1. **ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ**
   1. **Основные принципы ООП**

ООП - это стиль программирования, который появился в 80 годах 20 века. В отличие от процедурных языков, где данные и инструкции по их обработке существуют отдельно, в объектно-ориентированном программировании эта информация объединяется в единую сущность.

У объектно-программного программирования есть свои постулаты. Принципы ООП - это его основные идеи. Выделяют три самых главных из них: наследование, полиморфизм и инкапсуляция. Основы программирования на языках ООП заключаются в использовании объектов и классов.

*Инкапсуляция* — это способность скрывать внутренние детали при предоставлении открытого интерфейса к определяемому пользователем типу [1]. Иными словами инкапсуляция – это использование объединения данных и инструкций по их обработке в единую сущность – класс. Во время написания программ на одном из языков ООП происходит разграничение между информацией внутри сущности и снаружи. Таким образом, достигается обеспечение безопасности данных и методов их реализации от внешних воздействий, например, со стороны других классов, не относящихся к этому объекту. Внутри сущности данные успешно взаимодействуют друг с другом, но надежно защищены от несанкционированного доступа извне.



То есть пользователю предоставляется только интерфейс объекта.

Пользователь может взаимодействовать с объектом только через этот интерфейс, который реализуется с помощью ключевого слова: public.

Пользователь не может использовать закрытые данные и методы, реализующиеся с помощью ключевых слов: private, protected, internal.

Сокрытие реализации целесообразно применять в следующих случаях [2]:

– предельная локализация изменений при их необходимости;

– прогнозируемость изменений (какие изменения в коде надо сделать для заданного изменения функциональности) и прогнозируемость последствий изменений.

Второй принцип ООП – *наследование*. Наследованием называют возможность одного класса использовать методы другого без повторения их фактической реализации, то есть оно позволяет описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым, что помогает избавиться от избыточности исходного кода [3]. В некоторых языках программирования существует два вида наследования: простое и множественное. Простым называется наследование, при котором производный класс имеет одного родителя. При множественном наследовании у класса может быть более одного предка. В этом случае класс наследует методы всех предков. Достоинства такого подхода в большей гибкости.

Еще одним принципом ООП является *полиморфизм*. Полиморфизм — возможность объектов с одинаковой спецификацией иметь различную реализацию [4]. Иными словами, использование полиморфизма означает, что для манипуляции с объектами разной степени сложности можно создать один интерфейс, который будет по-разному реагировать на события и одновременно правильно реализовывать поставленные задачи.

Язык программирования поддерживает полиморфизм, если классы с одинаковой спецификацией могут иметь различную реализацию — например, реализация класса может быть изменена в процессе наследования[5].

* 1. **Основные объектно-ориентированные языки программирования**

Объектно-ориентированные языки программирования пользуются в последнее время большой популярностью среди программистов, так как они позволяют использовать преимущества объектно-ориентированного подхода не только на этапах проектирования и конструирования программных систем, но и на этапах их реализации, тестирования и сопровождения.

Первый объектно-ориентированный язык программирования Simula 67 был разработан в конце 60-х годов в Норвегии. Авторы этого языка очень точно угадали перспективы развития программирования: их язык намного опередил свое время. Однако современники (программисты 60-х годов) оказались не готовы воспринять ценности языка Simula 67, и он не выдержал конкуренции с другими языками программирования (прежде всего, с языком Fortran). Прохладному отношению к языку Simula 67 способствовало и то обстоятельство, что он был реализован как интерпретируемый (а не компилируемый) язык, что было совершенно неприемлемым в 60-е годы, так как интерпретация связана со снижением эффективности (скорости выполнения) программ.

Но достоинства языка Simula 67 были замечены некоторыми программистами, и в 70-е годы было разработано большое число экспериментальных объектно-ориентированных языков программирования: например, языки CLU, Alphard, Concurrent Pascal и др. Эти языки так и остались экспериментальными, но в результате их исследования были разработаны современные объектно-ориентированные языки программирования.

Обзор некоторых современных объектно-ориентированных языков приводится ниже.

