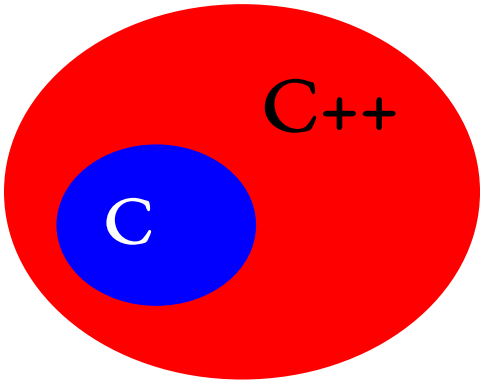
C $\not\subset$ C++

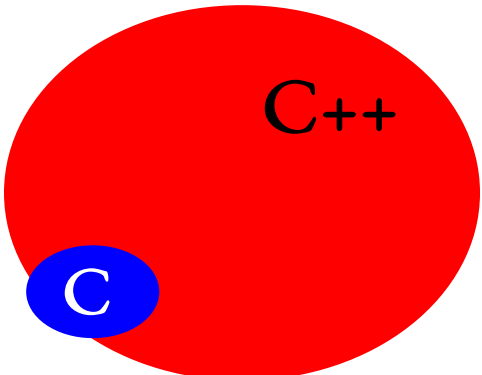
Frank Terbeck
ft@bewatermyfriend.org

February 5, 2020

C ⊂ C++ — Right!?



$C \not\subset C++$



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

Quiz

Code

```
%:include <stdio.h>
%:include <stdlib.h>
int main(int argc, const char *argv<::>) <%
    while (argc-- > 1) <%
        printf("%s\n", argv<:argc:>);
    %>
    return EXIT_SUCCESS;
%>
```

Questions

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

Quiz

Code

```
%:include <stdio.h>
%:include <stdlib.h>
int main(int argc, const char *argv<::>) <%
    while (argc-- > 1) <%
        printf("%s\n", argv<:argc:>);
    %>
    return EXIT_SUCCESS;
%>
```

Valid C and C++! — Digraphs and Trigraphs.

Questions

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

Quiz

Code

```
int class = 2;
int new = 3;
bool mutable = false;
```

Questions

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

Quiz

Code

```
int class = 2;
int new = 3;
bool mutable = false;
```

Questions

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

C

All valid, and likely good names too.

Quiz

Code

```
int class = 2;
int new = 3;
bool mutable = false;
```

Questions

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

C++

```
test.cpp:8:5: error: expected primary-expression before ‘int’
test.cpp:9:9: error: expected unqualified-id before ‘new’
test.cpp:10:18: error: expected unqualified-id before ‘=’ token
```

Quiz

Code

```
/* File level */
const unsigned int n = 32u;
char buffer[n];
```

Questions

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

Quiz

Code

```
/* File level */
const unsigned int n = 32u;
char buffer[n];
```

Questions

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

C

```
test.c:6:6: error: variably modified ‘buffer’ at file scope
```

Quiz

Code

```
/* File level */
const unsigned int n = 32u;
char buffer[n];
```

Questions

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

C++

Valid C++.

Quiz

Code

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

Questions

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

Quiz

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;
```

Quiz

Code

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

Questions

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

Quiz

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`

Quiz

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]
```

Quiz

Code

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

Questions

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

Quiz

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
```

Quiz

Code

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

Questions

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

C++

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

Quiz

Code

```
auto x = 2.25;
```

Questions

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

Quiz

Code

```
auto x = 2.25;
```

Questions

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

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]
```

Quiz

Code

```
auto x = 2.25;
```

Questions

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

C++

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

Quiz

Code

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

Questions

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

Quiz

Code

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

C

Valid C.

Questions

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

Quiz

Code

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

Questions

- Valid C?
- Valid C++?
- 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]
```

Quiz

Code

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

Questions

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

Quiz

Code

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

Questions

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

C

Valid C.

Quiz

Code

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

Questions

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

C++

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

Quiz

Code

```
void foo(void) {
  int x = 8;
  int a[x];
  /* ... */
```

Questions

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

Quiz

Code

```
void foo(void) {
  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.

Quiz

Code

```
void foo(void) {
  int x = 8;
  int a[x];
  /* ... */
```

Questions

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

C++

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

Quiz

Code

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

Questions

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

Quiz

Code

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

Questions

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

C

Valid C (since C99): Designated Initialisers

Quiz

Code

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

Questions

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

C++

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

Quiz

Code

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

Questions

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

Quiz

Code

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

Questions

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

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
```

Quiz

Code

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

Questions

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

Quiz

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]
```

Quiz

Code

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

Questions

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

Quiz

Code

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

Questions

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

C++

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

Quiz

Code

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

Questions

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

Quiz

Code

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

Questions

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

C

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

Quiz

Code

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

Questions

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

C++

In C++, a character literal is a `char`.

Quiz

Code

```
int
main(void)
{
    /* No previous decl! */
    quux();
}
```

Questions

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

Quiz

Code

```
int
main(void)
{
    /* No previous decl! */
    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`

Quiz

Code

```
int
main(void)
{
    /* No previous decl! */
    quux();
}
```

Questions

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

C++

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

Closing Thoughts

- These are not all differences!
  - Sometimes things just don't build.
  - Worse: Semantic (`const`, `inline` etc.)
- Almost everything is language version dependent.
- Almost everything is toolchain dependent.
- Idiomatic `C` is not idiomatic `C++`.
- Idiomatic `C++` is probably not valid `C`.
- Interfaces are the solution for interoperation.
  - Using `C` from `C++` is very easy.
  - Using `C++` from `C` is probably not.
- Compiling `C` with a `C++` compiler is not.

Thanks for your attention!