МБОУ «Средняя общеобразовательная школа № 117

имени М.В. Стрельникова»

г. Сорочинск Оренбургской области

**Исследовательская работа**

«Программирование. Создание скриптов!»

 Выполнил: Баранов Артем

 обучающийся СОШ № 117, 10 класс

Руководитель: Лупова Наталья Александровна

 учитель информатики и ИКТ СОШ № 117

г. Сорочинск

2022 год

**Содержание**

|  |  |
| --- | --- |
| Введение 3 | 3 |
| **Глава 1. Основная часть. Что такое программирование?** | 5 |
| 1.1 Понятие язык программирования  | 5 |
| 1.2 История развития языков программирования  | 5 |
| 1.3 Устройство языков программирования  | 6 |
| **Глава 2. Создание скриптов** | 10 |
| 2.1 Что такое скрипт? | 10 |
| 2.2 Программы для создания скриптов  | 10 |
| 2.3 Программы для написания кода  | 12 |
| 2.4 Программы для создания игр  | 14 |
| **Глава 3 .Практический аспект исследования. Создание программного продукта. Свой язык программирования** | 16 |
| Вывод. | 19 |
| Заключение | 20 |
| Список использованных источников | 21 |
| ПРИЛОЖЕНИЕ А | 23 |
| ПРИЛОЖЕНИЕ Б | 30 |
| ПРИЛОЖЕНИЕ В | 31 |
|  |  |
|  |  |

**Введение**

На современном этапе развития компьютерных технологий трудно представить высококвалифицированного специалиста, который не владеет информационными технологиями. Чтобы свободно ориентироваться в информационных потоках современному человеку необходимо уметь получать, обрабатывать информацию с помощью компьютеров, других новейших средств, а также необходимо уметь программировать на языках высокого уровня.

Программирование… Многим миллионам людей на нашей планете эти слова ни о чём не говорят. Несомненно, у всех это слово на слуху, но что оно обозначает, многие наверняка даже не задумывались.

Программисты-профессионалы сейчас утверждают о том, что научиться программировать может каждый. Программирование сегодня становится второй грамотностью. Кто умеет программировать, становятся обладателями «суперсилы», способной претворять в жизнь самые невероятные вещи, помогать себе и окружающим людям быстрее решать их проблемы и задачи, делать жизнь всего человечества лучше.

**Актуальность**

Языки программирования – является актуальной темой в IT индустрии. Они помогают реализовать разные виды задач и воплотить свои мысли в реальность. От простых мини-игр, до огромных проектов в IT индустрии. Поэтому я решил попробовать научиться программировать самостоятельно.

**Новизна**

На сегодняшний день существуют работы, посвященные теме программирования вообще. Однако я решил изучить эту тему намного глубже и создать свой язык программирования. В этом и заключается новизна моего исследования.

**Цель исследовательской работы:** изучение языков программирования, создание скриптов.

**Задачи исследования:**

познакомиться с историей программирования;

изучить литературу, интернет-источники по данной теме;

самостоятельно изучить среды программирования;

провести исследование;

создать самостоятельно программный продукт.

**Практическая значимость** моей работы состоит в том, что в дальнейшем материалы исследования могут быть использованы теми обучающимися, которые заинтересуются программированием.

**Гипотеза**

Я предполагаю, что у меня получится научиться программировать самостоятельно и создать собственный программный продукт. Поставленные цель и задачи, выдвинутая гипотеза определили **предмет исследования:** языки программирования;

**Объект исследования:** программный продукт.

**Методы исследования:**

анализ литературы и интернет - источников по теме;

анкетирование;

количественно-качественный анализ полученных результатов.

Исследование проходило на базе МБОУ «СОШ № 117» среди 7-11 классов.

