**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ России**

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования**

**«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Инженерно-технологическая Академия**

**Институт компьютерных технологий и информационной безопасности**

**Кафедра Систем Автоматизированного Проектирования**

**Отчет по лабораторной работе № 4**

**на тему: «Изучение операторов селекции и отбора»**

по дисциплине «Методы машинного обучения»

Выполнили:

студенты группы КТбо4-4

Батагов С.С.\_\_\_\_\_\_\_\_\_\_\_\_\_

Москаленко М.В.\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

доцент кафедры САПР

Лебедев О.Б. \_\_\_\_\_\_\_\_\_\_\_\_\_

Таганрог 2022

Содержание

[ВВЕДЕНИЕ 2](#_Toc122197185)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 3](#_Toc122197186)

[1.1 Типы хромосом 3](#_Toc122197187)

[1.2 Стратегии создания начальной популяции 5](#_Toc122197188)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 6](#_Toc122197189)

[2.1 Главная функция 6](#_Toc122197190)

[2.2 Функция создания начальной популяции 6](#_Toc122197191)

[2.3 Функция развития популяции 7](#_Toc122197192)

[2.4 Функция для синтеза видов хромосом 8](#_Toc122197193)

[ЗАКЛЮЧЕНИЕ 9](#_Toc122197194)

[Приложение А 10](#_Toc122197195)

ВВЕДЕНИЕ

***Цель работы:***

Ознакомиться с основными положениями селекции и отбора в ГА. Научиться производить операции с функцией селекции и отбора. Также разработать программу с функциями селекции и отбора.

***Задания к лабораторной работе***

Написать программу, реализующую различные виды селекции. Продемонстрировать и объяснить работу программы на примерах.

Написать программу, реализующую различные виды отбора. Продемонстрировать и объяснить работу программы на примерах.

На основе полученных знаний предложить новые модификации операторов селекции и отбора.

Написать программу, реализующую разработанные схемы. Продемонстрировать работу программы на примерах.

# ТЕ­ОРЕТИЧЕСКАЯ ЧАСТЬ

## Селекция

Применение селективных методов означает методический отбор генетического материала в соответствии с принятым критерием, что должно повысить скорость получения конечного результата и его качество. Отметим, что ускорение мутационного процесса ведет к получению все более разнообразного генетического материала.

Селекция представляет собой форму искусственного отбора, где в отличие от естественного отбора эволюция направляется факторами внешней среды. Селекция, как наука, создана Ч.Дарвином, который выделял три формы отбора:

естественный отбор, вызывающий изменения, связанные с приспособлением к новым условиям;

бессознательный отбор, при котором в процессе эволюции сохраняют лучшие экземпляры;

методический отбор, при котором проводится целенаправленное изменение популяций в сторону установленного идеала.

В учении об отборе Ч. Дарвин показал, что основой является отбор наилучших форм, требующий таких условий, как:

полный выбор исходного материала;

правильная постановка цели;

проведение селекции в достаточно широких масштабах и возможно более жесткий отбор материала;

отбор по одному основному признаку.

*Естественный отбор* – процесс, направленный к повышению (или понижению) вероятности оставления потомства одной формой организмов по сравнению с другими. Отбор, прежде всего, действует в пределах каждой популяции, отбирая те или иные входящие в ее состав генотипы.

При относительной стабильности внешних условий преобладающие генотипы все время будут сохранять свое доминирующее положение. Напротив, все уклонения от этой группы будут уничтожаться. Такая форма отбора названа центростремительным или стабилизирующим отбором. Однако при изменении условий существования может произойти отбор, ведущий к замене одних количественно преобладающих генотипов другими. Эта форма отбора названа движущей, или ведущей.

*Естественный отбор* – единственный направляющий эволюцию элементарный фактор. Его действие всегда направляется складывающимися условиями существования.

Рассмотренные выше факторы по степени влияния на эволюцию можно упорядочить следующим образом:

естественный отбор;

изоляция;

колебание численности популяции;

мутационные процессы.

В современной селекции широко применяют массовый и индивидуальный отбор. Для создания генетически устойчивых видов, обладающих желаемыми признаками, необходимо знать генетическую структуру исходных популяций. Массовый отбор, производимый по фенотипу, является менее эффективным, чем индивидуальный.

При индивидуальном отборе популяцию делят на отдельные части, причем для размножения выбирают носителей желаемых свойств. Иногда отбор ведут по боковым родственникам.

