Built-in Types

Floral has multiple built-in types. These include various integers and characters, strings, booleans, arrays, pointers and functions. This document aims to outline the various syntax and nomenclature for writing types, and also aims to serve as a guide for the author for parsing types within the Floral parser.

Many of Floral's types are integer-based. These include:

- `Int` (signed 64-bit), also `Int64`
- `UInt` (unsigned 64-bit), also `UInt64`
- `Char` (signed 8-bit), also `Int8`
- `UChar` (unsigned 8-bit), also `UInt8`
- `Short` (signed 16-bit)
- `UShort` (unsigned 16-bit)
- `Int32` (signed 32-bit)
- `UInt32` (unsigned 32-bit)
- `Int64` (signed 64-bit)
- `UInt64` (unsigned 64-bit)
- `Bool` (signed 32-bit)

There are also a handful of types to represent real numbers. These include:

- `Float` (32-bit)
- `Double` (64-bit)

There are thoughts about introducing Half or Quad (alongside `Int128` and `UInt128`)

Most other Floral types and nearly all user-defined types will use the aforementioned built-in integral types. The other built-in types expand on the functionality provided by these types:

- Pointers: `&Type` (64-bit)
  - Char pointer: `&Char`, also `CString`
- Arrays: `[Type]` (3-bytes)
- Strings: `String` (3-bytes)
- Tuples: `(Type, …)` (variable)

Function pointers are denoted as follows:

- `Void > Void` (no parameters, no return)
- `Type > Void` (single parameter, no return)
- `Void > Type` (no parameters, return)
- `Type > Type` (single parameter, return)
- `(Type, …) > Void` (multiple parameters, no return)
- `(Type, …) > Type` (multiple parameters, return)

For ease of access, the size of any type in bytes can be evaluated though the use of the `sizeof` function. The `alignof` function returns the alignment of the type. For example, `sizeof(Bool) == 4` but on 64-bit architectures `alignof(Bool) == 8`.