**Глава 1. Основная часть. Что такое программирование?**

**Теоретическая часть**

**1.1 Понятие язык программирования**

Язык программирования – этот система обозначений, служащая для точного описания программ или алгоритмов для компьютера. Языки программирования относятся к формальным языкам и от естественных языков они отличаются числом «слов» и строгими правилами записи команд (операторов), поэтому применение их по назначению они не допускают свободного толкования выражений, которые характерны для естественного языка. К формальным языкам предъявляется ряд требований это:

наглядность;

единство;

гибкость;

модульность;

однозначность.

Сейчас в мире существует несколько тысяч языков программирования и для каждого языка существует своя определённая область применения.

**1.2 История развития языков программирования**

Языки программирования условно можно разделить на пять поколений.

В первое поколение входят языки, созданные в начале 50-х годов, когда первые компьютеры только «появились на свет». Самым первым был язык ассемблера, созданный по принципу «одна инструкция – одна строка».

Второе поколение языков программирования приходится на конец 50-60-х годов, в этот период разрабатывается символический ассемблер, в котором появилось понятие переменной. В дальнейшем он стал первым полноценным языком программирования, благодаря чему возросла скорость разработки и надёжность программ.

Третье поколение языков программирования относится к 60-м годам XX столетия. В этот период появились языки программирования высокого уровня, при помощи которых можно решить задачи из любых областей. Такие качества новых языков, как относительная простота, независимость от конкретного компьютера и возможность использования мощных синтаксических конструкций, позволили резко повысить производительность труда программистов.

Структура этих языков была понятная большинству пользователей, поэтому многих привлекала к самостоятельному написанию небольших программ. Большинство языков этого поколения успешно применяется и сегодня.

Примерно с 70-х годов начинается четвертый период развития языков высокого уровня. В основном они предназначались для реализации крупных проектов, обычно ориентированы на специализированные области применения, где хороших результатов можно добиться, используя не универсальные, а проблемно-ориентированные языки, оперирующие конкретными понятиями узкой предметной области. Как правило, в эти языки встраиваются мощные операторы, позволяющие одной строкой описать такую функциональность, для реализации которой на языках младших поколений потребовались бы тысячи строк исходного кода.

Рождение языков пятого поколения произошло в середине 90-х годов. К ним относятся системы автоматического создания прикладных программ с помощью визуальных средств разработки, без знания основ программирования.

Главная идея, которая закладывается в эти языки, это возможность автоматического формирования результирующего текста на универсальных языках программирования (который потом требуется откомпилировать). Инструкции же вводятся в компьютер в максимально наглядном виде с помощью методов, наиболее удобных для человека, который не знаком с программированием.

**1.3 Устройство языков программирования**

Правила записи команд на конкретном языке называются синтаксисом языка. Синтаксис определяет, какая команда будет считаться правильной, а какая нет. Например, в языке Basic команды CLS и FOR I=1 TO 10 считаются правильными, а команды CLERSCREEN и FOR I FROM 1 TO 10 – неправильными.

Каждая команда, записанная на языке программирования, имеет определенное значение, то есть заставляет компьютер выполнять те или иные действия. Правила, определяющие смысл команд, называются семантикой языка. Например, команда CLS вызывает очистку экрана.

Каждый язык имеет алфавит – набор символов, которые можно использовать при записи программ на этом языке. Ранние версии одного и того же языка могут немного различаться алфавитом.

Программа, написанная на языке программирования, состоит из команд (операторов), задающих последовательность действий. Эти действия выполняются над некоторыми объектами. Объектами могут быть числа, текстовые строки, переменные и другие. Языки отличаются друг от друга множеством допустимых объектов и набором операций, которые можно выполнять над этими объектами.

Программа, написанная на языке программирования, представляет собой просто текст. Чтобы компьютер мог выполнять команды, содержащиеся в этой программе, надо перевести программу в набор понятных компьютеру инструкций, записанных в двоичной форме (в код). Такой перевод называется трансляцией.

По способу трансляции языки делятся:

компиляторы;

интерпретаторы.

