Title: Generic Data Structures Without Name Mangling
Author: Marcus Johnson
Company: Mad Scientist
Date: March 11th 2025
Document: n3520
Proposal Category: New Feature
Audience: Compiler authors, Linker authors

Abstract:

Allow programmers to create and name generic structures.

Rationale:

C++'s templates are a useful feature in creating generic data structures and in reducing duplication, but the feature doesn't mesh well with C's philosophy or syntax.

Syntax:

```
typedef struct GenericString {
   size_t NumCodeUnits;
   auto *CodeUnits;
} _Generic(CodeUnits, char8_t:String8, char16_t:String16, char32_t:String32);
```

Example:

```
size_t GetNumCodeUnits(GenericString *Generic) {
   return Generic->NumCodeUnits;
}

char8_t *GetCodeUnits(String8 *String) {
   return String->CodeUnits;
}

char8_t GetCodeUnit8(String8 *String, size_t Index) {
   assert(Index < String->NumCodeUnits);
   return String->CodeUnits[Index];
}

char16_t GetCodeUnit16(String16 *String, size_t Index) {
      assert(Index < String->NumCodeUnits);
   return String->CodeUnits[Index];
}

char32_t GetCodeUnit32(String32 *String, size_t Index) {
      assert(Index < String->NumCodeUnits);
```

```
    return String->CodeUnits[Index];
}

#define GetCodeUnit(String, Index) _Generic(String, String8:GetCodeUnit8,
String16:GetCodeUnit16, String32:GetCodeUnit32)(String, Index)

String8 *Generic = u8"Literal";

char8_t GenericChar = GetCodeUnit(Generic, 3); // GenericChar == 'e'
```