**Муниципальное бюджетное общеобразовательное учреждение**

**города Новосибирска**

**«Аэрокосмический лицей имени Ю.В. Кондратюка»**

****

**Лицейская конференция школьников**

**«Человек. Земля. Вселенная»**

**Секция: Информационные технологии**

**ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА**

**по теме:**

***Применение различных систем счисления в цифровой технике***

**Учащийся:**

**Бабушкин Сергей Сергеевич**

**Класс: 8А**

**Руководитель:** Крисько Анжелика Александровна

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc192781356)

[Глава 1. Системы счисления в теории 4](#_Toc192781357)

[1.1. Понятие системы счисления 4](#_Toc192781358)

[1.2. Двоичная система счисления 4](#_Toc192781359)

[1.3. Восьмеричная система счисления 5](#_Toc192781360)

[1.4. Шестнадцатеричная система счисления 6](#_Toc192781361)

[1.5. Компиляторы 7](#_Toc192781362)

[1.6. Переводы систем счисления в программирование на языке C++ 10](#_Toc192781363)

[ГЛАВА 2. Создание продуктов 12](#_Toc192781364)

[2.1. Продукт I 12](#_Toc192781365)

[2.2. Продукт II 13](#_Toc192781366)

[ЗАКЛЮЧЕНИЕ 20](#_Toc192781367)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 21](#_Toc192781368)

**ВВЕДЕНИЕ**

Различные системы счисления везде: от обычной математики до языков программирования и машинного кода, где есть цифры. Тему с системами счисления проходят на уроках информатики. И вся информация на цифровых устройствах закодирована с помощью систем счисления.

**Актуальность:** цифровой мир каждый день расширяется, а производительность, скорость работы, анализа данных и вычислительных функций напрямую зависят от скорости перевода из программного кода в двоичную систему счисления(генерации кода). Шифрование данных, которое защищает цифровое пространство и не позволяет «черным» хакерам получать доступ к скрытым файлам идет также в различных системах счисления, узнать основание которой является их целью.

**Цель:** создание программы для обработки систем счисления.

**Задачи:**

1. Изучить системы счисления и их применение.
2. Изучить компиляторы/практическое приминение систем счисления в современном мире
3. Создать продукт.

**Глава 1. Системы счисления в теории**

**1.1. Понятие системы счисления**

Система счисления — символический метод записи чисел, представление чисел с помощью письменных знаков.

Система счисления имеет ряд особенностей:

* даёт представления множества чисел (целых и/или вещественных);
* даёт каждому числу уникальное представление (или, по крайней мере, стандартное представление);
* отражает алгебраическую и арифметическую структуру чисел.

Выделяется два вида систем счисления:

* позиционные;
* непозиционные.

В позиционных системах счисления один и тот же числовой знак (цифра) в записи числа имеет различные значения в зависимости от того места (разряда), где он расположен.

В непозиционных системах счисления величина, которую обозначает цифра, не зависит от положения в числе. При этом система может накладывать ограничения на положение цифр, например, чтобы они были расположены в порядке убывания.

Основными используемыми понятиями, которые относятся к данной теме, можно назвать:

* основание системы – количество цифр в ней;
* разряд – индекс цифры, начинающийся с нуля и отсчитываемый «справа-налево»;
* бит – простая (минимальная) единица представления информации в бинарной системе, представленная 0 или 1.

Также с помощью различных систем счисления используется шифрование данных, идущих по открытым каналам. Человек, который захочет украсть ваши данные не сможет это сделать так как просто не будет знать основание системы счисления шифровки.

**1.2. Двоичная система счисления**

Двоичная система счисления — позиционная система счисления с основанием 2. Благодаря непосредственной реализации в цифровых электронных схемах на логических вентилях, двоичная система используется практически во всех современных компьютерах и прочих вычислительных электронных устройствах.

