# Notation for "reference to array" type

Asked 1 year, 3 months ago  Modified 1 year, 3 months ago  Viewed 66 times

1

In c++, type of a reference to array is shown like `int (&) [10]`. As an example, when I try to compile with g++ and clang the following code

```
template <typename T> void foo(T&);

int main() {
    int a[] {1, 2, 3};

    foo(a);
}
```

I get the following error:

```
undefined reference to `void foo<int [3]>(int (&) [3])'
```

In this error text, why is the type of argument shown as `int (&) [3]`? Why don't we denote array reference types like references to integral types or class types, i.e. `int [10] &`? What is the reason for using `(&)`?

I know we can define a 'reference to array' variable like this:

```
int (&b)[3] = a;
```

And this definition indeed 'looks like' the type of b. But is this the only reason? Is there a problem related to the notation `int [10] &`?

c++ g++ clang

Share Edit Follow Flag

edited Jan 10, 2021 at 21:24
dreamcrash
40.2k 23 72 99

asked Jan 9, 2021 at 12:48
Cem
1,156 2 15

@bloody There is no definition. I do not write the definition on purpose, to see the type of argument T& as an error. This method is from the book "Effective Modern C++" by Scott Meyers. – Cem Jan 9, 2021 at 12:55

1 But the type `T&` *isn't* an error? Is your question simply why the syntax isn't (e.g.) `void bar(int[10] &array)`? – Some programmer dude Jan 9, 2021 at 13:00

You can spare yourself some build time if you define the function as `= delete;`. Won't have to wait on the linker. – StoryTeller - Unslander Monica Jan 9, 2021 at 13:01

@Someprogrammerdude I only wonder the reason for the notation, i.e. why is the type shown as `int (&) [3]` and not `int [3] &`. – Cem Jan 9, 2021 at 13:04

1 Probably because it's simply not correct syntax. It would be rather bad when a compiler shows an error message containing invalid syntax. – Some programmer dude Jan 9, 2021 at 13:05

You are right. I thought initially that since this was a reference to `int[3]`, it was more natural to denote it as `int[3]&`. But as I was answering the comments it has become more and more clear to me that the type actually should look like `int (&) []`, since arrays are not declared as `int[30] array` but `int array[30]`. So this is probably more consistent with the rest of language. – Cem Jan 9, 2021 at 13:09

## 1 Answer

Sorted by: Highest score (default)

1

It comes from how the type of pointers to arrays look like: `int(*)[10]`. The `*` is just replaced by `&`, like in all reference types.

The reason that pointers to arrays look like that is how it looks like in C, and C++ had no reason to change it.

```
int (*a)[10];
// "(*a)[10]" is an int
// The type of `a` is `int (*)[10]`, just remove the name
```

I don't see any technical reasons why `int[10]*` and `int[10]&` would not be possible as the name of the types for "pointer or reference to an array of 10 `int`", other than compatibility with C

Share Edit Follow Flag

answered Jan 9, 2021 at 13:06
Artyer
19.8k 2 36 57

I see. ANSI C Standard actually simply says that `int (*) [3]` will be a pointer to array of `int`s. It was merely a decision by the C standard committee then. Thank you. – Cem Jan 9, 2021 at 13:22