МИНОБРНАУКИ РОССИИ Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования «Хакасский государственный университет им. Н.Ф. Катанова» Колледж педагогического образования, информатики и права

ПЦК естественнонаучных дисциплин, математики и информатики

РЕФЕРАТ

на тему:
Язык программирования C#

Автор реферата: _____ Бондарев А. Н.
(подпись) (инициалы, фамилия)

Специальность: 230115 - Программирование в компьютерных системах

Курс: II Группа: И-21
Зачет/незачет:_____
Руководитель: _____ Когумбаева О.П.
(подпись, дата) (инициалы, фамилия)

г. Абакан, 2017г.

Содержание

Введение

C# — это типизированный, объектно-ориентированный, простой и в то же время мощный язык программирования, который позволяет разработчикам создавать многофункциональные приложения. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как основной язык разработки приложений для платформы Microsoft .NET (программной платформы от компании Microsoft, предназначенной для создания обычных программ и вебприложений).

Символ # (октоторп) в названии языка печатается на клавиатуре как Shift+3, что символизирует третью реализацию C. С другой стороны # можно интерпретировать и как две пары плюсов ++;++, намекающие на новый шаг в развитии языка по сравнению с C++ (подобно шагу от C к C++), и как музыкальный символ диез, вместе с буквой C составляющий в английском языке название ноты до-диез (англ. C sharp). Октоторп # часто называют «шарпом» (от англ, sharp) из-за его схожести с диезом ?, отсюда и название языка — «Си шарп».

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. C# — это фактически гибрид разных языков. Переняв многое от своих предшественников — языков C++, Java, Delphi, Модула и Smalltalk - и опираясь на практику их использования, C# синтаксически не менее (если не более) чист, чем Java, так же прост, как Visual Basic, и обладает практически той же мощью и гибкостью, что и C++.

Актуальность: Язык программирования C# является одним из наиболее популярных за счет простоты его изучения.

Цель: Дать представление языка C#, особенности его работы.

Задача: Выявить преимущества языка C# перед другими языками, изучить историю C#, сравнить его с предшествующими языками: C и C#.

1. История возникновения

Язык C# появился на свет в июне 2000 г. в результате кропотливой работы большой группы разработчиков компании Microsoft, возглавляемой Андерсом Хейлсбергом (Anders Hejlsberg).
Появление языка C# и инициативы .NET отнюдь не случайно пришлось на начало лета 2000 г. Именно к этому моменту компания Microsoft подготовила промышленные версии новых компонентных технологий и решений в области обмена сообщениями и данными, а также создания Internet-приложений (COM+, ASP+, ADO+, SOAP, Biztalk Framework). Несомненно, лучшим способом продвижения этих новинок является создание инструментария для разработчиков с их полноценной поддержкой. В этом и заключается одна из главных задач нового языка C#. Кроме того, Microsoft не могла больше расширять все те же инструменты и языки разработки, делая их все более и более сложными (а это было необходимо, так как требования поддержки современного оборудования и требования обеспечения обратной совместимости с программными продуктами начала 90-х годов становились всё более конфликтующими). Наступил момент, когда необходимо начать с чистого листа для того, чтобы создать простой, но имеющий сложную структуру набор языков, сред и средств разработки, которые позволят разработчику легко создавать современные программные продукты.
C# и .NET являются той самой отправной точкой. Если говорить упрощенно, то .NET представляет собой новую платформу, новый API (англ. Application Programming Interface - интерфейс прикладного программирования)для программирования в Windows, а C# - новый язык, созданный с нуля, для работы с этой платформой, а также для извлечения всех выгод из прогресса сред разработки и достижений объектно-ориентированного программирования в течение последних 20 лет.

2. Общее описание языка

Последнее время C и C++ являются наиболее используемыми языками для разработки коммерческих и бизнес приложений. Эти языки устраивают многих разработчиков, но в действительности не обеспечивают должной продуктивности разработки. К примеру, процесс написания приложения на C++ зачастую занимает значительно больше времени, чем разработка эквивалентного приложения, скажем, на Visual Basic. Сейчас существуют языки, увеличивающие продуктивность разработки за счет потери в гибкости, которая так привычна и необходима программистам на C/C++. Подобные решения являются весьма неудобными для разработчиков и зачастую предлагают значительно меньшие возможности. Эти языки также не ориентированы на взаимодействие с появляющимися сегодня системами и очень часто они не соответствуют существующей практике программирования для Web. Многие разработчики хотели бы использовать современный язык, который позволял бы писать, читать и сопровождать программы с простотой Visual Basic и в то же время давал мощь и гибкость C++, обеспечивал доступ ко всем функциональным возможностям системы, взаимодействовал бы с существующими программами и легко работал с возникающими Web стандартами.