В компиляторах перевод всего текста программы в код осуществляется сразу, и создаётся исполняемый файл, который затем можно неоднократно запускать.

В интерпретаторах при запуске программы каждая ее строчка последовательно переводится в код и выполняется; затем переводится в код и выполняется другая строчка, и так далее.

По уровню (особенностям построения) языки делятся:

машинно-ориентированные (ассемблеры).

Первым значительным шагом представляется переход к языку ассемблера. Не очень заметный, казалось бы, шаг-переход к символическому кодированию машинных команд имел на самом деле огромное значение.

Программисту не надо было больше вникать в хитроумные способы кодирования команд на аппаратном уровне. Более того, зачастую одинаковые по сути команды кодировались совершенно различным образом в зависимости от своих параметров.

Появилась также возможность использования макросов и меток, что также упрощало создание, модификацию и отладку программ. Появилось даже некое подобие переносимости и существовала возможность разработки целого семейства машин со сходной системой команд и некоего общего ассемблера для них, при этом не было нужды обеспечивать двоичную совместимость.

Вместе с тем, переход к новому языку таил в себе и некоторые отрицательные (по крайней мере, на первый взгляд) стороны. Становилось почти невозможным использование всяческих хитроумных приемов сродни тем, что упомянутых выше.

Кроме того, здесь впервые в истории развития программирования появились два представления программы: в исходных текстах и в откомпилированном виде. Сначала, пока ассемблеры только транслировали мнемоники в машинные коды, одно легко переводилось в другое и обратно, но затем по мере появления таких возможностей, как метки и макросы, дизассемблирование становилось все более и более трудным делом. К концу ассемблерной эры возможность автоматической трансляции в обе стороны была утеряна окончательно. В связи с этим было разработано большой количество специальных программ-дизассемблеров, осуществляющих обратное преобразования, однако в большинстве случаев они с трудом могут разделить код и данные. Кроме того, вся логическая информация (имена переменных, меток и т.п.) теряется безвозвратно. В случае же задачи о декомпиляции языков высокого уровня примеры удовлетворительного решения проблемы и вовсе единичны.

Каждый оператор языка представляет собой мнемоническое (условное) обозначение машинной команды. Естественно, что каждый тип процессора имеет свой набор команд, а значит, свой ассемблер. Ассемблеры используются для создания драйверов, программирования различных устройств, а также для написания фрагментов программ, где очень важно время выполнения, так как на ассемблере можно написать максимально эффективную программу.

Универсальные

Иногда их делят на процедурно-ориентированные и объектно-ориентированные, но в настоящее время граница между этими видами совершенно стерлась. Эти языки используются чаще всего для решения самых разнообразных задач. И хотя каждый из языков имеет свои особенности, что делает его наиболее эффективными для решения определенного вида задач, но в принципе для решения любой задачи можно выбирать любой язык программирования.

Среди универсальных языков программирования в настоящее время наиболее распространены: С++, C#, Java, Python и др, которые также следует относить к языкам высокого уровня.

**Глава 2. Создание скриптов**

**2.1 Что такое скрипт?**

Скрипт (сценарий) — это последовательность действий, описанных с помощью скриптового языка программирования (JavaScript, PHP, Perl, Python и др.) для автоматического выполнения определенных задач.

Например, для функционирования какого-нибудь сервиса, инструмента на сайте.

Если говорить простым языком, скрипты — это то, что придает сайту динамику, приводит механизм в действие. Если с любой веб-страницы удалить все сценарии, это будет просто статичное изображение, набор определенных данных. Взаимодействовать с такой страницей будет невозможно. Например, нажатие кнопки «Отправить комментарий» ни к чему не приведет.

Чтобы происходило действие (сообщение появлялось под публикацией), необходимо прописать соответствующий script. При выполнении условий (в данном случае нажатие кнопки «Отправить») он будет запускаться. После чего пользователь увидит результат своих действий.

