Муниципальное автономное общеобразовательное

учреждение г. Рязани «Лицей №4»

# Нейронная сеть для профориентации

Выполнил ученик 11Б класса

Марюнин Никита

Руководитель

Мирохина Ольга Игоревна

Рязань

2023

# Аннотация

Цель проекта - создать программу, способную помочь подросткам определиться со своей будущей профессией. Для этого автор использует такое направление искусственного интеллекта, как нейронная сеть. В связи с этим в задачи проекта входит сбор сведений о принципах работы нейросетей и их устройстве, выделение основной информации, использование полученных данных на практике и вывод продукта в открытый доступ.

Проект знакомит с исследованием проблемы профессиональной ориентации у молодёжи. Для помощи в решении этого вопроса, был проведён разбор темы искусственного интеллекта, а именно: определение перспектив искусственного интеллекта, разбор устройства нейронных сетей и анализ способа их реализации для далее представленного практического воплощения идеи.

В ходе выполнения работы, были рассмотрены все главные сведения о проблемах в выборе профессии у подростков, собраны данные о нейронных сетях и написана и выложена в открытый доступ рабочая нейросеть под названием “Нейровыбор”, которую может использовать или доработать любой человек.

Как итог, исследование даёт базовое представление о нейронных сетях, а продукт проекта может помочь натолкнуть пользователя на решение поставленной проблемы.

Содержание

[Введение 4](#_Toc140783693)

[Основная часть 5](#_Toc1)

[1. Теоретическая часть 5](#_Toc140783695)

[1.1 Выбор профессии 5](#_Toc140783696)

[1.2 Перспективы ИИ 5](#_Toc140783697)

[1.3 История создания нейронных сетей 6](#_Toc140783698)

[1.4 Сущность нейронных сетей 7](#_Toc140783699)

[2. Аналитическая часть 8](#_Toc140783700)

[2.1 Выбор языка 8](#_Toc140783701)

[2.2 Основы 8](#_Toc140783702)

[2.3 Идея: ввод и вывод 9](#_Toc140783703)

[2.4 План кода](#_Toc140783704) 10

[3. Практическая часть 11](#_Toc140783705)

[3.1 Библиотеки и пространство имён 11](#_Toc2)

[3.2 Реализация ввода базы данных 12](#_Toc10)

[3.3 Русский язык в консоли 13](#_Toc11)

[3.4 Рандом и время 14](#_Toc12)

[3.5 Объявление переменных 14](#_Toc13)

[3.6 Прямое распространение сигнала 16](#_Toc14)

[3.7 Обратное распространение ошибки 18](#_Toc15)

[3.8 Примеры 19](#_Toc7)

[3.9 Проверка корректности работы нейросети 21](#_Toc8)

[3.10 Интерфейс 24](#_Toc9)

[3.11 Тестирование 26](#_Toc16)

[3.13 Теперь на GitHub 28](#_Toc21)

[Заключение 29](#_Toc140783706)

[Использованные материалы 31](#_Toc140783707)

[Приложения 32](#_Toc140783708)

# Введение

**Нейронная сеть** – математическая модель, а также её программное или аппаратное воплощение, построенные по принципу организации и функционирования биологических нейронных сетей.

**Актуальность** создания программы для профориентации заключается в частом возникновении **проблем** при определении человеком своего профессионального пути, что напрямую влияет на состояние рынка труда.

**Объектом исследования** являются увлечения и предпочтения людей.

**Предметом исследования** стал самообучающийся искусственный интеллект.

**Цель проекта:** создать программу, способную помочь подросткам определиться со своей будущей профессией.

**Задачи проекта:**

* Собрать сведения о принципах работы нейросетей и их устройстве
* Выделить основную информацию
* Использовать полученные данные на практике
* Выложить продукт в открытый доступ

**Методы исследования:**

* Анализ литературы
* Компьютерное моделирование

**Практическая значимость** проекта состоит в оказании помощи учащимся, не сделавшим свой выбор по той или иной причине.

# Основная часть­

## 1. Теоретическая часть

### 1.1 Выбор профессии

*Профориентация* – это комплекс действий для выявления у человека склонностей и талантов к определённым видам профессиональной деятельности, а также система действий, направленных на помощь в выборе карьерного пути людям всех возрастов.

*Выбор профессии* и способа получения необходимого для неё высшего образования – одно из самых главных решений в жизни, ведь от именно этого выбора зависит будущее каждого.

Чем раньше человек делает свой выбор, тем лучше. Однако, более половины всех школьников не знает на кого поступать после окончания школы. На самом деле, это очень печальная статистика. Большинству старшеклассников трудно понять к чему они обладают предрасположенностью и испытывают наибольший интерес. Для того, чтобы это сделать, необходимо изучить свои склонности и особенности, увидеть сильные и слабые стороны.

Если самостоятельно разобраться в себе не получается, следует прибегнуть к помощи со стороны. Хорошим вариантом будет узнать мнение своих близких и знакомых, побеседовать со студентами и профессорами. Или же воспользоваться бесплатными услугами или программами в интернете, которые, возможно, смогут натолкнуть на правильную мысль. В частности, такая программа может являться специализированным искусственным интеллектом.

### 1.2 Перспективы ИИ

*Искусственный интеллект* уверенно входит в нашу жизнь. Идея создания ИИ сформировалась из стремлений понять устройство человеческого мозга, из убеждений в том, что каждый отдельный элемент механизма мозговой системы можно описать языком цифр и, таким образом, создать машину, действующую по принципу работы мозга человека.

В данный момент искусственный интеллект ещё не способен мыслить, чувствовать и действовать, как человек, но современные технологии и наука развиваются в геометрической прогрессии, поэтому это лишь вопрос времени. В научном сообществе по большей части его будущая форма рассматривается исключительно, как программа с ограниченным функционалом, включающая в себя чёткое выполнение поставленных задач для упрощения жизни людей. Однако, вполне вероятно, что когда-то люди смогут создать ПО, многократно превосходящее по интеллекту человека, ставшее во много раз эффективнее. Чем умнее становится машина, тем быстрее она может развиваться. Так, машина сможет опережать нас на несколько ступень развития за несколько минут, каждый раз повышая количество ступеней и уменьшая время, за которое она это делает. Выходит, вполне реально, что человечество само создаст то, что превышает его развитие в миллионы или миллиарды раз.

Перспективы искусственного интеллекта уже давно стали определены в сфере бизнеса. Существуют ИИ, что планируют расписания, осуществляют автономное управление транспортом, проводят диагностику чего-либо или достойные партии в шахматы. Они упрощают работу экспертов и делают жизнь комфортнее.

Результаты работы некоторых программ, обладающих ИИ, люди используют ежедневно. Например, состав ленты в социальных сетях и показ рекламы на основе ваших предпочтений, всеми известная Алиса и другие элементы «Умного дома», приход уведомлений от банка при совершении транзакций с нового устройства. Всё это осуществляется или является искусственным интеллектом.

ИИ способен регулировать многие процессы без человеческой помощи, тем самым способствуя увеличению продуктивности самого человека. С каждым днём его влияние на нашу жизнь только усиливается.

### 1.3 История создания нейронных сетей

В роли одного из направлений искусственного интеллекта выступают *нейронные сети*. Именно благодаря им мир возможно сделать наиболее персонализированным. Их суть лежит в особенностях работы человеческого мозга, по аналогии с которыми они и были разработаны.