Учитывая все подобные пожелания, Microsoft разработала новый язык - C#. В него входит много полезных особенностей - простота, объектная ориентированность, типовая защищенность, "сборка мусора", поддержка совместимости версий и многое другое. Данные возможности позволяют быстро и легко разрабатывать приложения, особенно COM+ приложения и Web сервисы. При создании C#, его авторы учитывали достижения многих других языков программирования: C++, C, Java, SmallTalk, Delphi, Visual Basic и т.д. Надо заметить что по причине того, что C# разрабатывался с чистого листа, у его авторов была возможность (которой они явно воспользовались), оставить в прошлом все неудобные и неприятные особенности (существующие, как правило, для обратной совместимости), любого из предшествующих ему языков. В

результате получился действительно простой, удобный и современный язык, по мощности не уступающий C++, но существенно повышающий продуктивность разработок.

Очень часто можно проследить такую связь - чем более язык защищен и устойчив к ошибкам, тем меньше производительность программ, написанных на нем. К примеру рассмотрим две крайности - очевидно это Assembler и Java. В первом случае вы можете добиться фантастической быстроты своей программы, но вам придется очень долго заставлять ее работать правильно не на вашем компьютере. В случае же с Java - вы получаете защищенность, независимость от платформы, но, к сожалению, скорость вашей программы вряд ли совместима со сложившимся представлением о скорости, например, какого-либо отдельного клиентского приложения (конечно существуют оговорки - JIT компиляция и прочее). Рассмотрим C++ с этой точки зрения - на мой взгляд соотношение в скорости и защищенности близко к желаемому результату, но на основе собственного опыта программирования я могу с уверенностью сказать, что практически всегда лучше понести незначительную потерю в производительности программы и приобрести такую удобную особенность, как "сборка мусора", которая не только освобождает вас от утомительной обязанности управлять памятью вручную, но и помогает избежать вам многих потенциальных ошибок в вашем приложении. В действительности скоро "сборка мусора", да и любые другие шаги к устранению потенциальных ошибок стану отличительными чертами современного языка. В C#, как в несомненно современном языке, также существуют характерные особенности для обхода возможных ошибок. Например, помимо упомянутой выше "сборки мусора", там все переменные автоматически инициализируются средой и обладают типовой защищенностью, что позволяет избежать неопределенных ситуаций в случае, если программист забудет инициализировать переменную в объекте или попытается произвести недопустимое преобразование типов. Также в C# были предприняты меры для исключения ошибок при обновлении программного обеспечения. Изменение кода, в такой ситуации, может непредсказуемо изменить

суть самой программы. Чтобы помочь разработчикам бороться с этой проблемой C# включает в себя поддержку совместимости версий (vesioning). В частности, в отличии от C++ и Java, если метод класса был изменен, это должно быть специально оговорено. Это позволяет обойти ошибки в коде и обеспечить гибкую совместимость версий. Также новой особенностью является native поддержка интерфейсов и наследования интерфейсов. Данные возможности позволяют разрабатывать сложные системы и развивать их со временем.

В C# была унифицирована система типов, теперь вы можете рассматривать каждый тип как объект. Несмотря на то, используете вы класс, структуру, массив или встроенный тип, вы можете обращаться к нему как к объекту. Объекты собраны в пространства имен (namespaces), которые позволяют программно обращаться к чему-либо. Это значит что вместо списка включаемых файлов заголовков в своей программе вы должны написать какие пространства имен, для доступа к объектам и классам внутри них, вы хотите использовать. В C# выражение using позволяет вам не писать каждый раз название пространства имен, когда вы используете класс из него. Например, пространство имен System содержит несколько классов, в том числе и Console. И вы можете писать либо название пространства имен перед каждым обращением к классу, либо использовать using как это было показано в примере выше.

