An `enum` with a discriminant must specify a `#[repr(inttype)]`.

Erroneous code example:

```
#![feature(arbitrary_enum_discriminant)]

enum Enum { // error!
    Unit = 1,
    Tuple() = 2,
    Struct{} = 3,
}
# fn main() {}
```

A `#[repr(inttype)]` must be provided on an `enum` if it has a non-unit variant with a discriminant, or where there are both unit variants with discriminants and non-unit variants. This restriction ensures that there is a well-defined way to extract a variant's discriminant from a value; for instance:

```
#![feature(arbitrary_enum_discriminant)]

#[repr(u8)]
enum Enum {
    Unit = 3,
    Tuple(u16) = 2,
    Struct {
        a: u8,
        b: u16,
    } = 1,
}

fn discriminant(v : &Enum) -> u8 {
    unsafe { *(v as *const Enum as *const u8) }
}

fn main() {
    assert_eq!(3, discriminant(&Enum::Unit));
    assert_eq!(2, discriminant(&Enum::Tuple(5)));
    assert_eq!(1, discriminant(&Enum::Struct{a: 7, b: 11}));
}
```