Благодаря тому, что множеству символов {0,1} можно сопоставить два устойчивых состояния электронных вентилей «выключено/включено», основное применение двоичная система счисления нашла в вычислительной технике, в цифровых системах контроля и управления. Ядром любого процессора является арифметико-логическое устройство (АЛУ), которое воспринимает управляющие сигналы и операнды в виде двоичных кодов и чисел. Результаты обработки информации также формируются в виде двоичных чисел и результатов логических сопряжений. Этот процесс развился потому, что двоичные сигналы имеют высокую надёжность при передаче и хранении информации, а правила выполнения команд просты и удобны.

Однако стоит отметить, что в 1959 году в МГУ была создана «Се́тунь» — малая ЭВМ на основе троичной логики: где -1 это ложь, 0 это неопределённость, 1 это истина.

Явные минусы двоичной системы обусловлены тем, что на интуитивном уровне людям она чужда — в отличие, например, от десятичной. И это — первый недостаток. Отметим остальные:

* длинная запись, неудобство с большими числами;
* долгое время ручных вычислений;
* не применяется в повседневной жизни (если, конечно, вы не компьютер).

Её плюсы:

* позиционная система, имеет разряды;
* применимы арифметические действия;
* можно построить логику;
* подходит для шифровки данных;
* родной язык компьютерных систем.

**1.3. Восьмеричная система счисления**

Восьмеричная система счисления — позиционная целочисленная система счисления с основанием 8. Для представления чисел в ней используются цифры от 0 до 7.

Восьмеричная система чаще всего используется в областях, связанных с цифровыми устройствами. Характеризуется лёгким переводом восьмеричных чисел в двоичные и обратно, путём замены восьмеричных чисел на триплеты двоичных. Широко использовалась в программировании и компьютерной документации, однако позднее была почти полностью вытеснена шестнадцатеричной.

Восьмеричная система применяется при выставлении прав доступа к файлам и прав исполнения для участников в Linux-системах.

Восьмеричная система счисления имеет вспомогательный характер, ее удобно использовать для сокращенной записи бинарных комбинаций чисел. Она более удобна в работе чем двоичная, так как использует меньшее количество разрядов. Восьмеричная система применялась в свое время для программирования на машинном языке, а также в устройствах подготовки данных, вышедших из употребления с появлением персональных компьютеров.

Арифметические действия в системе счисления с основанием 8 выполняются так же, как и в десятичной. Удобнее всего складывать и вычитать большие числа столбиком. Только следует помнить, что после 7 идет 10, то есть сумма восьмеричных чисел 3 + 5 = 10, а не восемь.

**1.4. Шестнадцатеричная система счисления**

Шестнадцатеричная система счисления — позиционная система счисления по основанию 16.

В качестве цифр этой системы счисления обычно используются цифры от 0 до 9 и латинские буквы от A до F. Буквы A, B, C, D, E, F имеют значения 1010, 1110, 1210, 1310, 1410, 1510 соответственно.

Широко используется в низкоуровневом программировании и компьютерной документации, поскольку в современных компьютерах минимальной адресуемой единицей памяти является 8-битный байт, значения которого удобно записывать двумя шестнадцатеричными цифрами, а значение половины байта — полубайта — одной цифрой. Такое использование началось с системы IBM/360, где вся документация использовала шестнадцатеричную систему, в то время как в документации других компьютерных систем того времени (даже с 8-битными символами, как, например, PDP-11 или БЭСМ-6) использовали восьмеричную систему счисления.

В стандарте Unicod номер символа принято записывать в шестнадцатеричном виде, используя не менее 4 цифр (при необходимости — с ведущими нулями).

Шестнадцатеричный цвет — запись трёх компонентов цвета (R, G и B) в шестнадцатеричном виде. Например, цвет, имеющий шестнадцатеричный номер 00FF00, — ярко-зелёный.

**1.5. Компиляторы**

Компилятор получает на вход файл с кодом на каком-то языке программирования. Он преобразовывает конструкции языка в формат, понятный компьютеру, и возвращает файл, который тот сможет выполнить.

