() { cout << "Foo::Foo()\tthis=" << this << "\n"; }
    ~Foo() { cout << "~Foo::Foo()\tthis=" << this << "\n"; }
    void In() { cout << "Foo::In()\tthis=" << this << "\n"; }
};
Foo& Bar() {
    static Foo foo;
    return foo;
}
static Foo foo;

struct Qaz {
    static inline Foo foo;
    Qaz() { cout << "Qaz::Qaz()\tthis=" << this << "\n"; foo.In(); }
    ~Qaz() { cout << "~Qaz::Qaz()\tthis=" << this << "\n"; foo.In(); }
} qaz;

int main()
{
    cout << "\nint main()\n";
    cout << "Bar().In();\n";    Bar().In();
    cout << "foo.In();\n";      foo.In();
    cout << "\n//end of scope main()\n\n";
}
```



godbolt.org/z/3vahsaEr9

C++ Under the Hood - structure static footprint

```cpp
struct A {
    auto& Get() { return map; }
private:
    std::map<int,int> map;
};
struct B {
    static auto& Get() { return map; }
private:
    inline static std::map<int,int> map;
};
struct C {
    static auto& Get() { return *map; }
private:
    inline static auto map = std::make_shared<std::map<int,int>>();
};
struct D {
    static auto& Get() { static std::map<int,int> map; return map; }
};
struct E {
    static auto& Get() { static auto map = std::make_shared<std::map<int,int>>(); return *map; }
};
```

godbolt.org/z/563sbr99r

C++ Under the Hood - Pre-/Post-main() TODO

```
int a = print("global int a");
auto A = Print("global auto A");

auto shared = std::make_shared<Print>(
            "global shared make_shared<Print>");

C c;
auto d = c.Get2();

int main() {
    cout<<"main()\n";
    auto shared = std::make_shared<Print>(
          "main local shared make_shared<Print>");
    auto e = c.Get3();
    cout<<"end main()\n";
}
```

```
ctor: inline static C::p1
func:global int a
ctor: global auto A
ctor: global shared make_shared<Print>
func:C::Get2()
ctor: C::Get2()
dtor: C::Get2()
ctor: inline static Get1() C::p2
main()
ctor: main local shared make_shared<Print>
func:C::Get3()
ctor: C::Get3()
dtor: C::Get3()
ctor: inline static C::Get2() C::p3
end main()
dtor: inline static C::Get2() C::p3
dtor: main local shared make_shared<Print>
dtor: inline static C::Get2() C::p3
dtor: inline static Get1() C::p2
dtor: inline static Get1() C::p2
dtor: global shared make_shared<Print>
dtor: global auto A
dtor: inline static C::p1
```

godbolt.org/z/xxxxxxxxx

C++ Under the Hood