Также можно написать еще один скрипт (или дополнить предыдущий новыми условиями), чтобы система оповещала пользователя об успешной или неуспешной отправке. Если все условия были выполнены правильно, пользователь увидит сообщение: «Комментарий отправлен». Если что-то было сделано неверно, он увидит: «Заполните все поля корректно», «Заполните поля, отмеченные \*» и т. д.

Скрипты, как правило, взаимодействуют с различными базами (например, MySQL), позволяющими хранить данные и иметь быстрый доступ к ним.

**2.2 Программы для создания скриптов**

IDLE

PyCharm

Intellij IDEA

Eclipse

Netbeance

RubyMine

Notepad++

SublimeText

IDLE
Python — один из самых популярных в мире языков программирования, который поддерживает сценарий скриптов. Он легок в освоении, имеет быструю скорость выполнения кода, а также может стать один из главных языков для написания небольших скриптов. Из-за этого мы предлагаем сначала взять во внимание этот ЯП и его стандартную интегрированную среду разработки под названием IDLE. Данный инструмент не нужно загружать отдельно, поскольку его получение происходит вместе с основными компонентами Python при их скачивании с официального веб-ресурса. Хоть эта программа, как можно понять по представленному далее скриншоту, и не имеет особенностей внешнего вида или каких-то дополнительных элементов интерфейса, она отлично подойдет для того, чтобы писать небольшие скрипты. В ней подсвечивается синтаксис, есть продвинутая система отчета об ошибках, а компилирование кода можно запустить горячей клавишей, ознакомившись с результатом в новом окне.

PyCharm

PyCharm — продвинутая альтернатива предыдущему решению, созданная известной компанией под названием JetBrains. Разработчики постарались реализовать в своей программе те опции, которые пригодятся профессиональным разработчикам и окажут значимое влияние на комфортность взаимодействия с софтом. Самая главная из них, предназначенная для повышения удобства, — встроенный умный редактор. Как только вы начинаете набирать команду, редактор отображает возможные варианты автозаполнения, а вам остается лишь подтвердить его, нажав по клавише Enter. Это же позволяет исключить большинство опечаток или случайных ошибок еще до запуска компилирования, сэкономив тем самым время, которое могло быть затрачено на исправление в дальнейшем.

Intellij IDEA

Еще одно решение от тех же разработчиков, но уже ориентированное на работу с разными языками программирования. Из тех, которые поддерживают создание скриптов, можно отметить JavaScript, Python и Ruby, но если вы вдруг решите реализовать сложный сценарий на C++, никаких проблем с этим тоже быть не должно, однако не стоит забывать, что этот ЯП не совсем подходит для написания сценариев и лучше обратить внимание на другие методы выполнения задачи. В IntelliJ IDEA присутствует удобный инструмент компилирования, поддерживаются горячие клавиши, а найденные ошибки сразу же отображаются в консоли. Умный редактор, предлагающий дополнить вводимый код, функционирует немного иначе, чем в других средах разработки, поскольку в этом случае разработчики решили сделать алгоритм подстраиваемым под контекст, а не общие правила.

 **2.3 Программы для написания кода**

IDLE

PyCharm

Visual Studio

Android Studio

Intellij IDEA

Eclipse

Netbeans

Komodo

Ruby Mine

Xcode

PhoneGap

Visual Studio

Компания Microsoft заинтересована в том, чтобы пользователи создавали программы для их операционных систем, задействовав в этом язык программирования C++. Специально для таких целей они создали бесплатную среду разработки под названием Visual Studio. Именно ее чаще всего рекомендуют использовать новичкам, как отличный интерпретатор при изучении C++. Однако Visual Studio еще поддерживает JavaScript, C#, Python и другие популярные ЯП. Особенные инструменты разработки делают это решение идеальным для новичков. Возьмем за пример опцию загорающейся лампочки. Она иногда появляется в тех строках, которые можно оптимизировать. Кликните по ней, чтобы узнать, что лучше поменять для оптимальной читаемости кода и его оптимизации. Конечно, исправление ошибок и автозаполнение строк при написании здесь тоже присутствует.

