Part E - Polymorphism   
  
**Function Templates**   
  
Workshop 10

In this workshop, you will code a function template for validating a value between a minimum and maximum range.

**Learning Outcomes**

Upon successful completion of this workshop, you will have demonstrated your abilities to

* code a function template
* code an explicit specialization for a function template
* reflect on what you have learned from this workshop

**SUBMISSION POLICY**

The “in-lab” section is to be completed **during your assigned lab section**. It is to be completed and submitted by the end of the workshop. If you do not attend the workshop, you can submit the “in-lab” section along with your “at-home” section (a 20% late deduction will be assessed). The “at-home” portion of the lab is **due the day before your next scheduled workshop.**

**In lab – VALIDATE function**

Create a validate function that accepts three arguments (a minimum, a maximum, and a test-value) and returns a bool that is true only if the test-value is between the minimum and maximum arguments, and false otherwise, that is:

Return true if (testValue >= minimum && maximum >= testValue), return false otherwise

The prototype for the function should be as follows

|  |
| --- |
| **bool validate(const T& minimum, const T& maximum, const T& testValue);** |

Template

Store a template for your function is a header file named **Validate.h**.

Explicit Specialization

Code an explicit specialization of your template for **char** inputs, which ignores the case of all characters when comparing their values.

Note: int toupper(int) is available in <ctype.h>

Add your specialization to the header file named **Validate.h.**

Client Module

The main program that uses your implementation is listed below:

// OOP244 Workshop 10 - Templates

// File: w10\_in\_lab.cpp

// Version: 1.0

// Date: 2015/12/01

// Author: Fardad Soleimanloo

// Description:

// This file tests in-lab section of your workshop

/////////////////////////////////////////////

#include <iostream>

#include "Validate.h"

using namespace sict;

int main() {

int t1Values[] = { 28, 44, 12, 30 };

int t1Max = 30;

int t1Min = 20;

double t2Values[] = { 28.0, 22.5, 15.8, 21.6 };

double t2Max = 25.0;

double t2Min = 20.0;

char t3Values[] = { 'a', 'Z', 'M', 'm' };

char t3Max = 'Q';

char t3Min = 'g';

for (int i = 0; i < 4; i++)

{

if (validate(t1Min, t1Max, t1Values[i])){

std::cout << t1Values[i] << " is valid" << std::endl;

}

else {

std::cout << t1Values[i] << " is invalid" << std::endl;

}

if (validate(t2Min, t2Max, t2Values[i])){

std::cout << t2Values[i] << " is valid" << std::endl;

}

else {

std::cout << t2Values[i] << " is invalid" << std::endl;

}

if (validate(t3Min, t3Max, t3Values[i])){

std::cout << t3Values[i] << " is valid" << std::endl;

}

else {

std::cout << t3Values[i] << " is invalid" << std::endl;

}

}

std::cout << std::endl;

return 0;

}

Output:

**28 is valid  
28 is invalid  
a is invalid  
44 is invalid  
22.5 is valid  
Z is invalid  
12 is invalid  
15.8 is invalid  
M is valid  
30 is valid  
21.6 is valid  
m is valid**

**in-lab SUBMISSION (80%)**

If not on matrix already, upload **Validate.h** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account:

**Sections SAA and SBB:  
~fardad.soleimanloo/submit w10\_in\_lab <ENTER>**

**VALIDATE with a class – AT Home**

Without modifying Validate.h do the following:

Create a Weather class (in Weather.h and Weather.cpp) that stores a high temperature as a float, and has a one argument constructor that accepts a float that will be used to initialize the high temperature.

Define the proper comparison operators (<=, <, etc…) so your Weather object can work with the validate function template using the main function below

Client Module

// OOP244 Workshop 10 - Templates

// File: w10\_at\_home.cpp

// Version: 1.0

// Date: 2015/12/01

// Author: Fardad Soleimanloo

// Description:

// This file tests in-lab section of your workshop

/////////////////////////////////////////////

#include <iostream>

#include "Validate.h"

#include "Weather.h"

using namespace sict;

int main() {

Weather min(-30.0);

Weather max(50.0);

Weather test1(-100.0);

Weather test2(0.0);

Weather test3(100.0);

if (validate(min, max, test1)){

std::cout << "test1 is valid" << std::endl;

}

else {

std::cout << "test1 is invalid" << std::endl;

}

if (validate(min, max, test2)){

std::cout << "test2 is valid" << std::endl;

}

else {

std::cout << "test2 is invalid" << std::endl;

}

if (validate(min, max, test3)){

std::cout << "test3 is valid" << std::endl;

}

else {

std::cout << "test3 is invalid" << std::endl;

}

return 0;

}

Ouput:

test1 is invalid

test2 is valid  
test3 is invalid

**AT-Home SUBMISSION (20%)**

If not on matrix already, upload **Validate.h, Weather.h and Weather.cpp** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account:

**Sections SAA and SBB:  
~fardad.soleimanloo/submit w10\_at\_home <ENTER>**