Using compiler-generated int swapper:
Now i, j = 20, 10.
Original arrays:
07/04/1776
07/20/1969
Swapped arrays:
07/20/1969
07/04/1776

Explicit Specializations

Suppose you define a structure like the following:

```cpp
struct job
{
    char name[40];
    double salary;
    int floor;
};
```

Also, suppose you want to be able to swap the contents of two such structures. The original template uses the following code to effect a swap:

```cpp
temp = a;
a = b;
b = temp;
```

Because C++ allows you to assign one structure to another, this works fine, even if type Any is a job structure. But suppose you only want to swap the salary and floor members, keeping the name members unchanged. This requires different code, but the arguments to Swap() would be the same as for the first case (references to two job structures), so you can't use template overloading to supply the alternative code.

However, you can supply a specialized function definition, called an explicit specialization, with the required code. If the compiler finds a specialized definition that exactly matches a function call, it uses that definition without looking for templates.