Android Studio

Android Studio — официальная среда разработки от создателей Android, которая предназначена для написания приложений под эту операционную систему. В этом ПО код пишется на языке Java, поскольку он является официальным ЯП для Android, или же можно использовать Kotlin, C или C++ в качестве дополнительных модулей. В Android Studio, как и в других профессиональных интерпретаторах, редактор имеет интеллектуальный режим, заранее просчитывающий слово, которое вы хотите написать. Весь синтаксис подсвечивается, а его цвет может быть настроен через параметры. Дополнительно в редакторе имеется визуальный редактор макетов. С помощью него осуществляется представление определенных объектов или блоков текста на экранах с разным разрешением.

Eclipse

Eclipse — еще одна популярная среда разработки, которая понравилась множеству пользователей из-за огромного количества подключаемых модулей. Изначально Eclipse была предназначена для написания кода под Java, а со временем вышло много внешних модулей, которые и позволили реализовать программирование, основываясь на других языках. Сейчас вы без проблем сможете подключить необходимое количество плагинов, вспомогательных не только при написании кода, но и для расширения общей функциональности программного обеспечения. Именно эта важная особенность делает Eclipse такой привлекательной для определенного круга юзеров. Конечно, все стандартные возможности, о которых мы уже упоминали при обзоре предыдущих сред разработки, в этом решении тоже присутствуют, работая по идентичному алгоритму.

**2.4 Программы для создания игр**

Программы для создания игр:

Game Maker

Unity 3D

Construct 2

Game Editor

3D Rad

Kodu Game Lab

CryEngine

Game Maker

Game Maker – это простая программа-конструктор для создания 2D и 3D-игр, позволяющая создавать их для большого числа платформ: Windows, iOS, Linux, Android, Xbox One и других. Но для каждой ОС игру нужно будет настраивать, так как Гейм Мейкер не гарантирует везде одинаковую работу игры. Плюсом конструктора является то, что он имеет низкий порог вхождения. Это значит, что если вы никогда не занимались разработкой игр, можно смело качать Game Maker – он не требует никаких специальных знаний в области программирования.

Unity 3D

Один из мощнейших и самых популярных игровых движков — Юнити 3Д. С его помощью можно создавать игры любой сложности и любого жанра, используя все тот же интерфейс визуального программирования. Хотя изначально создание полноценных игр на Unity 3D подразумевало знание таких языков программирования, как JavaScript или C#, сейчас они нужны только для более масштабных проектов. Движок предоставит вам массу возможностей, нужно только научиться их использовать. Для этого в интернете вы найдете тонны обучающего материала. Да и сама программа всячески помогает пользователю в его творчестве.

Kodu game Lab

Kodu Game Lab – это, наверное, лучший выбор для тех, кто начинает знакомиться с разработкой трехмерных игр. Благодаря красочному и понятному интерфейсу создавать игры в этой программе интересно и совсем несложно. Изначально этот проект разрабатывался для обучения школьников, но все же он будет полезен даже для взрослых. Программа очень хорошо помогает разобраться с тем, как работают и какой алгоритм создания игр. Кстати, для создания игры вам даже не понадобится клавиатура – все можно делать с помощью одной лишь мыши. Здесь не нужно писать код, нужно просто нажимать на объекты и события.

**Глава 3. Практический аспект исследования. Создание программного продукта.**

Мне нужен был свой язык программирования для моего проекта, поэтому я начал изучать, как устроены языки. Я остановился на интерпретаторе. Делать язык компилируемым – это очень сложно. Прежде всего, стоит обратить внимание на то, что языки программирования обычно делятся на 3 вида: объектно-ориентированные, функциональные, гибридные. Я хотел сделать именно функциональный.

