C ⊄ C++
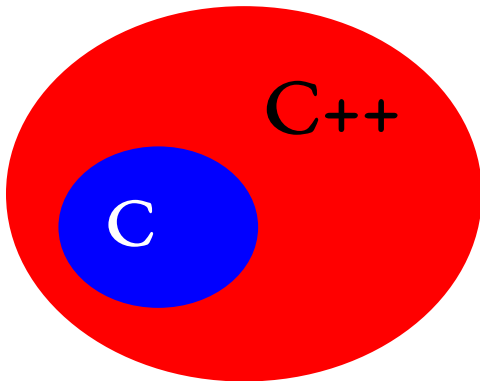
Frank Terbeck
ft@bewatermyfriend.org

January 9, 2020

$C \subset C++$ — Right!?

C++

C

In Reality: C $\not\subset$ C++

Code

```
main(void)
{
    /* ... */
}
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
main(void)
{
    /* ... */
}
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

C

```
test.c:1:1: warning: return type defaults to 'int' [-Wimplicit-int]
```

"Implicit int" should be an error since C99:   -Werror-implicit-int

Code

```
main(void)
{
    /* ... */
}
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

C++

```
test.cpp:1:10: warning: ISO C++ forbids declaration of 'main' with
               no type [-Wreturn-type]
```

Code

```
struct FooBar {
    int a;
    int b;
};
/* ... */
Foobar fb;
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
struct FooBar {
    int a;
    int b;
};
/* ... */
Foobar fb;
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

C

```
test.c:9:5: error: unknown type name 'FooBar'; use 'struct' keyword
          to refer to the type

typedef struct FooBar { int a; int b; } FooBar;
```

Code

```
auto x = 2.25;
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
auto x = 2.25;
```

C

```
test.c:4:10: warning: type defaults to 'int' in declaration of
             'x' [-Wimplicit-int]
test.c:4:14: warning: implicit conversion from 'double' to 'int'
             changes value from 2.25 to 2 [-Wliteral-conversion]
```

Code

```
auto x = 2.25;
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

C++

From C++11 x will be a double via type inference.

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
auto int y = 5;
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
auto int y = 5;
```

C

Valid C.

Questions

Code

- Valid C?
- Valid C++?

```
auto int y = 5;
```

- Both?
- Neither!?

C++

```
test.cpp:4:14: error: two or more data types in declaration of 'y'

test.cpp:4:5: warning: 'auto' storage class specifier is not permit-
              ted in C++11, and will not be supported in future re-
              leases [-Wauto-storage-class]
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
register int z = 5;
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
register int z = 5;
```

C

Valid C.

Questions

Code

- Valid C?
- Valid C++?
- Both?
- Neither!?

```
register int z = 5;
```

C++

```
test.cpp:4:5: error: ISO C++17 does not allow 'register' storage
              class specifier [-Wregister]
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
int x = 8;
int a[x];
```

Code

```
int x = 8;
int a[x];
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

C

- Valid C (since `C99`): Variable Length Arrays.
- Optional since `C11`.
- Also: "USING VLAs IS ACTIVELY STUPID!" – Linus T.

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
int x = 8;
int a[x];
```

C++

```
test.cpp:5:10: warning: variable length arrays are a C99
              feature [-Wvla-extension]
```

Code

```
FooBar fb = { .b = 1,
              .a = 2 };
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
FooBar fb = { .b = 1,
              .a = 2 };
```

C

Valid C (since C99): Designated Initialisers

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
FooBar fb = { .b = 1,
              .a = 2 };
```

C++

```
test.cpp:6:41: error: designator order for field 'FooBar::a' does
              not match declaration order in 'FooBar'
```

Code

```
_Atomic char f = 'f';
_Bool g = true;
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
_Atomic char f = 'f';
_Bool g = true;
```

C++

```
test.cpp:4:5: error: '_Atomic' was not declared in this scope
test.cpp:5:5: error: '_Bool' was not declared in this scope
```

Code

```
#define cbrt(x)
  _Generic((x),           \
      long double: cbrtl, \
      float: cbrtf        \
      default: cbrt,      \
      )(x)
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
#define cbrt(x)
  _Generic((x),            \
      long double: cbrtl, \
      float: cbrtf         \
      default: cbrt,       \
      )(x)
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

C++

```
test.cpp:8:5: warning: generic selections are a C11-specific
              feature [-Wc11-extensions]
```

Code

```
int *ii;
ii = malloc(sizeof(*ii));
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Questions

Code

- Valid C?
- Valid C++?
- Both?
- Neither!?

```
int *ii;
ii = malloc(sizeof(*ii));
```

C++

```
test.cpp:6:21: error: invalid conversion from 'void*' to 'int*'
                [-fpermissive]
```

Questions

Code

```
size_t chlitsize = sizeof('%');
```

- Valid C?
- Valid C++?
- Both?
- Neither!?

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
size_t chlitsize = sizeof('%');
```

C

In C, a character literal is an `int`.

Questions

Code

```
size_t chlitsize = sizeof('%');
```

- Valid C?
- Valid C++?
- Both?
- Neither!?

C++

In C, a character literal is a char.

Code

```
int
main(void)
{
    quux();
}
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
int
main(void)
{
    quux();
}
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

C

```
test.c:4:5: warning: implicit declaration of function 'quux'
```

Should be an error since C99:   -Werror-implicit-function-declaration

Code

```
int
main(void)
{
    quux();
}
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

C++

```
test.cpp:4:5: error: 'quux' was not declared in this scope
```

Code

```
int
main(void)
{
}
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

Code

```
int
main(void)
{
}
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

C

Valid since C99 (to match C++ sadly). Before:

```
test.c:4:1: warning: control reaches end of non-void function
```

Code

```
int
main(void)
{
}
```

Questions

- Valid C?
- Valid C++?
- Both?
- Neither!?

C++

Valid C++. Returns 0. Only applies to main().

- These are not all differences!
- Almost everything is language version dependent.
- Almost everything is toolchain dependent.
- Idiomatic C is not idiomatic C++
- …and visa-versa.
- Interfaces are the solution.
- Compiling C with a C++ compiler is not.

Thanks for your attention!