C++ (& C) Grab Bag

Final Project: Due in 2 Days

Complete Something

Parent destructors

```
struct Buffer {
  Buffer(int s) { buf = new char[s]; }
  ~Buffer() { delete [] buf; }
  char *buf;
};

struct FBuffer : public Buffer {
  FBuffer(int s) : Buffer(s) {
    f = fopen("file", "w");
  }
  ~FBuffer() { fclose(f); }
  void write() { fwrite(buf, 1, 40, f); }
  FILE *f;
};
```

```
struct Buffer {
  Buffer(int s);
  ~Buffer();
  char *buf;
};
```

```
struct FBuffer
: public Buffer {
  FBuffer(int s);
  ~FBuffer();
  void write();
  FILE *f;
};
```

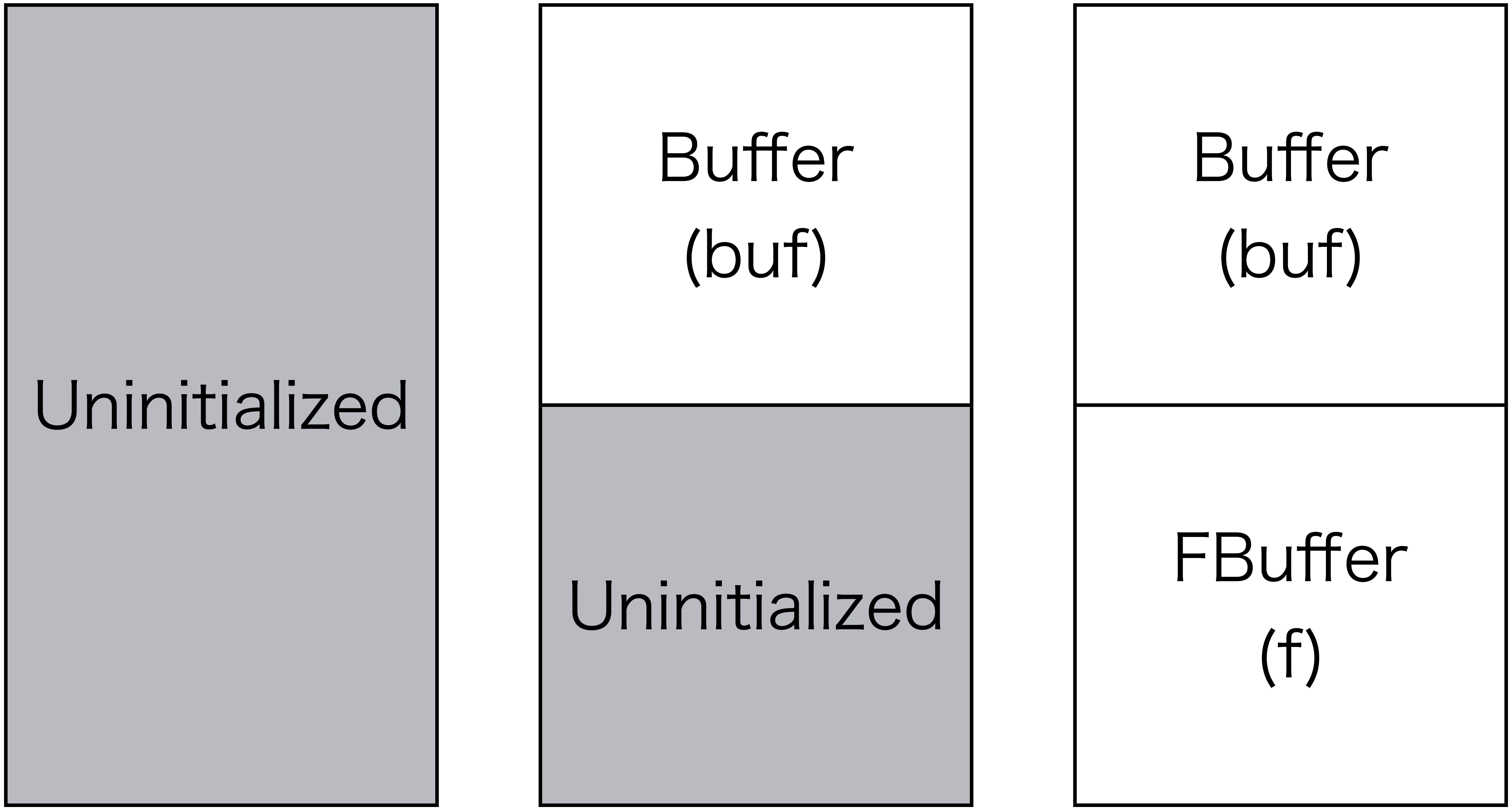
```
Buffer *buf = new Buffer(128);
delete buf;
// ✓
```

```
struct Buffer {
  Buffer(int s);
  ~Buffer();
  char *buf;
};
```