Моей задачей было сделать свой интерпретатор на языке Python. Я искал примеры, но их было мало. Большинство информации было на английском.

Свою задачу я начал с изучения теории создания интерпретатора. Я прочитал несколько статей и пролистал несколько книг по данной теме. В итоге я сформировал представление о том, как создают интерпретаторы.

При создании интерпретатора нужно думать абстрактно. Например, выражение (Expression) “1 + 1” – это выражение из: выражения; оператора; выражения. Все выражения вычисляются по порядку с помощью абстрактно-синтаксического дерева (AST). Готовая программа, написанная на интерпретаторе – это и есть большое абстрактно-синтаксическое дерево. Это дерево собирается при помощи лексического анализатора и парсера. Лексический анализатор обрабатывает исходный текст программы в массив токенов. Например “1 + 1” – это токены: NUM(1); PLUS; NUM(2). Дальше из этих токенов в парсере собирается абстрактно-синтаксическое дерево.

**Простой калькулятор**

Я искал материалы, посвящённые созданию интерпретатора. Этот калькулятор мог вычислить, например такое выражение –“(2 \* 2) / 4 - 1”. Это работает не сложно. Бинарное выражение принимает два выражения и возвращает результат. Прописываем приветствие и правила, для того чтобы пользователю было понятнее, как пользоваться калькулятор.

 Таким образом, можно сделать бесконечную вложенность выражений. *(Наглядно фрагмент простого калькулятора представлен в приложении А).*

**Переменные**

Прописываем цикл «while», чтобы пользователь мог совершать несколько операций подряд, а не запускать программу заново. Цикл прекратится, когда пользователь пропишет вместо действия(+-/\*) 0. Создаем переменные для хранения данных и запрашиваем у пользователя их значение(числа и действие).

 *(Наглядный пример представлен в приложении А)*

**Операторы**

Они нужны для оперирования над данными (выражениями). Операторы обычно принимают на вход разные выражения, но ничего не возвращают. Прописываем проверку переменной «do» с помощью «if-elif-else», это значит «если -иначе если - иначе» . if do == '\*’: - если пользователь ввел «\*», то переменные(числа) перемножаться   print(f'Ответ: {first \* second}’) и так с другими значениями «do». Когда пользователь пропишет «0», то калькулятор выключится*. (Наглядный пример представлен в приложении А).*

*(Результат мы можем с вами посмотреть в приложении А).*

**Отгадай число**

Следующим этапов в создании интерпретатора были функции. Я сделал небольшую игровую программу **«Отгадай число»**. Прописываем приветствие и импортируем рандом, т.к. игра основана на угадывание рандомное число. Т.к. в приветствие мы спрашивали пользователя «готов ли он сыграть?», то сейчас надо проверить его ответ, если он Y , что равно ответу ДА, то игра начнется, если ответ N(нет), то игра не начнется, если она напишет, что-то другое, то игра не поймет его. Прописываем правила игры и даем возможность игроку выбрать сложность. Проверяем значение переменной, в которую пользователь «положил» уровень сложности. Для каждого уровня сложности свое количество попыток. Сообщаем это пользователю.

Я не стал делать язык объектно-ориентированным. Это усложнило бы весь язык в целом. Поэтому я оставил всё как есть, но вышло так, что у меня уже было готовое решение с объектами. Это были ассоциативные массивы. Они очень хорошо подходят в виде контейнера для хранения переменных и функций.

Всё получилось просто и элегантно.

Наглядный пример представлен в приложении Б.

По завершению работы над своим программным продуктом, я провел исследование среди обучающихся моей школы.

Цель исследования:

выяснить, есть ли у обучающихся школы интерес к программированию.

 Задачи:

 - подготовить вопросы для исследования;

 - провести процедуру исследования;

 - проанализировать результаты исследования;

 - сделать выводы по результатам исследования.

