

Пензенский государственный университет

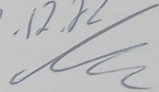
Кафедра «Вычислительная техника»

Пояснительная записка

к курсовому проектированию

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Реализация алгоритма по распределению ресурсов»

ОП.
28.12.22


Выполнил:

Студент группы 21ВВ1

Антропов Д.В.

Принял:

Митрохин М.А.

Пенза 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

« » 20

ЗАДАНИЕ

на курсовое проектирование по курсу

«Логика и основы алгоритмизации в вычислительных задачах»
Студенту Андрееву Даниилу Вадимовичу Группа 21ВВ1
Тема проекта «Оптимизация алгоритмов по распределению ресурсов»

Исходные данные (технические требования) на проектирование

Разработка алгоритмов и программного обеспечения в соответствии с
техническим заданием курсового проекта.

Техническое задание должно содержать:

1. Постановку задачи
2. Описание алгоритма постановки задачи
3. Пример ручного расчета задачи и вычислений
4. Описание самой программы
5. Тесты
6. Листинг программы
7. Результаты работы программы.

Объем работы по курсу

1. Расчетная часть

Полный расчет работы выполнен.

2. Графическая часть

Схема алгоритма в формате блок-схем.

3. Экспериментальная часть

Тестирование программы.

Визуализация работы программы на типовых данных.

Срок выполнения проекта по разделам

- 1 Исследование теоретической части курса
- 2 Разработка алгоритмов, программ
- 3 Разработка программы
- 4 Тестирование и завершение разработки программы
- 5 Оформление пояснительной записки
- 6
- 7
- 8

Дата выдачи задания "13" сентября 2022

Дата защиты проекта "20" декабря 2022

Руководитель Шенников М.А.

Задание получил " " 2022 г.

Студент Игнатов Д.В.

Содержание

Реферат	5
Введение	6
Постановка задачи	7
Описание алгоритма программы	9
Описание программы	16
Тестирование	18
Ручной расчёт задачи	18
Заключение	19
Приложение А. Листинг программы	19

Реферат

Отчёт 24 стр, рисунков 4

Цель исследования - разработка программы, способной работать с большими наборами данных и составлять наборы данных, удовлетворяющие нескольким условиям.

В работе рассмотрены правила работы с массивами и циклами, составлен алгоритм, выполняющий эффективный перебор входных данных и составление выходных наборов данных, удовлетворяющих фиксированным требованиям.

Введение

В современном мире информационные технологии стали неотъемлемой частью жизни практически любого человека.

Снижение трудоёмкости любых производственных, ускорение социальных процессов и, как следствие, повышение их эффективности сейчас связано с существенно изменившимися за последние десятилетия возможностями сбора, обработки, хранения, передачи и представления информации.

Реализация полученной в результате этих процессов информации совершенно нового качества позволяет обеспечить процесс создания и использования материальных и духовных ценностей на более высоком и принципиально новом уровне.

Таким образом изучение современных информационных технологий обеспечивает возможность человеку без проблем ориентироваться в современных тенденциях и путях развития общества, процессах производства и повышать качество социальной жизни.

В таком мире современные информационные системы имеют всё большее значение и применение. Очень часто перед современным человеком встаёт задача обработки огромного количества входных данных и переработка их в меньшее количество выходных данных, что является если и выполнимой задачей при обработке вручную, то затрачивающей невероятное количество временных ресурсов. Чаще же человеку в целом не представляется возможным обработать все эти данные самому. Именно поэтому написание программ, способных работать с большим количеством информации, является очень важной задачей, упрощающей труд других людей.

Постановка задачи

В России насчитывается 25 819 366 детей в возрасте до 14 лет. Из них 13 263 223 мальчиков и 12 556 143 девочек.

Дед Мороз с помощниками весь год трудились над созданием игрушек для детей, у них получилось 70 685 034 различных предметов трёх категорий.