Важной и отличительной от C++ особенностью C# является его простота. К примеру, всегда ли вы помните, когда пишите на C++, где нужно использовать "->", где "::", а где "."? Даже если нет, то компилятор всегда поправляет вас в случае ошибки. Это говорит лишь о том, что в действительности можно обойтись только одним оператором, а компилятор сам будет распознавать его значение. Так в C#, оператор"->" используется очень ограничено (в unsafe блоках, о которых речь пойдет ниже), оператор "::" вообще не существует. Практически всегда вы используете только оператор "." и вам больше не нужно стоять перед выбором.

Еще один пример. При написании программ на C/C++ вам приходилось думать не только о типах данных, но и о их размере в конкретной реализации. В

C# все упрощено - теперь символ Unicode называется просто char (а не wchar_t, как в C++) и 64-битное целое теперь - long (а не __int64). Также в C# нет знаковых и беззнаковых символьных типов.

В C#, также как и в Visual Basic после каждого выражения case в блоке switch подразумевается break. И более не будет происходить странных вещей если вы забыли поставить этот break. Однако если вы действительно хотите чтобы после одного выражения case программа перешла к следующему вы можете переписать свою программу с использованием, например, оператора goto.

Многим программистам (на тот момент, наверное, будущим программистам) было не так легко во время изучения C++ полностью освоиться с механизмом ссылок и указателей. В C# (кто-то сейчас вспомнит о Java) нет указателей. В действительности нетривиальность указателей соответствовала их полезности. Например, порой, трудно себе представить программирование без указателей на функции. В соответствии с этим в C# присутствуют Delegates - как прямой аналог указателя на функцию, но их отличает типовая защищенность, безопасность и полное соответствие концепциям объектно-ориентированного программирования.

Хотелось бы подчеркнуть современное удобство C#. Когда вы начнете работу с C#, а, надеюсь, это произойдет как можно скорее, вы увидите, что довольно большое значение в нем имеют пространства имен. Уже сейчас, на основе первого примера, вы можете судить об этом - ведь все файлы заголовков заменены именно пространством имен. Так в C#, помимо просто выражения using, предоставляется еще одна очень удобная возможность - использование дополнительного имени (alias) пространства имен или класса.

Современность C# проявляется и в новых шагах к облегчению процесса отладки программы. Традиционым средством для отладки программ на стадии разработки в C++ является маркировка обширных частей кода директивами #ifdef и т.д. В C#, используя атрибуты, ориентированные на условные слова, вы можете куда быстрее писать отлаживаемый код.

В наше время, когда усиливается связь между миром коммерции и миром разработки программного обеспечения, и корпорации тратят много усилий на планирование бизнеса, ощущается необходимость в соответствии абстрактных бизнес процессов их программным реализациям. К сожалению, большинство языков реально не имеют прямого пути для связи бизнес логики и кода. Например, сегодня многие программисты комментируют свои программы для объяснения того, какие классы реализуют какой-либо абстрактный бизнес объект. C# позволяет использовать типизированные, расширяемые метаданные, которые могут быть прикреплены к объекту. Архитектурой проекта могут определяться локальные атрибуты, которые будут связанны с любыми элементами языка - классами, интерфейсами и т.д. Разработчик может программно проверить атрибуты какого-либо элемента. Это существенно упрощает работу, к примеру, вместо того чтобы писать автоматизированный инструмент, который будет проверять каждый класс или интерфейс, на то, является ли он действительно частью абстрактного бизнес объекта, можно просто воспользоваться сообщениями основанными на определенных в объекте локальных атрибутах.

4. Особенности C#

Полный и хорошо определенный набор основных типов.

Встроенная поддержка автоматической генерации XML-документации.

Автоматическое освобождение динамически распределенной памяти.

Возможность отметки классов и методов атрибутами, определяемыми пользователем, (это может быть полезно при документировании и способно воздействовать на процесс компиляции - например, можно пометить методы, которые должны компилироваться только в отладочном режиме).

Полный доступ к библиотеке базовых классов .NET, а также легкий доступ к Windows API (если это действительно необходимо).

Указатели и прямой доступ к памяти, если они необходимы (однако язык разработан таким образом, что практически во всех случаях можно обойтись и без этого). Поддержка свойств и событий в стиле Visual Basic.

Простое изменение ключей компиляции. Позволяет получать исполняемые файлы или библиотеки компонентов .NET, которые могут быть вызваны другим кодом так же, как элементы управления ActiveX (компоненты COM).

