**Команда 5 А-09-18**

**Петров А.Д. – аналитик**

**Касаткин К.А. – тестировщик**

**Бондаренко Н.Д. - менеджер**

**Поздеев Д.А. – разработчик**

**Документация по лабораторной работе**

**Вариант 3**

1. **Задание:**

Разработка алгоритма для анализа и рационализации биматричных игр:

1. Ввод, вывод и представление матричных данных.
2. Задание опций анализа/рационализации.

Уточнение задания согласно Варианту 3:

Поиск равновесий по Нэшу и оптимумов по Парето в чистых стратегиях.

**2. Процесс разработки**

Было проведено два этапа разработки алгоритмов и соответственно 2 этапа тестирования для проверки фундаментальных и точечных ситуаций, требующих особого внимания.

**3. Запуск и работоспособность**

Разработчиком было предусмотрено создание меню для удобства эксплуатации программы. Оно проиллюстрировано на рисунке 1.

Пользователю предлагается нажать любую клавишу в соответствии с указаниями в меню для выполнения желаемого действия.

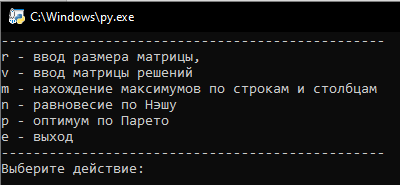


Рисунок 1. Стартовое меню программы.

Предполагается первоочередно выбрать первую опцию для ввода размера матрицы. Пользователь может ввести целочисленные значения для каждого из двух игроков.

Следующим действием нужно ввести матрицу весов или в нашей интерпретации матрицу решений. Вводятся названия стратегий и выигрыши для каждого игрока в соответствии с каждой стратегией. Демонстрация на рисунке 2.

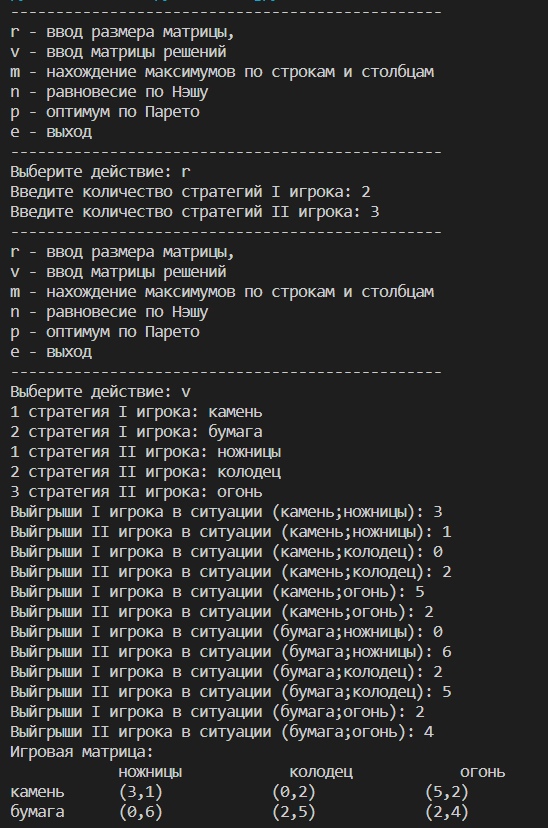


Рисунок 2. Ввод и вывод матрицы решений.

Существует несколько оптимумов по Парето. На рисунках 3 и 4 результаты работоспособности.

