|  |  |
| --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  РОССИЙСКОЙ ФЕДЕРАЦИИ  Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный исследовательский университет» | |
|  | Институт компьютерных наук и технологий |
| ОТЧЕТ  по лабораторной работе **№5**  по дисциплине Языки программирования  «**ООП**» | |
|  | Работу выполнила студентка  Группы ПМИ 1-2, курс 2  Института компьютерных наук и технологий  **Арланова Анна Андреевна** |
|  | Работу принял:  ассистент кафедры Информационных технологий  Ракина В.Д. |
| Пермь, 2023 | |

Оглавление

[1 Постановка задачи 3](#_Toc152648357)

[2 Проектирование 4](#_Toc152648358)

[2.1 Класс со строковым полем 4](#_Toc152648359)

[2.2 Класс Money 6](#_Toc152648360)

[3 Тестирование 8](#_Toc152648361)

[3.1 Класс со строковым полем 8](#_Toc152648362)

[3.2 Класс Money 10](#_Toc152648363)

# Постановка задачи

Написать 2 программы на С#, в которых необходимо:

1. Разработать класс с одним строковым полем. Создать конструктор копирования. Разработать метод, приписывающий к полю в начало три знака восклицания. Перегрузить метод ToString() для формирования строки из полей класса. Реализовать дочерний класс (его содержание предложить самостоятельно и описать в решении: какой содержательный смысл имеют поля; реализовать конструкторы; предложить и реализовать 2-3 метода). Протестировать все конструкторы и другие методы базового и дочернего классов.
2. Реализовать определение класса Money (поля: uint rubles и byte kopek, свойства конструкторы, перегрузка метода ToString() для вывода полей).

Реализовать методы:

* 1. Добавление произвольного количества копеек (uint) к объекту типа Money. Результат должен быть типа Money.
  2. Унарные операции:
     1. ++ добавление копейки к объекту типа Money
     2. -- вычитание копейки из объекта типа Money
  3. Операции приведения типа:
     1. uint (явная) – результатом является количество рублей (копейки отбрасываются);
     2. double (неявная) – результатом является копейки, переведенные в рубли, целые рубли отбрасываются, результат <1.
  4. Бинарные операции:
     1. + Money m, беззнаковое целое число (лево- и правосторонние операции). Результат должен быть типа Money.
     2. – Money m, беззнаковое целое число (лево- и правосторонние операции). Результат должен быть типа Money.

Протестировать все методы, включая конструкторы

# Проектирование

## Класс со строковым полем

Разделила программу на 3 класса:

1. StringMaker.cs
2. Register.cs
3. Program.cs

**Класс StringMaker:**

Поля:

* data: строка

Конструкторы:

* Без параметров: строка будет иметь значение «qwerty»
* С параметром string: строка будет иметь значение указанного параметра string
* Копирования: создается копия объекта с этим же классом, полями

Методы:

* AddThreeExclamation: Добавление трех знаков восклицания в начало строки

ToString:

* Возвращает строку, без всякого форматированного поля

Реализовав такой класс, решила, что будет уместно создать дочерний класс Register. Который будет отвечать за регистрацию пользователя на сайте со странными требованиями для регистрации и насильно предлагающий альтернтивные логины и пароли.

**Класс Register:**

Поля:

* строка

Конструкторы:

* Без параметров: строка будет иметь значение от метода RandomLogin
* С параметром string: строка будет иметь значение указанной string

Методы:

* CheckComplexity: Проверяет сложность пароля, если в нем меньше трех знаков восклицания создает новый пароль на основе старого, применяя метод родительского класса AddThreeExclamation.Если пароль сложный (3 и более знаков восклицания), то ничего не делает с паролем.
* CheckAllowChar: Проверяет содержится ли в логине запрещенные символы (кирилица). Если обнаруживается кирилица, то заменяет логин на новый при помощи метода RandomLogin
* RandomLogin: Возвращает строку формата «userXXXX», где XXXX это рандомные цифры

ToString:

* Возвращает строку в формате:

Логин: {login}

Пароль: {password}

**Класс Program:**

Описание: для создание объектов, манипуляции над ними и вывода

## Класс Money

Разделила программу на 3 класса:

1. Money.cs
2. ReadFromUser.cs
3. Program.cs

**Класс Money:**

Поля:

* rubles: безнаковое целое число размером в 4 байт
* kopeeks: безнаковое целое число размером в 1 байт

Свойства:

* Rubles: получает и устанавливает текущее значение для rubles
* Kopeeks: получает текущее значение для kopeeks. Устанавливает значение для kopeeks и rubles в зависимости от значения. Если больше или равно 100, то берется устанавливается в качестве значение его остаток от деления на 100, так же на rubles устанавливается значение равное целочисленному делению kopeeks на 100.

Конструкторы:

* Без параметров: оба поля будет иметь значение 0
* С параметрами uint и byte: поля будут иметь значения в зависимости от их свойст

Методы:

* AddKopeeks: Подсчитывает общее количество копеек с учетом параметра метода, затем подсчитывает количество копеек и рублей

Перегруженные операторы:

* Оператор инкремента ++: Этот оператор увеличивает количество копеек в объекте money на 1, возвращая новый объект Money с измененным значением.
* Оператор декремента --: Этот оператор уменьшает количество копеек в объекте money на 1. Если количество копеек становится равным 0, то проверяется количество рублей. Если и рублей тоже 0, то возвращается новый объект Money с нулевыми значениями, иначе уменьшается количество рублей на 1, а количество копеек становится равным 99.
* Оператор сложения Money + Число: Этот оператор позволяет складывать объект money с целым числом (uint), увеличивая количество рублей.
* Оператор сложения Число + Money: Этот оператор аналогичен предыдущему, но позволяет складывать целое число (uint) с объектом money.
* Оператор вычитания Money - Число: Этот оператор позволяет вычитать из объекта money целое число (uint), уменьшая количество рублей. Если результат становится отрицательным, возвращается новый объект Money с нулевыми значениями.
* Оператор вычитания Число - Money: Этот оператор аналогичен предыдущему, но позволяет вычитать из целого числа (uint) объект money. Если результат становится отрицательным, возвращается новый объект Money с нулевыми значениями.

Преобразования:

* Явное преобразование в uint: Этот метод позволяет явно (компилятор не может сам преобразовать значение из одного типа в другой, программист должен явно это указать) преобразовывать объект Money в тип uint, возвращая количество рублей.
* Неявное преобразование в double: Этот метод позволяет неявно (компилятор сам преобразует значение из одного типа в другой) преобразовывать объект Money в тип double, представляя количество копеек в виде десятичной дроби.

ToString:

* Возвращает строку с форматированием: {Rubles} рублей и {Kopeeks} копеек

**Класс** ReadFromUser**:**

Методы:

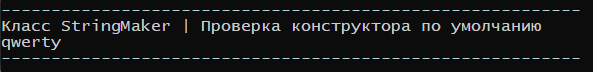
* UInt: запрашивает у пользователя ввод положительного числа типа uint. Функция пытается преобразовать введенное значение в uint. Если успешно и введенное число больше 0, возвращается результат. В противном случае выводится сообщение об ошибке, и цикл повторяется.
* Byte: запрашивает у пользователя ввод положительного числа типа byte. Функция пытается преобразовать введенное значение в byte. Если успешно, возвращается результат. Если введенное число не является положительным или выходит за пределы диапазона, выводится сообщение об ошибке, и цикл повторяется.

**Класс Program:**

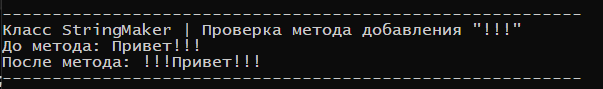
* Описание: для создание объектов, манипуляции над ними и вывода

