C++ Class Templates ☆

31/44 challenges solved

Rank: 6470 | Points: 605

Your C++ Class Templates submission got 20.00 points. Share Tweet

Try the next challenge | Try a Random Challenge

Problem Submissions Leaderboard Editorial

A class template provides a specification for generating classes based on parameters. Class templates are generally used to implement containers. A class template is instantiated by passing a given set of types to it as template arguments. Here is an example of a class, MyTemplate, that can store one element of any type and that has just one member function divideBy2, which divides its value by 2.

```
template <class T>
class MyTemplate {
T element;
public:
MyTemplate (T arg) {element=arg;}
T divideBy2 () {return element/2;}
};
```

It is also possible to define a different implementation of a template for a specific type. This is called Template Specialization. For the template given above, we find that a different implementation for type char will be more useful, so we write a function printElement to print the char element:

```
// class template specialization:
template <>
class MyTemplate <char> {
char element;
public:
MyTemplate (char arg) {element=arg;}
char printElement ()
{
return element;
}
};
```

You are given a main() function which takes a set of inputs. The type of input governs the kind of operation to be performed, i.e. concatenation for strings and addition for int or float. You need to write the class template AddElements which has a function add() for giving the sum of int or float elements. You also need to write a template specialization for the type string with a function concatenate() to concatenate the second string to the first string.

Input Format

The first line contains an integer n. Input will consist of $n+1$ lines where n is the number given in the first line of the input followed by n lines.

Each of the next n line contains the type of the elements provided and depending on the type, either two strings or two integers or

two floating point numbers will be given. The type will be one of int, float or string types only. Out of the following two elements, you have to concatenate or add the second element to the first element.

Constraints

$1 \leq n \leq 5 \times 10^5$

$1.0 \leq value_{float} \leq 10.0$, where $value_{float}$ is any float value

$1 \leq value_{int} \leq 10^5$, where $value_{int}$ is any int value

$0 \leq len_{string} \leq 10$, where len_{string} is the length of any string

The time limit for this challenge is 4 seconds.

Output Format

The code provided in the code editor will use your class template to add/append elements and give the output.

Sample Input

```
3
string John Doe
int 1 2
float 4.0 1.5
```

Sample Output

```
JohnDoe
3
5.5
```

Explanation

"Doe" when appended with "John" gives "JohnDoe". 2 added to 1 gives 3, and 1.5 added to 4.0 gives 5.5.

C++14

```
    std::is_same<Type, int>::value;

template<typename Type> class AddElements final
{
    static_assert(allowed_types<Type>,
        "This type is not allowed. Only (double and int are allowed)");
    Type _var1;
public:
    explicit constexpr AddElements(const Type &var1) noexcept
        : _var1{ var1 }
    {}

    constexpr Type add(const Type &var2) const noexcept { return _var1 + var2 ; }
};

template<> class AddElements<std::string> final
{
    std::string _var1;
public:
    explicit AddElements(const std::string& var1) noexcept
        : _var1{ std::move(var1) }
```

```
    {}

    std::string concatenate(const std::string& var2) const noexcept { return _var1 + var2; }
};


int main () {
  int n,i;
  cin >> n;
  for(i=0;i<n;i++) {
    string type;
    cin >> type;
    if(type=="float") {
        double element1,element2;
        cin >> element1 >> element2;
        AddElements<double> myfloat (element1);
```

Line: 19 Col: 13

Upload Code as File Test against custom input Run Code Submit Code

You have earned 20.00 points!

31/44 challenges solved.

70%

Congratulations

You solved this challenge. Would you like to challenge your friends? Next Challenge

Test case 0 ✓

Test case 1 ✓

Test case 2 ✓

Test case 3 ✓

Test case 4 ✓

Test case 5 ✓

Test case 6 ✓

Compiler Message

Success

Input (stdin) Download

```
3
string John Doe
int 1 2
float 4.0 1.5{-truncated-}
```

Download to view the full testcase

Expected Output Download

```
JohnDoe
3
5.5{-truncated-}
```