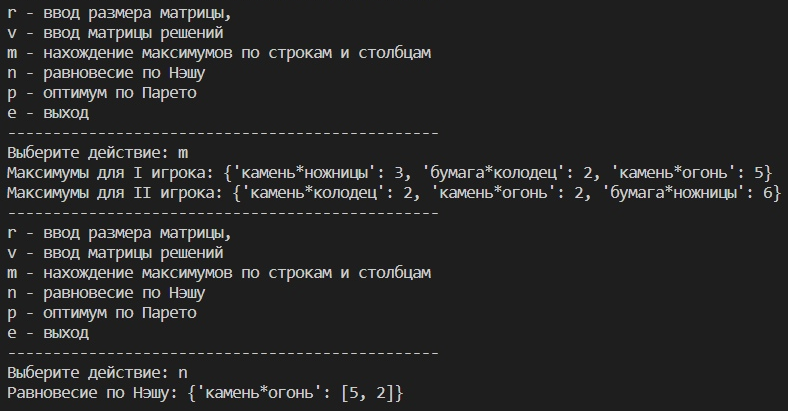


Рисунок 3. Нахождение максимумов

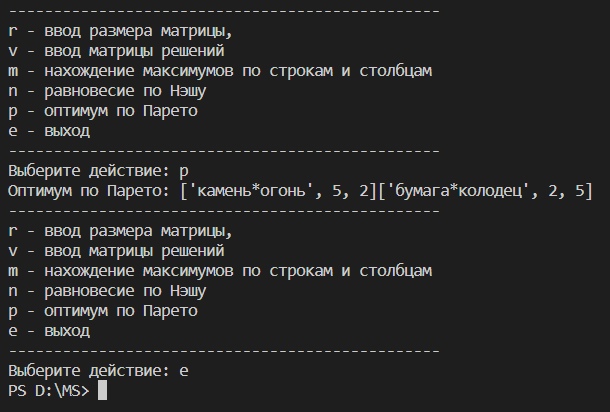


Рисунок 4. Оптимум по Парето и выход

Следующая разобранная ситуация так называемая «дилемма заключенного». Рисунки 5 и 6.

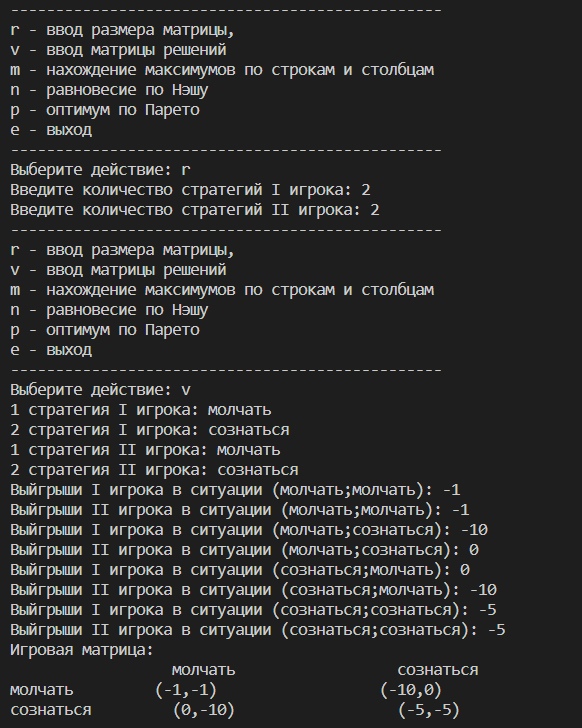


Рисунок 5. Ввод матрицы решений для «дилеммы заключенного»