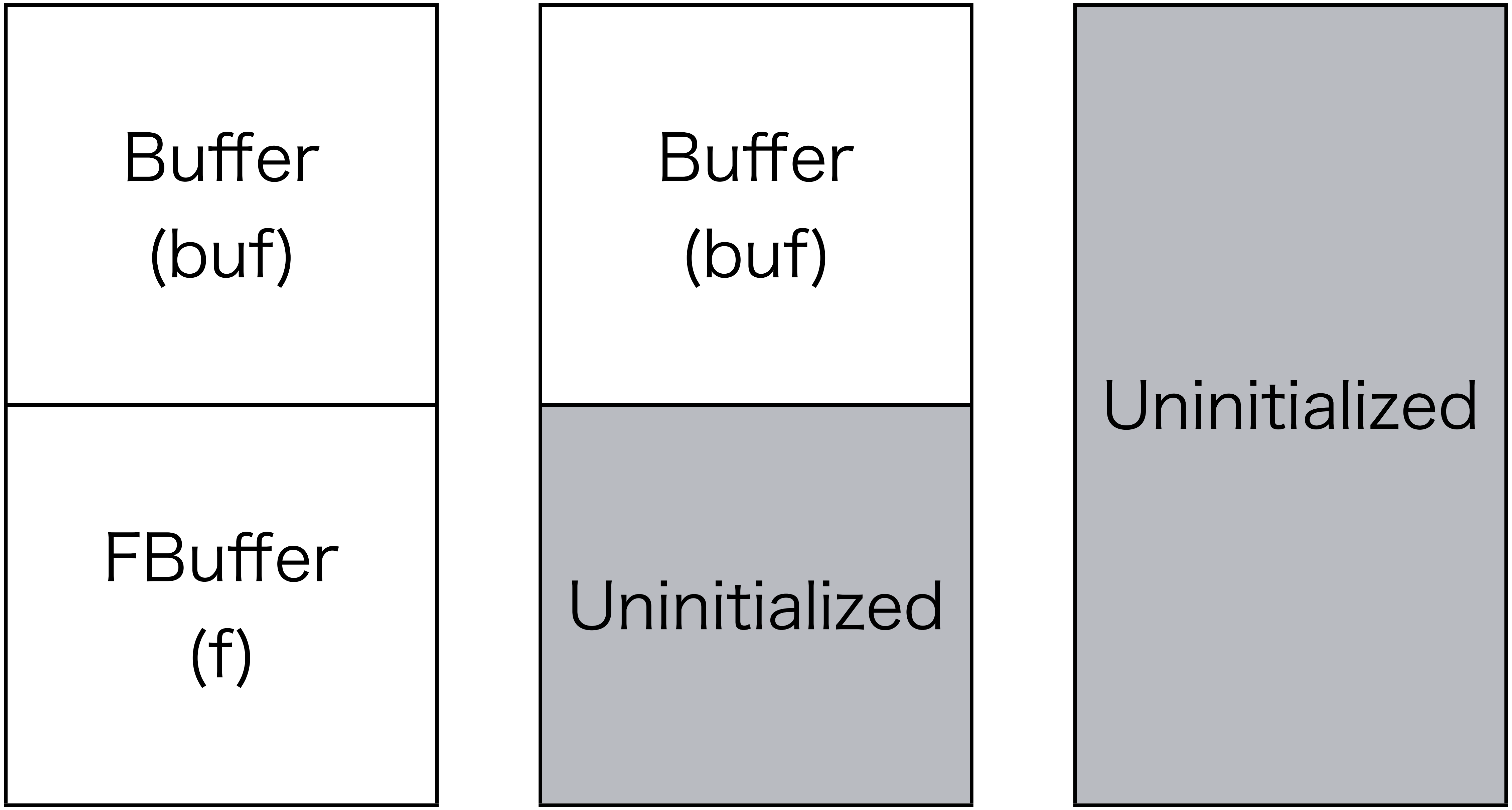
```
struct FBuffer
: public Buffer {
  FBuffer(int s);
  ~FBuffer();
  void write();
  FILE *f;
};
```

```
FBuffer *fbuf = new FBuffer(128);
delete fbuf;
// ✓
```

