

Lab 6.4.2 Interfaces and virtual functions: part 2

Objectives

Familiarize the student with:

- implementing interfaces according to specifications;
- polymorphism, or using objects of different types through a common interface.

Scenario

Now that we have a basic validator interface ready, we can practice using the interface.

Create the following validator classes:

- `MinLengthValidator`, which will consider a string valid only if it's longer than the required minimum;
- `MaxLengthValidator`, which will consider a string valid only if it's shorter than the required maximum;
- `PatternValidator`, which will consider a string valid only if **at least part of the string** matches the supplied pattern. To remind you, here are the rules we defined earlier regarding patterns:
 - a pattern will consist of non-whitespace characters;
 - the letter 'D' will match any decimal digit, so the pattern "DDDD" will match for strings "1234", "2309" etc.;
 - the letter 'A' will match any character of the English alphabet (upper and lower case), so the pattern "AAA" will match for strings "CAT", "dog", "ToC" etc.;
 - lower-case letters in a pattern will match according letters of the English alphabet, so the pattern "cat" will match for strings "Cat", "cat" "CAT", etc.;
 - The character '?' will match every character, including whitespace, so the pattern "a?b" will match for strings "A+B", "a0b", "Acb", "a B", etc.
 - Any punctuation except '?' will match exactly the same punctuation in a string, so the pattern "AA-DDD" will match for strings "NE-785", "am-236", etc.

```

#include <iostream>
#include <string>

class StringValidator
{
public:
    virtual ~StringValidator() {};
    virtual bool isValid(std::string input) = 0;
};

// Your code here

using namespace std;

void printValid(StringValidator "validator, string input)
{
    cout << "The string '" << input << "' is "
        << (validator.isValid("hello") ? "valid" : "invalid");
}

int main()
{
    cout << "MinLengthValidator" << endl;
    MinLengthValidator min(6);
    printValid(min, "hello");
    printValid(min, "welcome");
    cout << endl;

    cout << "MaxLengthValidator" << endl;
    MaxLengthValidator max(6);
    printValid(max, "hello");
    printValid(max, "welcome");
    cout << endl;

    cout << "PatternValidator" << endl;
    PatternValidator digit("D");
    printValid(digit, "there are 2 types of sentences in the world");
    printValid(digit, "valid and invalid ones");
    cout << endl;

    return 0;
}

```

Example output

```

MinLengthValidator
The string 'hello' is invalid
The string 'welcome' is valid

MaxLengthValidator
The string 'hello' is valid
The string 'welcome' is invalid

PatternValidator
The string 'there are 2 types of sentences in the world' is valid
The string 'valid and invalid ones' is invalid

```