Types and Pointers

C++, as we all know by now is a strongly, statically typed language. All variables must have a type and types cannot be changed at runtime.

```
int x = 10;
```

Auto Types

Even when we use the ‘auto‘ type we are still getting a strongly typed variable it is just decided by the compiler instead of us. This often saves time but where types can be ambiguous.

a proper use of auto is when the type the compiler will assign you is obvious

```
auto x = new std::vector<int>();
```

the return type of this expression is obviously std::vector<int>* so typing it all out is redundant.

Unclear Auto Usage

This function isnt explicit in its return type from what it is named so the auto return value is going to be hard to figure out unless you go to the definition of getValues/0

```
std::vector<int>* getValues() {
  // return values
}

auto vals = getValues();
```

It is often best to manually set the type of values that are returned from functions unless doing so would be redundant. Often times we over use auto because fancy editors will tell us with a simple hover what the type is. Never assume another developer has the same editor settings as you unless you have jedi mind tricked them into using your setup.

Proper type casting

When we have types that are A) set when the variable is declared and cant change we often times need to convert one type to another to satisfy an Apia

C style

C style casting is the old favorite from a bygone era.

```
int a = 10;
float f = (int)a;
```