Construction



Destruction



```
struct Buffer {
  Buffer(int s);
  ~Buffer();
  char *buf;
};
```

```
struct FBuffer
: public Buffer {
  FBuffer(int s);
  ~FBuffer();
  void write();
  FILE *f;
};
```

```
Buffer *fbuf = new FBuffer(128);
delete fbuf;
// ✗ only ~Buffer is called
```

```
struct Buffer {
  Buffer();
  virtual ~Buffer();
  char *buf;
};
```

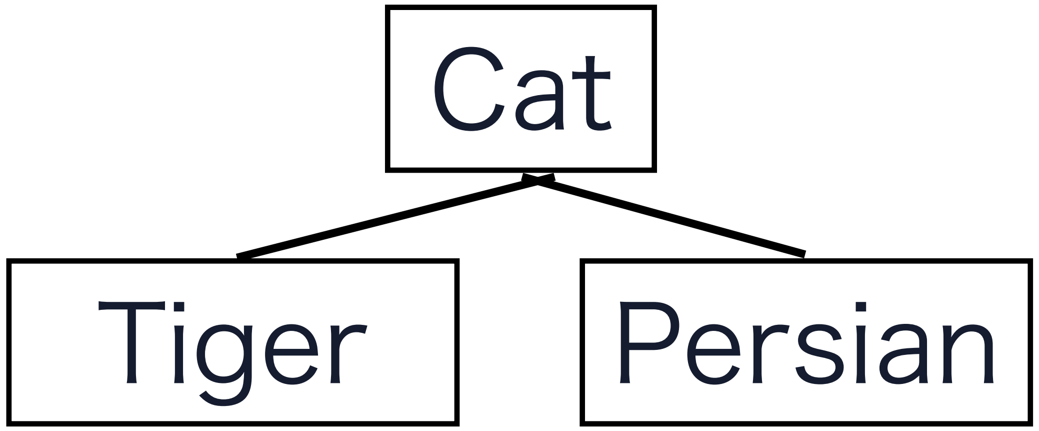
```
struct FBuffer
: public Buffer {
  FBuffer();
  virtual ~FBuffer();
  void write();
  FILE *f;
};
```

```
Buffer *fbuf = new FBuffer;
delete fbuf;
// ✓
```

