Jurassic Park – проект 8

Илина Гергова 45503

[Линк към github](https://github.com/IlinaGergova/Jurassic-Park.git)

Идеята на проекта е да се направи програма, с която да се контролира един зоопарк за динозаври – добавяне и махане на динозаври, създаване на клетки и осигуряване на храна и персонал.В следващите параграфи ще се разгледа какви класове има и какво съдържат.

В проекта има три класа Dinosaur, Cage и JurassicPark, main(), няколко функции,отделени в header файл и един header ParkEnumerables, в който има 6 променливи тип enum. Реших да използвам enum, защото в задачата има данни, които могат да имат определен брой стойности и това ще улесни работата.

Клас Dinosaur съдържа всички характеристики, които един динозавър има, а те са - име, пол, ера, група, вид и храна като член-данни. Вместо динамична памет, съм използвала string за името и вида; останалите променливи са от тип enum. Конструктура приема и шест параметъра, съответстващи на член-данните и с тях конструира обект от тип този клас(има си и стойности по подразбиране). Има и гетъри, които трябват за достъпване до член-данните чрез другите класове и функции. Предефиниран е оператор ==, за да мога да сравнявам два обекта от тип Dinosaur по всички характеристики. Предефиниран е и оператора << така, че да извежда на конзолата данните на един динозавър (за променливите от тип enum използвам switch, защото не можем за „изпринтим“ enum като текст променлива с <<). Друг метод в класа Dinosaur е saveDino, който приема поток/файл тип ostream, в който ще се записват данните на динозавъра, като enum данните се записват като числа, тъй като всяка променлива в enum-a отговаря на индекс. За по-лесно четене от файла, при всяко записване най-напред се записва символа „^“, с който обозначавам, че съм записала динозавър. loadDino е метода, който чете от файл и съответно приема поток/файл тип istream. Създавам две променливи от тип string, където след прочит ще се запишат името и вида. За останалите характеристики са създадени променливи от тип int, защото са записани във файла като числа, а не като текст.Със switch случаи и в завизимост от прочетеното число, се определят пола, ерата, групата и храната след което се създава обект от тип Dinosaur като прочетените данни в съответните променливи се подават на конструктура и накрая този нова обект се връща от метода.

Клас Cage съдържа вектор от динозаври(тип Dinosaur), размер на клетката (от тип int), климат(тип enum),размер за колко динозавъра има вътре(тип int) и ера(тип enum) като член-данни. Конструктура приема само два параметъра- размер на клетката и климат, защото според условието тези две неща определят една клетка, като размера на клетката се приема като enum и със switch се преобразува в число; ерата е зададена да е „NoEra“, защото тя се определя от ерата на първия динозавър добавен в клетката, а при създаването на таква тя е празна. Има метод bool addAnimalInCage, който приема нов динозавър, които е const, защото няма да го променяме и има &, за да работим с оригинала и казва дали е добавено животно в клетката.Проверява се дали има място в клетката, дали и дали климата на клетката съответства на групата на динозавъра; проверява се ако клетката е празна, тоест няма динозаври вътре, ерата на клетката да бъде ерата на динозавъра; последната проверка е дали ерата на динозавъра съответства на ерата на останалите динозаври, тоест ерата на клетката и накрая ако проверките позволяват новия динозавър се добавя във вектора и бройката на динозаврите се увеличава с 1 и се връща true като резултат, тоест динозавъра е успешно добавен. Друг метод, който има в класа е за премахване на динозавър от клетка bool removeAnimalFromCage , който приема динозавър, с един цикъл се обхождат динозаврите във вектора и с един if и предефинирания оператор в класа Dinosaur == се проверява дали има същия динозавър като подадения, ако има тогава му запазваме индекса в променливата index. За премахването на динозавъра от вектора използвам erase като подавам позицията- началото на вектора + с колко отместена от началото, тоест + index и броя на динозаврите спада с 1, а ако броя стане 0, то това значи, че клетката е празна и не би трябвало да има ера, затова ерата е „NoEra“.Първоначално index = -1, защото в случай, че няма такъв динозавър метода да върне false и е -1,а не 0, защото ако няма динозавъра, който търсим, то тогава ще изтрием този на индекс 0. Предефиниран е оператор <<, за да извежда информацията за клетката и понеже enum не може да се изведе просто така, има switch, който проверява коя опция е използвана от съответия enum, и извежда текст, след което един цикъл, който извежда динозаврите от вектора (улесненто е, защото оператор << е предефиниран в клас Dinosaur).Следващ метод е saveCage, който приема файл/поток, в който записва информацията на клетката. Подобно на записването на динозавър, в началото преди всичко се записва обозначителен символ „#“, които казва, че след него е записана информацията на клетката; записва се и символа за нов ред, които помага при четенето на информацията.С един цикъл на всеки динозавър от вектора се вика метода, който е за запазване на динозавър от клас Dinosaur.Последен метод е loadCage, който отново приема файл/поток, от който чете информацията.Има две променливи тип int, в които записваме размера на клетката и климата и двете са тип enum в класа, затова се изпозват switch случаи, така че да ги преобразуваме в подходящ вид за класа. След като сме ги запазили, използваме get(), за да преместим указателя с едно място, тоест да го преместим на следващия ред. Създаваме обект от тип Cage и подаваме на конструктура размера и климата.С помощта на променлива от тип char и цикъл, започваме обхождането на елементите, докато не срещнем обозначителния знак за динозавър, а именно „^“ или докато не сме стиганли края на файла. Използваме peek(), който гледа елемент и го проверява, ако е търсения, използваме get(), за да отидем на следващия след „^“ елемент. После в новосъздадената клетка с метода addAnimalInCage добавяме динозавър, като на addAnimalInCage подаваме метода на Dinosaur за четене от файл loadDino, а пък на loadDino подаваме файла, от които се чете в метода loadCage.След това отново използваме get(), за да отидем на следващия ред, тоест да пропуснем символа за нов ред.С peek() гледаме следващия символ и така до края на цикъла.Накрая метода връща новосъздадената клетка.

Клас JurassicPark има вектор от клетки, бройка на персонал,бройка на клетки и бройки на различните храни.Конструктора приема само максимален размер на клетката и в него генерираме с rand() и srand(time(NULL)) (служи за това всеки път да дава различни числа) произволно число, което казва колко клетки да има, когато парка се конструира. По същия начин, но с помощта на switch и цикъл, се избира по случаен начин климат и размер за всяка клетка от масива(като % 3 + 1, означава, че ще избираме число от 1 до 3 включително).След което създаваме обект от тип Cage, подаваме на конструктура по случаен начин генерираните стойности и след това го добавяме ввъв вектора от клетки. Има метод за добавяне на животно addAnimalInPark, който приема нов динозавър и където проверяваме дали има достатъчно храна за динозавъра и после проверяваме всяка клетка за подходящо място за динозавъра и ако го добавим, ще вземем от общата бройка на съответната храната.Следващ метод е makeCage, които при подаден размер на клетка и климат прави нова такава и я слага във вектора от клетки и увеличава броя на клетките, единствената причина една клетка да не може да бъде създадена е да няма достатъчно персонал. Друг метод е addCageWithData, които приема вече готова клетка, тоест обект тип Cage, const е защото няма да го променяме и има &, за да работим с оригинала. Подобно на makeCage има проверка дали има достатъчно персонал и ако има тази клетка да се добави към вектора с клетки и броя да се увличи. Метода removeAnimal премахва животно от парка, като с цикъл на всяка клетка се вика метода removeAnimalFromCage и после се извежда името на динозавъра, който е бил махнат. Метода foodDelivery добавя храна, когато няма достатъчно, като се подава колко храна да се добави от потребителя. Метода addStaff е аналогичен на foodDelivery. Предефиниран е оператор <<, който приема освен файл/поток и обекти тип JurassicPark и служи за вкарване на информацията във файл, данните се записват на един ред, разделени със спейсове, а с цикъл на всяка клетка се вика метода saveCage от Cage, който ще запази информацията за клетките . Предефиниран е оператор >> за изкарване на информацията във файл. Приема освен файл/поток и обекти тип JurassicPark. Отново подобно на методи в другите класове, има няколко променливи от тип int, където се запзват съответните данни и после се подават на други методи.Следва търсенето на символа за клетка „#“. Преди да влезнем в цикъла, преместваме указателя един път с get(), за да отидем на следващия ред след което използваме peek(),за да вземем символа и в цикъла вече се проверява символа дали е търсения, добавянето на клетка, която е вече готова с метода addCageWithData, на който подаваме loadCage, a пък на него файла, от който четем. Има метод print(), който извежда информацията за парка.

