Rvalue Arrays
N835/J11 98-034
June 18, 1998
Randy Meyers

1. Introduction

A year or two ago, the committee discussed that certain array expressions were fairly useless since the committee was overly specific in defining the conversion of arrays to pointer types. There was general agreement that the Standard was incorrect, but no one ever drafted new words. This paper provides the new words.

2. Example

```
struct S {int a[2];};
struct S f(void)
{
  struct S x;
  x.a[0] = x.a[1] = 1;
  return x;
}
…
printf("answer is %d\n", f().a[0]);
```

The expression **`f().a[0]`** is undefined. The **`[]`** operator requires a pointer operand (Subclause 6.5.2.1, paragraph 1). However, **`f().a`** does not undergo the usual conversion of array to pointer since Subclause 6.3.2.1 Paragraph 3 says:

Except when it is the operand of the **`sizeof`** operator or the unary **`&`** operator, or is a character string literal used to initialize an array of character type, or is a wide string literal used to initialize an array with element type compatible with **`wchar_t`**, an lvalue that has type ‘‘array of *type*’’ is converted to an expression that has type ‘‘pointer to *type*’’ that points to the initial element of the array object and is not an lvalue. If the array object has register storage class, the behavior is undefined.

The expression **`f().a`** is not an lvalue since the member selection operator is only an lvalue if its left operand is an lvalue, and a function call is not an lvalue.

Note that C++ does not have this problem. Subclause 4.2 of the C++ Standard says:

An lvalue or rvalue of type "array of **`N T`**" or "array of unknown bound of **`T`**" can be converted to an rvalue of type "pointer to **`T`**." The result is a pointer to the first element of the array.

3. Change to C9x

Change Subclause 6.3.2.1 Paragraph 3 to:

Except when it is the operand of the **`sizeof`** operator or the unary **`&`** operator, or is a character string literal used to initialize an array of character type, or is a wide string literal used to initialize an array with element type compatible with **`wchar_t`**, an expression that has type ‘‘array of *type*’’ is converted to an expression that has type ‘‘pointer to *type*’’ that points to the initial element of the array object and is not an lvalue. If the array object has register storage class, the behavior is undefined.

(The above edit changes “lvalue” in the original text to “expression.”)

The above change permits pointers to be formed to rvalue arrays in structs, and those pointers might be used to form lvalues to store into such arrays or access the array after its natural lifetime unless we plug the holes. The holes occur with any operator that yields an rvalue struct. The following changes plug the holes.

Add a new paragraph after paragraph 10 in Subclause 6.5.2.2 Function calls:

If an attempt is made to modify the result of a function call, the behavior is undefined. If an attempt is made to access the result of a function call after the next sequence point, the behavior is undefined.

Add a new paragraph after paragraph 6 in Subclause 6.5.15 Conditional Operator:

If an attempt is made to modify the result of a conditional operator, the behavior is undefined. If an attempt is made to access the result of a conditional operator after the next sequence point, the behavior is undefined.

Add a new paragraph after paragraph 3 in Subclause 6.5.16 Assignment operators:

If an attempt is made to modify the result of a assignment operator, the behavior is undefined. If an attempt is made to access the result of a assignment operator after the next sequence point, the behavior is undefined.

Add a new paragraph after paragraph 2 in Subclause 6.5.17 Comma operator:

If an attempt is made to modify the result of a comma operator, the behavior is undefined. If an attempt is made to access the result of a comma operator after the next sequence point, the behavior is undefined.

ERROR: undefined
OFFENDING COMMAND: --------------2D72C25CFEC5DC079EB98954--

STACK: