Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Конструирование программ и языки программирования»

Инструкция

по выполнению лабораторной работы №17
«Разработка, отладка и испытание программ с лямбда-выражениями»

Минск 2024 г.

Лабораторная работа № 17

Тема работы: «Разработка, отладка и испытание программ с лямбда-выражениями»

1 Цель работы

Сформировать умения работать с лямбда-выражениями и использовать их в программах.

2 Задание

Номер варианта соответствует номеру по списку в журнале.

Необходимо реализовать методы по вариантам из лабораторной работы № 4 и реализовать их с помощью лямбда-выражений.

3 Оснащение работы

ПК, среда Visual Studio 2019, MSword.

4 Основные теоретические сведения

Так же экземпляр делегата можно инициализировать **лямбда-оператором** (lambda-operator) или **лямбда-выражением** (lambda-expression). Так как в целом это одно и то же, то далее по тексту я буду их просто называть «лямбды» в местах, где не нужно подчеркивать их различия. Стоит упомянуть, что они были введены в C# 3.0, а до них существовали анонимные-функции появившиеся в C# 2.0. Отличительной чертой лямбд является оператор =>, который делит выражение на левую часть с **параметрами** и правую с **телом метода**. Допустим у нас есть делегат:

```
delegate string MyDeleg (string verb);
```

Тогда общий синтаксис лямбда-оператора будет следующим:

```
MyDeleg myDeleg = (string x) => { return x; };
```

Это именно **Лямбда-оператор,** так как мы обрамляем его тело в фигурные скобки, что позволяет нам поместить в него более одного оператора:

```
MyDeleg myDeleg = (string x) => { var z = x + x; return z; };
```

Допускается не указывать типы аргументов, ведь компилятор и так знает тип и сигнатуру вашего делегата, но можно и указать для простоты чтения кода другим человеком:

```
MyDeleg myDeleg = (x) => { return x; };
```

В случае если имеется лишь один аргумент то можно опустить обрамляющие его скобки:

```
MyDeleg myDeleg = x => { return x; };
```

Если в сигнатуре делегата аргументов нет то необходимо указать пустые скобки:

```
AnotherDeleg myDeleg = () => { return x; };
```

Если тело лямбды состоит лишь из одного выражения, то оно является **Лямбда-выражением**. Это очень удобно, так как у нас появляется возможность использовать упрощенный синтаксис в котором:
— можно опустить фигурные скобки, обрамляющие тело лямбды;
— без вышеупомянутых фигурных скобок нам не нужно использовать ключевое слово return перед оператором и точку запятой после оператора в теле лямбды. В итоге код определения лямбды может стать крошечным:

```
MyDeleg myDeleg = x => x+x;
```

А что о лямбдах думает компилятор?

Важно понимать, что лямбда выражения не являются волшебными строками, передающимися напрямую в делегат. На самом деле на этапе компиляции каждое такое выражение превращается в анонимный private метод с именем, начинающимся на "<", что исключает возможность вызова такого метода напрямую. Этот метод всегда является **членом типа в котором вы используете данное лямбда выражение**, и передается в конструктор делегата явно в CIL коде.
Причем компилятор анализирует содержит ли выражение в своем теле операции с экземплярными полями типа в котором выражение обновлено или нет. Если да, то генерируемый метод будет экземплярным, а если нет, то метод будет статическим. Использование в лямбда выражении статических полей данного типа, а так же экземпляров и экземплярных полей других типов на это не влияют.

Вы можете спросить почему бы CLR не генерировать экземплярный метод в обоих случаях, ответ прост — такому методу нужен дополнительный параметр this, что делает его выполнение более трудоемким по сравнению со статическим. Помимо этого, CLR создает конструкцию, которая **кэширует** делегат с нашим методом в анонимном закрытом поле (все там же в нашем типе где было использовано лямбда выражении) при первом обращении к нему, а при последующих просто читает из его из поля. И

действительно, нет никакого толка создавать его заново каждый раз, ведь информация о методе, заданном выражением неизменна на этапе выполнения программы.

5. Порядок выполнения работы

1. Выделить ключевые моменты задачи.
2. Построить алгоритм и теоретическую объектную модель решения задачи.
3. Запрограммировать полученные алгоритмы и объектную модель.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____
Фамилия, инициалы учащегося _____
Дата выполнения работы _____
Тема работы: _____
Цель работы: _____
Оснащение работы: _____
Результат выполнения работы: _____

7. Контрольные вопросы и задания

1. Поясните функциональное назначение лямбда-выражений.
2. Каким образом лямбда-выражения обрабатывет компилятор?
3. Поясните использование упрощенного ситаксиса лямбда-выражений.

8. Рекомендуемая литература

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. СПб. : Изд-во Питер, 2021. 896 с.

2. Прайс, М. Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка / М. Дж. Прайс. СПб : Изд-во Питер, 2023. 848 с.

3. Васильев, А.Н. Программирование на C# для начинающих. Особенности языка / А.Н. Васильев. М. : Эксмо, 2022. 528 с.

4. Фримен, А. ASP.NET Core 3 с примерами на C# для профессионалов / А. Фримен. СПб. : Изд-во Вильямс, 2021. 1184 с.