C++ Casts

```
// C cast
char *buf = (char *)malloc(128);

// C-style cast
float b = 98.6;
int a = int(b);
```

```
// C-style cat casts
class Cat { };
class Tiger : public Cat { };
class Persian : public Cat { };
Cat *c = new Persian;
Tiger *t = (Tiger *)c; // whoops!
```



```
// valid up-cast

Tiger *t = new Tiger;

Cat *c1 = (Cat *)t;
Cat *c2 = static_cast<Cat *>(t);
Cat *c3 = dynamic_cast<Cat *>(t);
```

```
// almost valid down-cast

Cat *c = new Tiger;

Tiger *t1 = (Tiger *)c;
Tiger *t2 = static_cast<Tiger *>(c);
Tiger *t3 = dynamic_cast<Tiger *>(c);

// compile error
// valid down-cast

class Cat { virtual void purr() { } };
class Tiger : public Cat { };
class Persian : public Cat { };

Cat *c = new Tiger;

Tiger *t1 = (Tiger *)c;
Tiger *t2 = static_cast<Tiger *>(c);
Tiger *t3 = dynamic_cast<Tiger *>(c);
```

```
// invalid down-cast

Cat *c = new Persian;

Tiger *t1 = (Tiger *)c;
Tiger *t2 = static_cast<Tiger *>(c);
Tiger *t3 = dynamic_cast<Tiger *>(c);

// t1 & t2 are invalid pointers
// t3 is NULL
```

References

```c
void swap(int *a, int *b) {
  int tmp = *a;
  *a = *b;
  *b = tmp;
}

int main() {
  int x = 2, y = 3;
  swap(&x, &y);
}
```

```
void swap(int &a, int &b) {
  int tmp = a;
  a = b;
  b = tmp;
}

int main() {
  int x = 2, y = 3;
  swap(x, y);
}
```

Hello, World!

```cpp
#include <iostream>

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

```
#include <iostream>

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

Namespaces

```
SNDFILE *open(const char *);
count_t seek(SNDFILE *, count_t);
int error(SNDFILE *);
```

```
SNDFILE *sf_open(const char *);
count_t sf_seek(SNDFILE *, count_t);
int sf_error(SNDFILE *);
```

```
namespace sf {
  SNDFILE *open(const char *);
  count_t seek(SNDFILE *, count_t);
  int error(SNDFILE *);
}
```

```
#include <iostream>

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

```
std::cout << "Hello, World!"
          << std::endl;
```

```
using namespace std;
cout << "Hello, World!"
     << endl;
```

```
using std::cout;
using std::endl;
cout << "Hello, World!"
     << endl;
```

g++ -E hello.cpp

iostream

```
namespace std {
  extern istream cin;
  extern ostream cout;
  extern ostream cerr;
}
```

```
namespace std {
  extern istream cin;
  extern ostream cout;
  extern ostream cerr;

  class ActionLawsuit {
  };
}
```

```
extern istream cin;
extern ostream cout;
extern ostream cerr;

class ActionLawsuit {
};
```

```
namespace super {
  namespace std {
    extern istream cin;
    extern ostream cout;
    extern ostream cerr;

    class ActionLawsuit {
    };
  }
}

super::std::ActionLawsuit;
```

`extern`

iostream

```
namespace std {
  extern istream cin;
  extern ostream cout;
  extern ostream cerr;
}
```

iostream

```
ostream cout;
```

main.c

```
#include <iostream>

int main() {
  cout << "i";
  foo();
}
```

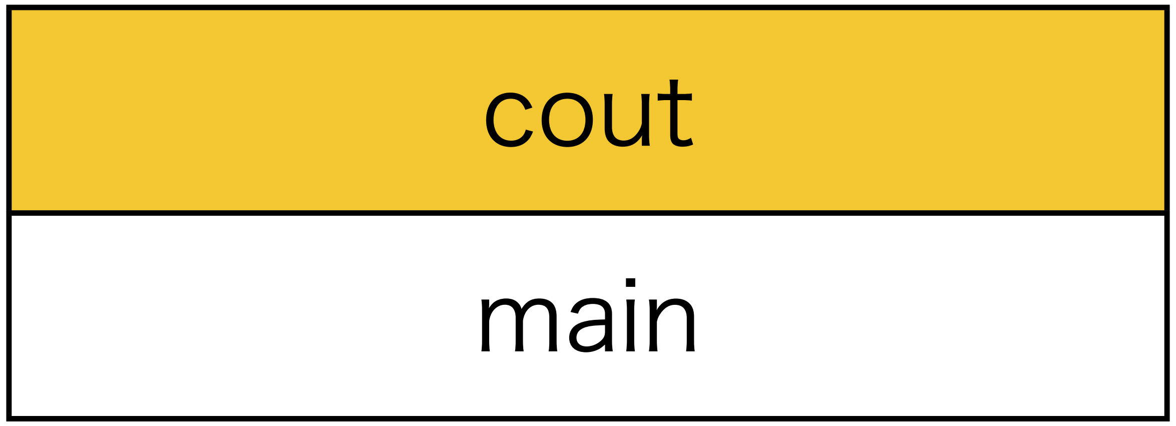
foo.c

