Options for safer void* handling

C allows implicit conversions between `void*` and other pointer types, as per section 6.3.2.3.1 of the standard. Making these implicit conversions explicit in Cforall would provide significant type-safety benefits, and is precedented in C++. A weaker version of this proposal would be to allow implicit conversions to `void*` (as a sort of "top type" for all pointer types), but to make the unsafe conversion from `void*` back to a concrete pointer type an explicit conversion. However, `int *p = malloc( sizeof(int) );` and friends are hugely common in C code, and rely on the unsafe implicit conversion from the `void*` return type of `malloc` to the `int*` type of the variable - obviously it would be too much of a source-compatibility break to disallow this for C code. We do already need to wrap C code in an `extern "C"` block, though, so it is technically feasible to make the `void*` conversions implicit in C but explicit in Cforall.

As a possible mitigation for calling C code with `void*`-based APIs, pointers-to-dtype are calling-convention compatible with `void*`; we could read `void*` in function signatures as essentially a fresh dtype type variable, e.g:

```
void* malloc( size_t )
    => forall(dtype T0) T0* malloc( size_t )
void qsort( void*, size_t, size_t, int (*)( const void*, const void* ) )
    => forall(dtype T0, dtype T1, dtype T2)
       void qsort( T0*, size_t, size_t, int (*)( const T1*, const T2* ) )
```

In this case, there would be no conversion needed to call `malloc`, just the polymorphic type binding. This should handle many of the uses of `void*` in C.

This feature would even allow us to leverage some of Cforall's type safety to write better declarations for legacy C API functions, like the following wrapper for `qsort`:

```
extern "C" { // turns off name-mangling so that this calls the C library
    // call-compatible type-safe qsort signature
    forall(dtype T)
    void qsort( T*, size_t, size_t, int (*)( const T*, const T* ) );

    // forbid type-unsafe C signature from resolving
    void qsort( void*, size_t, size_t, int (*)( const void*, const void* ) )
        = delete;
}
```