Возможность использования C# для написания динамических web-страниц ASP.NET. Одной из областей, для которых не предназначен этот язык, являются критичные по времени и высокопроизводительные программы, когда имеет значение, занимать на исполнение цикла 1000 или 1050 машинных циклов, и освобождать ресурсы требуется немедленно. C++ остается в этой области наилучшим из языков высокого уровня. В C# отсутствуют некоторые ключевые моменты, необходимые для создания высокопроизводительных приложений, в частности подставляемые функции и деструкторы, выполнение которых гарантируется в определенных точках кода.

5. **Сравнение C# с другими языками программирования**

C#, являясь последним из широко распространенных языков программирования, должен впитать в себя весь имеющийся опыт и вобрать лучшие стороны существующих языков программирования, при этом являясь специально созданным для работы в .NET. Сама архитектура .NET продиктовала ему (как и многим другим языкам, на которых можно писать под .NET) объектно-ориентированную направленность. Конечно, это не является правилом, возможно создание компиляторов даже функциональных языков по .NET, на эту тему существуют специальные работы.

Свой синтаксис C# во многом унаследовал от C++ и Java. Разработчики, имеющие опыт написания приложений на этих языках, найдут в C# много знакомых черт. Но вместе с тем он является во многом новаторским - аттрибуты, делегаты и события, прекрасно вписанные в общую идеологию языка, прочно заняли место в сердцах .NET - разработчиков. Их введение позволило применять принципиально новые приемы программирования.

Конечно, излюбленным объектом для сравнения с C# у мировой коммьюнити является Java. Также разработанный для работы в виртуальной среде выполнения, имеющей объектно-ориентированную архитектуру и сборщик мусора, осноыванный на механизме ссылок. При сравнении с этим языком сразу выделаются такие особенности, как возможность объявлять несколько классов в одном файле, из чего следует синтаксическая поддержка иерархической системы пространств имен. Из реализации ООП-концепций сходство в механизме наследования и реализации (и в Java и в C# возможно единичное наследование, но множественная реализация интерфейсов, в отличие от C++). Но в Java отсутствуют свойства и индексаторы (а также делегаты и события, но они отсутствуют еще много где). Также есть возможность перечисления контейнеров.

Из вещей, включенных в спецификацию языка, но не являющихся чисто "программистскими" необходимо отметить возможность использование комментариев в формате XML. Если комментарии отвечают специально

описанной стркутуре, компилятор по ним может сгенерировать единый XML-файл документации.

Но C# внес и свои уникальные черты, которые уже были упомянуты - это события, индексаторы, аттрибуты и делагаты. Все эти элементы будут обсуждены в следующих частях, сейчас лишь отмечу, что они предоставляют собой очень полезные возможности, которые не останутся невостребованными.

6. Интерфейсы

Интерфейсы объявляются тем же способом, что и классы, только вместо ключевого слова class используется ключевое слово interface. Ключевые слова для модификации доступа public и internal используются точно так же, поэтому, например, для того, чтобы сделать интерфейс общедоступным, следует использовать ключевое слово public.

Для интерфейсов ключевые слова abstract и sealed использовать нельзя, так как ни один модификатор для интерфейсов не имеет смысла (у них отсутствует реализация, следовательно, для них не могут создаваться экземпляры в явном виде).

Наследование для интерфейсов определяется аналогично наследованию для классов. Основное отличие здесь в том, что мы можем использовать интерфейсы с множественными базами, например: public interface IMylnterface : IMyBaselnterface, IMyBaselnterface2

Интерфейсы, как и классы, наследуются от System.object. Этот механизм допускает полиморфизм интерфейсов. Однако, как отмечалось ранее, нельзя создать экземпляр интерфейса таким же способом, как и экземпляр класса.

7. Сфера применения C#

C# — один из языков программирования, который может использоваться для создания приложений. Создавать приложения на C# легче, чем на C+ + , поскольку синтаксис языка C# более простой, чем синтаксис C+ + .

Некоторые наиболее часто встречающиеся типы приложений:

Приложения Windows. Это приложения вроде Microsoft Office, имеющие знакомый "Windows-подобный" вид и представление. Создавать такие приложения достаточно просто с помощью модуля .NET Framework, который называется Windows Forms и представляет собой библиотеку управляющих элементов (кнопок, панелей инструментов, меню и т. п.); эта библиотека может использоваться для создания пользовательского интерфейса (user interface, Ul) Windows.