Чтобы преобразовать исходный код, компилятор использует собственный словарь с определениями — например, оператор «if» меняет на 11010011100110, а сложение — на 101011. Он делает это, пока не закончатся все строки в файле. Получается исполнительный файл, который выглядит так:

001011011010010101110101010101010100001100001110111100110100001010001001110…

В таком формате компьютеру уже удобно читать инструкции и выполнять их. А значит, компилятор сделал свою работу хорошо.

Компилирование состоит из пяти этапов: синтаксического анализа, парсинга, семантического анализа, оптимизации и генерации кода. Давайте разберём каждую стадию.

Синтаксический анализ. Это что-то вроде разбора грамматики языка. Когда мы пишем код, то следуем определённым правилам — синтаксису. Например, в Java между командами ставим точку с запятой. Если этого не сделать, то получим ошибку.

На этапе синтаксического анализа компилятор проверяет, соответствует ли код правилам конкретного языка программирования. И пока он не думает о том, что именно написано, — проверка идёт только по формальным признакам.

Парсинг. На этом этапе компилятор разбивает код на маленькие кусочки — токены. Каждый токен — это какое-то слово или символ, например if, while, int или (.

Из токенов строится синтаксическое дерево, которое содержит слова и символы, и пригодится на следующем этапе — семантическом анализе. Каждый узел дерева — это либо операция, например сложение, либо переменная. Обычно, когда мы доходим до переменной, то дальше ветви не разрастаются.

Ниже рассмотрим, как выглядит такое дерево.

Допустим, у нас есть простой код со сложением двух чисел:

Здесь пять токенов: x, =, 5, + и 3. Пробелы считать не будем. Ниже представлено дерево, построенное из рассмотренных токенов (рис. 1).