Сам термин «нейронная сеть» появился в середине XX века. В 1943 году нейрофизиолог Уоррен Мак-Каллок и математик Уолтер Питтс в своей научной работе предложили математическую модель нейрона и создали нейронную сеть на электрических схемах. В 1949 году физиолог Дональд Хебб первым предположил, что обучение заключается в изменениях силы *синаптических связей*. К 1957 году Фрэнк Розенблатт представил *персептрон* – модель машинного обучения, созданную для помощи компьютерам в обучении на разных данных. Данная *однослойная* нейронная сеть научилась складывать и вычитать без единой строчки программного кода. Уже в 1959-м появилась первая нейронная сеть, которая смогла реально помогать человеку. Таковой являлась MADALINE (улучшенная версия другой модели ADALINE), созданная Бернардом Уидроу и Марсианом Хоффом, которая убирала эхо из телефонной линии. На этой стадии развития направления разработчики увидели в нём перспективы, однако, из-за недостатка вычислительной мощности компьютеров того времени, развитие в этой области продолжилось только спустя десять с лишним лет. Так, в 1975 году Кунихикой Фукусимой был разработан *когнитрон*, обладающий конкурентным обучением, который стал одной из первых *многослойных* нейронных сетей, чуть позже им также была разработана более продвинутая сеть – *неокогнитрон,* относящаяся к сетям *свёрточного* типа.

В нулевых стали доступны к использованию действительно большие объёмы данных, в следствие чего стали разрабатываться алгоритмы глубокого обучения. Это стало большим толчком в развитии нейронный сетей и привело к началу распространения их практического применения. В 2020-м году кампания OpenAl создала модель GPT-3, а в конце 2022 года выпустила Chat-GPT, который получил статус самого быстрорастущего онлайн-сервиса в истории. В связи с этим, другие мировые корпорации стали активно реализовывать свои новые проекты.

### 1.4 Сущность нейронных сетей

Как же работает это чудо науки? Нейросети воссоздают процессы, происходящие в мозге, пока человек над чем-либо работает. Они способны *самостоятельно* обучаться и развиваться, строя свой опыт на совершённых ошибках.

Но любой будущей нейросети нужен некий толчок, в качестве которого выступает так называемая *обучающая выборка*. Это ряд примеров, ответы на которые известны заранее. Ответы должны быть достоверными, потому что, если предоставить плохой пример, она и будет им руководствоваться. Уже на основе выборки можно написать *главную функцию*, ошибающуюся очень редко.

Важным звеном для верности работы *главной функции* является *функция ошибки*. В самом простом случае данная функция - это количество примеров, содержащих ошибку.

Часто нейронную сеть приходится дообучать. Процесс дообучения повторяется до тех пор, пока нейронная сеть используется человеком.

Хотя нейронная сеть - это также всего лишь математическая модель, потенциал этого направления очень велик. В ближайшем будущем на нейросети будет возложена большая часть рутинной работы многих областей. В том числе они смогут помочь и в решении поставленной проблемы.

## 2. Аналитическая часть

### 2.1 Выбор языка

Нейронные сети можно программировать на любом языке. Всё же для их создания большинство использует либо *Python*, либо *C++* за счёт их удобности во множестве областей. На Python есть возможность сделать любую нейронную сеть за полчаса, но надо чётко понимать, что это не написание нейронной сети с нуля, а использование уже готовой библиотеки. Если же мы хотим создать её с чистого листа, для этого больше подходит *C++*, так как он обладает наиболее высокой скоростью (имеются в виду широкие возможности для оптимизации), когда *Python* никогда этим не выделялся.

### 2.2Основы

Нейросети бывают нескольких видов:

• многослойные или персептроны – обрабатывают числовые данные

• рукуррентные – собирают и обрабатывают информацию, которая со временем меняется

• свёрточные – работают с изображениями

• генеративные – создают тексты и изображения

В данном случае будет использоваться *сеть прямого распространения* (Feed Forward), которая относится к многослойным нейронным сетям. Нейроны в сети прямого распространения никогда не образуют цикла.

У всех нейронных сетей есть *слои*. В нашем случае их три вида: входной, скрытый и выходной. Значение первого и последнего понятно из их названия. Скрытый слой называется так, потому что он недоступен пользователю, однако обязателен для программы. Все слои содержат нейроны, связанные с нейронами следующего слоя с помощью *синапсов*. Каждый синапс имеет *вес* (*weight* на англ.) – коэффициент, определяющий важность нейрона. По синапсам передаётся *сигнал* (см. приложение 1). Обработка сигналов происходит так: сигналы всех нейронов перемножаются на собственные веса и складываются. Например, вход на b1 = (a1 w[1; 1]) + (a2 w[2; 1]) + (a3 w[3; 1]) + bias, где w[...; ...] – вес синапса, связывающего конкретный нейрон одного слоя с конкретным нейроном другого, а *bias* – нейрон смещения, который нужен для того, чтобы иметь возможность получать выходной результат путём сдвига графика функции активации вправо или влево.

Кроме того, нейрон необходимо «возбудить». Для этого можно использовать любую функцию, имеющую значение, к которому она стремится, но никогда не достигнет. Такие функции называются *функциями активации*. Хорошо подходит *сигмоида* – гладкая монотонная возрастающая нелинейная функция, имеющая форму буквы «S»: f (x) = (см. приложение 2), где x, соответствует входу на b1, f(x) - выходное значение нейрона, а E (f) = (0; 1). Сигмоида хороша ещё и тем, что имеет очень простую производную: f ' (x) = f (x) \* (1 – f (x)).

Теперь для примера представим игровую ситуацию (см. приложение 3). Мышка находится в лабиринте перед развилкой в состоянии S1. Если пройти налево в состояние S2, то появится выбор: пойти к мышеловке или к сыру. Если же пройти направо в состояние S3, придётся выбирать между водой и мукой. Каждый из этих предметов обладает определённой полезностью для мышки. Так, мышеловка полностью бесполезна (вес 0), сыр является наиболее привлекательным лакомством (вес 4), а вода и мука равнозначны между собой и оба могут принести некую пользу (вес 2). При осуществлении обучения, функция будет «говорить», куда нужно пойти, чтобы получить наибольшую выгоду, поэтому мышка перейдёт в состояние S2, а после дойдёт до сыра.

### 2.3 Идея: ввод и вывод

Пришло время добавить конкретики и решить, как именно должна будет выглядеть программа. На вход будет даваться степень заинтересованности человека, для начала, в двенадцати различных пунктах: интроверсия (здесь скорее не заинтересованность, а отношение); биология; география; иностранные языки; информатика; история; литература; математика; обществознание; физика; физическая подготовка; химия.

Пользователь будет сам оценивать пункты: от 0 до 2 в отношении интроверсии, где 0 - не относится, 1 - затрудняюсь ответить, 2 – относится; от 0 до 3 в отношении всех остальных направлений, где 0 - абсолютно не заинтересован, 1 - слабо заинтересован, 2 - достаточно заинтересован, 3 - представляет наибольший интерес. В саму нейронную сеть уже будет заложена база примеров. Нейросеть проанализирует полученные данные, сравнит с уже имеющимися, после чего отправит на выход наиболее подходящие варианты. После успеха во всём вышеперечисленном, теоретически её можно усовершенствовать, добавив больше дисциплин на вход, занеся в архив больше направлений, дав пользователю возможность рассказывать о себе не числами, определяющими уровень его знаний, а текстом, состоящим из слов, который нейронная сеть должна будет распознавать.

