Chapter 6 introduces templates, which are a C++ feature that easily permits the reuse of existing code for new purposes.

This presentation shows how to implement and use the simplest kinds of templates: template functions.

Finding the Maximum of Two Integers

- Here’s a small function that you might write to find the maximum of two integers.

```cpp
int maximum(int a, int b) {
    if (a > b) return a;
    else return b;
}
```

Finding the Maximum of Two Doubles

- Here’s a small function that you might write to find the maximum of two double numbers.

```cpp
int maximum(double a, double b) {
    if (a > b) return a;
    else return b;
}
```

Finding the Maximum of Two Knafns

- Here’s a small function that you might write to find the maximum of two Knafns.

```cpp
int maximum(Knafn a, Knafn b) {
    if (a > b) return a;
    else return b;
}
```

One Hundred Million Functions...

- Suppose your program uses 100,000,000 different data types, and you need a maximum function for each...

A Template Function for Maximum

- This template function can be used with many data types.

```cpp
template <class Item>
Item maximum(Item a, Item b) {
    if (a > b) return a;
    else return b;
}
```
A Template Function for Maximum

When you write a template function, you choose a data type for the function to depend upon...

```cpp
template <class Item>
Item maximum(Item a, Item b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

A Template Function for Maximum

A template prefix is also needed immediately before the function’s implementation:

```cpp
// A template prefix is also needed immediately before the function’s implementation:

template <class Item>
Item maximum(Item a, Item b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

Using a Template Function

Once a template function is defined, it may be used with any adequate data type in your program...

```cpp
// Once a template function is defined, it may be used with any adequate data type in your program:

template <class Item>
Item maximum(Item a, Item b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

Finding the Maximum Item in an Array

Here’s another function that can be made more general by changing it to a template function:

```cpp
int array_max(int data[], size_t n)
{
    size_t i;
    int answer;
    assert(n > 0);
    answer = data[0];
    for (i = 1; i < n; ++i)
        if (data[i] > answer) answer = data[i];
    return answer;
}
```

Finding the Maximum Item in an Array

Here’s the template function:

```cpp
// Here’s the template function:

template <class Item>
Item array_max(Item data[], size_t n)
{
    size_t i;
    Item answer;
    assert(n > 0);
    answer = data[0];
    for (i = 1; i < n; ++i)
        if (data[i] > answer) answer = data[i];
    return answer;
}
```

Summary

A template function depends on an underlying data type.

More complex template functions and template classes are discussed in Chapter 6.