1. Игрушки для мальчиков - Легковой автомобиль, Грузовик, Пожарная машина, Самолёт, Вертолёт на радиоуправлении, Конструктор "Радиолюбитель".
2. Игрушки для девочек - Кукла, Игрушечный домик, Мягкая игрушка, Набор "Кухня", Детские украшения, Детская косметика.
3. Универсальные игрушки - Поп ит, Пластилин, Слайм, Набор "Выращиваем кристаллы", Книга, Фломастеры.

Количество игрушек ограничено, каждая игрушка имеет определённую стоимость и вес, представленные в таблице.

Таблица 1 - Характеристики игрушек

Название игрушки	Вес	Стоимость	Количество
Игрушки для мальчиков			
Легковой автомобиль	300	500	5843536
Грузовик	500	400	3983569
Пожарная машина	450	700	2434427
Самолёт	200	200	8564275
Вертолёт на радиоуправлении	350	1200	1768341
Конструктор «Радиолюбитель»	700	1000	945376
Игрушки для девочек			
Кукла	450	300	8159634
Игрушечный домик	800	1500	1759267
Мягкая игрушка	250	300	4264527
Набор «Кухня»	500	650	2874123

Название игрушки	Вес	Стоимость	Количество
Детские украшения	200	800	1689522
Детская косметика	200	400	3896345
Универсальные игрушки			
Поп ит	50	50	4952347
Пластилин	100	100	2589632
Слайм	50	50	3256876
Набор «Выращиваем кристаллы»	200	150	1958632
Книга	200	200	8589632
Фломастеры	100	100	3154973

Теперь из этих игрушек необходимо сформировать подарки для каждого из детей. Такой подарок должен отвечать определенным требованиям:

1. В подарках для мальчиков не должно быть игрушек для девочек.
2. В подарках для девочек не должно быть игрушек для мальчиков.
3. Вес подарка не должен превышать 1000 грамм.

Основная задача - написать программу, формирующую 25 819 366 подарков, отвечающих требованиям, имеющих максимальную суммарную стоимость. Количество каждого вида игрушек в подарках не должно превышать количества сделанных игрушек, указанных в таблице.

Список сформированных подарков необходимо сохранить в виде файла.

Формат файла:

1 столбец – номер подарка

2 столбец – стоимость подарка

3 столбец – вес подарка

и далее столбцы – название игрушки

Столбцы отделяются запятыми.

Описание алгоритма программы

Поскольку в условии задачи сказано, что игрушки для мальчиков могут быть только у мальчиков, также, как игрушки для девочек могут быть только у девочек, а общие доступны для всех, то сначала распределим игрушки, относящиеся к конкретному полу, а потом займёмся общими.

Поставим условие, что в каждом подарке для мальчика должна быть игрушка для мальчика, и, соответственно, в каждом подарке для девочки должна быть игрушка для девочки.

Ещё одним условием, которое будем проверять при распределении игрушек является то, что для добавления игрушки к сформированному подарку он не должен содержать эту игрушку, т.е. каждая игрушка может входить в подарок только один раз.

Начинаем распределять игрушки для мальчиков:

- Сначала сортируем список игрушек для мальчиков по убыванию количества игрушек и записываем верхнюю строку с игрушкой, которой больше всего, в список сформированных подарков, при этом установив количество этой игрушки в изначальном списке в 0 (так как этой игрушки больше всех остальных, то она будет самой часто встречающейся).

- Затем начинаем в цикле брать каждую следующую строку с игрушкой из списка игрушек и проверять, можно ли добавить эти игрушки в подарок и не нарушить условие с массой подарка ≤ 1000 грамм. Если можно, то добавляем новую игрушку в подарок, корректируя массу и стоимость, при этом если количество добавляемых игрушек меньше, чем количество игрушек в подарке, то строка в списке с подарками делится на две: в верхней остаётся одна изначальная игрушка в количестве, равном разнице её изначального количества и количества добавляемых игрушек, а в нижней строчке подарка содержатся уже изначальная и добавленная игрушки со скорректированными стоимостью и весом и в количестве, равном числу добавленных игрушек.