### 2.4 План кода

Сначала подключаются все необходимые *библиотеки*, затем пишется *функция* *ввода*, *сигмоиды* и её *производной*, *функция потерь* или *вычисления ошибки*, а также *главная функция* с *прямым распространением сигнала* и *обратным распространением ошибки*.

В самом начале главной функции объявляются все переменные. Далее идёт *цикл*, отсчитывающий *количество повторений* обучения. Число, обозначающее количество повторений, называют количеством *эпох*. В цикле будет задействован параметр, обозначающий *скорость*, который придётся подбирать самостоятельно. Если скорость слишком низкая – качество работы сети будет очень высокое, но перед её использованием придётся слишком долго ждать. К тому же, есть шанс, что нейросеть переобучится, то есть запомнит все варианты наизусть, из-за чего не сможет верно определять уникальные параметры входа. Если же скорость слишком высокая – качество её работы в итоге будет на низшем уровне. Поэтому параметр скорости очень важен.

При написании нейросети понадобятся уже существующие формулы. Ниже представлена их запись в векторной форме.

Полученный сигнал от первого слоя:

Формула 1

rec1 = traindata \* w1 + bias1,

где *traindata* – входной сигнал, *w* – вес соответствующего синапса, *bias* – смещение.

Обработанный сигнал от первого слоя:

Формула 2

proc1 = sigmoid (rec1),

где *sigmoid ()* – выбранная функция активации, то есть сигмоида.

Полученный сигнал от второго слоя:

Формула 3

rec1 = proc1 \* w2 + bias2.

Для получения второго массива обработанных сигналов будет использоваться функция *софтмакс* (*softmax*), что превращает произвольный массив в массив вероятностей.

Формула 4

proc2i = softmax (rec2) = exp (rec2i) / (rec2j),

где *rec* – полученный сигнал.

Как функция потерь будет использоваться *кросс-энтропия* (*cross-entropy*). Вычисление ошибки с её помощью:

Формула 5

error = cross\_entropy (proc2, resultdata) =

-i \* log(proc2i),

где *resultdata* – верные ответы.

Теперь всё, что нужно для перехода к практической части, собрано воедино.

## 3. Практическая часть

### 3.1 Библиотеки и пространство имён

В языке программирования *C++* *библиотека* означает коллекцию уже написанных классов и функций. Ключевое слово для подключения - *#include* (см. рисунок 1). Для данного проекта точно будут использоваться следующие библиотеки для следующих целей (однако это не значит, что ими всё обязательно ограничится):

*<iostream>* - ввод и вывод (*cin*, *cout*)

*<fstream>* - считывание файлов (*ifstream*)

*<cstdlib>* - генерация (псевдо)случайных чисел (*rand()*, *srand()*)

*<cmath>* - математические функции

*<ctime>* - время (*time()*)

*Пространство имён* в *C++* - это группа связанных между собой различных компонентов программы. Ключевая фраза для подключения: *using namespace*. Данный механизм позволяет группировать идентификаторы в отдельное место, что упрощает организацию кода. Воспользуемся пространством имён std (см. рисунок 1), так как оно содержит в себе большую часть стандартных функций *C++*. В противном случае придётся печатать *std::* перед каждой относящейся к этому пространству имён функцией.

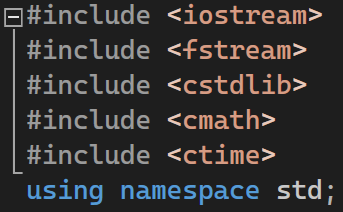


Рисунок 1

### 3.2 Реализация ввода базы данных

Для ввода используется *void* функция, то есть функцией, не возвращающей значение. Переменной строкового типа данных присваивается название файла, содержащего базу данных. Содержание данного файла будет описано в пункте 3.8, сперва напишем корректно работающую программу (см. рисунок 2).

С помощью класса *ifstream* создаётся переменная, отсылающая к ранее указанному файлу.

Если файл открыть удаётся, то с помощью цикла *for* и счётчика с 0 по общее количество примеров сначала вводится номер профессии, который заносится в одномерный массив ответов для тренировочных примеров, а далее последовательно вводятся ответы на каждый из вопросов, в свою очередь заносящиеся в двумерный массив данных для тренировки. Существует мнение, что нейронную сеть лучше обучать в разнобой, поэтому с помощью ещё одного цикла *for* и рандома осуществляется случайная перестановка элементов массивов. Конкретнее, происходит это следующим образом: в цикле со счётчиком с нуля по общее количество примеров создаётся переменная, значение которой равно случайному числу, полученному функцией *rand()*,делённому с остатком на общее количество примеров. Деление связано с тем, что функция *rand()* сама по себевозвращает число от 0 до 3276, а в данном случае нас интересует диапазон от 0 до размерности массивов. Для замены элементов с индексами счётчиков цикла на элементы с индексами “переменной рандома” используется функция *swap()*, принимающая два соответствующих значения.

Если файл открыть не удаётся, в консоль выводится оповещающее об этом сообщение.

В завершение функции ввода файл закрывается функцией *close()*.

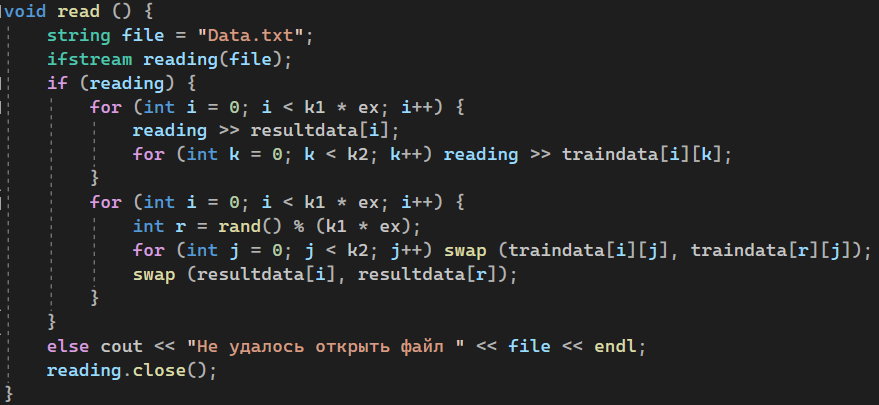


Рисунок 2

Выводиться эта функция будет в главной функции.

Объявление главной функции:

*int main(*\*параметры функции\**){*

\*тело функции\*

*}*

Использование функции ввода:

*read();*

### 3.3 Русский язык в консоли

Если ввести в консоль текст на русском языке (см. рисунок 3), то по умолчанию вместо осмысленного текста взору предстанет бессмысленный набор букв (см. рисунок 4).