Web-приложения. Эти приложения представляют собой web-страницы, которые могут просматриваться любым web-браузером. В состав .NET Framework входит мощная система динамического создания содержимого web-страниц, позволяющая идентифицировать пользователя, обеспечивать безопасность и пр. Эта система называется Active Server Pages.NET (ASP.NET — активные серверные страницы .NET); для создания приложений ASP.NET можно применять Web Forms языка C#.

Web-службы. Это новый замечательный способ создания гибких распределенных приложений. С помощью web-служб можно обмениваться по Интернету практически любыми данными с использованием единого простого синтаксиса независимо оттого, какой язык программирования применялся при создании web-службы и на какой системы она размещена. Приложениям всех перечисленных типов может потребоваться доступ к базам данных, что осуществляется с помощью раздела .NET Framework, называемого Active Data Objects.NET (ADO.NET — активные объекты с данными .NET). Также можно использовать и многие другие ресурсы, например, инструменты для создания

сетевых компонентов, графического вывода, выполнения сложных математических вычислений и т.д.

Заключение

На основании материала, изученного в данной работе, можно сформулировать следующие выводы:

Возникновение языка C# было обусловлено прогрессом технологий и возрастающими достижениями в области программирования и разработки

C# относится к C-подобным языкам, поэтому многие синтаксические конструкции являются заимствованными из более ранних реализаций C. Также C# имеет некоторое сходство с Java, так как создан на общем базисе для решения сходной задачи (создание переносимого кода)

Язык C# выгодно отличается от своих предшественников и конкурентов широким спектром поддерживаемых возможностей и относительной простотой в использовании C# популярен, так как применяется для создания приложений, пользующихся большим спросом (а также вследствие распространения ОС Windows и созданной для неё NET Framework)

В данной работе реализованы следующие цели:

Описана история создания C#.

Рассмотрены особенности языка и реализация основных его конструкций. Представлены сферы применения языка C#.

Проведено сравнение C# с предшествующими ему языками: C и C++.

Библиографический список

1. C Sharp. Википедия - свободная энциклопедия, - [Электронный ресурс] URL: http://ru.wikipedia.org/wiki/Mac_OS (Дата обращения: 24.01.17)
2. Введение в язык C# и .net Framework, - [Электронный ресурс] URL: https://msdn.microsoft.com/ru-ru/library/z1zx9t92.aspx (Дата обращения: 24.01.17)
3. Лахатин, А.С. Языки программирования. Учеб. пособие / А. С. Лахатин, Л.Ю. Исакова. – Екатеринбург, 1998 – 548с.: ил.
4. Уэйт, М. Язык С. Руководство для начинающих. / М. Уэйт, С. Прага, Д. Мартин. - М.: Мир, 1995. - 521с.: ил.
5. Богатырев, А. Язык программирования С, - [Электронный ресурс] URL: http://www.refby.com. (Дата обращения: 24.01.17)
6. C# 2005 для профессионалов / К. Нейгел, Б. Ивьен, Д. Глин и др. - Москва, Санкт-Петербург, Киев: «Диалектика», 2007
7. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах: Пер. с англ. - М.: Издательско-торговый дом "Русская Редакция", 2002.
8. Лабор В. В. СиШарпсозданиеприложенийдляWindows. - Минск :Харвест, 2003
9. Марченко А. Л. Основы программирования на C# 2.0. - М.: БИНОМ (Лаборатория знаний, Интернет-университет информационных технологий - ИНТУИТ.ру), 2007
10. Кариев Ч. А. Разработка Windows-приложений на основе Visual C#. - М.: БИНОМ (Лаборатория знаний, Интернет-университет информационных технологий - ИНТУИТ.ру), 2007

11. C Sharp. Википедия - свободная энциклопедия, - [Электронный ресурс] URL: http://ru.wikipedia.org/wiki/Mac_OS (Дата обращения: 24.01.17)

12. Воройский Ф. С. Информатика. Новый систематизированный толковый словарь-справочник. - 3-е изд. - М.: ФИЗМАТЛИТ, 2003

13. Brown E. Windows Forms Programming with C#. - Manning Publications Co., 2002