В main() функцията се проверява какъв вид команда е казана от потребителя и се извиква съответната функция от Commands.h.

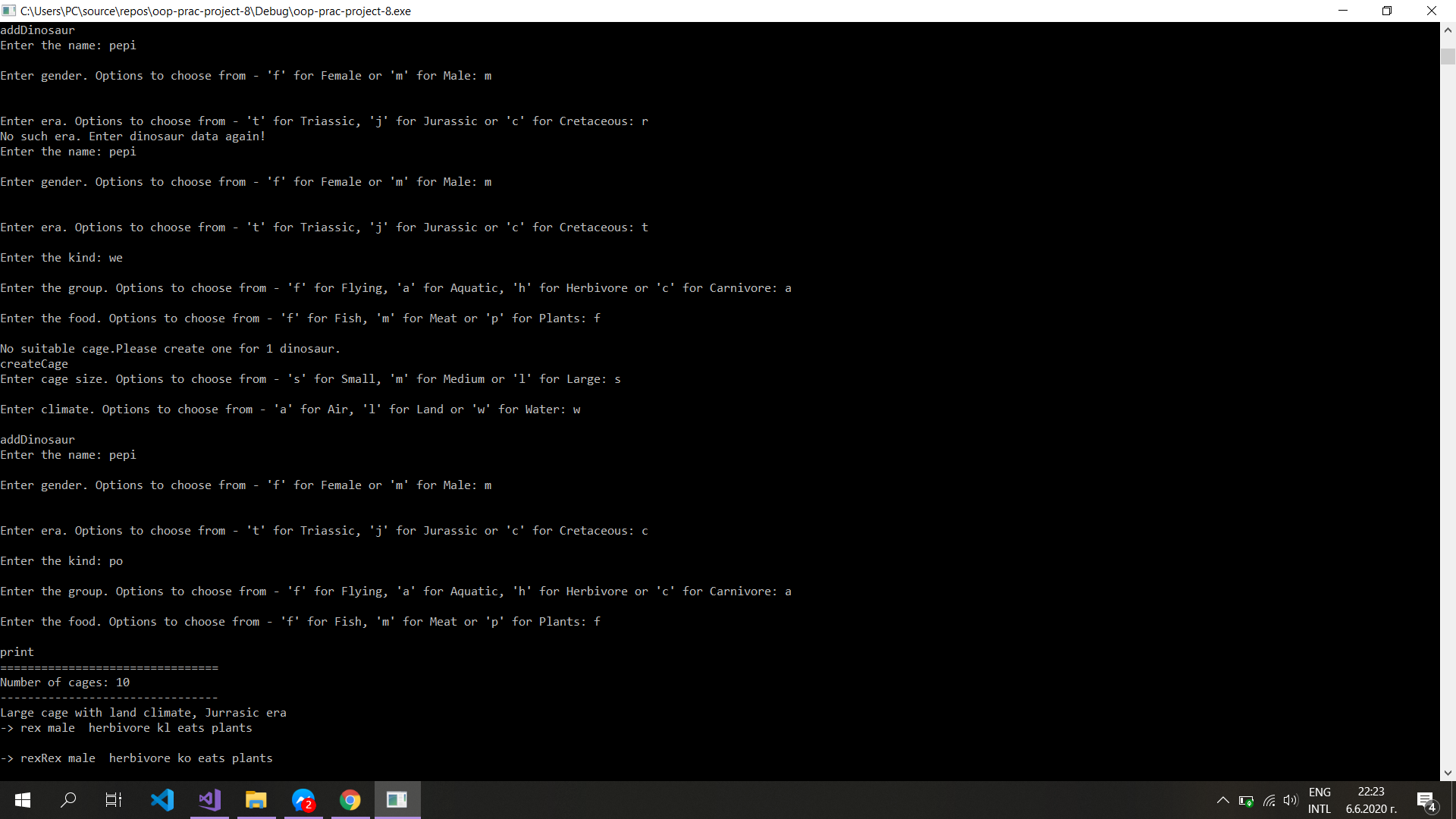
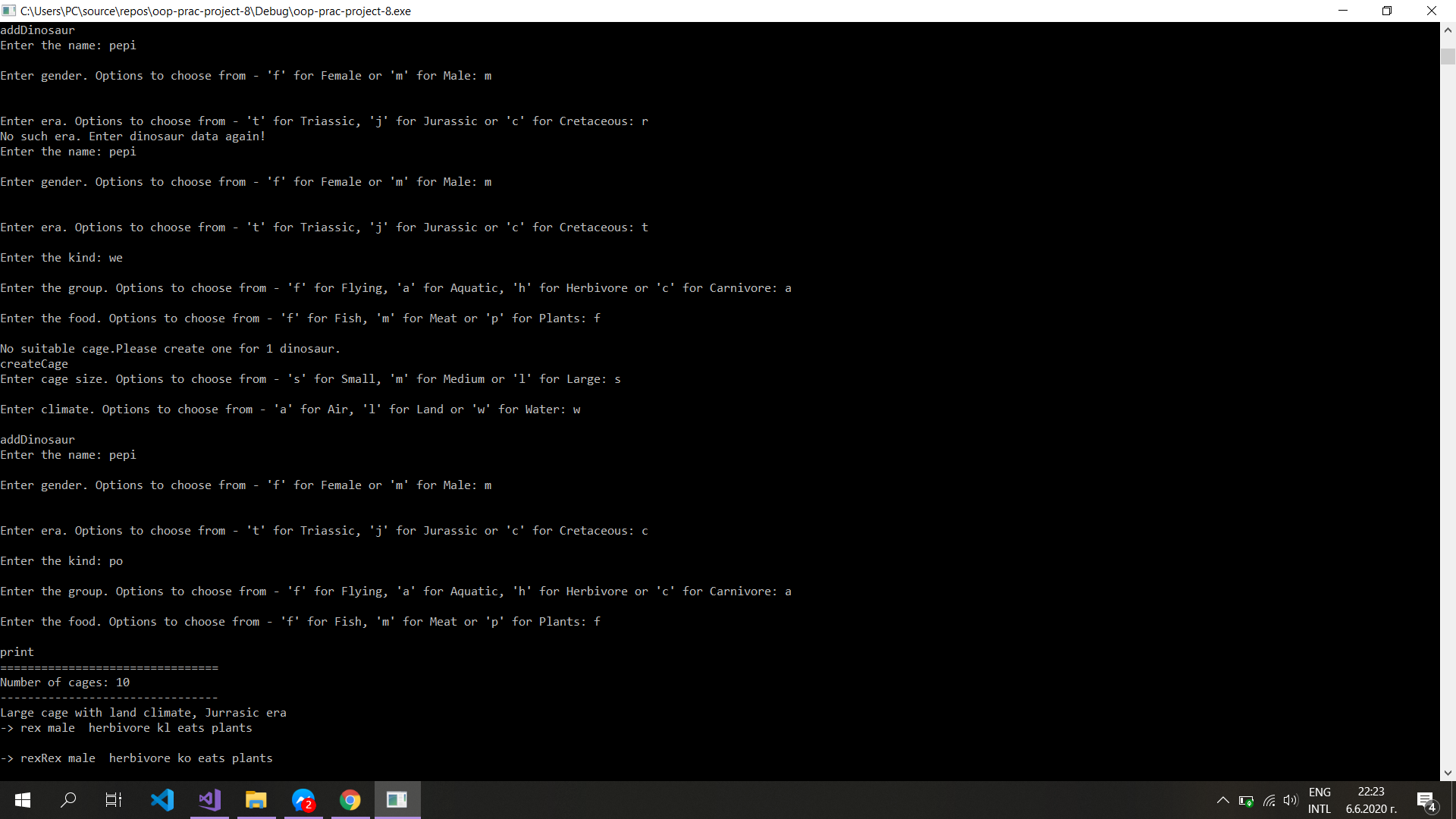
За всяка команда има външна функция като командите могат да са:

1. addDinosaur - adds a new dinosaur
2. createCage - creates a new cage
3. remove - removes a dinosaur
4. addFood - adds food to the park's reserves
5. addStaff - adds staff to the park
6. print - prints the information of the park.
7. exit - terminates the program and saves data.