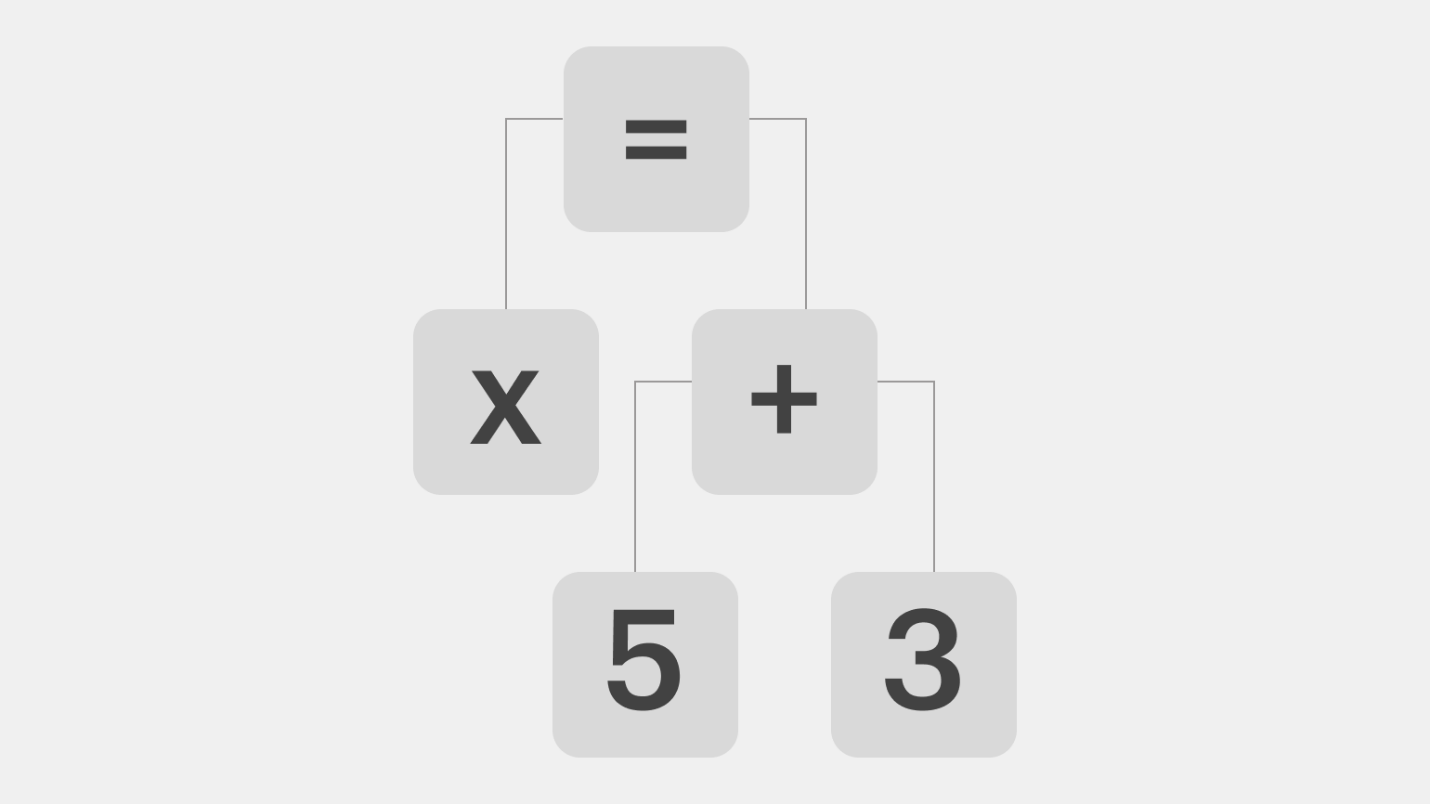


Рисунок 1 – Дерево сложения чисел

Мы видим, что на вершине находится главная операция — присваивание переменной x результата сложения двух чисел. От неё отходит две ветви — сама переменная x и символ сложения, который ветвится на слагаемые числа.

В процессе парсинга компилятор не понимает, зачем нужен каждый из токенов. Пока что он машинально выполняет свою работу — думать будет на следующем этапе.

Семантический анализ. Компилятор начинает вдумываться в то, что написано в коде, анализируя составленное синтаксическое дерево. Например, если мы объявили переменную, он понимает, что это значит и какие операции можно с ней выполнить.

Ещё компилятор на этом этапе может предполагать, какие именно действия с переменной возможны. Если он видит, что у нас есть переменная неизменяемого типа, например константа, то при попытке кода её изменить, выдаст ошибку.

Оптимизация. Когда синтаксис разобран и стало понятно, что делает программа, время ускорить работу кода. Компилятор ищет способы повысить скорость его выполнения или уменьшить количество занимаемой им памяти.

Самый простой пример оптимизации — умножение на ноль. Например, у нас есть фрагмент кода:

Чтобы определить значение переменной y, потребуется сначала вычислить сложную формулу для переменной x. Но мы, люди, сразу видим, что при умножении на ноль, результат будет нулём, а значит, смысла считать переменную x нет. Компилятор тоже видит такие вещи — и не будет вычислять то, что вычислять бесполезно. Он просто заменит эти две строки кода на одну:

Это действительно удобнее, но это сработает только в том случае, если переменная x не пригодится нам в программе дальше.

Это возможно из-за особенностей работы компилятора — он не выполняет код, а сначала читает его и ищет способы оптимизации программы.

Генерация кода. Синтаксис разобран, анализ проведён, код оптимизирован — пора перевести его на язык компьютера. На этом этапе все команды, что мы писали на языке программирования, переводятся в машинные инструкции.

После перевода мы получаем исполняемый файл, например в формате «.exe», который можно запустить и проверить работу программы. На этом компиляция завершается.

Самый первый компилятор написан на языке ассемблера, потому что программистам нужно было как-то упростить себе работу с машинным кодом. Работают они так:

* разработчик пишет код на ассемблере;
* компилятор переводит его в машинные инструкции;
* компьютер запускает эти инструкции.