```
#include <iostream>

int foo() {
  cout << "Phone";
}
```

main.c (preprocessed)

```
ostream cout;

int main() {
  cout << "i";
  foo();
}
```

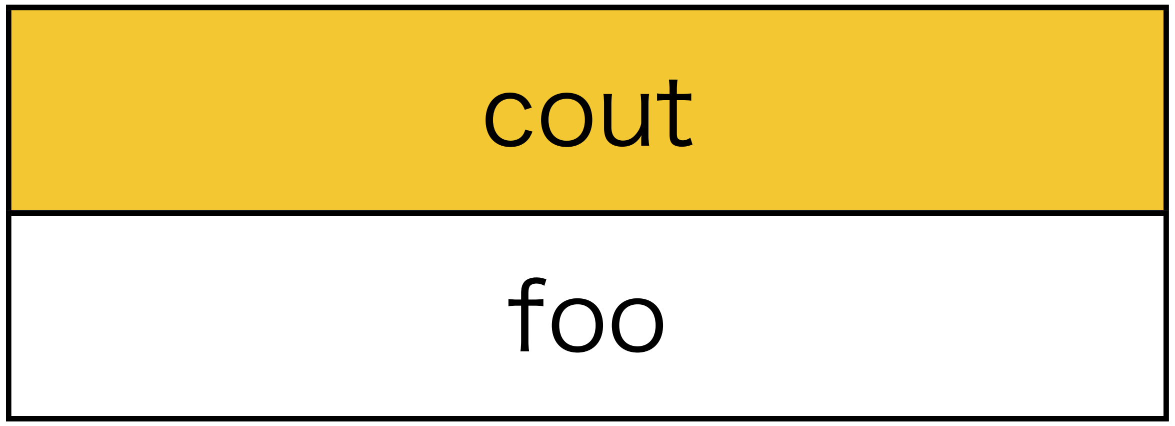
main.o



foo.c (preprocessed) -

```
ostream cout;