* Съотношението на храна и динозавър е 1:1, както и на персонал и клетки

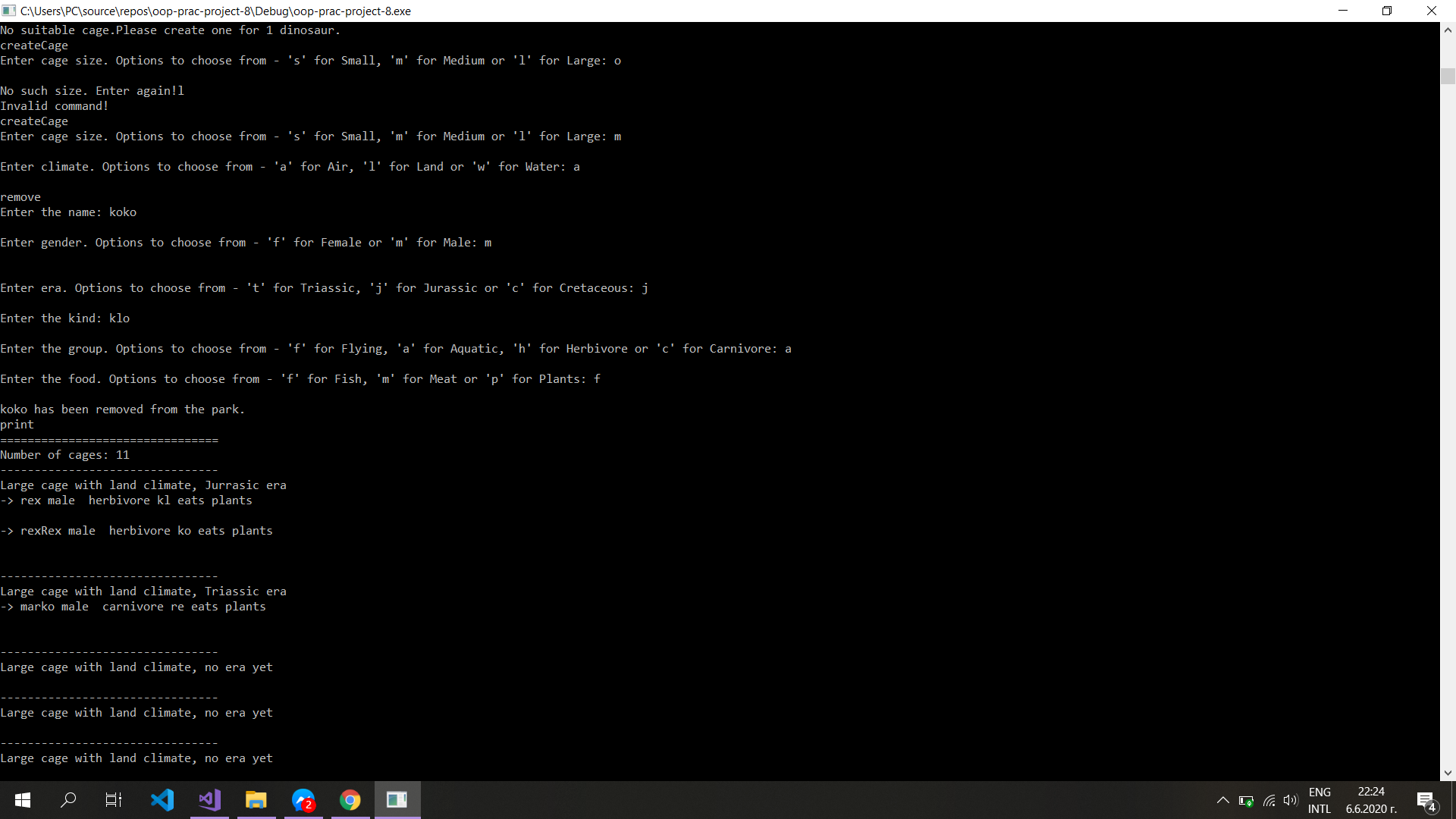
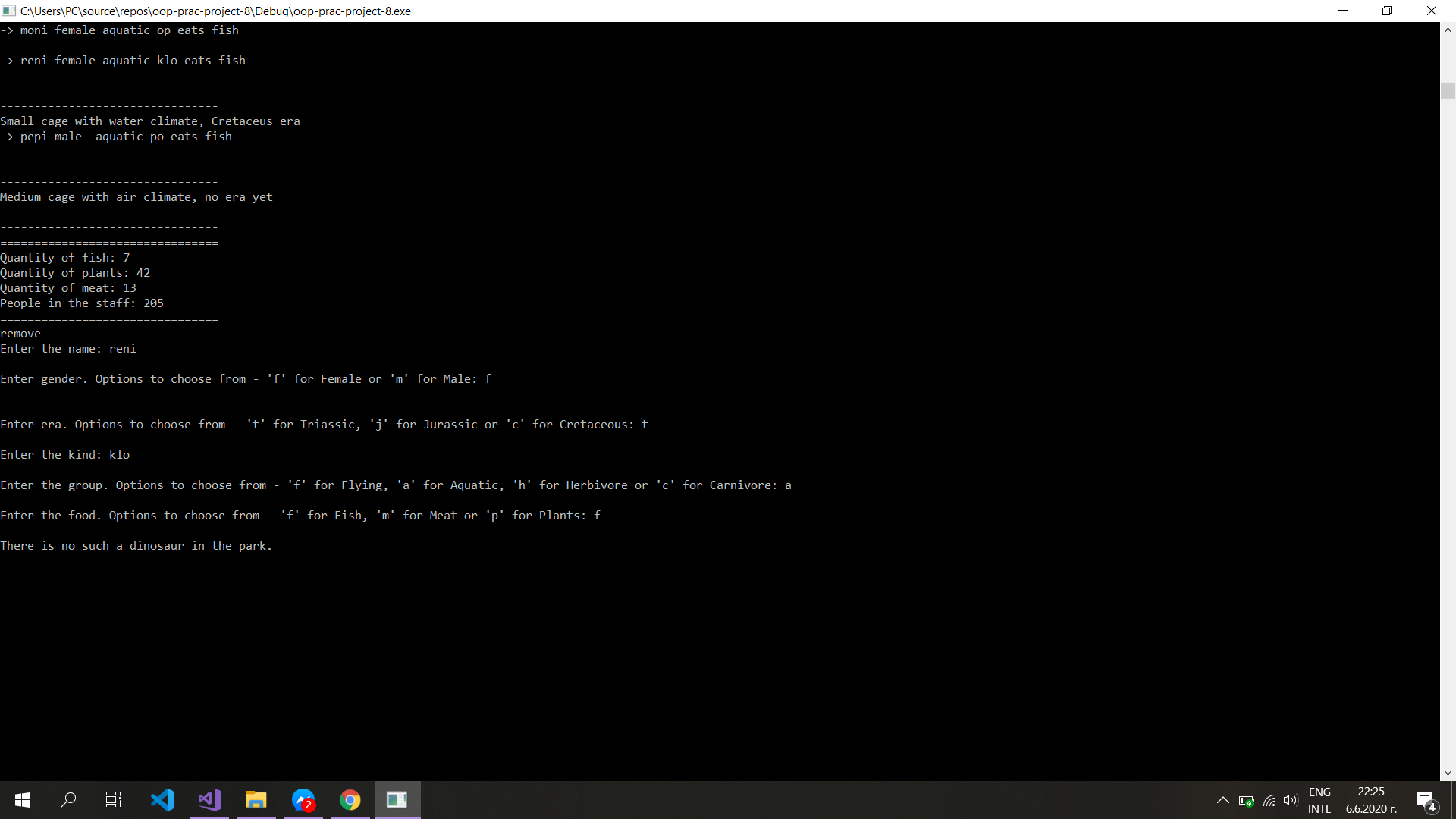
