

1

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 the internal C++ class mechanisms:

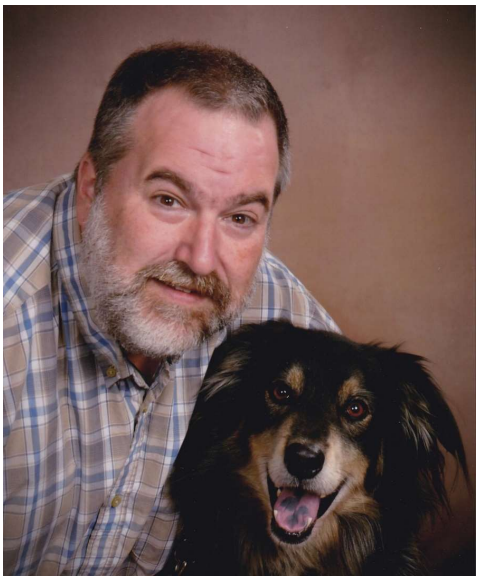
- C++ Onion
- Object Footprints
- Inheritance and Polymorphic mechanisms,
- Construction / Destruction order,
- Data Pointers / Member Data Pointers,
- Function Pointers / Member Function Pointers,

2

2

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

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



- Classically trained in Computer Systems Engineering (SW & HW),
- Well experienced in Modern C++ & Classic 'C',
- WG21 / ISO C++ Standards Committee member, EWG.

- Large Scale, **Complex** Problem Spaces, **Simple** Solutions
- Firmware & Embedded (large & small),

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

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

I believe in reducing complexity through simplification

3

3

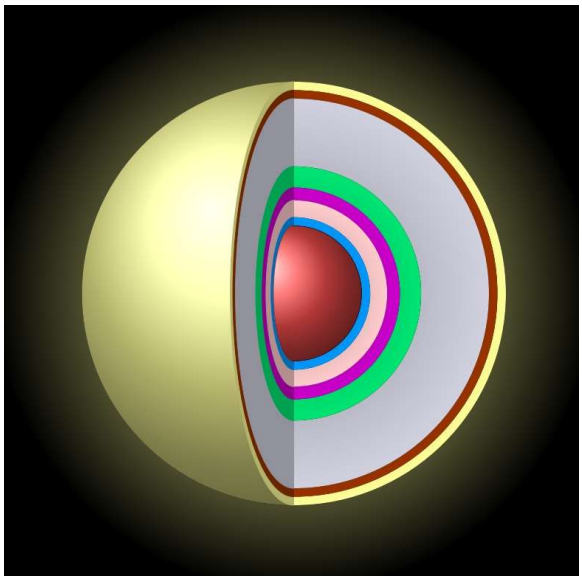
C++ Under the Hood – The Onion



4

4

C++ Under the Hood – The Onion



5

5

C++ Under the Hood – The Onion



6

6

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

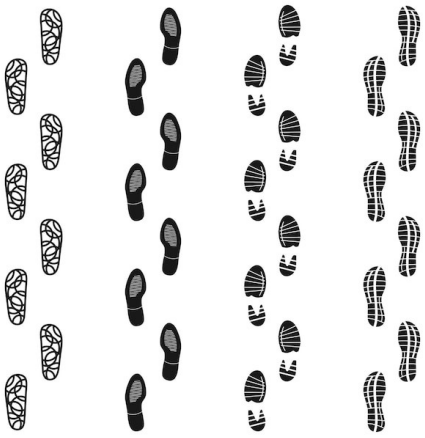


7

7

C++ Under the Hood

Object Memory Footprint



v-table Mechanism



8

8

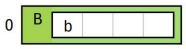
C++ Under the Hood

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



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

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



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

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



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

9

9

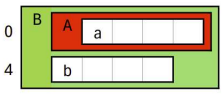
C++ Under the Hood

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



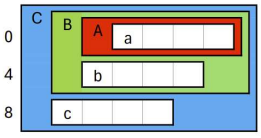
A::Foo(...) {...}

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



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

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



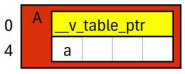
C::Foo(...) {...}

10

10

C++ Under the Hood

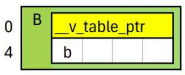
```
struct A {
  int a;
          A(...) {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



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

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

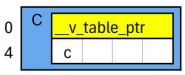
```
struct B {
  int b;
          B(...) {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



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

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

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



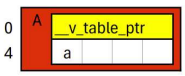
C::'v_table'= {C::Bar};

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

11

C++ Under the Hood

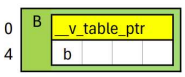
```
struct A {
  int a;
          A(...) {...};
  virtual     ~A() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



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

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

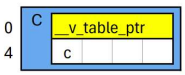
```
struct B {
  int b;
          B(...) {...};
  virtual     ~B() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



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(...) {...};
};
```



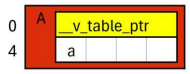
C::'v_table'= {C::~C, C::Bar};

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

12

C++ Under the Hood

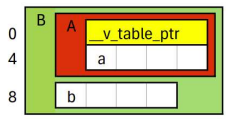
```
struct A {
  int a;
          A(...) {...};
  virtual ~A() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



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

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

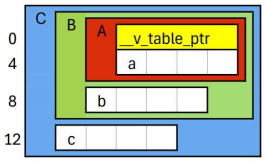
```
struct B : A {
  int b;
          B(...) {...};
  virtual ~B() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



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};

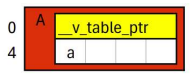
C::C(...) {...}
C::~C() {...}
C::Bar(...) {...}
C::Foo(...) {...}

13

13

C++ Under the Hood

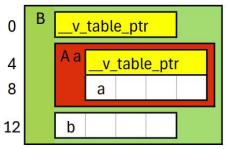
```
struct A {
  int a;
          A(...) {...};
  virtual ~A() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



