# Objectives - Tue 3/22/2022

- Const
- Static
- Linked lists

# const

```
int const n = 5;   // n is a read-only value
n = 7;   // ✗
// "assignment of read-only variable 'a_n'"
```

**«TYPE» const «NAME»**

**Const makes whatever *variable* comes after it read-only.**

*We will expand this into a stronger statement before we're done here.*

```
const int n = 5;   // w is a read-only value (same as above)
```

**const «TYPE»** **…is equivalent to…** **«TYPE» const**

Rule: You can switch the position of const and a type name that is directly adjacent to const.

# const*

```
int const* a_n = ░; // *a_n is read-only

const int* a_n = ░; // *a_n is read-only

a_n = &q;   // 👌

*a_n = 4;   // ✗ "assignment of read-only location '*a_n'"


int* const a_n = ░; // a_n is read-only

a_n = &q;   // ✗ "assignment of read-only variable 'a_n'"

*a_n = 4;   // 👌
```

# const**

```
int const** a_a_n = ░; // **a_a_n is read-only

const int** a_a_n;   // Same as above


a_a_n = &a_q; // 👌

*a_a_n = &q;  // 👌

**a_a_n = 4;  // ✗ "assignment of read-only location '*a_n'"
```

# const***

```
int const*** a_a_a_n = ░;    // **a_a_a_n is read-only

const int*** a_a_a_n; // " (same as above)

a_a_a_n = &a_a_q; // 👌

*a_a_a_n = &a_q;  // 👌

**a_a_a_n = &q;    // 👌

***a_a_a_n = 4;    // ✗
// ERROR: "assignment of read-only location '***a_a_a_n'"
```

## const***

```
int const*** a_a_a_n = ░;     // ***a_a_a_n is read-only

int* const** a_a_a_n = ░;     // **a_a_a_n is read-only

int** const* a_a_a_n = ░;     // *a_a_a_n is read-only

int*** const a_a_a_n = ░;     // a_a_a_n is read-only
```

**Const makes the variable—including all * <u>after</u> the const—read-only.**

# const* const

```
int const* const a_n; // *a_n and a_n are read-only

const int* const a_n; // *a_n and a_n are read-only (same)

a_n = ░;    // ✗ assignment of read-only variable 'a_n'

*a_n = ░;   // ✗ assignment of read-only location '*a_n'
```

# const* const*

```
int const* const* a_n; // **a_n and *a_n are read-only

const int* const* a_n; // **a_n and *a_n are read-only (same)

*a_n = ░;   // ✗ assignment of read-only location '*a_n'

**a_n = ░; // ✗ assignment of read-only location '**a_n'
```

# const* const* const

```
int const* const* const a_n; // **a_n,*a_n, a_n are read-only

const int* const* const a_n; // (same as ↑)

a_n = ░;     // ✗ assignment of read-only variable 'a_n'

*a_n = ░;    // ✗ assignment of read-only location '*a_n'

**a_n = ░;   // ✗ assignment of read-only location '**a_n'
```

## Rules

- Const is a promise you can't break [*].

  [*] Okay, there are tricks, but let's not go there.

- Const makes the variable—including all `*` <u>after</u> the const—read-only.

- `const` *«TYPE»* ⇔ *«TYPE»* `const`
  - You can switch the position of const and a type name that is directly adjacent to const.