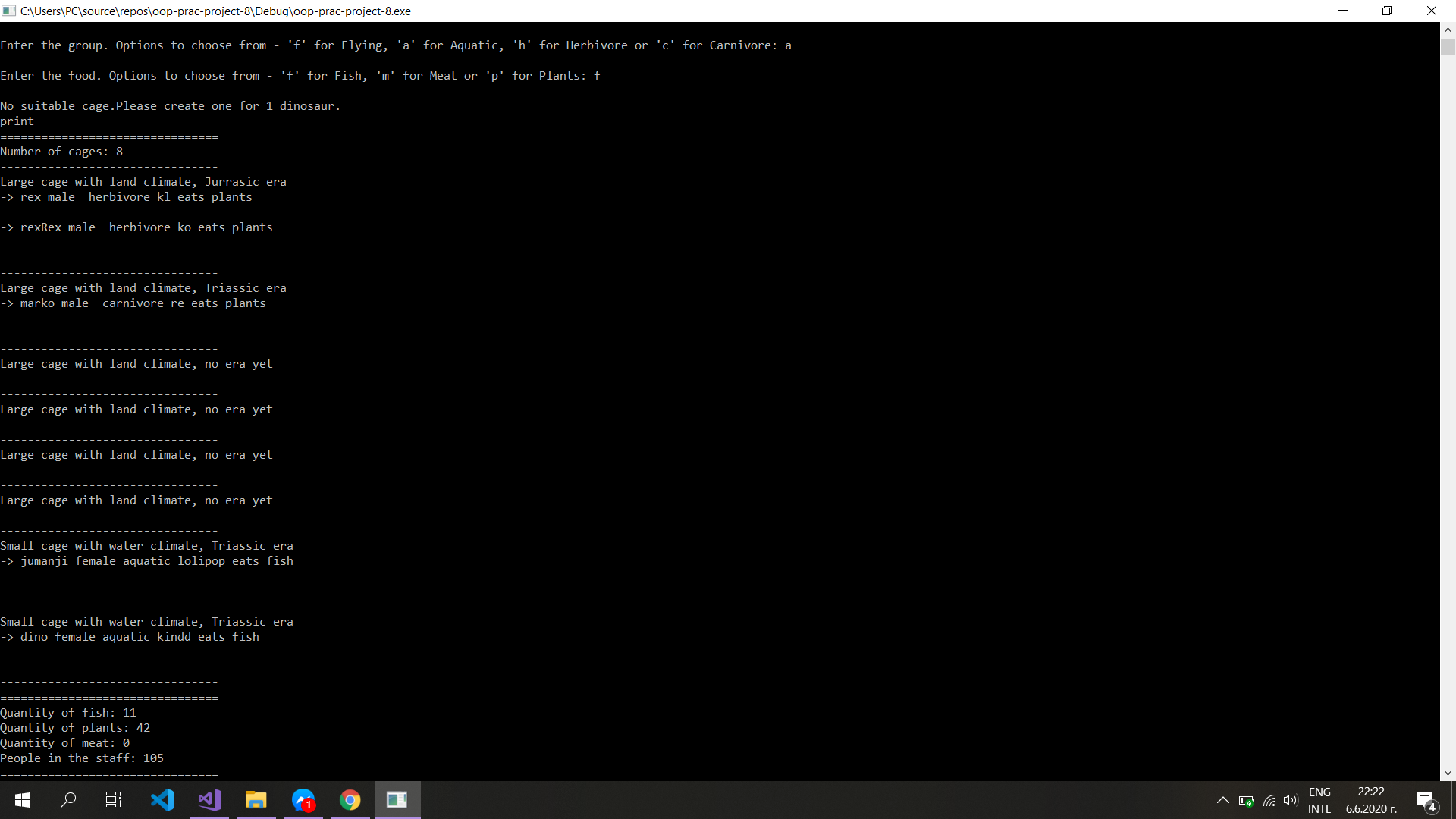
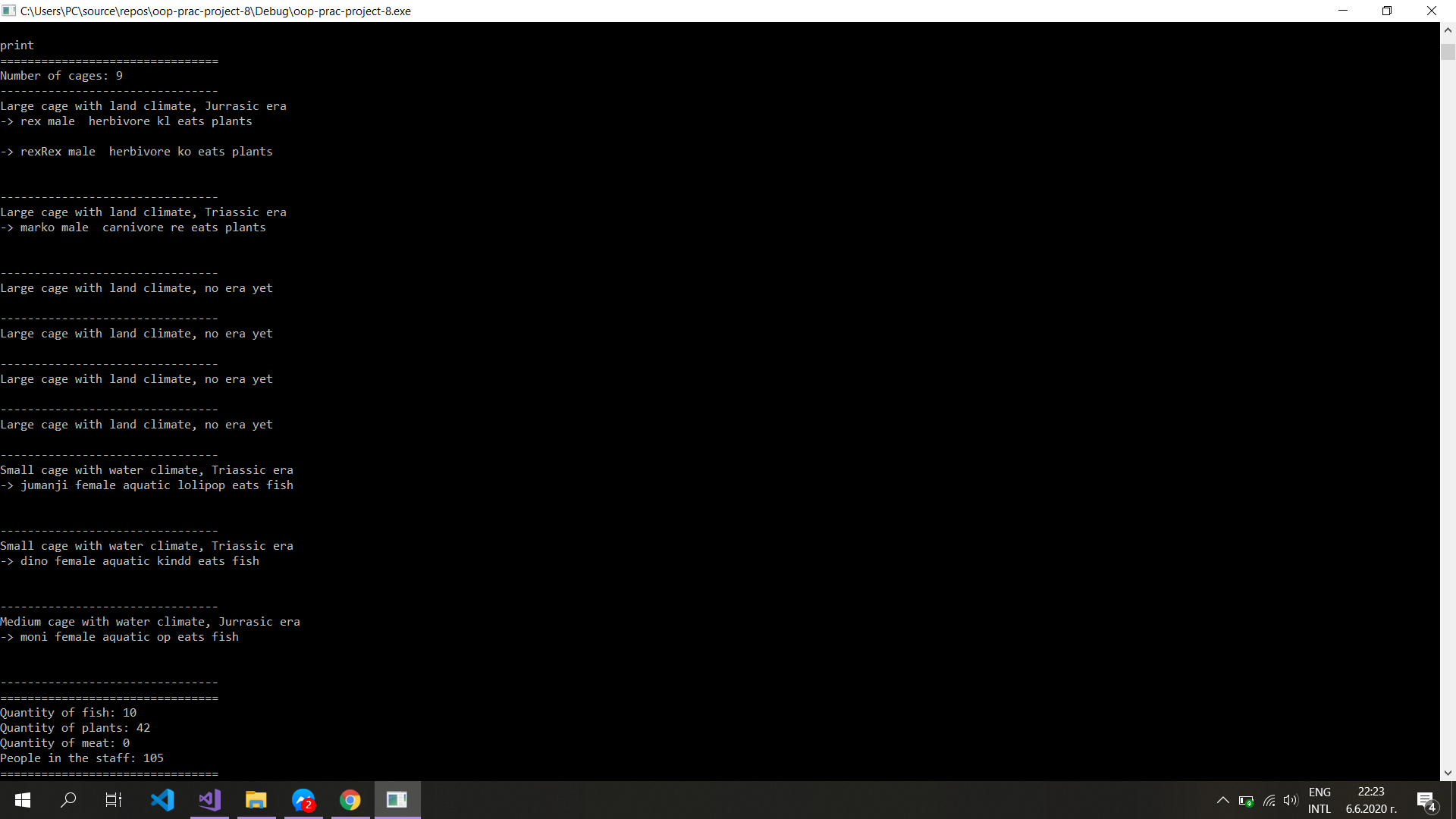
Примери от това как работят функциите:

1.addDinosaur

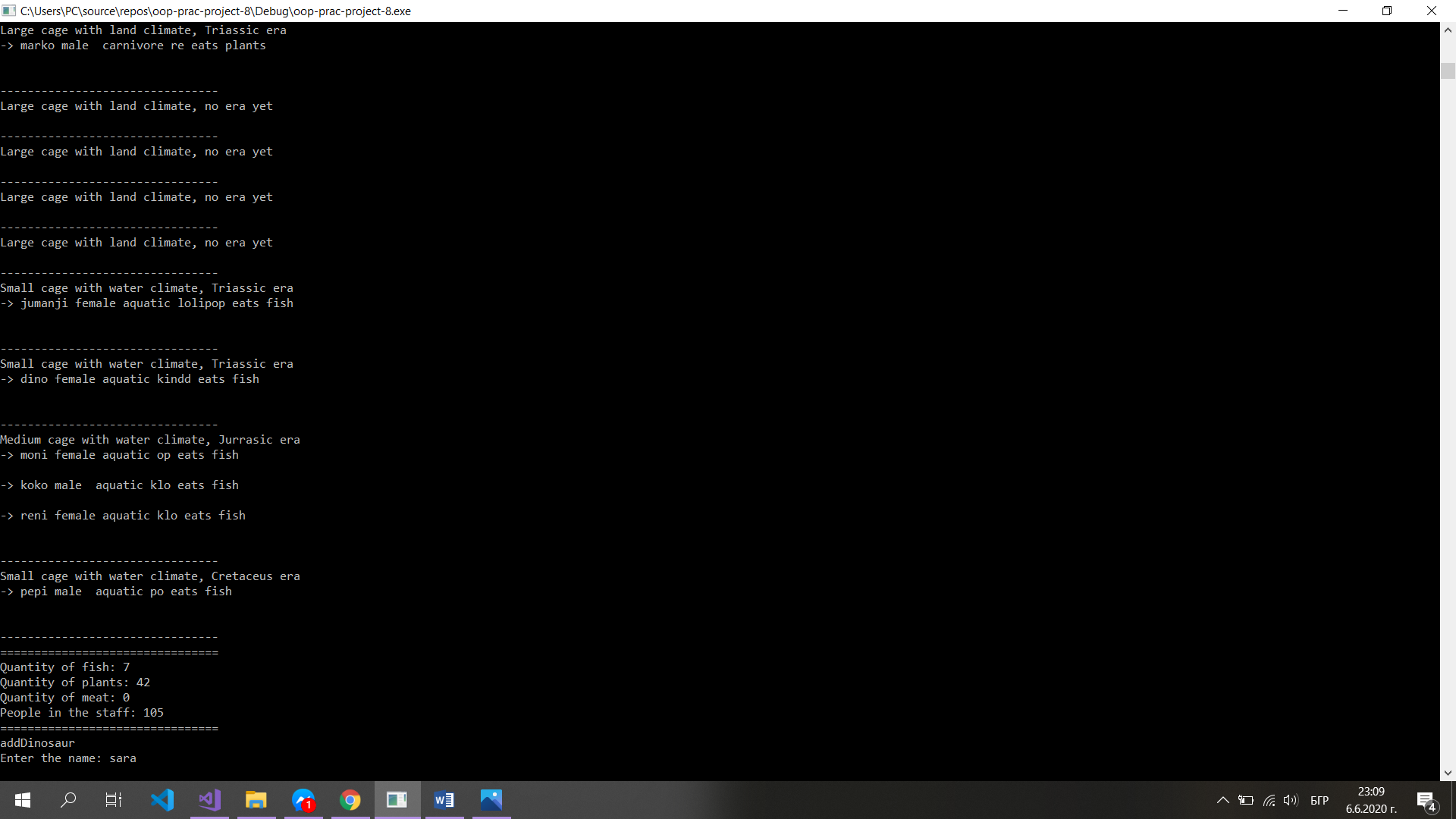


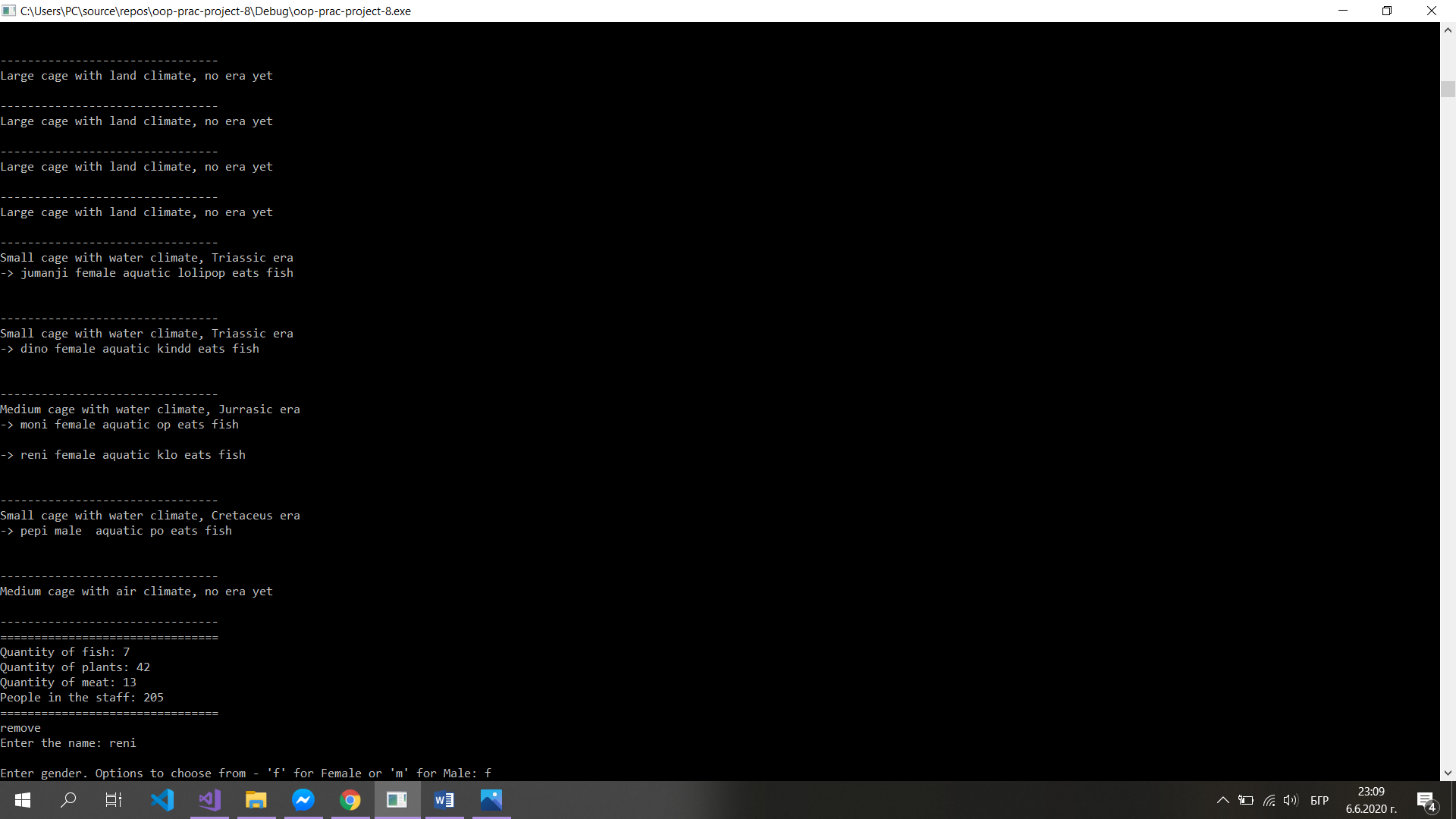
2.createCage

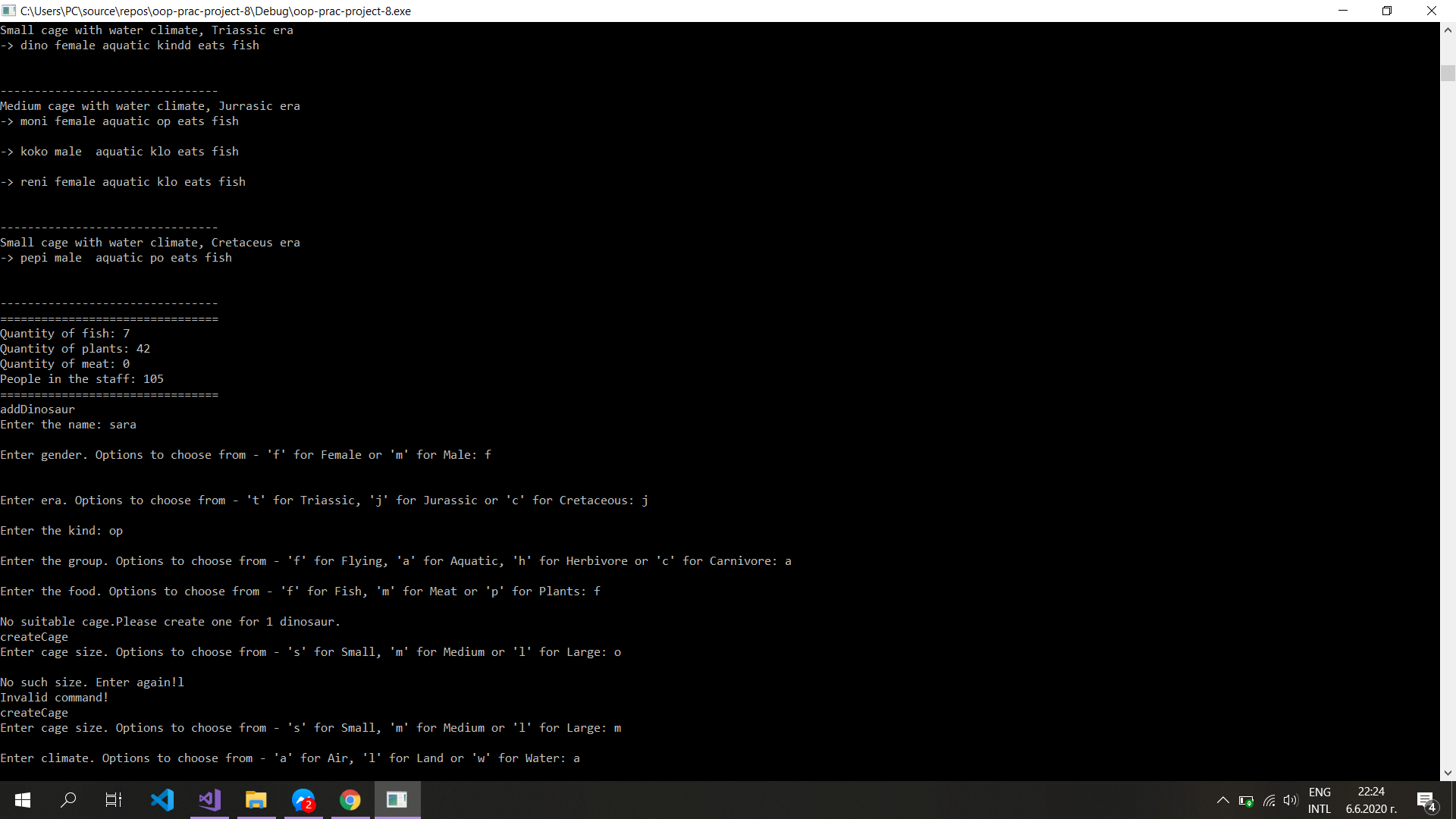
След като сме създали подходяща клетка, отново добавяме динозавър с addDinosaur като 1. При всяко добавяне на динозавър, храната, която той яде, спада с 1.