Получается, что компилятор на ассемблере — это другая программа на нём же, которая умеет переводить код. Например, она подставляет вместо команды jmp строку 001110111, которая запускает нужные шестерёнки внутри процессора.

После уже появились языки более высокого уровня — например, C. Компилятор для C написан на том же ассемблере. Работает он похожим образом:

* разработчик пишет программу на C;
* компилятор переводит команды на языке программирования с C в машинные инструкции;
* компьютер запускает эти инструкции.

Дальше — вверх по высокоуровневости языков программирования. Компилятор на С++ написан на C, а для JavaScript — на C++. Но если спускаться по цепочке, то мы рано или поздно придём к ассемблеру.

**1.6. Переводы систем счисления в программирование на языке C++**

В языке программирования C++ присутствует оператор автоматического конвертера при вводе/выводе чисел из восьмеричной, десятичной, шестнадцатеричной систем счисления.

Допустим вводится целочисленная переменная «n». Ввод этой переменной в десятичной системе счисления выглядит так:

cin>>dec>>n;

или так(ввод по умолчанию):

cin>>n;

Ввод этой переменной в восьмеричной системе счисления выглядит так:

cin>>oct>>n;

Ввод этой переменной в шестнадцатеричной системе счисления выглядит так:

cin>>hex>>n;

Допустим выводится целочисленная переменная «n». Вывод этой переменной в десятичной системе счисления выглядит так:

cout<<dec<<n;

или так (вывод по умолчанию):

cout<<n;

Вывод этой переменной в восьмеричной системе счисления выглядит так:

cout<<oct<<n;

Вывод этой переменной в шестнадцатеричной системе счисления выглядит так:

cout<<hex<<n;

Но эти преобразования ограничиваются целыми и положительными числами.

Я в качестве Продукта 2 сделал программный код и сайт, где можно загрузить себе код или библиотеку, для преобразования чисел от 2 до 16 системы счисления.

**ГЛАВА 2. Создание продуктов**

**2.1. Продукт I**

В качестве продукта проекта было решено провести опрос среди учеников 8А класса МБОУ «АКЛ им. Ю.В. Кондратюка». Вопросы были направлены на то, чтобы проверить уровень знаний учеников по теме «системы счисления».

Список вопросов представлен в приложении (см. Приложение 1), где правильный ответ начинается с плюса.

Всего было опрошено-18 человек и результаты представлены в приложении (см. Приложение 2).

Одинаковые доли опрошенных (11%) людей, написавших на 0% вообще никак не интересуются системами счисления, (28%) людей, написавших на 20% знают немного информации о этой теме, (33%) людей, а именно большинство, написавших на 40% знают про системы счисления на среднем уровне, (16%) людей, написавших на 60% знают про системы счисления выше среднего, (0%) написавших на 80%, (11%) написавших на 100% очень хорошо понимают тему системы счисления, по пройденной информации в школе или после дополнительного самосовершенствования

**2.2. Продукт II**

Программный код на языке C++ для перевода из систем счисления от двоичной до шестнадцатеричной в десятичную системы счисления и из десятичной в от двоичной до шестнадцатеричной системы счисления.

(См. Приложение 3)

Также я создал заголовочный файл, состоящий из функций этого кода (кроме main()).

И выложил код и библиотеку на сайт, адаптированный под различные утсройства и различные системные настройки(в частности настройки темы по умолчанию браузера).

Преимуществами этого кода/заголовочного файла в сравнении с остальными сервисами является:

* Open source;
* Возврат результата в переменную, а не обычный вывод;
* Работа с отрицательными и дробными числами;
* Использование как строчных так и заглавных символов;
* Быстрый и низкоуровневый C++;
* Вводимые числа могут быть лимитом типа данных «int»;
* Удобство использования заголовочного файла в программах;
* Легкодоступность скачивания.