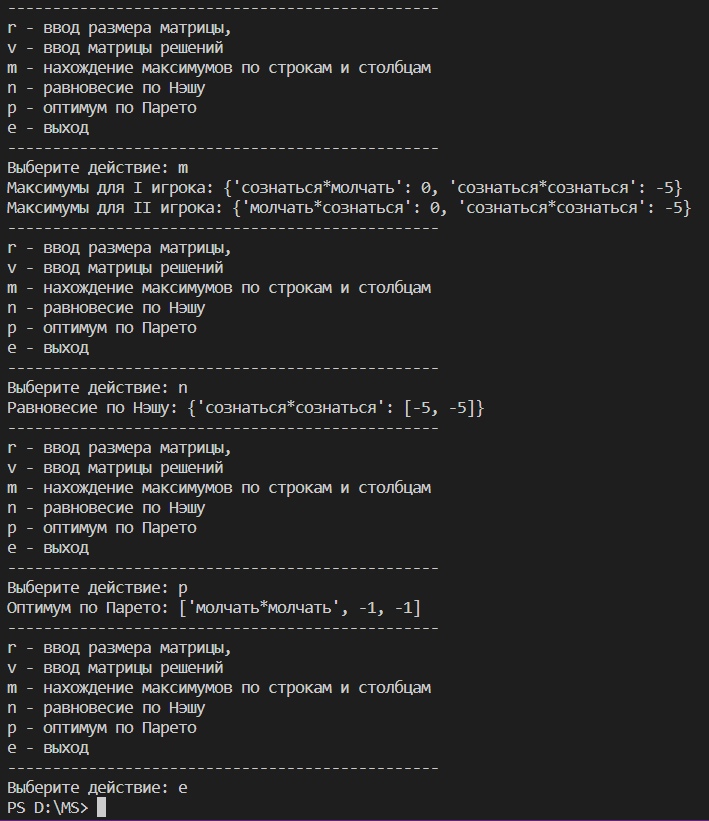
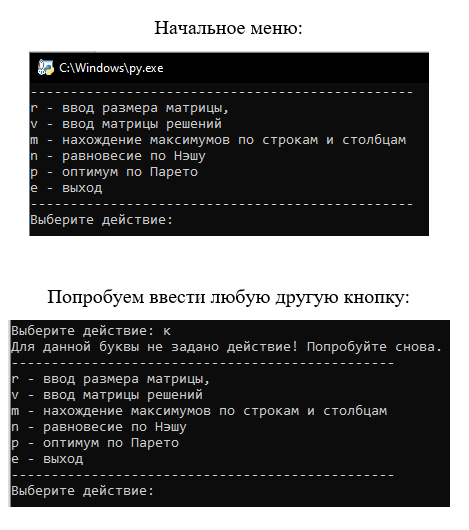
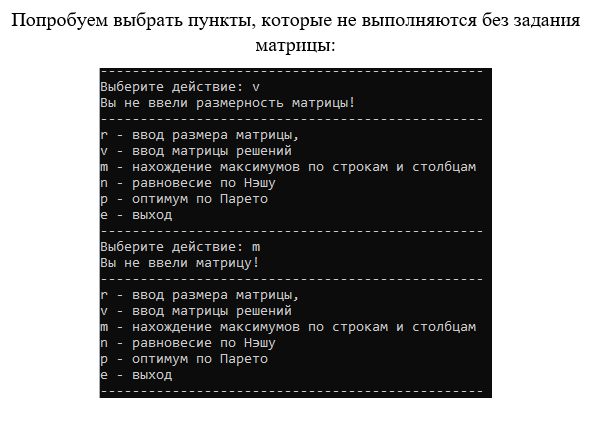


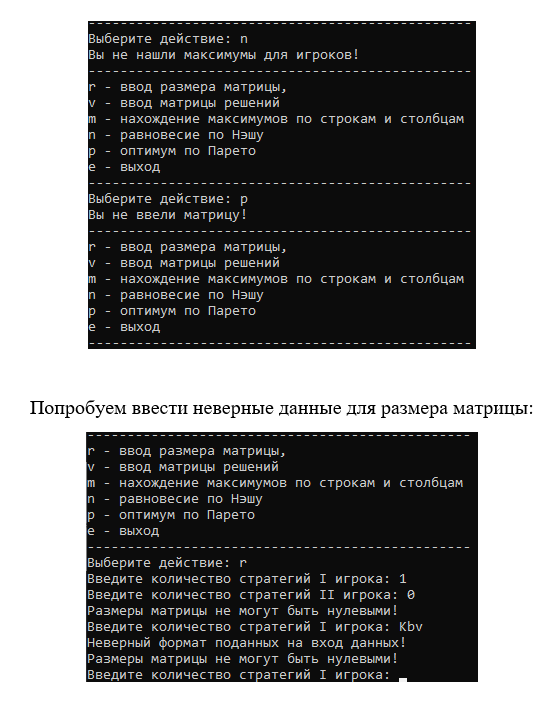
Рисунок 6. Нахождения равновесия по Нэшу и оптимума по Парето.