- Действуем так, пока не встречаем строчку с игрушкой, при распределении которой количество оставшихся игрушек будет меньше, чем количество оставшихся собрать подарков (количество собранных подарков равно сумме количеств из списка сформированных подарков). Когда встречаем эту строчку, то находим количество подарков, которое осталось сформировать (число мальчиков минус сумма сформированных подарков), и распределяем максимально допустимое значение игрушек из этой строчки (разность суммы оставшегося количества игрушек и оставшегося для формирования числа подарков) в соответствии с предыдущим пунктом. Это значение находится как разница количества всех оставшихся игрушек и количества оставшихся для формирования подарков. Оставшиеся строчки с игрушками записываем в новые строчки списка сформированных подарков.

В соответствии с данным алгоритмом все игрушки для мальчиков были распределены, количество сформированных подарков в списке соответственно равно количеству мальчиков - всё верно.

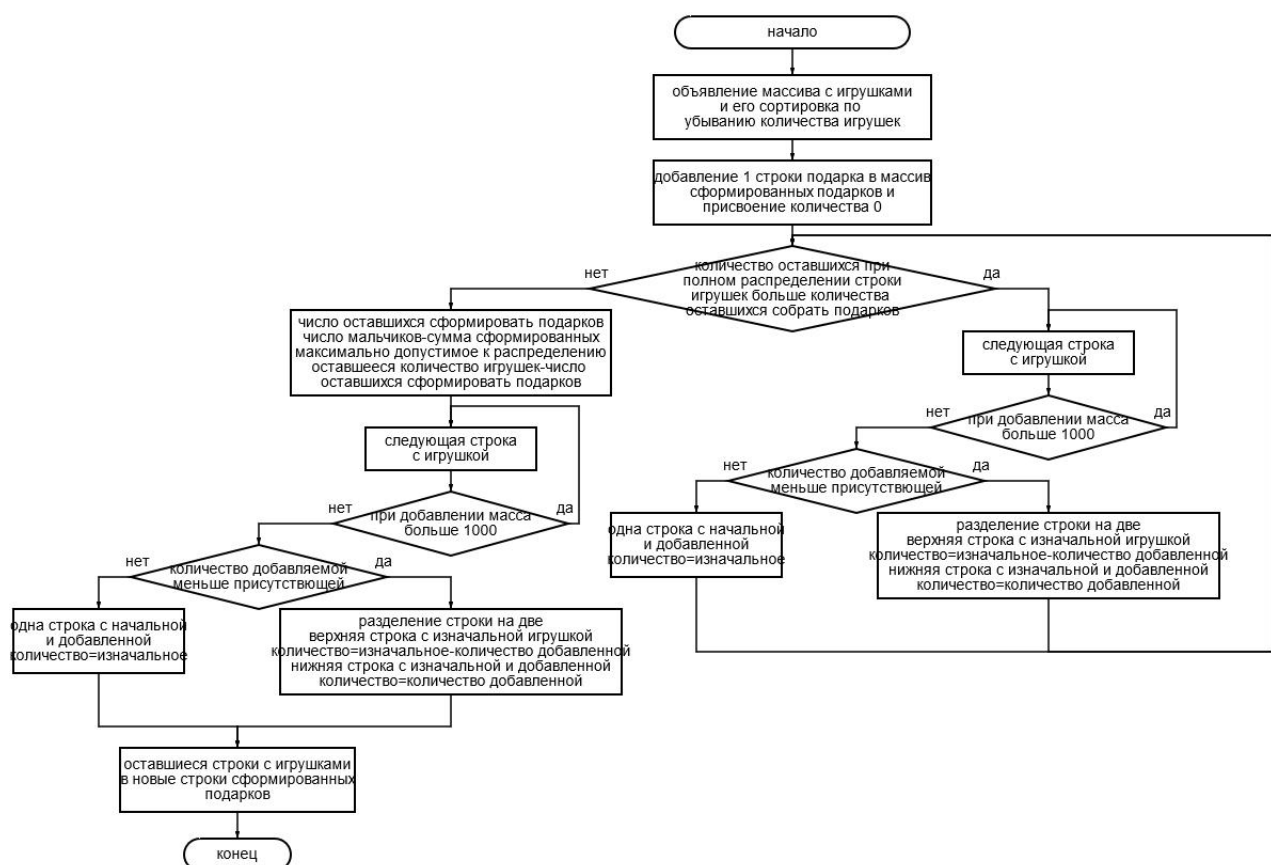


Рисунок 1 - Алгоритм распределения для мальчиков

Далее приступаем к распределению подарков для девочек. Алгоритм, используемый для распределения игрушек для мальчиков в данном случае не сработал, так как при переходе к шагу с частичным распределением максимально допустимого значения игрушки, при полном распределении которой наблюдается нехватка игрушек на подарки, эту игрушку распределить частично не представляется возможным из-за нарушения условия на ограничение массы подарка. Поэтому было принято решение использовать другой алгоритм, работающий с парами доступных для объединения в одном подарке игрушек.

Начинаем распределять игрушки для девочек:

- Из списка игрушек выбираем пары тех игрушек, которые можно объединить без нарушения условия массы. Эти пары игрушек записываем в список пар.

- Потом формируем отсортированный по убыванию значения стоимости список наборов и записываем туда сформированные из этих пар наборы, которые содержат количество наборов(минимальное число игрушек из двух объединяемых), массу(складывается масса объединяемых игрушек), цену(складывается цена объединяемых игрушек), названия объединяемых игрушек и стоимость(произведение минимального количества игрушек из объединяемых(так как объединяем по минимальному) на их общую цену). Далее берём набор с наибольшей стоимостью и записываем в список сформированных подарков, а также корректируем данные о количестве игрушек в исходном списке с игрушками и списке пар.

