**C++ Templates**

Templates are reusable 🡪 faster coding

1. **Function Template**

Say we want to implement swap function for all types

Option 1: Write a function for each type (nah)

Option 2: Write a template where the type can change

A template is a C++ tool that allows programmers to add data types as parameters.

* **Templates declaration and implementations must entirely placed in the header files**.
* Templates let us choose the type implementation right when you call the function.
* The type we choose may apply to the return type, a parameter type, or both.

template <typename Type>

void function\_name (Type var) {

// Function body

}

Then, we can call the function for any type of parameter passed, as long as it works with the function body.

**Note:** Templates will slow down the program’s compile time, but speed up the execution time.

1. **Class Templates**

Applying templates to Classes. For example, we want a type Pair that can have 2 members as a key-value pair, with the key-value can have any types.

**Graphical user interface, text, application, Teams

Description automatically generatedDeclaration and Implementation**

Text

Description automatically generated

**Using Templates**

When using templates for classes, we **must** specify the types of the variables while declaring.

pair<int, float> bruh1 (1, 2.5f);

pair<std::string, double> bruh2 ("Mario", 48.5);

**In C++, there is a Standard Template Library (C++ STL), using the concept of templates to create useful containers (pair, vectors, queue, stack, list, etc.)**

* **This is why containers can store various types (int, float, objects, other containers, etc)**