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(...) {...};
};
```

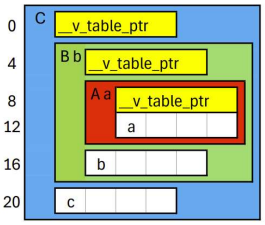


B::'v_table'= {B::~B, B::Bar};

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};

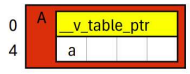
C::C(...) {...}
C::~C() {...}
C::Bar(...) {...}
C::Foo(...) {...}

14

14

C++ Under the Hood

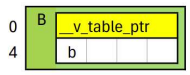
```
struct A {
  int a;
          A(...) {...};
  virtual ~A() {...};
  virtual int Bar(...) {...};
};
```



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

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

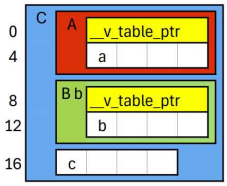
```
struct B {
  int b;
          B(...) {...};
  virtual ~B() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



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(...) {...};
};
```



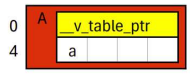
C::A_'v_table'= {C::~C, C::Bar};

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

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

C++ Under the Hood

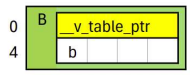
```
struct A {
  int a;
          A(...) {...};
  virtual ~A() {...};
  virtual int Bar(...) {...};
};
```



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

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

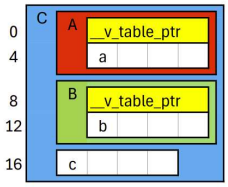
```
struct B {
  int b;
          B(...) {...};
  virtual ~B() {...};
  virtual int Bar(...) {...};
          int Foo(...) {...};
};
```



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 – One object, 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() {
    C c;    C* pC = &c;        B* pB = &c;        A* pA = &c;
    cout << "\npA=" << pA << " pA->Bar();\n";     pA->Bar();
    cout << "\npC=" << pC << " pC->Foo();\n";     pC->Foo();
    cout << '\n';
}
```

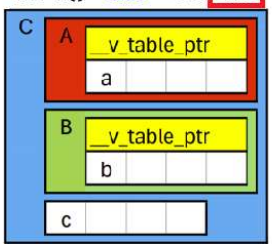
godbolt.org/z/77WGcq4er

```
A::A()   this = B3CFF6F8
B::B()   this = B3CFF708
C::C()   this = B3CFF6F8

pA = B3CFF6F8 pA->Bar();
C::Bar() this = B3CFF6F8
B::Foo() this = B3CFF708

pC = B3CFF6F8 pC->Foo();
B::Foo() this = B3CFF708

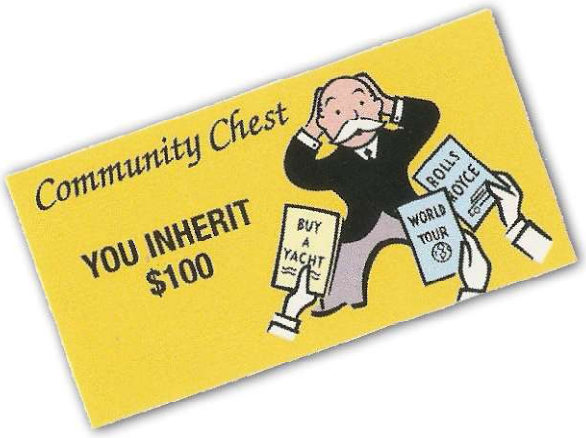
C::~C()  this = B3CFF6F8
B::~B()  this = B3CFF708
A::~A()  this = B3CFF6F8
```



17

17

C++ Under the Hood – Inheritance & Construction



18

18

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;
    // ...
};
```

19

19

C++ Under the Hood – Inheritance & Initialization

```
struct LegacyDependency
{
    int legacyId;
    LegacyDependency(int id) : legacyId(id) {}
};

class Widget : public LegacyDependency
{
private:
    const int m_id;
    std::string& m_externalRef;
    struct NoDefault
    {
        double val;
        NoDefault(double val) :val(val) {}
    } m_noDefaultObj;

    std::string m_name;
    std::vector<int> m_largeData;
    int m_width;
    int m_area;
    std::unique_ptr<int> m_ptr;
    int m_version = 1;
    static const int s_maxLimit = 100;
public:
    Widget() {}
};
```

godbolt.org/z/77WGcq4er

20

20

C++ Under the Hood – ~~Initialization List~~ Construction List

```
struct LegacyDependency
{
    int legacyId;
    LegacyDependency(int id) : legacyId(id) {}
};

class Widget : public LegacyDependency 1
{
private:
    const int m_id; 2
    std::string& m_externalRef; 3
    struct NoDefault
    {
        double val;
        NoDefault(double val) :val(val) {}
    } m_noDefaultObj; 4

    std::string m_name;5
    std::vector<int> m_largeData;6
    int m_width;
    int m_area;
    std::unique_ptr<int> m_ptr;7
    int m_version = 1;8
    static const int s_maxLimit = 100;
public:
```

```
public:
    ~~Widget() {}~~

    Widget(int id, std::string& ref, std::string name,
           std::vector<int> data, int width, int baseId)
      1 : LegacyDependency(baseId)
      2 , m_id(id)
      3 , m_externalRef(ref)
      4 , m_noDefaultObj(3.14)

    {}
};
```

godbolt.org/z/77WGcq4er

21

C++ Under the Hood – ~~Initialization List~~ Construction List

```
struct LegacyDependency
{
    int legacyId;
    LegacyDependency(int id) : legacyId(id) {}
};

class Widget : public LegacyDependency 1
{
private:
    const int m_id;2
    std::string& m_externalRef; 3
    struct NoDefault
    {
        double val;
        NoDefault(double val) :val(val) {}
    } m_noDefaultObj; 4

    std::string m_name;5
    std::vector<int> m_largeData;6
    int m_width;7
    int m_area;8
    std::unique_ptr<int> m_ptr; 9
    int m_version = 1;10
    static const int s_maxLimit = 100;
public:
```

```
public:
    ~~Widget() {}~~

    // The Constructor demonstrating all 10 flavors
    Widget(int id, std::string& ref, std::string name,
           std::vector<int> data, int width, int baseId)
      1 : LegacyDependency(baseId)
      2 , m_id(id)
      3 , m_externalRef(ref)
      4 , m_noDefaultObj(3.14)

      5 , m_name(std::move(name))
      6 , m_largeData(std::move(data))
      7 , m_width(width)
      8 , m_area(m_width * m_width)
      9 , m_ptr(std::make_unique<int>(42))
    {}
};
```

godbolt.org/z/77WGcq4er

22

C++ Under the Hood – More construction/initialization examples

```
struct Person
{
    std::string name{};
    std::string address{};
    int age{};