Рисунок 3

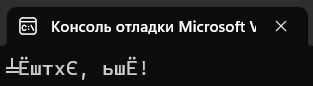


Рисунок 4

Для предотвращения проблем с выводом используют функцию *setlocale()*, принимающую два параметра: категорию, на которую влияет языковой стандарт и указатель языкового стандарта. Категория *LC\_ALL* включает в себя всё, что есть во всех остальных категориях - она станет первым параметром. Указатель языкового стандарта для русского языка - *“RUS”* - второй параметр. Теперь русский текст в консоль выводится исправно (см. рисунок 5).



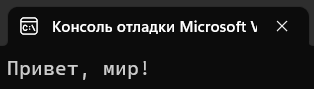


Рисунок 5

### 3.4 Рандом и время

Изначально *rand()* генерирует только псевдослучайные числа. Псевдослучайные числа не являются по-настоящему случайными, так как между ними остаются связывающие их закономерности. Чтобы это исправить, можно связать функции *srand()* и *time()*. Получится следующая конструкция: *srand (time (0));* . Первая функция задаёт начальное значение для генератора псевдослучайных чисел, используемого функцией *rand()*. Вторая функция принимает указатель на место хранения для времени, который можно заменить на 0, если хранить время не нужно, и возвращает время в секундах, прошедших с полуночи 1 января 1970 года. Таким образом каждую секунду мы получаем новое значение.

### 3.5 Объявление переменных

Переменные слоёв, имеющие названия *datasize*, *insize*, *hidesize* и *outsize*, которые означают размеры обучающей базы данных, входного, скрытого и выходного слоёв, объявляются как константные выражения типа данных *unsigned short int* (сокращённый целочисленный без отрицательных значений), поскольку не планируется сразу использовать такое большое количество примеров, которое не поместится в пределы данного типа. Равны эти переменные соответственно количеству профессий, умноженному на количество примеров для каждой профессии, количеству вопросов с прибавкой единицы, количеству профессий с прибавкой единицы и выражению 2 \* *insize* / 3 + 7. Добавление единицы необходимо, потому что индексация в *C++* начинается с нуля, а последнее выражение было определено путём подбора, при нём возможен наилучший результат при конкретных примерах, которые будут описаны в пункте 3.8.

Веса, биас, полученные сигналы, обработанные сигналы и все производные представлены динамическими массивами и типом данных *double* (расширенный с плавающей запятой), так как нам важна высокая точность.

Скорость типа float (вещественный) и количество эпох типа *unsigned short int* определяются так же, как и размер скрытого слоя - наугад.

Всё объявление происходит стандартно в соответствии с синтаксисом *C++* (см. рисунок 6).

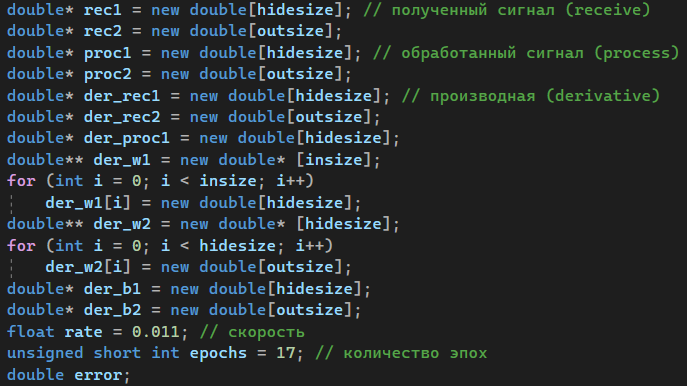
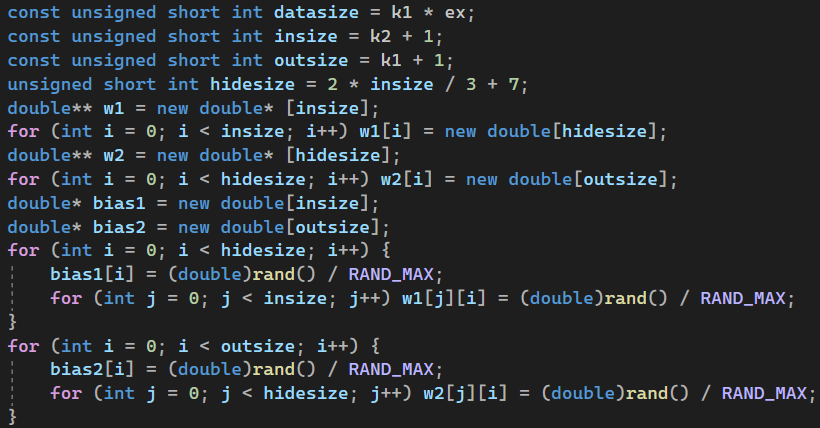


Рисунок 6

### 3.6 Прямое распространение сигнала

Переход к самому главному. Создаётся вешний цикл *for* по эпохам. В него включается цикл *for* по всем обучающим примерам (*datasize*), в теле которого и будут выполняться последующие действия.

Прямой проход осуществляется от входного сигнала до выходного.

Полученный сигнал считается в цикле по скрытому слою: элемент массива по индексу приравнивается нулю, к этому элементу прибавляется произведение входных сигналов на матрицу весов и в заключение прибавляется соответственное смещение (формула 1, рисунок 7). В массив обработанных сигналов заносится значение, возвращаемое функцией *sigmoid* (формула 2) с входным параметром полученного нового значения сигнала. Кстати, функции объявляются всегда передфункцией, в которой они используются (в данном случае перед функцией *main()*).

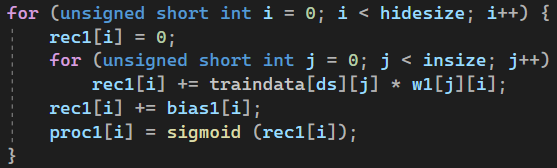


Рисунок 7

Функция *sigmoid* содержит только ранее упомянутое возвращение значения результата расчётов по формуле сигмоиды (формула прописана в пункте 2.2) с помощью оператора *return* (рисунок 8).

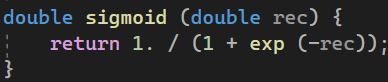


Рисунок 8

Аналогично тело следующего цикла по выходному слою (формула 3), но вместо обработки сигнала функцией активации к предварительно созданной переменной *sum* типа *double,* определённой значением 0, на каждой итерации прибавляется экспонента полученного сигнала (формула 4, рисунок 9).

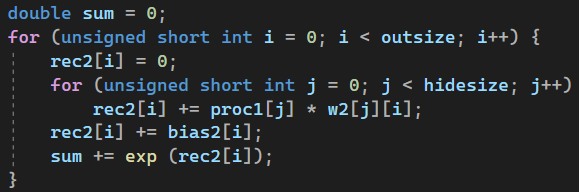


Рисунок 9

В новом цикле по выходному слою вычисляется обработанный сигнал для второго массива полученных сигналов (формула 4, рисунок 10).



Рисунок 10

Данных достаточно, чтобы высчитать ошибку. Сначала пишется функция (формула 5, рисунок 11), а после этого осуществляется её вызов (формула 5, рисунок 12).

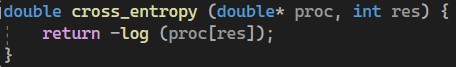


Рисунок 11



Рисунок 12

### 3.7 Обратное распространение ошибки

При обратном проходе вычисляются все производные с помощью уже существующих для их вычисления формул (рисунок 14). В одной из них фигурирует и производная сигмоиды, которую для удобства восприятия кода записывается, как отдельная функция (рисунок 13).