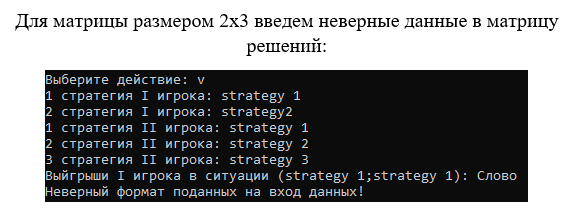
**4. Тесты**

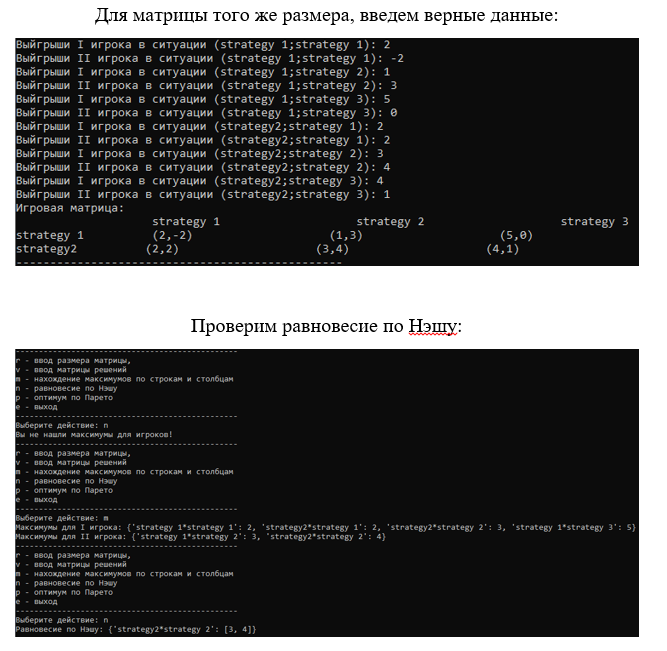
Первая итерация тестов была проведена преимущественно на ввод не валидных команд для программы. Ниже будет сплошная выкладка из отчёта нашего тестировщика.

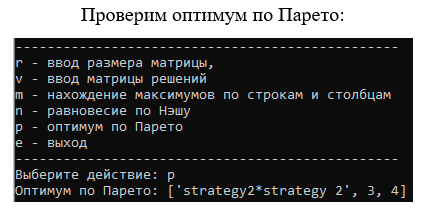












Была выявлена полная работоспособнасть в случаях ввода неуместных команд.

Вторая итерация тестов выявила уязвимость при выводе нескольих вариантов оптимума по Парето.

Разработчиком была исправлена проблема, что отразилось на итоговой эстимации задания. О временных трудозатратах и проблемных моментах в следующем разделе касательно аналитической части работы.

**5. Анализ проделанной работы**

Разработчиком было объявлено о планируемых 4,5 часах на написание функционала.

Части тестировщика, менеджера, а также аналитика предположительно должны были занять суммарно еще 7 часов работы.

Действительные затраты на завершение всех циклов работы превысили ожидаемые на 6 часов в связи с нахождением дополнительной уязвимости, о которой было написано выше, корректированием рабочего процесса остальных участников команды в связи с этой ситуацией.

В целом по опросу участников трудности возникали в виду необходимости дополнительного ознакомления с документациями языка программирования Python. Эта проблема заняла 1,5 часа из выше упомянутых 6.

**6. Код программы**

def razmer():  
fl=False  
while(fl==False):  
try:  
c=int(input('Введите количество стратегий I игрока: '))  
d=int(input('Введите количество стратегий II игрока: '))  
fl=True  
except ValueError:  
print('Неверный формат поданных на вход данных!')  
fl=False  
return c,d  
  
def vvod():  
max\_sl=0  
tablic=[]  
strat1=[]  
strat2=[]  
for i in range (0,a):  
strat1.append(input(str(i+1) + ' стратегия I игрока: '))  
for i in range(0,a):  
if max\_sl<len(strat1[i]):  
max\_sl=len(strat1[i])  
for i in range (0,b):  
strat2.append(input(str(i+1) + ' стратегия II игрока: '))  
for i in range (0,a):  
tablstr=[]  
for j in range (0,b):  
fl=False  
while(fl==False):  
try:  
win=[0,0]  
win[0]=int(input('Выйгрыши I игрока в ситуации (' + strat1[i] + ';' + strat2[j] + '): '))  
win[1]=int(input('Выйгрыши II игрока в ситуации (' + strat1[i] + ';' + strat2[j] + '): '))  
fl=True  
except ValueError:  
print('Неверный формат поданных на вход данных!')  
tablstr.append(win)  
tablic.append(tablstr)  
  
qw=max\_sl\*2\*' '  
print('Игровая матрица: ')  
print(max\_sl\*2\*' ' + qw.join(strat2))  
for i in range(0,a):  
stroka=''  
for j in range(0,b):  
stroka=stroka + '(' + str(tablic[i][j][0]) + ',' + str(tablic[i][j][1]) + ')' + 2\*max\_sl\*' '  
print(strat1[i] + max\_sl\*' ' + stroka)  
return strat1,strat2,tablic  
  