*Инбридинг* — близкородственное скрещивание — позволяет быстро выделить части, несущие желаемые гены. Здесь применяется жесткий отбор экземпляров, а также введение в части популяции нового генетического материала.

Генетический материал обладает такими универсальными свойствами, как дискретность, непрерывность, линейность и относительная стабильность, выявляемыми в ходе генетического анализа.

*Дискретность* — существование гена, хромосомы, генома - проявляется в виде множества аллелей, соответствующих гену, множества генов, соответствующих группе сцепления (хромосоме), множества групп сцепления, соответствующих геному. Это значит, что можно выделять в исходном генетическом материале отдельные фрагменты, контролирующие те или иные функции.

*Непрерывность* — физическое единство групп сцепления - проявляется в существовании множества генов, обнаруживающих сцепление между собой. Это свойство находит выражение также в различных типах эффекта положения гена. Это свойство означает, что определенные комбинации генов совместно контролируют некоторую функцию.

*Линейность* — одномерность записи генетической информации – проявляется в определенной последовательности генов в пределах группы сцепления.

*Относительная стабильность генетического материала* — его способность к изменениям — проявляется в мутациях, что означает способность генов к воспроизведению и изменению с последующим воспроизведением измененных вариантов.

## Оператор отбора

*Репродукция* — процесс, в котором хромосомы копируются пропорционально значению их ЦФ. Копирование хромосом с «лучшим» значением ЦФ имеет большую вероятность для попадания в следующую генерацию. Рассматривая эволюцию Ч. Дарвина, можно отметить, что оператор репродукции (ОР) является искусственной версией натуральной селекции — «выживание сильнейших». Он представляется в алгоритмической форме различными способами. Самый простой — создать модель «колеса рулетки», в которой каждая хромосома имеет поле, пропорциональное значению ЦФ.

*Оператор редукции* — это языковая конструкция, позволяющая на основе анализа популяции после одной или нескольких поколений генетического алгоритма уменьшать ее размер до заданной величины. Рассмотрим способы реализации оператора редукции. Он выполняется для устранения неудачных решений. В некоторых генетических алгоритмах, в частности в ПГА, этот оператор применяется для сохранения постоянного размера начальной популяции. Основная проблема здесь – это нахождение компромисса между разнообразием генетического материала и качеством решений. Сначала формируют репродукционную группу из всех решений, образовавшихся в популяции . Далее выполняют отбор решений в следующую популяцию.

Численность новой популяции

Nt+1= Nt + Noк + Noм + Nou + Noт + Noтр + Noc + Noy + Noв,

где Nt+1 – численность новой популяции,

Nt – численность популяции на предыдущем шаге (поколении) t,

Noк , Noм , Noт, Nou, Noc, Noтр Noy, Noв – потомки, полученные в результате применения операторов – кроссинговера, мутации, инверсии, транслокации, транспозиции, сегрегации, удаления, вставки.

Отметим, что оператор редукции может применяться после каждого оператора или после всех в одной генерации ГА. Выделяют две основных схемы редукции (иногда их называют схемы отбора):

– Элитная схема редукции. В группу удаления из популяции включаются такие хромосомы, как и только те потомки, для которых выполняется условие:

, 

где  – потомки (решения), полученные после применения ГО.

– Последовательная схема редукции позволяет варьировать методы выбора хромосом для удаления из популяции:

* случайный выбор,
* выбор «лучших» и «худших»,
* «близкое» родство,
* «дальнее» родство,
* на основе кода Грея для бинарных хромосом,
* на основе Хэммингова расстояния,
* на основе «турнира».

Случайный выбор хромосом позволяет разнообразить генофонд на ранних этапах ГА. Вероятность этого выбора должна снижаться при эволюции поколений.

По аналогии с оператором репродукции известны следующие модификации операторов редукции:

1. равновероятностный отбор с вероятностью

,

где N – размер популяции;

2) пропорциональный отбор с вероятностью

.

Подведем итоги. С помощью операторов редукции на ранних стадиях работы ГА происходит выбор хромосом без учета значений их ЦФ (Pk), т.е. случайный отбор. На заключительной стадии – определяющий фактор при отборе значения ЦФ(Pk). Чем выше ЦФ(Pk), тем выше вероятность отбора Pk в следующую популяцию. На заключительной стадии проводится уменьшение случайных операций и увеличивается процент направленных.