В анкетировании принимали участие обучающиеся 7-11 классов, т.е. те обучающиеся, которые изучают информатику в рамках школьного предмета. Всего в исследовании принимали участие 55 человек. Проанализировав полученные результаты я выяснил, что 53 человека из числа всех опрошенных хорошо знают кто такой программист и чем он занимается. 14 человек желают изучать языки программирования, затрудняются с ответом 13 человек. 15 человек думают, что языки программирования изучать сложно, а 27 человек затрудняются ответить на этот вопрос. 13 человек считают, что языки программирования изучать не вовсе не сложно. Могут фантазировать и воображать свой мир 36 человек, не умеют фантазировать 10 человек, не смогли ответить на вопрос 9 человек.

*(Итоговая диаграмма представлена в приложении В).*

**Вывод**

Если подвести итог по всей работе, то можно сказать что у меня получился вполне неплохой функциональный язык программирования. На его разработку у меня ушёл месяц. На протяжении всей разработки я изучал информацию в интернете и на её основе добавлял функционал в свой язык. И это оказалось не так сложно, как казалось в самом начале.

По результатам исследования можно сделать вывод о том, что в нашей школе есть обучающиеся, которым интересна тема программирования. Это 14 человек из общего числа опрошенных, которые уверены в своих силах, но их может быть гораздо больше, так как я выяснил, что нестандартным мышлением наделены 36 обучающихся, а раз они могут воображать свой мир, и у них развита фантазия, они вполне могут обучиться программированию. Это можно сделать, повторив мой путь к программированию, или пройти специальные курсы по программированию.

Гипотеза подтвердилась. Я научился программировать самостоятельно и создал свой программный продукт (язык программирования).

Я не собираюсь останавливаться на достигнутом и продолжу работу над своим проектом.

Таким образом, задачи решены, цель исследовательской работы достигнута.

**Заключение**

К 2020 году программирование настолько глубоко вошло в быт, что люди перестали замечать, как изменилась жизнь. Сотни тысяч привычных вещей не существовали бы без программирования или были бы гораздо менее удобными в использовании. Привычные бытовые приборы: микроволновая печь, стиральная машина — работают благодаря заложенным в них программам.

Всего 50 лет назад невозможно было представить, как легко будет найти любую необходимую информацию, насколько экономнее станет использоваться время, затрачиваемое на решение некоторых задач.

Поначалу для каждого крохотного действия приходилось создавать программу с нуля. Сейчас же программ написано такое множество, что в их разнообразии трудно ориентироваться. Например, чтобы смонтировать клип, придётся потратить часы на изучение существующих видеоредакторов и выбор подходящего.

Наше время диктует новые условия буквально в каждой области деятельности. Если компания хочет облегчить жизнь себе и своим клиентам, обойти конкурентов, то будет внедрять компьютерные технологии. Музыкальная индустрия и кино, мультипликация и игры, мобильная связь, банковское дело, дизайн и журналистика — все области тесно переплетены с программированием и используют специальные программы на разных этапах работы.

**Список использованных источников**

1.C\_Sharp [Электронный ресурс]‑URL:http://ru.wikipedia.org/wiki/C\_Sharp (Дата обращения 28.01.2019)

2.Java [Электронный ресурс]‑URL:<http://ru.wikipedia.org/wiki/Java> (Дата обращения 22.02.2019)

3.Python [Электронный ресурс]‑URL:<http://ru.wikipedia.org/wiki/Python> (Дата обращения 22.02.2019)

4.Ассемблер [Электронный ресурс]URL:[https://ru.wikipedia.org/wiki/Ассемблер](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80) (дата обращения 02.02.2019)