Рисунок 13

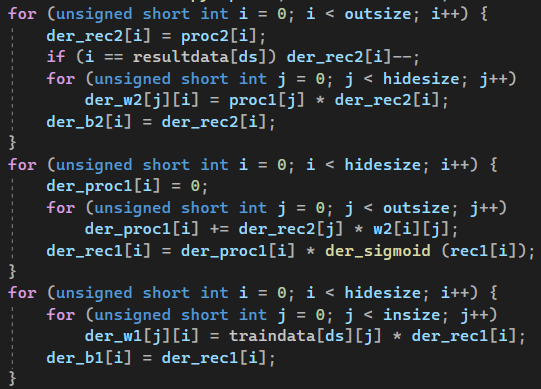


Рисунок 14

Производные же нужны для того, чтобы корректировать веса и смещения, путём вычитания из последних произведения скорости, ошибки и этих самых производных (рисунок 15).

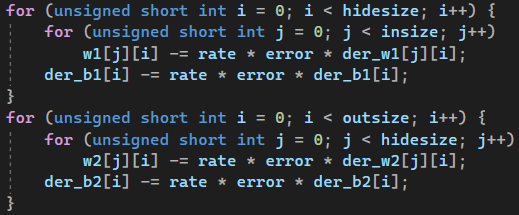


Рисунок 15

На этом моменте все три внешних цикла закрываются.

### 3.8 Примеры

Предоставим нейросети выбор для начала из 10 профессий и 20 примеров к каждой из них при условии, что в каждом примере содержится ответ на 10 вопросов. Вопросы и форма ответов на них указаны в пункте 2.3. Чем больше профессий и вопросов или, другими словами, размеры выходного и входного слоя, тем труднее нейросети определить верный ответ, а чем больше примеров к профессиям или же размер базы данных, тем нейросети, наоборот, легче не ошибиться. Естественно, количество профессий, как и вопросов можно увеличить во много раз, но надо учитывать, что и количество примеров придётся увеличить, чтобы количество правильных ответов удовлетворяло ожиданиям.

Профессии при их ограниченном количестве специально выбраны так, чтобы направления их деятельности были различны, и при этом сами профессии были популярны.

*Список профессий* с соответствующими им индексами:

1. Артист
2. Дизайнер
3. Журналист
4. Инженер
5. Врач
6. Переводчик
7. Полицейский
8. Преподаватель
9. Строитель
10. IT-специалист

С сильным *обобщением* (какие-то предметы важны больше, какие-то меньше, что, конечно, учитывается, но здесь указаны конкретно те предметы, по которым можно сдать ЕГЭ и поступить по специальности) ориентация выглядит следующим образом:

* Дизайнер: иностранные языки, информатика, история, литература, математика, обществознание, физика;
* Журналист: не интроверт, география, иностранные языки, информатика, история, литература, обществознание;
* Инженер: география, иностранные языки, информатика, история, математика, обществознание, физика, химия;
* Врач: биология, иностранные языки, математика, физика, химия;
* Переводчик: не интроверт, иностранные языки, информатика, история, литература, обществознание;
* Полицейский: не интроверт, математика, обществознание, физическая подготовка;
* IT-специалист: информатика, математика, физика.

Исключения составляют такие профессии, как артист и преподаватель: артисту требуется только талант, поэтому оценки его предметов на текущих вопросах малы (0-1) и расставлены случайно, лишь физическая подготовка иногда получает 2 балла; преподаватель зачастую заточен на один предмет (или на большинство сразу, если это, например, учитель начальной школы), оценки указаны соответственно.

Текстовый документ с примерами должен находиться в папке проекта.

Пример заполнения файла (рисунок 16):

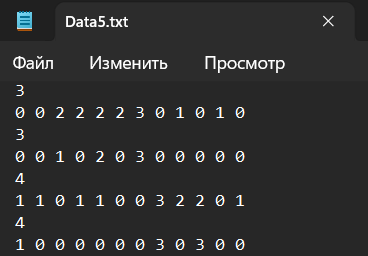


Рисунок 16

После подключения пространства имён *std* объявляются глобальные константные переменные количества профессий, примеров и вопросов с присвоением значений и массивы (рисунок 17).



Рисунок 17

### 3.9 Проверка корректности работы нейросети

Теперь, чтобы понять насколько нейросеть эффективна, можно написать проверку на количество угаданных ответов. Для этих целей используем примеры, на которых она обучалась. Конечно, правильнее было бы использовать новые примеры, но поскольку в конкретном случае примеры не однозначны, ничего страшного в этом нет.

Итак, целочисленная переменная correct инициализируется нулём и будет использоваться, как счётчик попаданий нейросети в верный ответ. В цикле по каждому элементу базы осуществляется проход сначала по первому слою, потом по второму, аналогично предыдущему коду (рисунок 18).

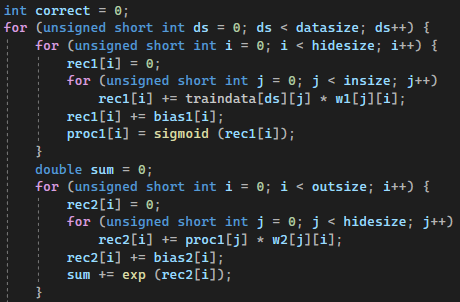


Рисунок 18

Далее, параллельно использованию софтмакса ищется индекс максимального обработанного сигнала. Если индексы совпадают, показатель correct увеличивается на единицу (рисунок 19).

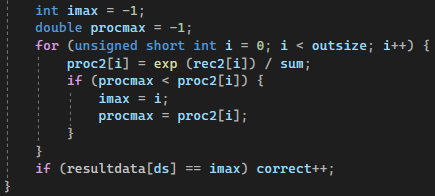


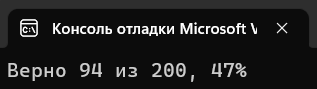
Рисунок 19

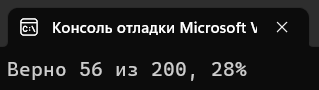
Для вывода в консоль используется стандартный выходной поток *cout* и операция поразрядного сдвига влево, означающая помещение в выходной поток, а для перехода на новую строку вызывается функция *endl,* которая, кстати, визуально даёт тот же результат, что и символ конца строки *‘\n’*, но на самом деле помимо этого ещё и очищает буфер, однако в данном случае не принципиально, что конкретно использовать (рисунок 20).



Рисунок 20

Запуская программу трижды, можно заметить, что процент правильных ответов всегда отличается (рисунок 21). К слову то, как можно поднять процент верных ответов, описано в пункте 3.8.





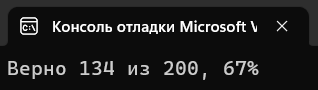


Рисунок 21

Чтобы узнать усреднённое значение, всё, что написано в главной функции, начиная с переменных, заключается в цикл на 100 повторений, подсчитывается среднее арифметическое с помощью специальных переменных, а предыдущая строка с выводом заносится в комментарий с помощью *//* (рисунок 22, рисунок 23) . Символы комментария делают строку просто текстом, никак не влияющим на программу. Кроме того, существует также многострочный комментарий, который также можно использовать: */\** \*тело\* *\*/* .