    Person() {};




};
```

23

C++ Under the Hood – More construction/initialization examples

```
struct Person
{
    std::string name{};
    std::string address{};
    int age{};

    Person() {};
    Person(std::string name, std::string address, int age)
    {
        name = name;
        age = age;
        address = address;
    }
};
```

24

C++ Under the Hood – More construction/initialization examples

```
struct Person
{
    std::string name{};
    std::string address{};
    int age{};

    Person() {};
    Person(std::string name, std::string address, int age)
        : name(name), age(age), address(address) {}
};
```

25

25

C++ Under the Hood – More construction/initialization examples

```
struct Person
{
    std::string name{};
    std::string address{};
    int age{};

    Person() {};
    Person(std::string name, std::string address, int age)
        : name(name), age(age), address(address) {}
};

struct Student : public Person
{
    float marks{};

    Student() {};




};
```

26

26

C++ Under the Hood – More construction/initialization examples

```
struct Person
{
    std::string name{};
    std::string address{};
    int age{};

    Person() {};
    Person(std::string name, std::string address, int age)
        : name(name), age(age), address(address) {}
};

struct Student : public Person
{
    float marks{};

    Student() {};
    Student(std::string name, std::string address, int age, float marks) :
       name(name), address(address), age(age),
        marks(marks)
    {
    }


};
```

27

27

C++ Under the Hood – More construction/initialization examples

```
struct Person
{
    std::string name{};
    std::string address{};
    int age{};

    Person() {};
    Person(std::string name, std::string address, int age)
        : name(name), age(age), address(address) {}
};

struct Student : public Person
{
    float marks{};

    Student() {};
    Student(std::string name, std::string address, int age, float marks) :

        marks(marks)
    {
        Person::name = name;
        Person::age = age;
        Person::address = address;
    }
};
```

28

28

C++ Under the Hood – More construction/initialization examples

```
struct Person
{
    std::string name{};
    std::string address{};
    int age{};

    Person() {};
    Person(std::string name, std::string address, int age)
        : name(name), age(age), address(address) {}
};

struct Student : public Person
{
    float marks{};

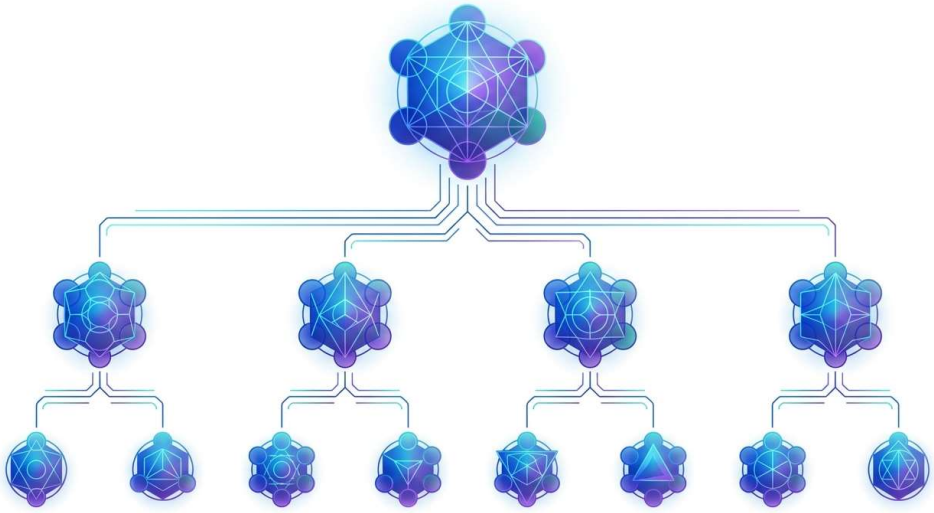
    Student() {};
    Student(std::string name, std::string address, int age, float marks) :
        Person(name, address, age),
        marks(marks)
    {
    }


};
```

29

29

C++ Under the Hood – Inheritance Hierarchy



30

30

C++ Under the Hood – Inheritance Hierarchy

```
class A
{


    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }
    virtual ~A()        { Bar();  }

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{

    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }
    virtual ~B()        { Bar();  }

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

31

31

C++ Under the Hood – Inheritance Hierarchy

```
class A
{

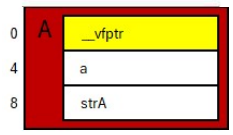

    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }
    virtual ~A()        { Bar();  }

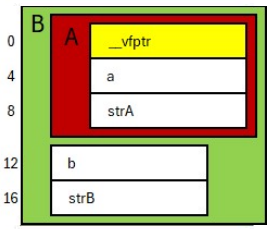
    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{

    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }
    virtual ~B()        { Bar();  }

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```





32

32

C++ Under the Hood – __vfptr

```
class A
{

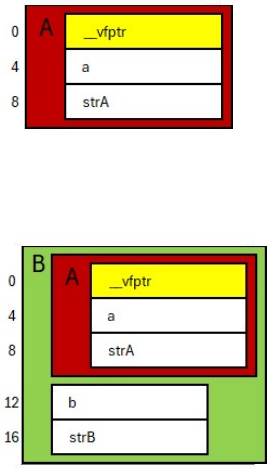
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }
    virtual ~A()        { Bar();  }

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{

    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }
    virtual ~B()        { Bar();  }