## Оператор рекомбинации

*Оператор рекомбинации* — это языковая конструкция, которая определяет, как новая генерация хромосом будет построена из родителей и потомков. Другими словами, оператор рекомбинации – это технология анализа и преобразования популяции при переходе из одной генерации в другую. Существует много путей выполнения рекомбинации. Один из них состоит из перемещения родителей в потомки после реализации каждого генетического оператора (ГО). Другой путь заключается в перемещении некоторой части популяции после каждой генерации.

Часто в ГА задается параметр W(Р), который управляет этим процессом. Так, Np(1-W(Р)) элементов в популяции Р, выбранных случайно, могут «выжить» в следующей генерации. Здесь Np − размер популяции. Величина W(Р) = 0 означает, что целая предыдущая популяция перемещается в новую популяцию в каждой генерации. При дальнейшей реализации алгоритма лучшие или отобранные элементы из родителей и потомков будут выбираться для формирования новой популяции.

В инженерных задачах используются различные механизмы и модели этого процесса. Приведем несколько из них:

* М1 — вытеснение (crowding factor). Этот механизм определяет способ и порядок замены родительских хромосом из генерации t хромосомами потомками после генерации t+1. Механизм реализован таким образом, что стремится удалять «похожие» хромосомы из популяции и оставлять отличающиеся.
* М2 — разделение (sharing). Этот механизм вводит зависимость значения ЦФ хромосомы от их распределения в популяции и поисковом пространстве. Это позволяет копиям родительских хромосом или близких к ним не появляться в популяциях.
* М3 — введение идентификаторов (tagging). Специальным хромосомам присваиваются метки. Операторы ГА применяются только к помеченным хромосомам.

Отметим, что оператор редукции является частным случаем оператора рекомбинации.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

Для написания программ использовался язык JavaScript и библиотека readline-sync для ввода символов в консоль. Исходный код программы представлен в приложении А.

### Главная функция

На рисунке 1, представлен класс, отвечающий за генерацию популяции. Пользователь сам задет размер популяции и количество хромосом.

Изображение выглядит как текст

Автоматически созданное описание

### Функция видов селекции

Были реализованы 2 вида селекции. Метод шкалы (Рисунок 2) состоит в предварительной сортировке от лучшей к худшей на основе заданного критерия. Каждому элементу назначается определенное число и тогда селекция выполняется согласно этому числу.

Изображение выглядит как текст

Автоматически созданное описание

Второй функций селекции является метод «элитная селекция». В этом случае выбираются лучшие (элитные) элементы на основе сравнения значений ЦФ. Далее они вступают в различные преобразования, после которых снова выбираются элитные элементы. Процесс продолжается аналогично до тех пор, пока продолжают появляться элитные элементы. На рисунке 3, представлен алгоритм работы. В ознакомительных целях была использована простая линейная функция 5x+3.

Изображение выглядит как текст

Автоматически созданное описание

### Функции реализации отбора

Далее будут рассмотрены 2 вида отбора.

В функции случайного отбора (Рисунок 4) формируется список (массив) с индексами, перемешивается случайным образом, далее выбирается несколько первых значений.