Язык **Smalltalk** был разработан командой Xerox Palo Alto Research Center Learning Research Group как программная часть Dynabook - фантастического проекта Алана Кея. В основу были положены идеи Simula. Smalltalk является одновременно и языком программирования, и средой разработки программ. Это чисто объектно-ориентированный язык, в котором абсолютно все рассматривается как объекты; даже целые числа - это классы. Вслед за Simula, Smalltalk является важнейшим объектно-ориентированным языком, поскольку он не только оказал влияние на последующие поколения языков программирования, но и заложил основы современного графического интерфейса пользователя, на которых непосредственно базируются интерфейсы Macintosh, Windows и Motif. В основу языка положены две простые идеи:

- все является объектами;

- объекты взаимодействуют, обмениваясь сообщениями.

***Microsoft Visual C++.*** [Язык программирования C++](http://bourabai.ru/C-Builder/cpp/index.htm) был разработан Бьерном Страустрапом, сотрудником AT&T Bell Laboratories. Непосредственным предшественником C++ является С with Classes, созданный тем же автором в 1980 году. Язык С with Classes, в свою очередь, был создан под сильным влиянием [С](http://bourabai.ru/C-Builder/c/index.htm) и Simula. C++ - это в значительной степени надстройка над С. В определенном смысле можно назвать C++ улучшенным С, тем С, который обеспечивает контроль типов, перегрузку функций и ряд других удобств. Но главное в том, что C++ добавляет к С объектную ориентированность.

*Достоинства.*

* ничем не ограниченные в рамках Windows возможности;
* очень приличный, хотя и довольно медленный компилятор С++;
* отличный отладчик, мощная библиотека MFC;
* правка кода в режиме отладки и последующее его выполнение без полной перекомпиляции и прерывания отладочной сес­сии;
* технология подсказок, работающая быстрее;
* полноценный браузер классов;
* пол­но­ценный менеджер исходного кода, отображающий файлы и папки так, как вы хотите;
* исчерпывающая справочная система.

*Недостатки.*

* необходимость запоминания методов работы с каждым объектом;
* довольно длинных и абсолютно не­произ­носимых идентификаторов;
* необходимость в предварительном изучении технической литературы, прежде чем у вас заработает приложение.

***Visual Basic*** широко используется для интеграции офисных пакетов и различных типов СУБД.

*Достоинства.*

* простота;
* отличная среда разработки;
* способность работы с СУБД;
* позволяет создавать Web-страницы совместно с HTML и ASP;
* выполняет функцию системного интегратора.

*Недостатки.*

* более низкая скорость выполнения приложений;
* отсутствие возможности в создании компонентов.

***Delphi*** является мощным, универсальным языком программирования быстрой разработки приложений (RAD).

*Достоинства.*

* простота, скорость и эффективность объяс­ня­ют попу­ляр­ность Delphi;
* имеет один из самых быстрых компиляторов, по­рож­да­ющих неплохой объектный код;
* программы, написанные на Delphi, не требуется снабжать дополнительными библиотеками;
* модель програм­ми­ро­вания в Delphi‑ компонентная, что поз­во­ляет пользоваться компо­нента­ми, написанными другими разработчиками, даже не имея их исходного кода.

*Недостатки.*

* статическое присоединение библиотеки визуальных компонентов (VCL) и ком­по­нентов к исполняемому файлу;
* вся информа­ция о форме (включая свойства, настройки компонентов, значения по умолчанию) хранится в EXE-файле неоптимальным образом.

## Java. С 1995 года стал широко распространяться новый объектно-ориентированный язык программирования [Java](http://bourabai.ru/alg/java.htm), ориентированный на сети компьютеров и, прежде всего, на Internet. Синтаксис этого языка напоминает синтаксис языка C++, однако эти языки имеют мало общего. [Java](http://bourabai.ru/alg/java.htm) - интерпретируемый на конечной стадии язык: для него определены внутреннее представление (прекомпиляция в байткод - bytecode) и постинтерпретатор этого представления на целевой машине, которые уже сейчас реализованы на большинстве платформ. Положительные свойства [технологии Java](http://bourabai.ru/alg/java.htm) позволяют использовать ее для программ, распространяемых по сетям (в частности, по сети Internet).