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```



33

33

C++ Under the Hood – __vfptr & vftable (virtual function pointer & table)

```
class A
{
                                            static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }
    virtual ~A()        { Bar();  }

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                            static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }
    virtual ~B()        { Bar();  }

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

34

34

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

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

35

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

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

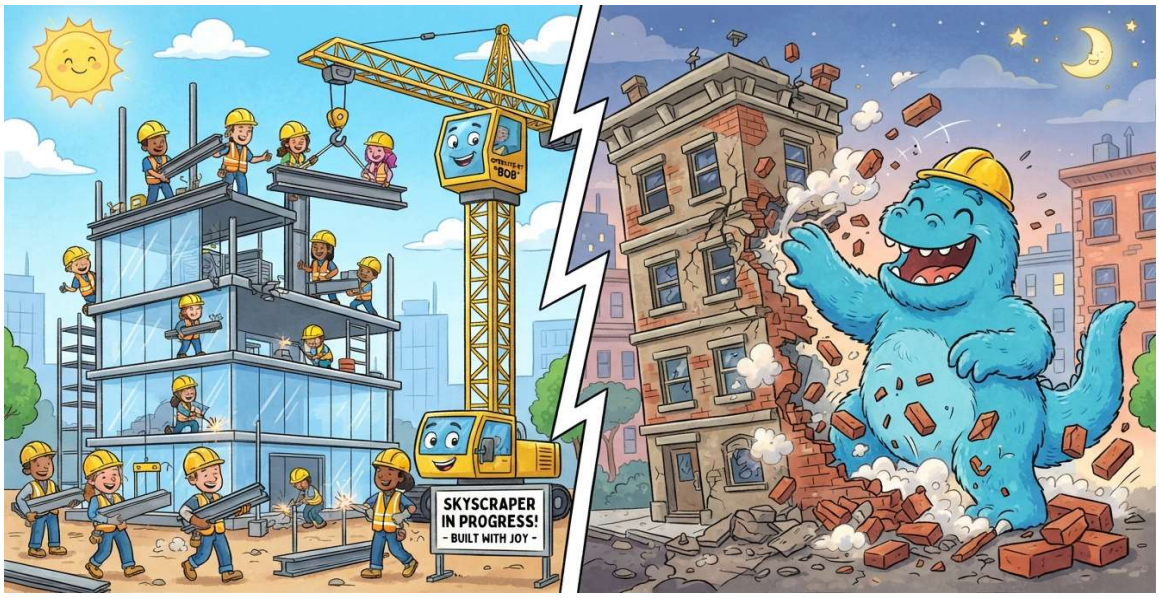
    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

36

C++ Under the Hood – Construction & Destruction



37

37

C++ Under the Hood – Construction

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

38

38

C++ Under the Hood – Construction

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

39

39

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

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

40

40

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

```
class A
{
                                          static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()               { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()      { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                          static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()               { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()      { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

41

41

C++ Under the Hood – Construction

```
class A
{
                                          static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()               { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()      { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                          static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()               { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()      { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

42

42

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

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

43

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

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

44

C++ Under the Hood – Polymorphism During Construction

```
class A
{
                                          static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()               { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()      { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()        { Foo(); }
};

class B : public A
{
                                          static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()               { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()      { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

45

C++ Under the Hood – Polymorphism During Construction

```
class A
{
                                          static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()               { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()      { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()        { Foo(); }
};

class B : public A
{
                                          static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()               { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()      { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

46

C++ Under the Hood – Polymorphism During Construction

```
class A
{
                                          static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }       A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }       A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                          static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }       B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }       B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

47

47

C++ Under the Hood – Polymorphism During Construction

```
class A
{
                                          static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }       A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }       A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                          static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }       B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }       B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

48

48

C++ Under the Hood – Construction

```
class A
{
                                            static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                            static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

49

49

C++ Under the Hood – Construction

```
class A
{
                                            static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                            static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

50

50

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

```
class A
{
                                          static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()               { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()      { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                          static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()               { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()      { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

51

51

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

```
class A
{
                                          static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()               { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()      { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                          static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()               { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()      { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

52

52

C++ Under the Hood – Polymorphism During Construction

```
class A
{
                                        static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar(); }          A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar(); }          A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                        static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar(); }          B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar(); }          B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

53

53

C++ Under the Hood – Construction

```
class A
{
                                        static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar(); }          A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar(); }          A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                        static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar(); }          B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar(); }          B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

54

54

C++ Under the Hood – Construction

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

55

C++ Under the Hood – Fully Constructed

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

56

C++ Under the Hood – Destruction

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

57

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

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

58

C++ Under the Hood – Polymorphism During Destruction

```
class A
{
                                        static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }     A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }     A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                        static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }     B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }     B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

59

59

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

```
class A
{
                                        static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }     A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }     A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                        static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }     B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }     B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

60

60

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

```
class A
{
                                            static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                            static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

61

61

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

```
class A
{
                                            static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                            static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

62

62

C++ Under the Hood – Destruction

```
class A
{
                                        static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }     A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }     A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                        static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }     B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }     B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

63

63

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

```
class A
{
                                        static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }     A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }     A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                        static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }     B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }     B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

64

64

C++ Under the Hood – Polymorphism During Destruction

```
class A
{
                                            static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                            static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

65

65

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

```
class A
{
                                            static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }         A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()          { Foo(); }
};

class B : public A
{
                                            static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }         B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

66

66

C++ Under the Hood – Destruction

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() :  vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

67

67

C++ Under the Hood – Destruction

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() :  vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

68

68

C++ Under the Hood – Destruction

```
class A
{
                                            static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }         A::~A() :   vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                            static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }         B::~B() :   vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

69

69

C++ Under the Hood – Destruction

```
class A
{
                                            static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }         A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }         A::~A() :   vfptr(A::'vftable'), { Bar(); ... }, ~str1()

    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                            static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }         B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }         B::~B() :   vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

70

70

C++ Under the Hood – Fully Destructed

```
class A
{
                                                static A::'vftable'[] = { A::~A(), A::Foo() }
            hidden v_table * '__vfptr';
    int a{};
    std::string strA{};
public:
    A()                 { Bar();  }             A::A()  : __vfptr(A::'vftable'), a(0), strA(), { Bar(); ...; }
    virtual ~A()        { Bar();  }             A::~A() : __vfptr(A::'vftable'), { Bar(); ... }, ~str1()