Изображение выглядит как текст

Автоматически созданное описание

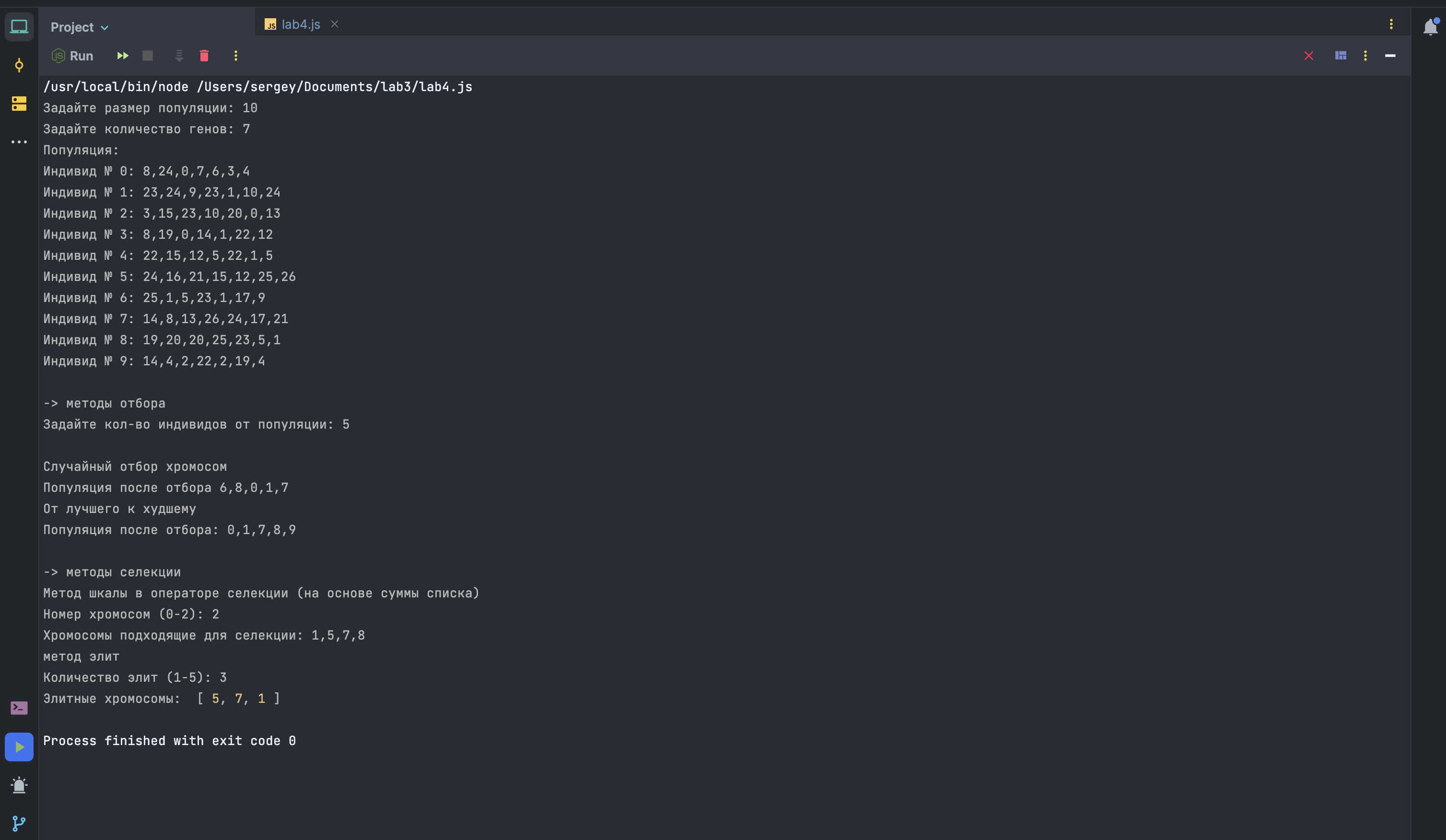
Второй метод, является «лучшее и худшее» (Рисунок 5). Алгоритм данной функции выбирает из предложенного массива худшие и лучшие элементы, то есть элементы начала списка и его конец, в зависимости от количества требуемой новой популяции.

Изображение выглядит как текст

Автоматически созданное описание

### Результат работы всех функций

На рисунке 6 представлен результат работы программы. Сначала, создается экземпляр класса и выводится на экран. Затем поочередно вызываются все функции, с передачей полученной популяции в аргументы.



ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были изучены теоретические материалы, с последующей упрощённой программной реализацией нескольких видов селекции, а именно: элитной секции и шкалы. И видов отбора: случайный отбор и «хорошо и плохо».

Приложение А

*const* readline = require('readline-sync');  
  
  
*const* sizePopulation = Number(  
 readline.question('Задайте размер популяции: ')  
)  
*const* sizeChr = Number(  
 readline.question('Задайте количество генов: ')  
)  
  
*class* Population {  
 *constructor*() {  
 *this*.population = [];  
 *for* (*let* item=0; item < sizePopulation; item++) {  
 *const* tempArr = []  
 *for* (*let* popul = 0; popul < sizeChr; popul++) {  
 tempArr.push(Math.floor(Math.random()\*27))  
 }  
 *this*.population.push(tempArr)  
 }  
 }  
  
 getPopulation = () => {  
 *return this*.population  
 }  
  
 printPopulation = () => {  
 console.log('Популяция:')  
 *for* (*let* item = 0; item < *this*.population.length; item++) {  
 console.log(`Индивид № ${item}: ${*this*.population[item]}`)  
 }  
 }  
}  
  
*const* shuffle = (array) => {  
 *return* array.sort(() => Math.random() - 0.5);  
}  
  