def maxab():  
maxa={}  
maxb={}  
for i in range(0,b):  
maxb[stratI[0] + '\*' + stratII[i]]=tabl[0][i][0]  
for j in range(0,a):  
keyss={}  
for key in maxb:  
keyss[key]=0  
for key in keyss:  
if (int(maxb[key]) == int(tabl[j][i][0])) and str(key).endswith(stratII[i]):  
maxb[stratI[j] + '\*' + stratII[i]] = tabl[j][i][0]  
if (int(maxb[key]) < int(tabl[j][i][0])) and str(key).endswith(stratII[i]):  
maxb.pop(key)  
maxb[stratI[j] + '\*' + stratII[i]] = tabl[j][i][0]  
  
for i in range(0,a):  
maxa[stratI[i] + '\*' + stratII[0]]=tabl[i][0][1]  
for j in range(0,b):  
keyss={}  
for key in maxa:  
keyss[key]=0  
for key in keyss:  
if (int(maxa[key]) == int(tabl[i][j][1])) and str(key).startswith(stratI[i]):  
maxa[stratI[i] + '\*' + stratII[j]] = tabl[i][j][1]  
if (int(maxa[key]) < int(tabl[i][j][1])) and str(key).startswith(stratI[i]):  
maxa.pop(key)  
maxa[stratI[i] + '\*' + stratII[j]] = tabl[i][j][1]  
maxbb=str(maxb)  
maxaa=str(maxa)  
print('Максимумы для I игрока: ' + maxbb)  
print('Максимумы для II игрока: ' + maxaa)  
return maxa, maxb  
  
def naash():  
nash={}  
for keya in max\_a:  
for keyb in max\_b:  
if keya==keyb:  
nash[keya]=[int(max\_b[keya]),int(max\_a[keyb])]  
nshh=str(nash)  
print('Равновесие по Нэшу: ' + nshh)  
return nash  
  
def paretto():  
pareto={}  
max\_sum={'0,0':[int(tabl[0][0][0])+int(tabl[0][0][1]),stratI[0],stratII[0]]}  
for i in range(0,a):  
for j in range(0,b):  
abcd={}  
for key in max\_sum:  
abcd[key]={'0'}  
for keys in abcd:  
if max\_sum[keys][0]==int(tabl[i][j][0])+int(tabl[i][j][1]):  
max\_sum[f'{i},{j}']=[int(tabl[i][j][0])+int(tabl[i][j][1]),stratI[i],stratII[j],i,j]  
if max\_sum[keys][0]<int(tabl[i][j][0])+int(tabl[i][j][1]):  
max\_sum.pop(keys)  
max\_sum[f'{i},{j}']=[int(tabl[i][j][0])+int(tabl[i][j][1]),stratI[i],stratII[j],i,j]  
for key in max\_sum:  
pareto[key]=[max\_sum[key][1] + '\*' + max\_sum[key][2],tabl[max\_sum[key][3]][max\_sum[key][4]][0],tabl[max\_sum[key][3]][max\_sum[key][4]][1]]  
prtt=''  
for key in pareto:  
prtt=prtt + str(pareto[key])  
print('Оптимум по Парето: ' + prtt)  
return pareto  
  
char=''  
itog=[]  
while char!='e':  
print('------------------------------------------------' + '\n' + 'r - ввод размера матрицы,' + '\n' + 'v - ввод матрицы решений' + '\n' + 'm - нахождение максимумов по строкам и столбцам' + '\n' + 'n - равновесие по Нэшу' + '\n' + 'p - оптимум по Парето' + '\n' + 'e - выход' + '\n' + '------------------------------------------------')  
char=input('Выберите действие: ')  
  
if char=='r':  
itog.clear()  
itog.append('r')  
a,b=razmer()  
  
if char=='v' and 'r' in itog:  
itog.append('v')  
stratI,stratII,tabl=vvod()  
if char=='v' and 'r' not in itog:  
print('Вы не ввели размерность матрицы!')  
  
if char=='m' and 'v' in itog:

itog.append('m')  
max\_a,max\_b=maxab()  
if char=='m' and 'v' not in itog:  
print('Вы не ввели матрицу!')  
  
if char=='n' and 'm' in itog:  
itog.append('n')  
nsh=naash()  
if char=='n' and 'm' not in itog:  
print('Вы не нашли максимумы для игроков!')  
  
if char=='p' and 'v' in itog:  
itog.append('p')  
prt=paretto()  
if char=='p' and 'v' not in itog:  
print('Вы не ввели матрицу!')  
  
if char=='e':  
exit()  
  
if char!='r' and char!='v' and char!='m' and char!='n' and char!='p' and char!='e':  
print('Для данной буквы не задано действие! Попробуйте снова.')