*Достоинства:*

* высокая безопасность;
* к языку существует огромное количество бесплатных программных библиотек, написанных программистами всего мира;
* приложения компилируются в специальный байт-код и выполняются в специальной виртуальной машине в любой операционной системе и на любом оборудовании.

*Недостатки:*

* чрезмерная нагрузка на оперативную память оборудования;
* время выполнения одних и тех же задач в полтора-два раза у продуктов этого языка медленнее, чем на тех же С.

**Visual Basic .NET** (VB.NET) — это объектно-ориентированный язык программирования, который можно рассматривать как очередной виток эволюции Visual Basic (VB), реализованный на платформе Microsoft .NET.   
В отличие от «классического» VB, VB.NET — полностью объектно-ориентированный язык программирования, поддерживающий полиморфизм, наследование и другие ключевые возможности ООП.

*Достоинства:*

* Высокая скорость разработки графических программ для Windows;
* Простой синтаксис;
* Возможность отладки и редактирования кода при приостановке программы без перекомпиляции;
* Защита от ошибок, связанных доступом к памяти;
* Возможность использования WinAPI.

*Недостатки:*

* Поддержка операционных систем Windows и Mac OS;
* Низкая скорость работы.

***Вывод***. Практически все объектно-ориентированные языки программирования являются развивающимися языками, их стандарты регулярно уточняются и расширяются. Следствием этого развития являются неизбежные различия во входных языках компиляторов различных систем программирования. Каждый язык имеет свои преимущества и недостатки, а стало быть, различные области эффективного применения. Это обосновывает необходимость изучения и освоения различных языков программирования.

* 1. **Язык программирования С# (Sharp)**

C# (произносится *си-шарп*) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как основной язык разработки приложений для платформы Microsoft .NET. Сторонники C# называют его самым мультипарадигменным, универсальным, продвинутым и удобным в использовании языком программирования. Учитывая тот факт, что за ним стоит платформа Microsoft .NET, число таких сторонников достаточно велико. Компилятор с C# входит в стандартную установку самой .NET, поэтому программы на нём можно создавать и компилировать даже без инструментальных средств, вроде Visual Studio. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML. Переняв многое от своих предшественников — языков C++, Java, Delphi, Модула и Smalltalk — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем: так, C# не поддерживает множественное наследование классов (в отличие от C++).

C# поддерживает все три «столпа» объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм. Кроме того, в нем была реализована автоматическая «сборка мусора», обработки исключений, динамическое связывание.

Рассмотрим некоторые особенности языка C#. Данный язык разрабатывался как язык программирования прикладного уровня для Common Language Runtime (CLR) и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает Base Class Library (BCL). Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR.

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 7.0, которая вышла 7 марта 2017 года вместе с Visual Studio 2017.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей, как, например, лямбды, динамическое связывание, асинхронные методы и т.д.

### Роль платформы .NET

Когда говорят C#, нередко имеют в виду технологии платформы .NET (WPF, ASP.NET). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Как-то Билл Гейтс сказал, что платформа .NET - это лучшее, что создала компания Microsoft. Возможно, он был прав. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

* Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках.
* Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Framework поддерживается на большинстве современных ОС Windows (Windows 10/8.1/8/7/Vista). А благодаря проекту Mono можно создавать приложения, которые будут работать и на других ОС семейства Linux, в том числе на мобильных платформах Android и iOS.
* Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.
* Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF. Для создания веб-сайтов - ASP.NET и т.д.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

**Список** **литературы**

1. Пол А. Объектно-ориентированное программирование на C++. – А.Пол. – 2001. – 476с.
2. Основные принципы ООП: инкапсуляция, наследование, полиморфизм. – Электронный ресурс. URL: <http://mylektsii.ru/14-6957.html>.
3. Основные принципы ООП и их использование. – Эелектронный ресурс. URL: <http://fb.ru/article/91926/osnovnyie-printsipyi-oop-i-ih-ispolzovanie>.
4. Использование полиморфизма в объектно-ориентированном программировании. – Электронный ресурс. URL: <http://mykonspekts.ru/2-298.html>.
5. Инкапсуляция, наследование, полиморфизм. – Электронный ресурс. URL: <http://codrob.ru/lesson/26>.