# Тестирование

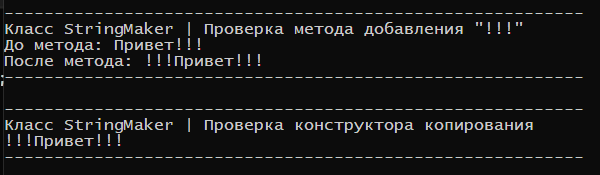
## Класс со строковым полем



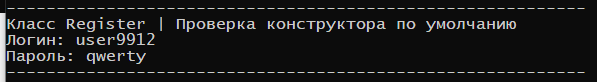
1ый тест



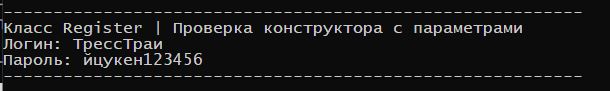
2ой тест



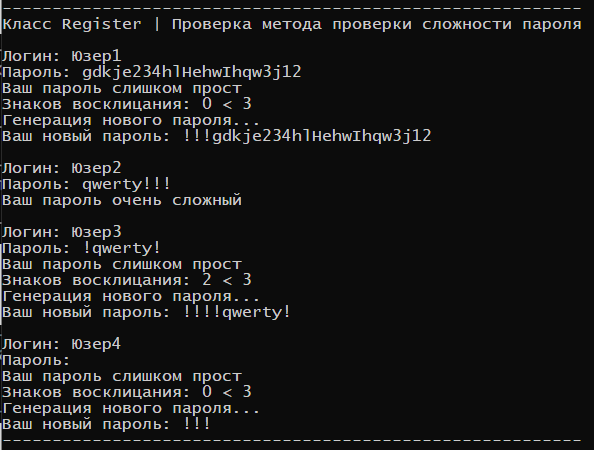
3ий тест



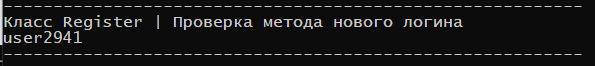
4ый тест



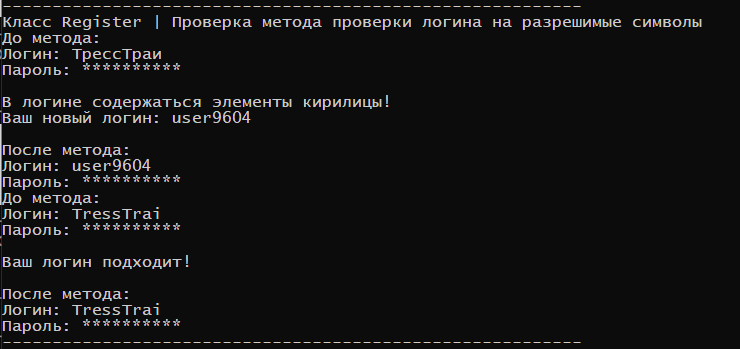
5ый тест



6ой тест



7ой тест

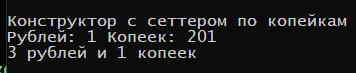


8ой тест

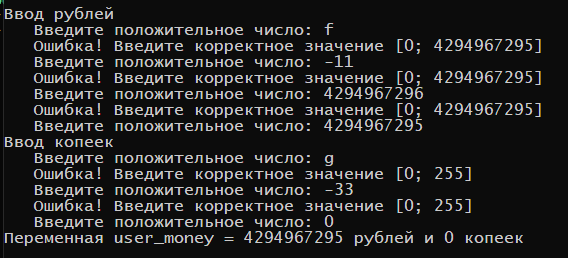
## Класс Money



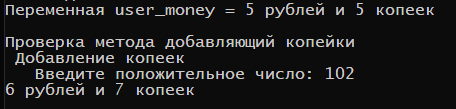
1ый тест



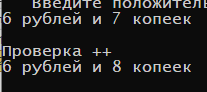
2ой тест



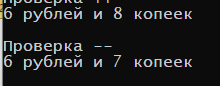
3ий тест



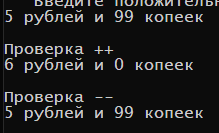
4ый тест



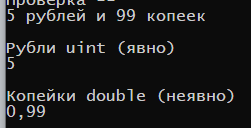
5ый тест



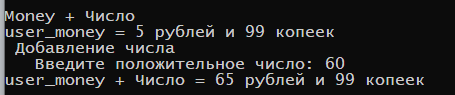
6ой тест



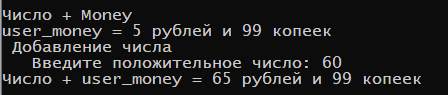
7ой тест



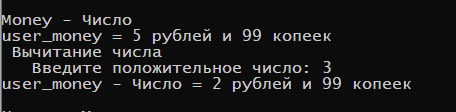
8ой тест



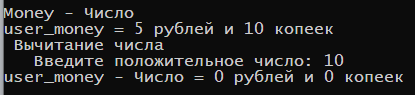
9ый тест



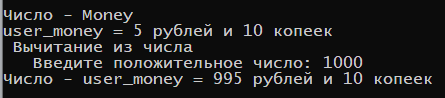
10ый тест



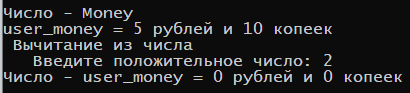
11ый тест



12ый тест



13ый тест



14ый тест