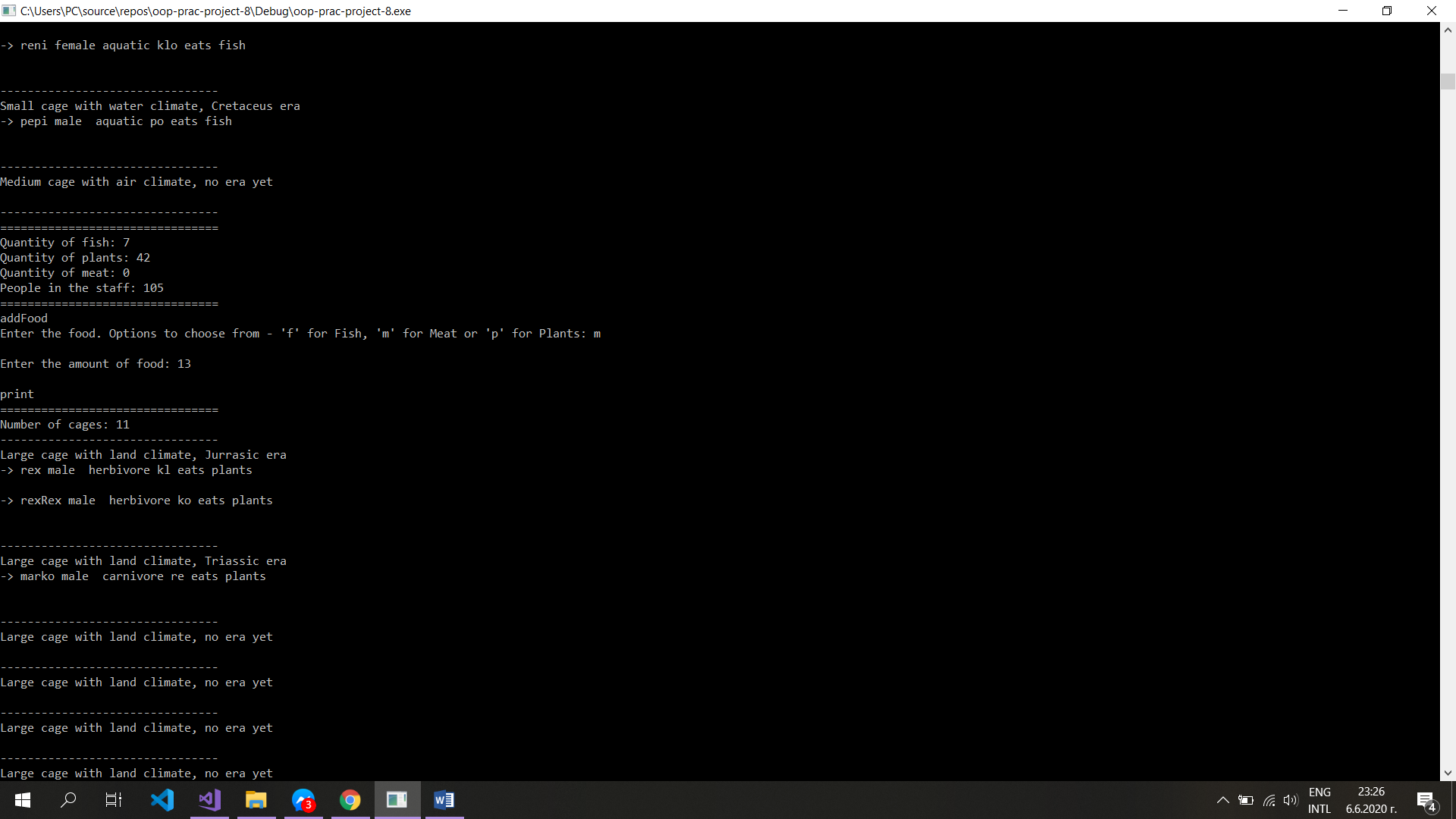


3.remove

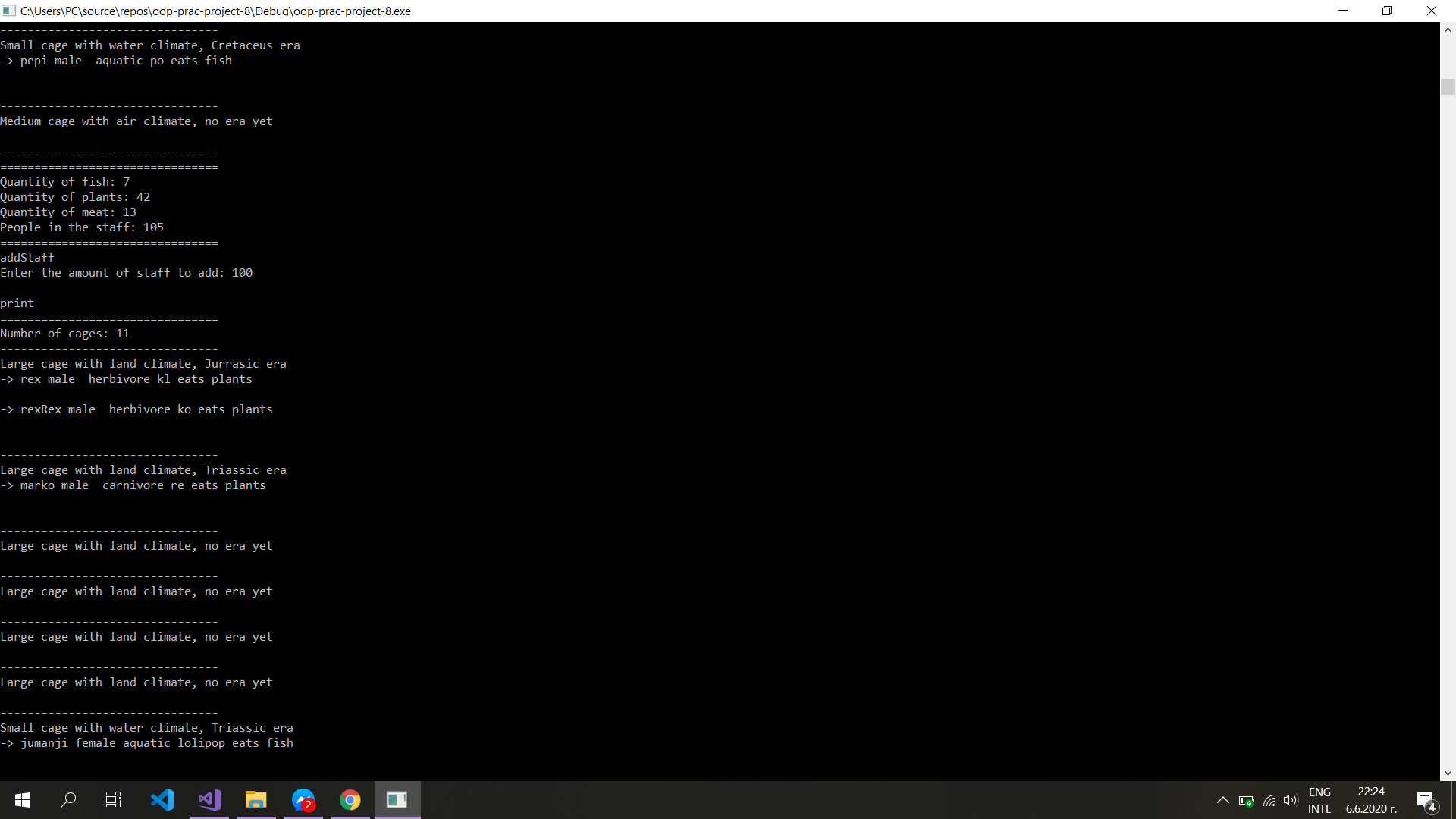


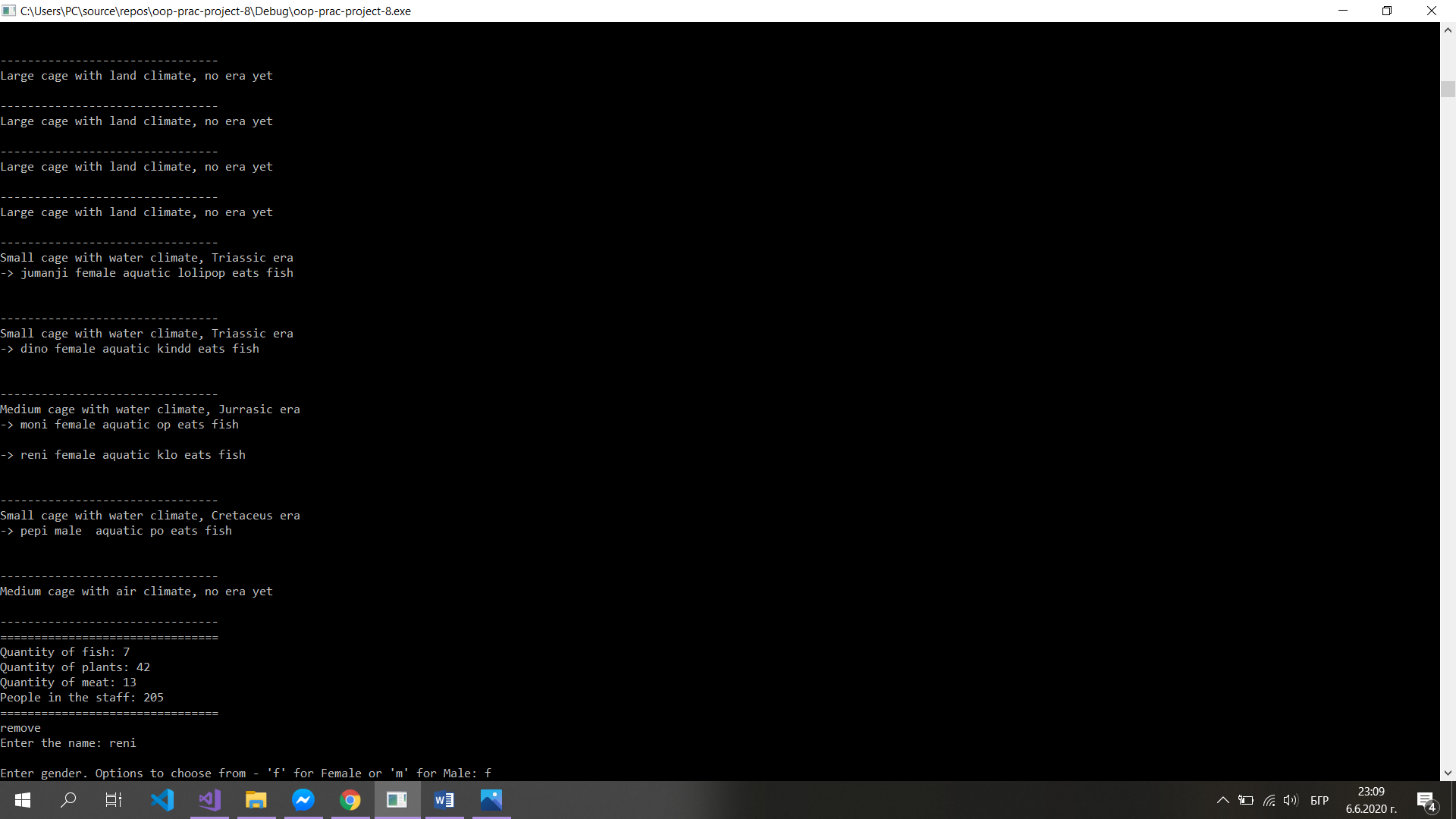


4.addFood