*const* randomSelection = (numberIndivid) => {  
 console.log("\nСлучайный отбор хромосом")  
 *const* indexPopulation = [];  
 *for* (*let* item = 0; item < sizePopulation; item++) {  
 indexPopulation.push(item);  
 }  
 *const* shuffleArray = shuffle(indexPopulation)  
 *const* tempIndexPopulationArray = []  
 *for* (*let* item = 0; item < sizePopulation; item++) {  
 indexPopulation.push(item);  
 }  
 *for* (*let* item = 0; item < numberIndivid; item++) {  
 tempIndexPopulationArray.push(shuffleArray[item]);  
 }  
 console.log(`Популяция после отбора ${tempIndexPopulationArray}`)  
}  
  
*const* fromWorstToBest = (population, number) => {  
 console.log("От лучшего к худшему")  
 *const* average = []  
 *for* (*let* item = 0; item < population.length; item++) {  
 average.push(population[item].reduce((accumulator, currentValue) => {  
 *return* accumulator + currentValue  
 }, 0))  
 }  
 average.sort((a, b) => a - b )  
 *const* chunk = [];  
 *const* tempArrPart1 = average.slice(0, Math.floor(number/2))  
 *const* tempArray = [...average]  
 *const* tempArrPart2 = tempArray.reverse().slice(0, number/2 === 0 ? number/2 : Math.floor(number/2)+1)  
 *for* (*let* item = 0; item < Math.floor(number/2); item++) {  
 chunk.push(tempArrPart1[item])  
 chunk.push(tempArrPart2[item])  
 }  
 number/2 !== 0 ? chunk.push(tempArrPart2[Math.floor(number/2)]) : *null* chunk.sort((a, b) => a - b)  
 *const* arrayIndex = [];  
 *for* (*const* item *of* chunk) {  
 arrayIndex.push(average.findIndex(x => x === item))  
 }  
 console.log(`Популяция после отбора: ${arrayIndex}`)  
}  
  
*const* methodScale = (population) => {  
 console.log("Метод шкалы в операторе селекции (на основе суммы списка)")  
 *const* tmp = [];  
 *for* (*let* item = 0; item < population.length; item++) {  
 tmp.push(population[item].reduce((accumulator, currentValue) => {  
 *return* accumulator + currentValue  
 }, 0))  
 }  
  
 *const* array = [];  
 *for* (*let* i = 0; i < population.length; i++) {  
 *for* (*let* j = 0; j < population.length; j++ ) {  
 *const* tmpSumJ = population[j].reduce((accumulator, currentValue) => {  
 *return* accumulator + currentValue  
 }, 0)  
 *if* (tmp[i] === tmpSumJ) {  
 *if* (tmpSumJ <= 60){  
 array.push([j, tmp[i], 0])  
 } *else if* (tmpSumJ > 60 && tmpSumJ <= 90) {  
 array.push([j, tmp[i], 1])  
 } *else if* (tmpSumJ > 90) {  
 array.push([j, tmp[i], 2])  
 }  
 }  
 }  
 }  
  
 *const* usr = Number(readline.question('Номер хромосом (0-2): '))  
 *const* select = []  
 *for* (*let* i = 0; i < population.length; i++) {  
 *if* (usr === array[i][2]){  
 select.push(array[i][0])  
 }  
 }  
 console.log(`Хромосомы подходящие для селекции: ${select}`)  
}  
  
*const* methodElite = (population) => {  
 console.log('метод элит')  
 *const* kof\_chr = []  
 *for* (*let* i = 0; i < population.length; i++) {  
 kof\_chr.push(population[i].reduce((accumulator, currentValue) => {  
 *return* accumulator + (currentValue\*2+2)  
 }, 0))  
 }  
 *const* tmp = [...kof\_chr]  
  
 tmp.sort((a,b) => a - b).reverse()  
  
 *const* numberChromosome = readline.question('Количество элит (1-5): ');  
 *const* tempArrayChromosome = []  
  
 *for* (*let* i = 0; i < numberChromosome; i++) {  
 tempArrayChromosome.push(kof\_chr.findIndex(x => x === tmp[i]))  
 }  
 console.log(`Элитные хромосомы: `, tempArrayChromosome)  
}  
  
*const* population1 = *new* Population();  
population1.printPopulation();  
*const* printedPopulation = population1.getPopulation()  
  
console.log("\n-> методы отбора");  
*const* numberIndivid = Number(readline.question('Задайте кол-во индивидов от популяции: '))  
randomSelection(numberIndivid)  
fromWorstToBest(printedPopulation, numberIndivid)  
  
console.log("\n-> методы селекции");  
methodScale(printedPopulation)  
methodElite(printedPopulation)