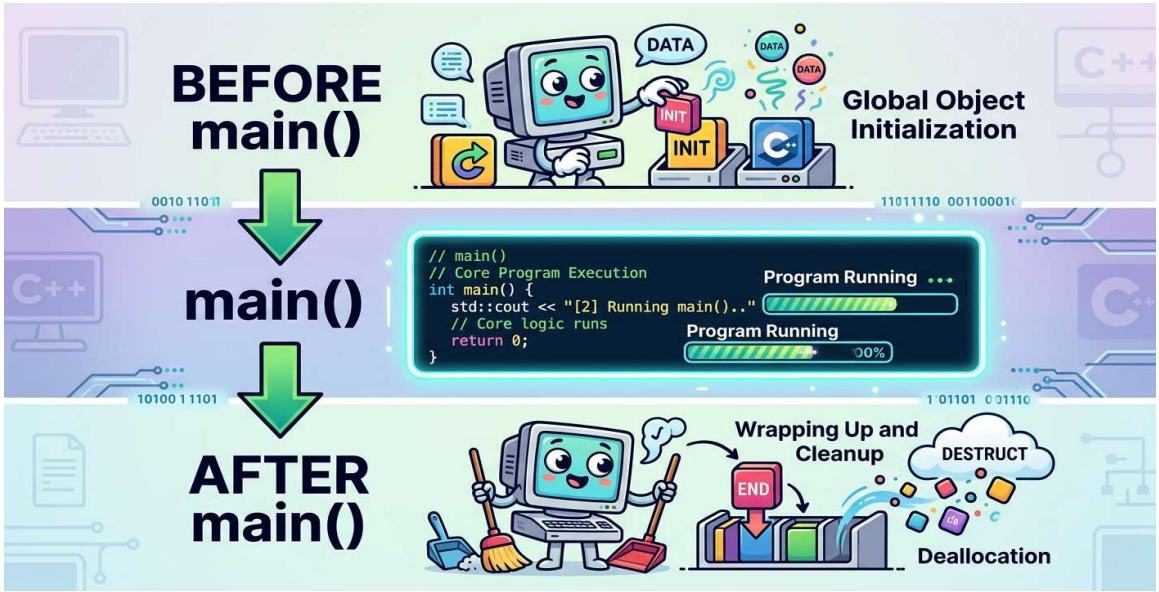
    virtual void Foo() { std::cout << "A::Foo()"; }
    void Bar()         { Foo(); }
};

class B : public A
{
                                                static B::'vftable'[] = { B::~B(), B::Foo() }
    int b{};
    std::string strB{};
public:
    B()                 { Bar();  }             B::B()  : A(), __vfptr(B::'vftable'), b(0), strB(), { Bar(); ...; }
    virtual ~B()        { Bar();  }             B::~B() : __vfptr(B::'vftable'), { Bar(); ... }, ~str2(), ~A()

    virtual void Foo() { std::cout << "B::Foo()"; }
};
```

71

C++ Under the Hood – The First will be Last and Inside-Out / Outside-In



72

C++ Under the Hood – Running code before/after main()

```
struct MyObject
{

    MyObject()
    {
        std::cout << "MyObject::MyObject()  this=" << this << '\n';  Foo();
    }
    ~MyObject()
    {
        std::cout << "MyObject::~MyObject() this=" << this << '\n';  Foo();
    }
    void Foo() { std::cout << "MyObject::Foo()       this=" << this << '\n'; }
};



MyObject myObject1;
MyObject myObject2;

int main()
{
    std::cout << "start main()\n";
    std::cout << "end main()\n";
}
```

```
MyObject::MyObject()  this = 0x404151
MyObject::Foo()       this = 0x404151
MyObject::MyObject()  this = 0x404152
MyObject::Foo()       this = 0x404152
start main()
end main()
MyObject::~MyObject() this = 0x404152
MyObject::Foo()       this = 0x404152
MyObject::~MyObject() this = 0x404151
MyObject::Foo()       this = 0x404151
```

73

73

C++ Under the Hood – Running c…ain()

Circular Gotcha

```
struct MyObject
{
    MyObject& myObject;
    MyObject(MyObject& myObject) : myObject(myObject)
    {
        std::cout << "MyObject::MyObject()  this=" << this << '\n';  myObject.Foo();
    }
    ~MyObject()
    {
        std::cout << "MyObject::~MyObject() this=" << this << '\n';  myObject.Foo();
    }
    void Foo() { std::cout << "MyObject::Foo()       this=" << this << '\n'; }
};

extern MyObject myObject2;

MyObject myObject1(myObject2);
MyObject myObject2(myObject1);

int main()
{
    std::cout << "start main()\n";
    std::cout << "end main()\n";
}
```

```
MyObject::MyObject()  this = 0x404158
MyObject::Foo()       this = 0x404160
MyObject::MyObject()  this = 0x404160
MyObject::Foo()       this = 0x404158
start main()
end main()
MyObject::~MyObject() this = 0x404160
MyObject::Foo()       this = 0x404158
MyObject::~MyObject() this = 0x404158
MyObject::Foo()       this = 0x404160
```

74

74

C++ Under the Hood – Array Construction & Destruction

```
#include <iostream>

struct Bar
{
    int i{}, j{};
    Bar(int j) : i(Next()), j(j){ std::cout << "Bar::Bar()  [" << i << "] j=" << j << " this=" << this << '\n';  }
    ~Bar()                      { std::cout << "Bar::~Bar() [" << i << "] j=" << j << " this=" << this << '\n';  }
    static int Next()           { static int next{}; return next++; }
};

int main()
{
    std::cout << "start main()\n";
    Bar bar[5]{2,8,3,6,7};
    std::cout << "end main()\n";
}
```

```
start main()
Bar::Bar()  [0] j = 2 this = 0x740d490
Bar::Bar()  [1] j = 8 this = 0x740d498
Bar::Bar()  [2] j = 3 this = 0x740d4a0
Bar::Bar()  [3] j = 6 this = 0x740d4a8
Bar::Bar()  [4] j = 7 this = 0x740d4b0
end main()
Bar::~Bar() [4] j = 7 this = 0x740d4b0
Bar::~Bar() [3] j = 6 this = 0x740d4a8
Bar::~Bar() [2] j = 3 this = 0x740d4a0
Bar::~Bar() [1] j = 8 this = 0x740d498
Bar::~Bar() [0] j = 2 this = 0x740d490
```

75

75

C++ Under the Hood – Static cleanup

```
struct Bar
{
    int i{};
    struct Foo
    {
        int j{};
        Foo()   { std::cout << "Foo::Foo()  this=" << this << '\n'; }
        ~Foo()  { std::cout << "Foo::~Foo() this=" << this << '\n'; }
    };
    Bar()       { std::cout << "Bar::Bar()  this=" << this << '\n'; }
    ~Bar()      { std::cout << "Bar::~Bar() this=" << this << '\n'; }
    Foo& Get()  { std::cout << "Bar::Get()  this=" << this << '\n';
                  static Foo foo; return foo; }
};

int main()
{
    std::cout << "start main()\n";
    Bar bar;
    auto& foo1 = bar.Get();
    std::cout << "in main()\n";
    auto& foo2 = bar.Get();
    std::cout << "end main()\n";
}
```