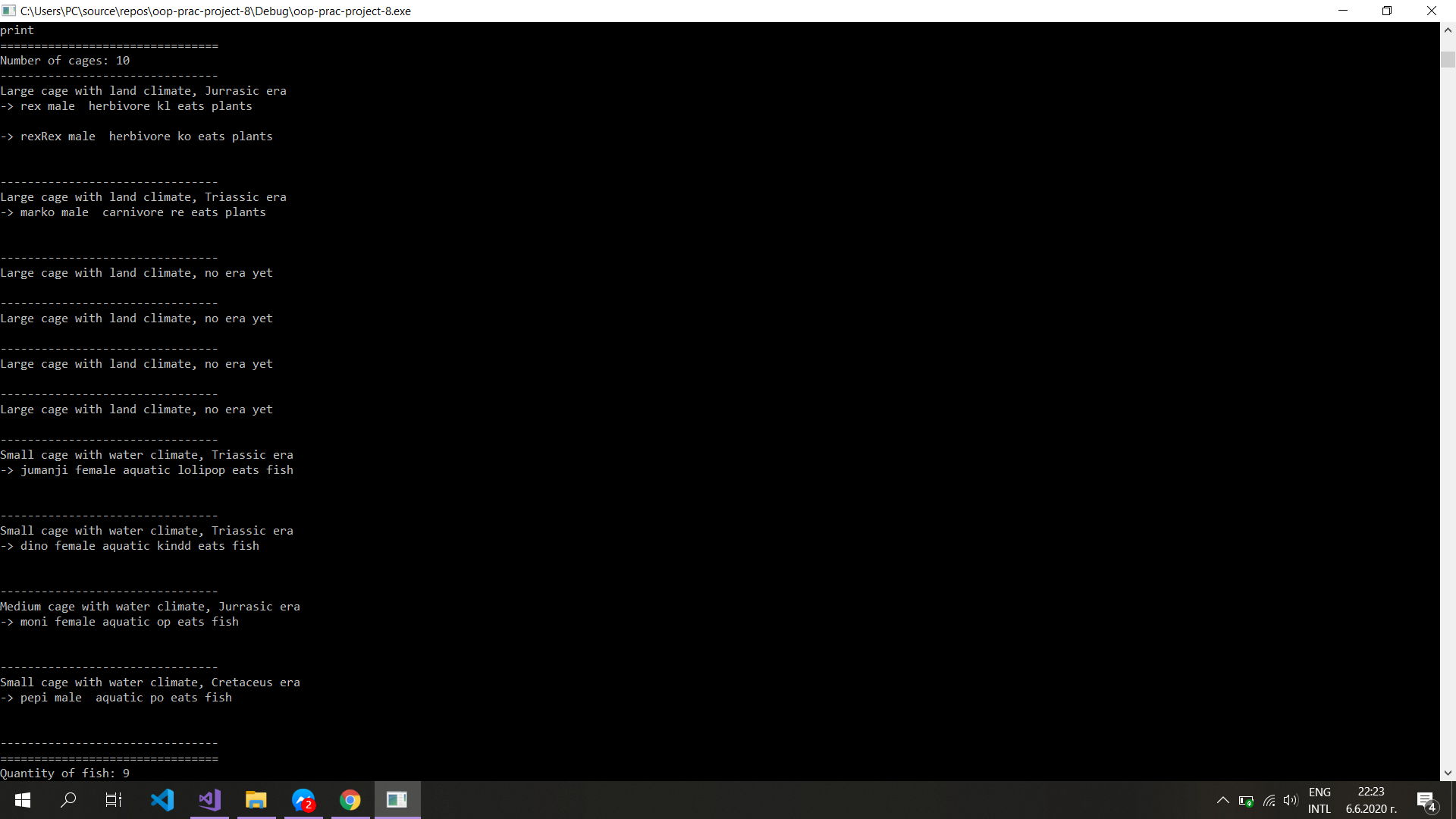


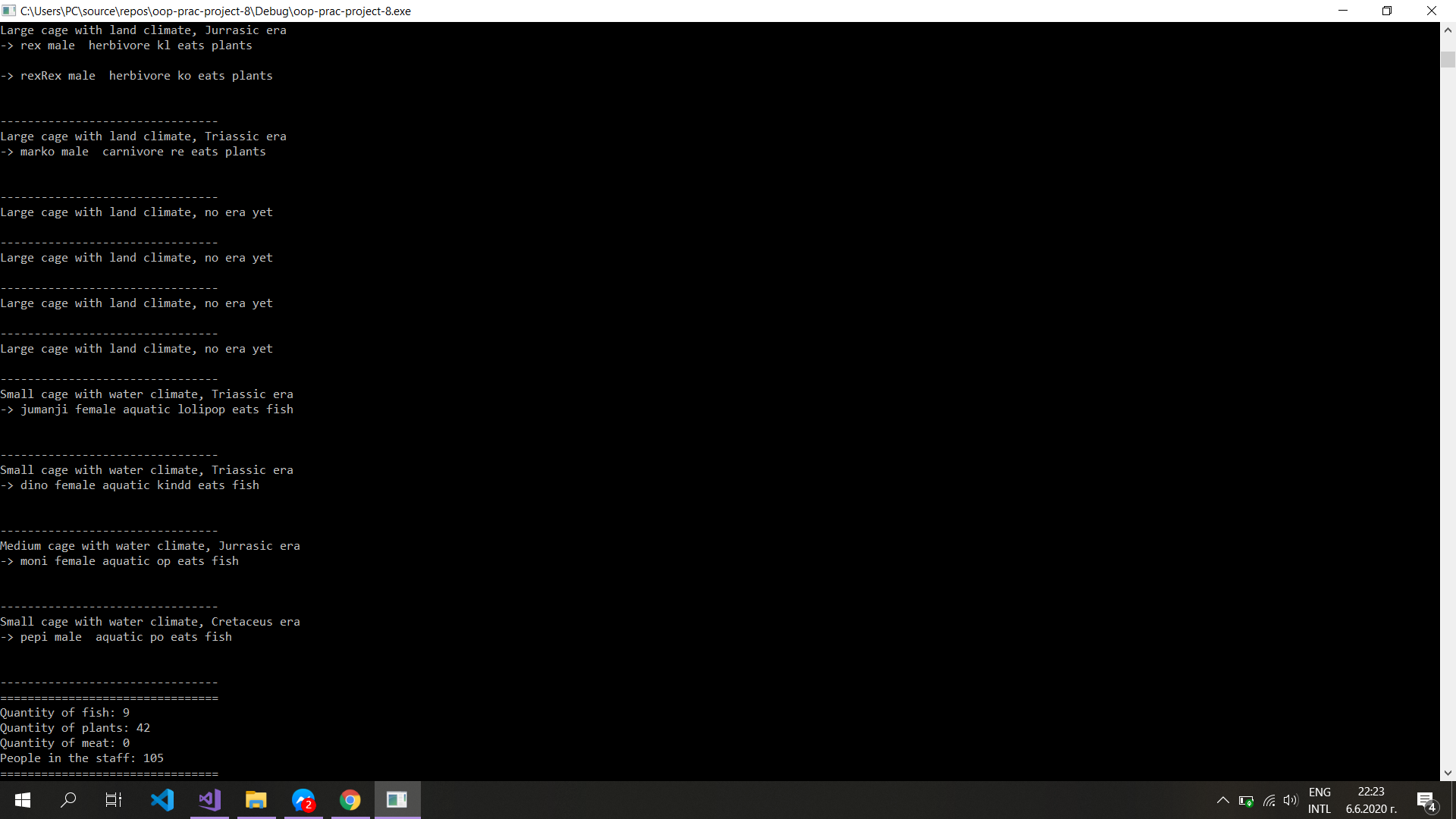
5.addStaff





6.print



Новодобавената клетка с новия динозавър от 1.