**ЗАКЛЮЧЕНИЕ**

Системы счисления встречаются во всем, что связано с цифровой электроникой. Они активно помогают нам в различных задачах и без них множество целей получались сложнее, чем могли бы быть. Многие люди немного знают о системах счисления, хотя пользуются ими в разном виде каждый день. Системы счисления являются основополагающими математики и программирования. Без них можно было бы забыть о цифровом мире.

У этого проекта среди вариантов дальнейшего развития могут быть: анализ "словаря" компилятора; проведение лекций по системам счисления, просмотр на то, как люди усваивают эту тему; попытка программирования на двоичном коде; создание новых операций на другом основании системы счисления; создание полноценного приложения или сайта с переводом вводимых данных; создание собственного компилятора.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Система счисления // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Система\_счисления (дата обращения: 14.11.2024).
2. Системы счисления и их особенности // OTUS URL: https://otus.ru/journal/sistemy-schisleniya-i-ih-osobennosti/ (дата обращения: 14.11.2024).
3. Двоичная система счисления // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Двоичная\_система\_счисления (дата обращения: 14.11.2024).
4. Двоичная (бинарная) система счисления: что это и как ей пользоваться // Skillbox URL: https://skillbox.ru/media/code/dvoichnaya-binarnaya-sistema-schisleniya-chto-eto-i-kak-ey-polzovatsya (дата обращения: 14.11.2024).
5. Сетунь (компьютер) // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Сетунь\_(компьютер) (дата обращения: 14.11.2024).
6. Восьмеричная система счисления // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Восьмеричная\_система\_счисления (дата обращения: 14.11.2024).
7. Восьмеричная система счисления // Образовака URL: https://obrazovaka.ru/informatika/vosmerichnaya-sistema-schisleniya-kak-perevodit.html (дата обращения: 14.11.2024).
8. Шестнадцатеричная система счисления // ВикипедиЯ URL: https://ru.wikipedia.org/wiki/Шестнадцатеричная\_система\_счисления (дата обращения: 14.11.2024).
9. Двоичная система счисления // Знание.Вики URL: https://znanierussia.ru/articles/Двоичная\_система\_счисления#:~:text=Современная%20двоичная%20система%20была%20полностью,и%20в%20современной%20двоичной%20системе (дата обращения: 14.11.2024).
10. Что такое компилятор и как он работает // Skillbox URL: https://skillbox.ru/media/code/chto-takoe-kompilyator-i-kak-on-rabotaet/ (дата обращения: 14.11.2024).

**Приложение 1**

**Вопросы опроса**

Как может применяться двоичная система счисления?

Применяется при выставлении прав доступа к файлам и прав исполнения для участников в Linux-системах;

Широко используется в низкоуровневом программировании и компьютерной документации;

+ Ядром любого процессора является АЛУ, которое воспринимает управляющие сигналы и операнды в виде двоичных кодов и чисел. Результаты обработки информации также формируются в виде…

Как может применяться восьмеричная система счисления?

+ Применяется при выставлении прав доступа к файлам и прав исполнения для участников в Linux-системах;

Широко используется в низкоуровневом программировании и компьютерной документации;

Ядром любого процессора является АЛУ, которое воспринимает управляющие сигналы и операнды в виде двоичных кодов и чисел. Результаты обработки информации также формируются в виде…

Как может применяться шестнадцатеричная система счисления?

Применяется при выставлении прав доступа к файлам и прав исполнения для участников в Linux-системах;

+ Широко используется в низкоуровневом программировании и компьютерной документации;

Ядром любого процессора является АЛУ, которое воспринимает управляющие сигналы и операнды в виде двоичных кодов и чисел. Результаты обработки информации также формируются в виде…

Расшифруйте из двоичной системы счисления.

1101001011001110110011011100110111000000

А-11000000; Д-11000100; Е-11000101; К-11001010; Н-11001101; О-11001110; Т-11010010; Ч-11010111

+ тонна

Закодируйте в восьмеричную систему счисления. Слово-КОНЕЧНО.