```
start main()
Bar::Bar()  this = 0x70869c
Bar::Get()  this = 0x70869c
Foo::Foo()  this = 0x404194
in main()
Bar::Get()  this = 0x70869c
end main()
Bar::~Bar() this = 0x70869c
Foo::~Foo() this = 0x404194
```

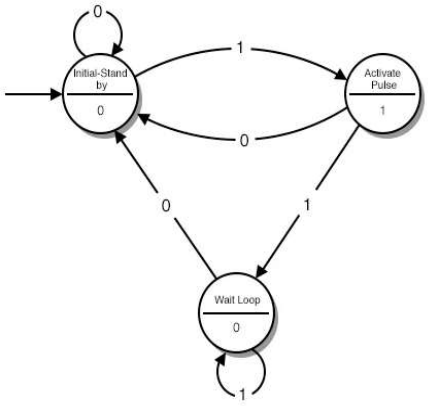
76

76

C++ Under the Hood – Running code before/after main()

```
struct MyApp {
    enum Event { timeout, event0, event1, stop, };
    struct FiniteStateMachine
    {
        Event WaitForEvent() { Event event{ stop }; /*...*/ return event; }
        bool handleEvent(Event event) {
            switch (event) {
            default:      {/*...*/} return false;
            case timeout:           break;
            case event0:  {/*...*/} break;
            case event1:  {/*...*/} break;
            }
            return true;
        }
    } fsm;
    MyApp() {
        init();
        while(run());
    }
    ~MyApp()              {/*...*/}
    bool BackgroundTask() {/*...*/ return true; }
    void init()           {/*...*/}
    bool run() {
        if (!BackgroundTask()) return false;
        return fsm.handleEvent(fsm.WaitForEvent());
    }
} myApp;

int main() {}
```



77

77

C++ Under the Hood - It is not polite to point.



78

78

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

79

79

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

```
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

80

80

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

81

81

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

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

#define count 5

int main()
{
    int array[count] = { 10, 5, 4, 6, 9 };

    qsort(array, count, sizeof(int), compare);

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

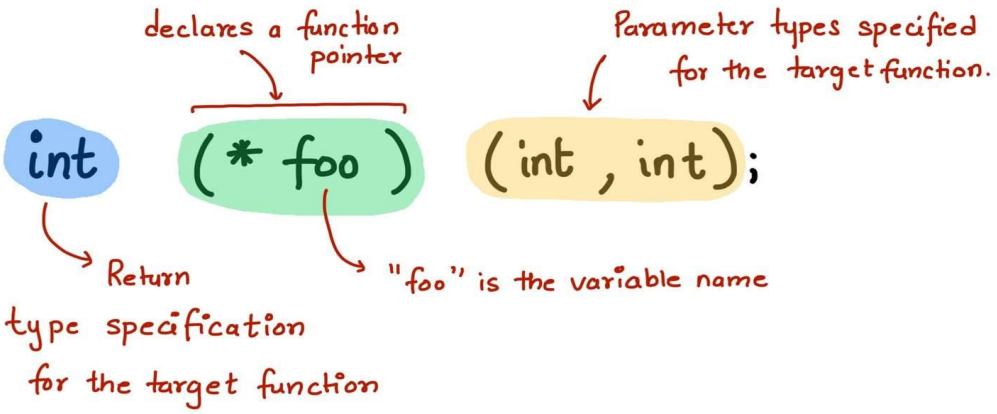
    return 0;
}
```

godbolt.org/z/6cKGG397h

82

82

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



83

83

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

84

84

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

85

85

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

86

86

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

87

87

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

88

88

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**

89

89

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

90

90

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

```
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

91

91

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

```
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

92

92

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

93

93

C++ Under the Hood – C++ Pointer to mixed use PFN & PMF

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

    int i;
    Foo(int i) : i(i) {}

    int Bar(int j) { return i + j; }
    static auto Qaz(Foo& obj, PMF pmf, int param) { return (obj.*pmf)(param); }
};




int main()
{
    Foo::PMF pMF = &Foo::Bar;
    Foo::PFN pFN = &Foo::Qaz;
    Foo obj{ 10 };

    return pFN(obj, pMF, 20);
}
```

godbolt.org/z/jTqzxGeeo

94

94

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

```
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

95

95

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

```
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

96

96

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

```
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

97

97

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/hE3nMhvET

98

98

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

```
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

99

99

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

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                            /* compiler supplied */ VFTable* vfptr;
                            /* compiler supplied */ static VFTable Foo::vftable[]={ &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i + j; }

};

struct Foo2 : Foo
{
                            /* compiler supplied */ static VFTable Foo2::vftable[]={ &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

100

100

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

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                            /* compiler supplied */ VFTable* vfptr;
                            /* compiler supplied */ static VFTable Foo::vftable[]={ &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i + j; }
                            /* compiler supplied */ int Foo::Bar_Thunk(int i) { return (this->vfptr[0])(int i); }
};

struct Foo2 : Foo
{
                            /* compiler supplied */ static VFTable Foo2::vftable[]={ &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i * j; }
};

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

godbolt.org/z/jTqzxGeeo

101

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

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                            /* compiler supplied */ VFTable* vfptr;
                            /* compiler supplied */ static VFTable Foo::vftable[]={ &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i + j; }
                            /* compiler supplied */ int Foo::Bar_Thunk(int i) { return (this->vfptr[0])(int i); }
};

struct Foo2 : Foo
{
                            /* compiler supplied */ static VFTable Foo2::vftable[]={ &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i * j; }
};

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

godbolt.org/z/jTqzxGeeo

102

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

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                            /* compiler supplied */ VFTable* vfptr;
                            /* compiler supplied */ static VFTable Foo::vftable[]={ &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i + j; }
                            /* compiler supplied */ int Foo::Bar_Thunk(int i) { return (this->vfptr[0])(int i); }
};

struct Foo2 : Foo
{
                            /* compiler supplied */ static VFTable Foo2::vftable[]={ &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i * j; }
};