Рисунок 22

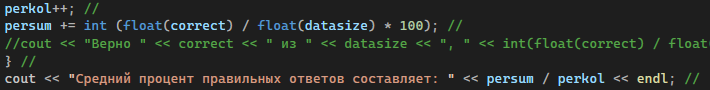


Рисунок 23

Итак, нейросеть, подкреплённая текущим набором примеров, выдала самый что ни на есть средний результат (рисунок 24).

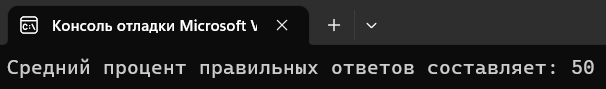


Рисунок 24

Чтобы двигаться дальше, закомментируем всё, что было в пункте 3.1.9.

### 3.10 Интерфейс

Теперь оформим программу, дав пользователю возможность вводить информацию конкретно о себе.

Сначала устанавливается метка ( \*название метки\**:* ) для возвращения в данное место кода при некорректном вводе данных. Далее создаётся массив, как хранилище ответов на выбранные вопросы. После этого уже следует сам вывод и соответствующее предоставление возможностей ввода. Пользователю будет предоставлен выбор между вводом в формате теста и сплошным вводом без пояснительных приписок о том, какое знание следует оценить, однако этот выбор предоставляется не столько для пользователя, сколько для проверки работоспособности программы на известных примерах, поэтому обычному пользователю рекомендован первый вариант. Когда все ответы введены, юзера либо отсылает к установленной ранее метке с помощью оператора *goto*, если эти данные некорректны, либо после ещё одного повторения обучения, только на новых данных, выводится результат.

Данный отрывок кода представлен на рисунке 25 (на изображении некоторые предложения обрезаны из-за большой длины). По завершении главной функции пишется оператор *return* *0*, однако это писать необязательно, потому что *main()* возвращает это значение по-умолчанию.

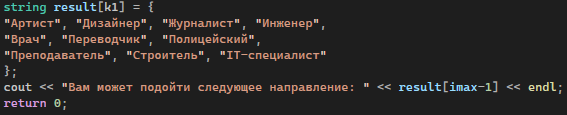
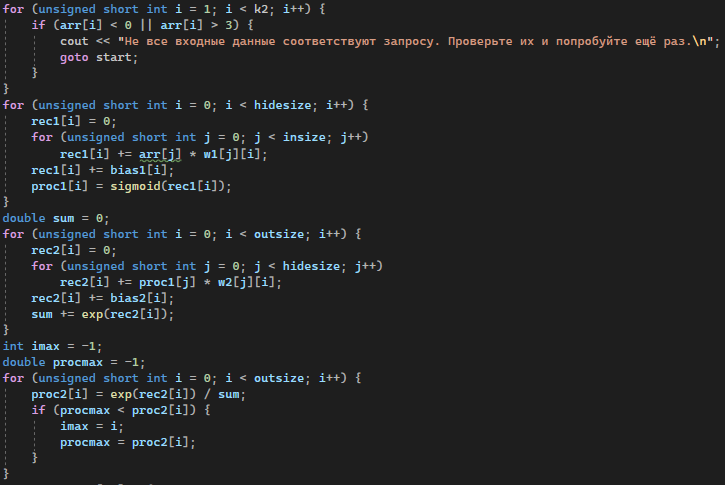
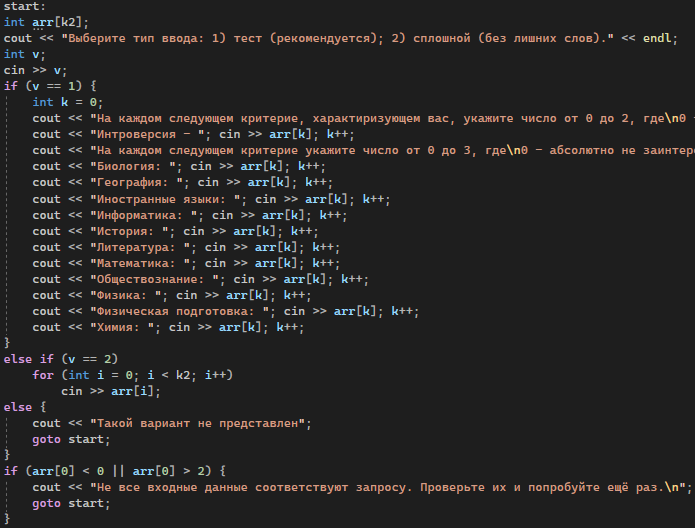


Рисунок 25

### 3.11 Тестирование

Автор проекта уже определился - он идёт по пути IT-специалиста, а также увлекается моделированием, что можно отнести к дизайну. Посмотрим, сможет ли нейросеть угадать его увлечение (см. приложение 4).

Нейронная сеть смогла верно определить специальность. Одно выполнение программы на текущей базе данных заняло 2,133 секунды - неплохой результат. Чтобы проверить, была ли это счастливая случайность, или же нейросеть была уверенна в своём ответе, заключим программу в цикл на 10 повторений и сопоставим результаты (см. приложение 5).

Из консоли можно наблюдать следующее: преподаватель наиболее частый ответ, IT-специалист и дизайнер идут следом в сумме составляя такой же процент, а строитель и журналист оказались единичными случаями. На самом деле, всё очень логично: преподаватель довольно обобщённая профессия, которая может относиться к любому предмету, а на тесте как раз было очень много неоднозначных ответов; IT-специалист и дизайнер действительно угаданы верно; журналист включён в список из-за проявленного интереса к иностранным языкам и наличии, хотя и слабой, заинтересованности в обществознании; строителю нужны точные науки, такие как математика, что удовлетворяет оценке математики как предмета на три балла из трёх.

3.12 Дополненный вывод

Для того, чтобы дать более развёрнутый ответ, используем идею из пункта 3.11. Увеличим количество итераций цикла до 50, занесём кусок кода с метки до последнего оператора *goto* в *void*-функцию и вызовем её перед циклом после чтения файла (рисунок 26). Также понадобится массив величины количества профессий, заполненный нулями (рисунок 27).



Рисунок 26

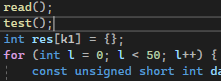


Рисунок 27

Вместо моментальной выдачи результата, теперь этот результат будет выглядеть, как плюс один балл к соответствующей ячейке массива (рисунок 30).



Рисунок 28

Массив со строковыми названиями специальностей вынесем за циклы. Чтобы вывод был упорядочен, используем контейнер map, для которого потребуется подключить ещё одну библиотеку - *#include <map>.* Этот контейнер содержит в себе пары элементов, упорядочивает данные по возрастанию по первому элементу пары. Данные из массива вводятся в *map* с условием - элемент массива не равен нулю. Пара будет содержать сам элемент и его индекс. Пройдясь итератором с его конца и до начала с помощью *rbegin()* и *rend()*, можно вывести профессию с процентным отношением её выбора ко всем выборам. Таким образом, вывод будет представлять собой не одно, а сразу несколько предложений. Весь остальной код, вне главного цикла, представлен на рисунке 29.