А-300; Д-304; Е-305; К-312; Н-315; О-316; Т-322; Ч-327

+312316315305327315316

**Приложение 2**

**Результаты опроса**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Опрос | | | | | | |
| Результат в процентах | 0 | 20 | 40 | 60 | 80 | 100 |
| Количество людей | 2 | 5 | 6 | 3 | 0 | 2 |

Объект OLE

**Приложение 3**

**Код для перевода между системами счисления, реализованный на C++**

#include <iostream>

#include <cstdlib>

#include <cmath>

using namespace std;

string binary(double k, int s){

string e, ee;

string m;

int kk=int(k);

if(kk<0){

kk\*=(-1);

m='-';

}

if(kk==0){

e='0';

}

else{

int z;

while(kk!=0){

z=kk%s;

kk/=s;

ee=to\_string(z);

if(z>9){

ee=char(z+55);

}

e=ee+e;

}

if(int(k)!=k){

e=e+'.';

double drch=k-int(k);

while(drch!=0){

drch\*=s;

z=int(drch);

drch=drch-z;

ee=to\_string(z);

if(z>9){

ee=char(z+55);

}

e=e+ee;

}

}

}

return e;

}

double bbinary(int k[], int w, int s, int x){

double z, y;

int j=x+1;

y=0;

for(int h=0; h<x; h++){

z=k[h]\*pow(s, (x-h-1));

y+=z;

}

for(x=-1; j<w; x--){

z=k[j]\*pow(s, x);

y+=z;

j++;

}

return y;

}

double ibinary(string n, int s){

int q;

double l;

q=n.length();

int x=q;

if(n[0]=='-'){

q--;

x--;

}

char pp[q];

if(n[0]!='-'){

for(int k=0; k<q; k++){

pp[k]=n[k];

}

}

else{

for(int k=0; k<q; k++){

pp[k]=n[k+1];

}

}

int ch[q];

for(int k=0; k<q; k++){

if(int(pp[k])>64 and int(pp[k])<55+s){

ch[k]=int(pp[k])-55;

}

if(int(pp[k])>96 and int(pp[k])<87+s){

ch[k]=int(pp[k])-87;

}

if(pp[k]=='.' or pp[k]==','){

x=k;

if((q-1)!=k){

ch[k]=-1;

}

else{ch[k]=0;}

}

if(int(pp[k])>47 and int(pp[k])<58){

ch[k]=pp[k]-'0';

}

}

l=bbinary(ch, q, s, x);

if(n[0]=='-'){

l\*=(-1);

}

return l;

}

int main(){

int s;

double m, l;

string n, j;

start:

cout<<"Введите систему счисления: ";

cin>>s;

if(s>16 or s<2){

cout<<"Введено некорректное значение."<<endl;

goto start;

}

cout<<"Введите 1 число в "<<s<<" системе счисления: ";

cin>>n;

if(s<=10){

for(int z=0; z<(n.length()); z++){

if(not(int(n[z])>47 and int(n[z])<s+48 or (n[z]=='.' or n[z]==','))){

cout<<"Введено некорректное значение."<<endl;

goto start;

}

}

}

if(s>10){

for(int z=0; z<(n.length()); z++){

if(not((int(n[z])>64 and int(n[z])<s+55) or (int(n[z])>96 and int(n[z])<s+87) or (int(n[z])>47 and int(n[z])<s+48) or (n[z]=='.' or n[z]==','))){

cout<<"Введено некорректное значение."<<endl;

goto start;

}

}

}

cout<<"Введите 2 число в 10 cистеме счисления: ";

cin>>m;

l=ibinary(n, s);

string sss=to\_string(l);

while(sss.back()=='0'){

sss.erase(sss.length()-1, 1);

}

if(sss.back()=='.'){

sss.erase(sss.length()-1, 1);

}

cout<<"1 число в 10 системе счисления:"<<endl<<sss<<endl;

j=binary(m, s);

cout<<"2 число в "<<s<<" системе счисления:"<<endl<<j;

return 0;

}