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

godbolt.org/z/jTqzxGeeo

103

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

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                            /* compiler supplied */ VFTable* vfptr;
                            /* compiler supplied */ static VFTable Foo::vftable[]={ &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i + j; }
                            /* compiler supplied */ int Foo::Bar_Thunk(int i) { return (this->vfptr[0])(int i); }
};

struct Foo2 : Foo
{
                            /* compiler supplied */ static VFTable Foo2::vftable[]={ &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i * j; }
};

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

godbolt.org/z/jTqzxGeeo

104

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

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                            /* compiler supplied */ VFTable* vfptr;
                            /* compiler supplied */ static VFTable Foo::vftable[]={ &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i + j; }
                            /* compiler supplied */ int Foo::Bar_Thunk(int i) { return (this->vfptr[0])(int i); }
};

struct Foo2 : Foo
{
                            /* compiler supplied */ static VFTable Foo2::vftable[]={ &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i * j; }
};

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

godbolt.org/z/jTqzxGeeo

105

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

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                            /* compiler supplied */ VFTable* vfptr;
                            /* compiler supplied */ static VFTable Foo::vftable[]={ &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i + j; }
                            /* compiler supplied */ int Foo::Bar_Thunk(int i) { return (this->vfptr[0])(int i); }
};

struct Foo2 : Foo
{
                            /* compiler supplied */ static VFTable Foo2::vftable[]={ &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i * j; }
};

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

godbolt.org/z/jTqzxGeeo

106

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

```
struct Foo
{
    using PMF = int(Foo::*)(int);
                            /* compiler supplied */ VFTable* vfptr;
                            /* compiler supplied */ static VFTable Foo::vftable[]={ &Foo::Bar, };
    int i;
    Foo(int i) : i(i) {}
    virtual int Bar(int j) { return i + j; }
                            /* compiler supplied */ int Foo::Bar_Thunk(int i) { return (this->vfptr[0])(int i); }
};

struct Foo2 : Foo
{
                            /* compiler supplied */ static VFTable Foo2::vftable[]={ &Foo2::Bar, };
    Foo2(int i) : Foo(i) {}
    int Bar(int j) override { return i * j; }
};

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

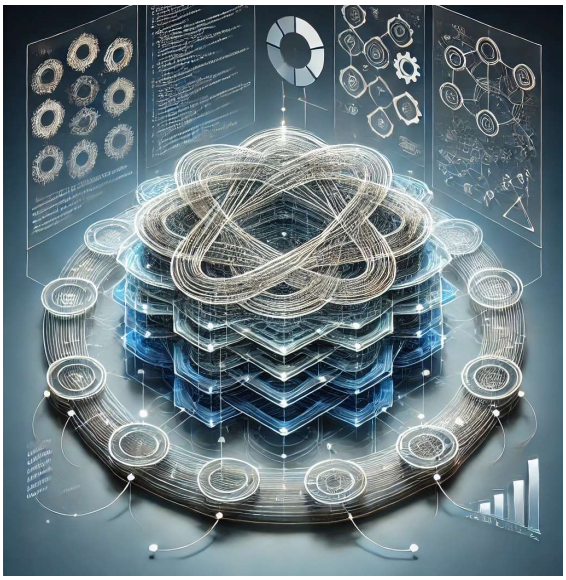
godbolt.org/z/jTqzxGeeo

107

C++ Under the Hood

Fundamental Theorem of Software Engineering:

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



108

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

109

109

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

```cpp
#include <iostream>

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;


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


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

godbolt.org/z/K5nf9jYPv

110

110

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

```cpp
#include <iostream>

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

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

godbolt.org/z/K5nf9jYPv

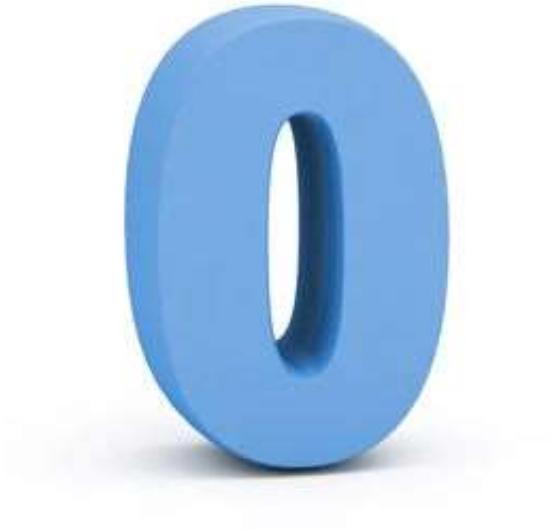
111

C++ Under the Hood

NULL

112

C++ Under the Hood



113

113

C++ Under the Hood

nullptr

114

114

C++ Under the Hood



115

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

```
#include <iostream>

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

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




}
```

godbolt.org/z/K5nf9jYPv

116

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

```
#include <iostream>

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

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

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



}
```

godbolt.org/z/K5nf9jYPv

117

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

```
#include <iostream>

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

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

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

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

godbolt.org/z/K5nf9jYPv

118

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

```
#include <iostream>

int main()
{

    struct Foo { char q; int a, b, c, d; };

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

    p = &Foo::a;    std::cout << "*(int*)&p = " << *(int*)&p << '\n';     //p = 4
    p = &Foo::b;    std::cout << "*(int*)&p = " << *(int*)&p << '\n';     //p = 8
    p = &Foo::c;    std::cout << "*(int*)&p = " << *(int*)&p << '\n';     //p = 12
    p = &Foo::d;    std::cout << "*(int*)&p = " << *(int*)&p << "\n\n";   //p = 16

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

godbolt.org/z/K5nf9jYPv

119

119

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

```
#include <iostream>

int main()
{
#pragma pack(1)
    struct Foo { char q; int a, b, c, d; };

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

    p = &Foo::a;    std::cout << "*(int*)&p = " << *(int*)&p << '\n';     //p = 1
    p = &Foo::b;    std::cout << "*(int*)&p = " << *(int*)&p << '\n';     //p = 5
    p = &Foo::c;    std::cout << "*(int*)&p = " << *(int*)&p << '\n';     //p = 9
    p = &Foo::d;    std::cout << "*(int*)&p = " << *(int*)&p << "\n\n";   //p = 13

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

godbolt.org/z/K5nf9jYPv

120

120

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

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

#define count 5

int main()
{
    int array[count] = { 10, 5, 4, 6, 9 };

    qsort(array, count, sizeof(int), compare);

