

Programming While You Wait

Const Correctness in C++

The `const` keyword is used to indicate that the data being referred to **should not change** beyond its initial initialization. It is used as a reference for the programmer indicating what they may do with the data, and how it should be used..

Your policy should be to **use `const` everywhere**, only removing the keyword when it becomes apparent that the data must be altered for the program to work *correctly*.

Const Variables and Parameters

Replace magic numbers with `const` variables. This takes advantage of type checking, and allows the scope to be reduced if needed.

```
#define SIZE      2    // Nasty #define to remove a magic number
const uint SIZE = 2;  // const with benefits of type checking etc.
```

Pass `const` variables and return `const` values to explicitly indicate what the method expects and what it does. A `const` parameter cannot change in the method that uses it.

```
// The 'player' object is guaranteed to be the same after the call to IsWithinRange
const bool IsWithinRange(const CPlayer& player);
```

Const Data Members

`Const` data members allows us to place information within the scope of a class while still having all the benefits of type checking, scope reduction and avoiding `#defined` values.

```
class CPlayer
{
    static const uint maxNumberOfWeapons = 10;
};
```

Const Pointers

```
CPlayer*      player; // Every thing and anything can change – not safe
const CPlayer* player; // The object contents are const – object cannot change
CPlayer* const player; // The pointer itself is const – the object can change
const CPlayer* const player; // Both the object and pointer are const
```

Const Methods

`Const` methods tell us that the method will not alter the *visible* structure of the object.

```
CPlayer::GetName(); // This function can do anything to the object
CPlayer::GetName() const; // Function guarantees object won't change in a visible way
```

Summary

- `Const` correctness helps **improve the readability and usability** of your code
- Code is more **secure and reliable** as it is used in ways it was intended
- Errors in code will be **detected at compile time** and not as runtime errors