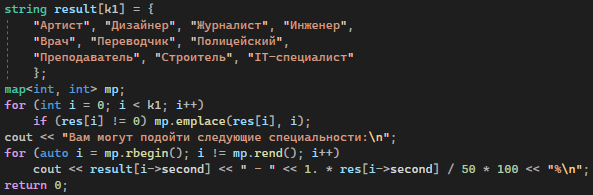


Рисунок 29

Попробуем ещё раз протестировать программу, используя ту же информацию, что и в прошлом тесте (см. приложение 6).

Всё прошло успешно! Выполнение программы заняло 3,867 секунды.

## 3.13 Теперь на GitHub

Так как нейросеть ещё нуждается в доработках, настоящая её версия будет выложена только на *GitHub* - крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Другими словами - это социальная сеть для разработчиков, где кроме размещения кода, который могут опробовать другие люди, участники могут общаться, комментировать правки друг друга, а также следить за новостями.

Чтобы имя проекта было уникально, написанная нейронная сеть названа “Нейровыбор”.

Проект доступен для бесплатного использования по ссылке:

<https://github.com/N1cklier/NeuroChoice>

# Заключение

Актуальность моего проекта заключается в том, что на данный момент многие люди испытывают проблемы при выборе своей будущей профессии. В работе были изучены увлечения и предпочтения людей как объект исследования, а как предмет - самообучающийся искусственный интеллект. Таким образом, целью было создать программу, способную решить поставленную проблему. Эта цель была достигнута - готовая нейронная сеть исправно запускается и выдаёт ожидаемые результаты. В задачи входил сбор основной информации о нейросетях и выделение главных аспектов из полного объёма данных, а также создание самого искусственного интеллекта с его публикацией в свободном доступе. Каждая из задач была выполнена.

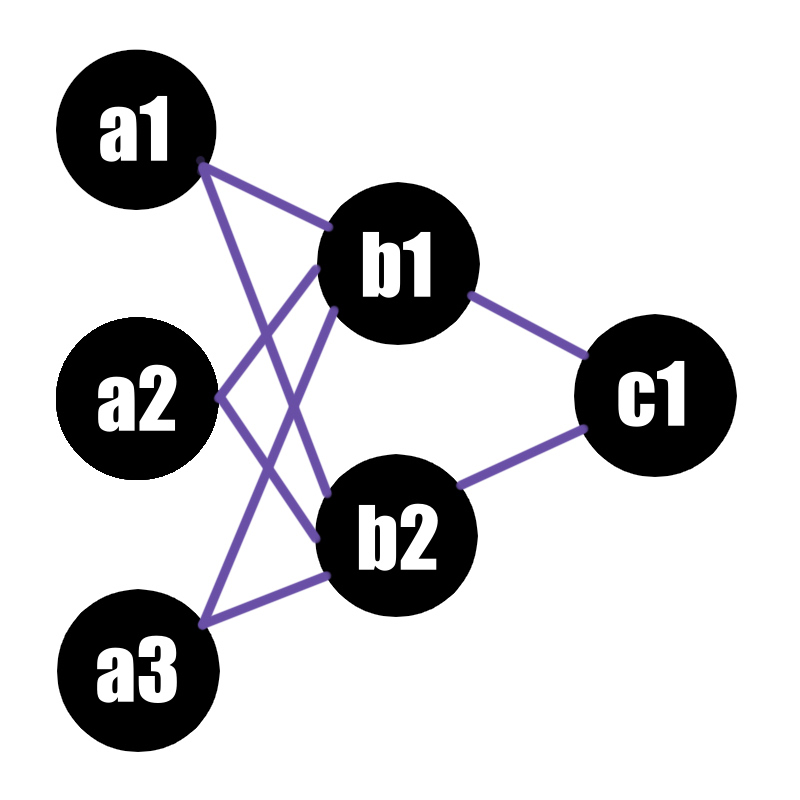
В теоретической части была указана информация, необходимая для того, чтобы лучше понимать с чем придётся работать. В это вошли как данные о заботах молодых людей в сфере соответствующей темы проекта, так и перспективы выбранного пути решения проблемы. Далее в теоретической части мной были расписаны основные события в направлении нейронных сетей и самые базовые понятия, нужные для понимания того, с чем придётся работать. Аналитическая часть включила в себя ряд решений о том, что стоит использовать в настоящей задаче, и конкретику о решении задачи, а именно был выбран язык программирования, и выбрано устройство кода, объяснён задуманный ввод и вывод и, конечно, составлен план, по которому впоследствии выполнялась практическая часть. В завершающей части объяснён и показан сам код, состоящий из двух условных частей: главной части и интерфейса. Главная часть строго подчинена математике - она является ядром работы нейросети. Вторая часть, интерфейс, более творческая и скорее включает в себя моё видение - то самое описание ввода и вывода, прописанное в аналитической части, которое по этой причине неоднократно изменялось в ходе работы. В итоге, написанный код, который уже стал являться продуктом проекта, был выложен в Интернет.

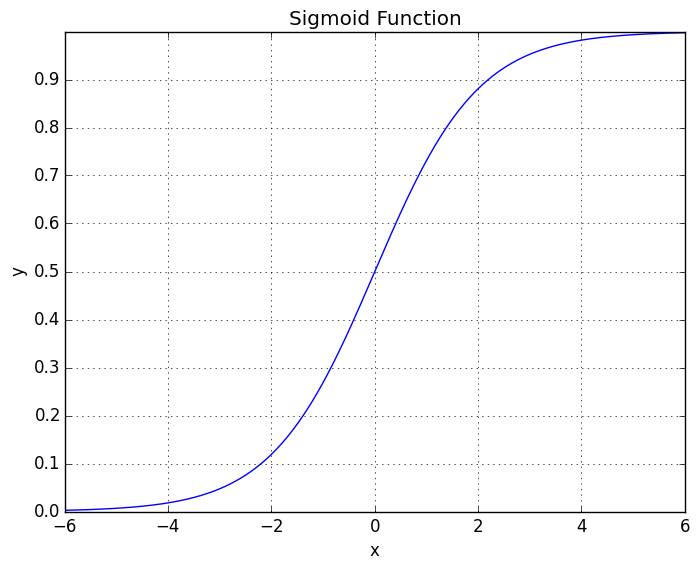
Практическая значимость моего исследования состоит в том, что каждый желающий теперь может попробовать пройти тест и узнать, что о нём думает нейронная сеть “Нейровыбор”. Это может помочь в выборе профессии и при этом почти не отнимает время. Кроме того, так как код открыт для всех, любой может лично поразбираться в программе и, при умении, попробовать разработать свою собственную нейросеть. В будущем планируется создать уже генеративную нейронную сеть, которая будет считывать и генерировать целостный текст, а не только числа. В случае успеха, подобный искусственный интеллект уже будет теоретически способен стать бесплатным психологом. А из всего вышеперечисленного можно сделать простой, напрашивающийся сам собой вывод, который заключается в том, что за нейросетями будущее, а зная на что уже способны многие популярные нейросети, можно смело утверждать, что мы уже живём в этом абстрактном понятии, мы живём в будущем.