int foo() {
  cout << "Phone";
}
```

foo.o



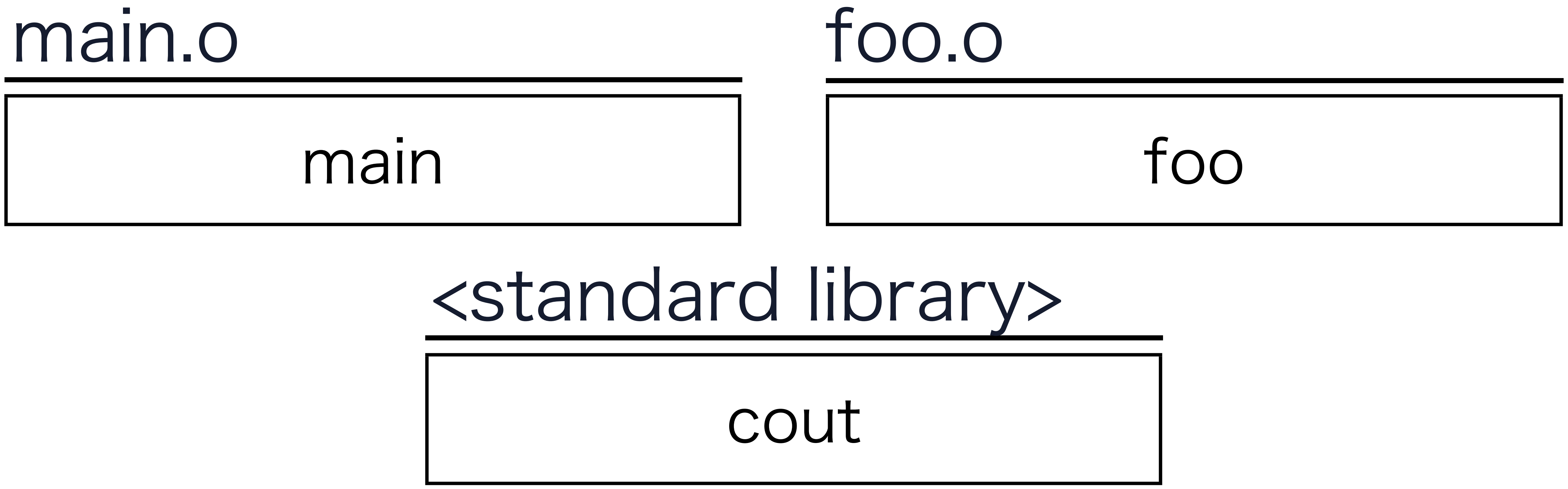
```
ld: 1 duplicate symbol for architecture x86_64
```

main.c (preprocessed)

```
extern ostream cout;

int main() {
  cout << "i";
  foo();
}
```

foo.c (preprocessed)

```
extern ostream cout;

int foo() {
  cout << "Phone";
}
```



```
#include <iostream>

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

```
#include <iostream>

int main() {
  std::cout << "Hello, World!"
            << std::endl;
  int a = 2 << 1;
  return 0;
}
```

Operator Overloading

```
struct vec2 {
  vec2(float x, float y)
    : x(x), y(y) { }
  float x, y;
};

int main() {
  vec2 a(1, 0);
  vec2 b(1, 3);
  vec2 c = a + b; // compile error
}
```

```
vec.cpp: In function ‘int main()’:
vec.cpp:12: error: no match for ‘operator+’ in ‘a + b’
```

```
vec2 vec2::add(const vec2 &o) {
  return vec2(x + o.x, y + o.y);
}

int main() {
  vec2 a(1, 0), b(1, 3);
  vec2 c = a.add(b);
}
```

```
vec2 vec2::operator +(const vec2 &o) {
  return vec2(x + o.x, y + o.y);
}

int main() {
  vec2 a(1, 0), b(1, 3);
  vec2 c = a + b;
  vec2 d = a.operator+(b);
}
```

```
vec2 operator +(vec2 &v, const vec2 &o) {
  return vec2(x + o.x, y + o.y);
}

int main() {
  vec2 a(1, 0), b(1, 3);
  vec2 c = a + b;
}
```

```
a + b       a != b
a - b       a && b
a * b       a || b
a / b       a & b
a % b       a | b
a < b       a ^ b
a <= b      a << b
a == b      a >> b
a >= b      a, b
a > b       a[b]
```

```
vec2 operator+(const vec2 &o);
```

```
+a
-a
++a
a++
--a
a--
!a
~a
*a
&a
```

```
vec2 operator+();
```

```
a = b          a &= b
a += b         a |= b
a -= b         a ^= b
a *= b         a <<= b
a /= b         a >>= b
a %= b         a = (b += c)
```

```
vec2 &vec2::operator+=(const vec2 &o)
{
  x += o.x;
  y += o.y;
  return *this;
}
```

(**this** is a pointer to the object)

Streams

```
struct Foo {
  char *str() const {
    return "Foo!";
  }
};

ostream &
operator<<(ostream &os, const Foo &f) {
  return os << f.str();
}

int main() {
  Foo f;
  std::cout << f << std::endl;
}
```