5.Записная книжка программиста-новичка, C#, SQL, PHP и все-все-все [Электронный ресурс]‑URL: <http://nullpro.info/2013/samouchitel-po-c-dlya-nachinayushhix-01-osnovy-yazyka-peremennye-logika-cikly/> Записная книжка программиста-новичка, C#, SQL, PHP и все-все-все (Дата обращения 26.02.2019)

6.Интерпретатор [Электронный ресурс]URL:[https://ru.wikipedia.org/wiki/Интерпретатор](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80) (дата обращения 02.02.2019)

7.Компилятор[Электронный ресурс]‑URL:https://ru.wikipedia.org/wiki/Компилятор (Дата обращения 28.01.2019)

8.Курносов, А. П., Кулев, С. А., Улезько А. В. и др. Информатика /под ред. А. П. Курносова. М.: КолосС. 2005. – 272 с.

9.Макарова, Н. В. Информатика /под ред. Проф. Н. В. Макаровой. – М.: Финансы и статистика. 1997. – 768 с.

10.Малышев, Р. А. Локальные вычислительные сети: Учебное пособие/ РГАТА. – Рыбинск, 2005. – 83 с.

11.Обучение программированию с нуля – пошаговое руководство для начинающих. Десять шагов от чайника к программисту-профессионалу [Электронный ресурс]‑URL: <http://geek-nose.com/obuchenie-programmirovaniyu-s-nulya/> Обучение программированию с нуля – пошаговое руководство для начинающих. Десять шагов от чайника к программисту-профессионалу (Дата обращения 13.02.2019)

12.Островский, В. А. Информатика: учеб. для вузов. М.: Высшая школа, 2000. –511 с.

13.Полное руководство по языку программирования С# 7.0 и платформе .NET 4.7 [Электронный ресурс]‑URL: <https://metanit.com/sharp/tutorial/> Полное руководство по языку программирования С# 7.0 и платформе .NET 4.7 (Дата обращения 21.01.2019)

14.Семакин, И. А. Информатика: Базовый курс /Семакин И. А ., Залогова Л ., Русаков С., Шестакова Л. – Москва: БИНОМ.2005. – 105с.

15.Симонович, С. В. Информатика. Базовый курс/Симонович С. В. и др. – СПб.: издательство Питер. 2000. – 640 с.

16.СС++ [Электронный ресурс]‑URL:http://ru.wikipedia.org/wiki/СС++ (Дата обращения 31.01.2019

ПРИЛОЖЕНИЕ А

Простой калькулятор

 

(Рисунок 1)

Переменные

 

(Рисунок 2)

Операторы



(Рисунок 3)

Результат









**Отгадай число**







 





Результат















 

ПРИЛОЖЕНИЕ Б

Вопросы анкетирования

1.Знаете ли Вы кто такой программист?

а) Да б) Нет в) Затрудняюсь ответить

2.Хотели бы Вы начать изучать программирование?

а) Да б) Нет в) Затрудняюсь ответить

3.Знаете ли Вы, чем занимаются программисты?

а) Да б) Нет в) Затрудняюсь ответить

4.Часто ли Вам удается фантазировать, воображать, создавать свой мир?

а) Да б) Нет в) Затрудняюсь ответить

5.Как Вы думаете, сложно ли изучать языки программирования?

а) Да б) Нет в) Затрудняюсь ответить

6. Интересна ли Вам тема программирования?

а) Да б) Нет в) Затрудняюсь ответить

ПРИЛОЖЕНИЕ В

1. Знаете ли Вы, кто такой программист?

( Рисунок 1)

2. Хотели бы Вы начать изучать программирование?

(Рисунок 2)

3. Знаете ли Вы, чем занимаются программисты?

(Рисунок 3)

4. Часто ли Вам удается фантазировать, воображать, создавать свой мир?

(Рисунок 4)

5. Как Вы думаете, сложно ли изучать языки программирования?

(Рисунок 5)

6. Интересна ли Вам тема программирования?

(Рисунок 6)

Итоговая диаграмма по результатам исследования

(Рисунок 7)