

Lab 6.4.1 Interfaces and virtual functions: part 1

Objectives

Familiarize the student with:

- using virtual functions to define the desired interfaces to be filled by objects;
- using objects through interfaces instead of concrete implementations.

Scenario

Let's get back to the idea of password validation.

We're going to create a general interface to validate a piece of text.

Look at the code below and then implement a validator class that will take a single string in the constructor and consider the input valid if and only if it matches that string exactly.

```
#include <iostream>
#include <string>

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

// Write your code here

class DummyValidator : public StringValidator {
public:
    virtual bool isValid(std::string input);
};

bool DummyValidator::isValid(std::string input)
{
    return true;
}

using namespace std;

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

int main()
{
    DummyValidator dummy;
    printValid(dummy, "hello");
    cout << endl;

    ExactValidator exact("secret");
    printValid(exact, "hello");
    printValid(exact, "secret");
    return 0;
}
```

Example output

```
The string 'hello' is valid

The string 'hello' is invalid
The string 'secret' is valid
```