# Использованные материалы

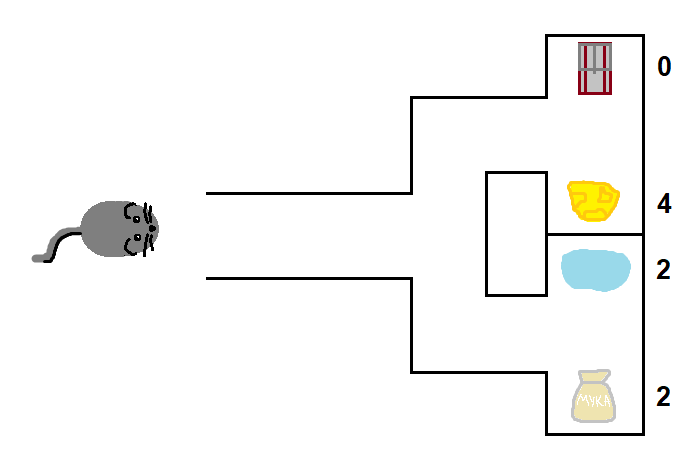
1. Катерина Швецова – Нейросети: ловитесь данные большие и малые, 2021
2. Студенческий научный форум, URL: <https://scienceforum.ru/2022/article/2018031957>
3. Pikabu, URL: <https://pikabu.ru/story/istoriya_vozniknoveniya_neyronnyikh_setey_7152766>
4. Дзен, URL: [https://dzen.ru/a/YhNKY8jF8RgrKCH8#:~:text=Математик%20Уолтер%20Питтс%20и%20нейрофизиолог,книгу%20The%20Organization%20of%20Behaviour](https://dzen.ru/a/YhNKY8jF8RgrKCH8%23:~:text=Математик%20Уолтер%20Питтс%20и%20нейрофизиолог,книгу%20The%20Organization%20of%20Behaviour)
5. Vc.ru, URL: <https://vc.ru/future/606777-ochen-kratkaya-istoriya-neyrosetey-ot-razrabotok-20-go-veka-do-chatgpt>
6. Блог Яндекс Практикума, URL: <https://practicum.yandex.ru/blog/chto-takoe-neyronnye-seti/>
7. Википедия, URL: <https://ru.wikipedia.org/wiki/Нейронная_сеть>
8. Хабр, URL: <https://habr.com/ru/company/vk/blog/397081/>
9. Хабр, URL: <https://habr.com/ru/company/yandex/blog/307260/>
10. Хабр, URL: <https://habr.com/ru/post/514372/>
11. Хабр, URL: <https://habr.com/ru/post/514626/>
12. David Kriesel - A Brief Introduction to Neural Networks, URL: <http://www.dkriesel.com/en/science/neural_networks>, 2018
13. Vuzopedia, URL: <https://vuzopedia.ru>
14. ID-Lab, URL: <https://id-lab.ru/posts/developers/funkcii/>
15. Helenkapatsa, URL: <https://www.helenkapatsa.ru/funktsiia-potieri/>
16. GitHub, URL: <https://github.com/dkorobchenko/nn-python/blob/main/train.py>

# Приложения

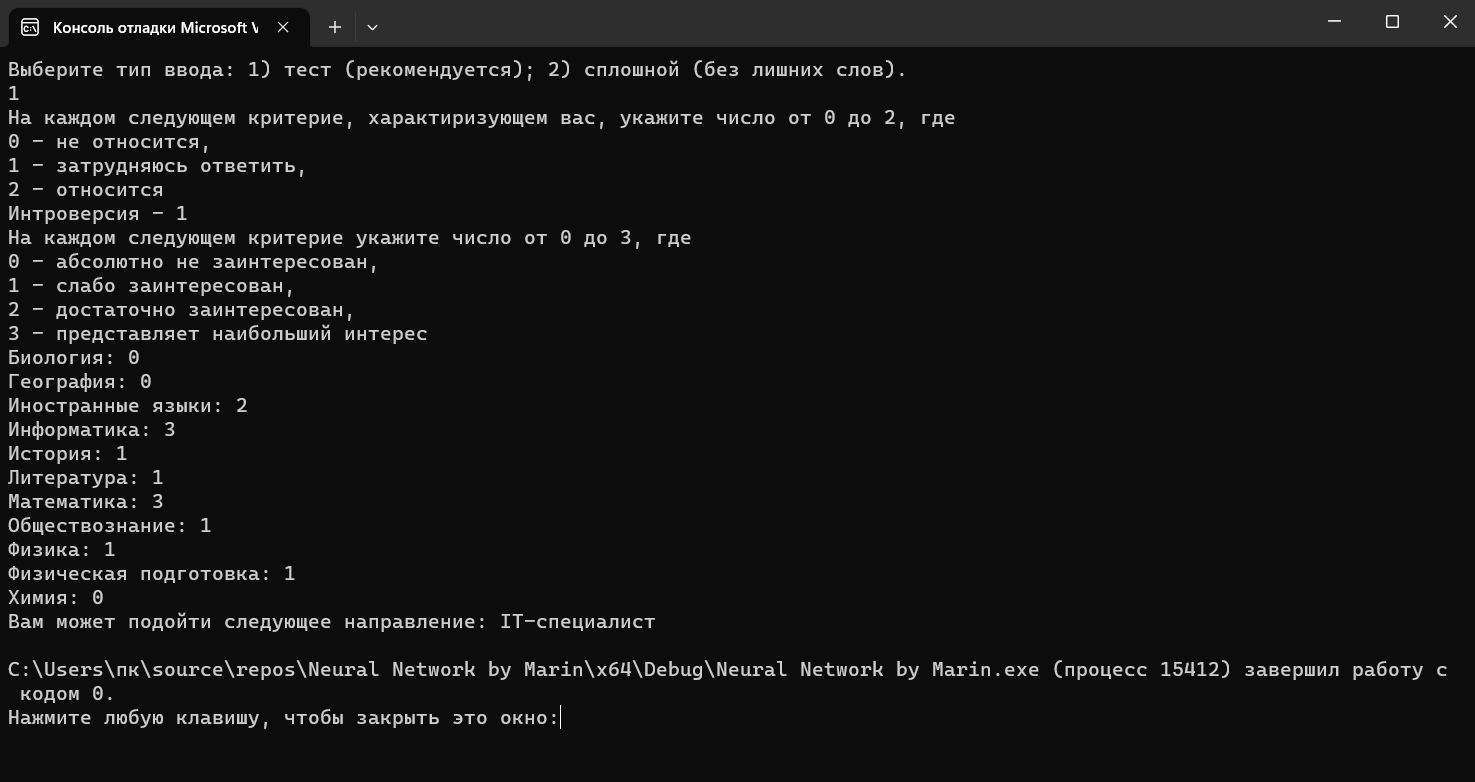
1. *Слои нейронов*
2. *Сигмоида*

**

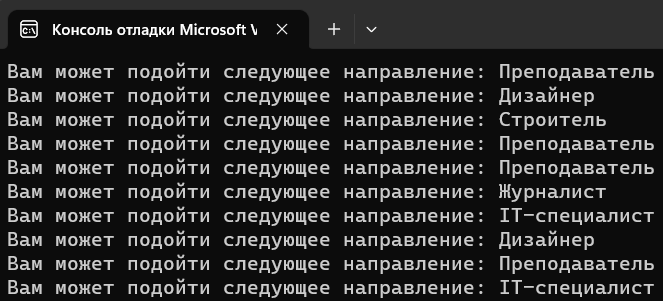
1. *Веса*

**

1. *Тест*



1. *Проверка*



1. *Вывод с процентами*