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

    return 0;
}
```

godbolt.org/z/6cKGG397h

121

121

C++ Under the Hood – C++ style Sort(...) using a lambda

```
int main()
{
    int array[] = { 10, 5, 4, 6, 9 };

    std::sort(std::begin(array), std::end(array), [](int const lhs, int const rhs) { return lhs < rhs; });

    std::cout << "sorted ascending:";
    for (const auto& item :  array)
        std::cout << ' ' << item;
    std::cout << '\n';

    std::sort(std::begin(array), std::end(array), [](int const lhs, int const rhs) { return lhs > rhs; });

    std::cout << "sorted descending:";
    for (const auto& item : array)
        std::cout << ' ' << item;
    std::cout << '\n';

    return 0;
}
```

godbolt.org/z/6Y4YvdWf7

122

122

C++ Under the Hood – C++ style Sort(...) with lambda

```
struct Foo {
    int a, b, c, d;
    friend std::ostream& operator<<(std::ostream& os, const Foo& obj)
        { return os << std::format("{} {} {} {}\n", obj.a, obj.b, obj.c,obj.d); }
};

std::ostream& operator<<(std::ostream& os, std::vector<Foo>& vFoo)
{
    os << "a b c d\n";
    for (const auto& item : vFoo)
        os << item;
    return os << '\n';
}
auto GetData()
{
    return std::vector<Foo>{{3, 8, 8, 4}, { 1,0,9,0 }, { 9,2,9,6 }, { 3,6,9,1 }, { 2,9,7,3 }, { 1,5,9,0 }, { 1,9,4,5 }};
}
void Sort(auto& data, auto compare) { std::sort(data.begin(), data.end(), compare); }

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

    Sort(data, [](Foo& lhs, Foo& rhs)  { return lhs.a < rhs.a; });    std::cout << "sort .a\n" << data << '\n';
    Sort(data, [](Foo& lhs, Foo& rhs)  { return lhs.b < rhs.b; });    std::cout << "sort .b\n" << data << '\n';
    Sort(data, +[](Foo& lhs, Foo& rhs) { return lhs.c < rhs.c; });    std::cout << "sort .c\n" << data << '\n';
    Sort(data, +[](Foo& lhs, Foo& rhs) { return lhs.d < rhs.d; });    std::cout << "sort .d\n" << data << '\n';
}
```

godbolt.org/z/YTPnjaaee

123

123

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

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

godbolt.org/z/nx7snTnqo

124

124

C++ Under the Hood – C++ Sort using Multiple Pointers to Member Data

```
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 << "raw data (unsorted)\n" << data;
    Shuffle(data);      Sort(data, &Data::a);                    std::cout << "Sort on a\n"           << data;
    Shuffle(data);      Sort(data, &Data::b, &Data::c);          std::cout << "Sort on b,c\n"         << data;
    Shuffle(data);      Sort(data, &Data::b, &Data::a);          std::cout << "Sort on b,a\n"         << data;
    Shuffle(data);      Sort(data, &Data::c, &Data::d, &Data::e);  std::cout << "Sort on c,d,e\n"       << data;
}
```

godbolt.org/z/nz3M1bTYT

125

125

C++ Under the Hood – C++ structs/tuple and operator<=>

```
struct MyStruct
{
    int a;
    int b;
    int c;

    auto operator<=>(const MyStruct&) const = default; //auto generates <, ==, >, <=, !=, >=
    friend std::ostream& operator<<(std::ostream& os, const MyStruct& obj)
        { return os << std::format("( {}, {}, {}, )", obj.a, obj.b, obj.c); }
};

int main()
{
    MyStruct s1{ 2, 3, 5 };
    MyStruct s2{ 2, 3, 4 };

    if (s1 < s2)
        std::cout << s1 << " < " << s2 << '\n';
    else if (s1 > s2)
        std::cout << s1 << " > " << s2 << '\n';
    else
        std::cout << s1 << " = " << s2 << '\n';
}
```

godbolt.org/z/Wso6E9b5o

126

126

C++ Under the Hood – C++ Sort using a better compare

```
template<typename T>
bool comp(const T& lhs, const T& rhs, auto T::* key, auto...keys)
{
    if constexpr (sizeof...(keys) > 0)
    {
        if (lhs.*key < rhs.*key) return true;
        if (lhs.*key > rhs.*key) return false;
        return comp(lhs, rhs, keys...);
    }
    return (lhs.*key < rhs.*key);
}

void Sort_old(auto& data, auto ...keys)
{
    std::sort(std::begin(data), std::end(data),
        [keys...](const auto& lhs, const auto& rhs)
            { return comp(lhs, rhs, keys...); }
    );
}

void Sort_new(auto& data, auto ...keys)
{
    std::sort(std::begin(data), std::end(data),
        [keys...](const auto& lhs, const auto& rhs)
            { return std::tie(lhs.*keys...) < std::tie(rhs.*keys...); }
    );
}
```

godbolt.org/z/nz3M1bTYT

127

127

C++ Under the Hood – C++ Sort using Tie vs Tuple using Pointers to Member Data

```
void Sort_new1(auto& data, auto ...keys)
{
    std::sort(std::begin(data), std::end(data),
        [keys...](const auto& lhs, const auto& rhs)
            { return std::tie(lhs.*keys...) < std::tie(rhs.*keys...); }
    );
}


void Sort_new2(auto& data, auto ...keys)    //&Data::c, &Data::d, &Data::e
{
    std::sort(std::begin(data), std::end(data),
        [keys...](const auto& lhs, const auto& rhs)
            { return std::tuple(lhs.*keys...) < std::tuple(rhs.*keys...); }
                //lhs. data { 3, 5, 1, 8, 2 } < rhs. data { 6, 4, 1, 8, 6} (exmaple data)
                //lhs. aka Data:: *c, *d, *e  < rhs. aka Data:: *c, *d, *e
                //tuple    {        1, 8, 2 } < tuple  {          1, 8, 6}
                //bool Tuple::operator<(tuple&)
    );
}
```

godbolt.org/z/nz3M1bTYT

128

128

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:
  - Member Initialization/Initialization List
  - Set up the v-table pointer before constructing member objects in the C'tor,
  - Set up the v-table pointer before destructing member objects in 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~~).

129

129

C++ Under the Hood

Questions?

Thank You!

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

130

130