- Предыдущий пункт повторяем в цикле, пока в списке пар и списке наборов количества доступных объединений не станут равны 0, после этого подсчитываем количество нераспределённых игрушек и количество подарков, которое необходимо сформировать(количество девочек минус количество сформированных подарков), и сортируем список игрушек по убыванию цены.

- Затем в цикле пока количество сформированных подарков меньше необходимого добавляем в новые строчки списка сформированных подарков максимально допустимое количество нераспределённых игрушек. После этого в списке ещё остаются нераспределённые игрушки одного вида. Также выполняем сортировку списка подарков по убыванию цены.

- Далее выполняем поиск в сформированных подарках такого набора, который по стоимости количества нераспределённой игрушки(произведение цены этого подарка на количество нераспределённой игрушки) меньше, чем стоимость этой нераспределённой игрушки(произведение цены этой игрушки на её количество), и заменяем первый набор на второй, скорректировав количество игрушек в списке игрушек. После этого распределяем оставшиеся нераспределёнными игрушки по алгоритму для мальчиков(начинаем в цикле брать каждую строку с игрушкой, количество которой не равно 0, из списка игрушек и проверять, можно ли добавить эти игрушки в подарок и не нарушить условие с массой подарка ≤ 1000 грамм. Если можно, то добавляем новую игрушку в подарок, корректируя массу и стоимость, при этом если количество добавляемых игрушек меньше, чем количество игрушек в подарке, то строка в списке с подарками делится на две: в верхней остаётся одна изначальная игрушка в количестве, равном разнице её изначального количества и количества добавляемых игрушек, а в нижней строчке подарка содержатся уже изначальная и добавленная игрушки со скорректированными стоимостью и весом и в количестве, равном числу добавленных игрушек).

В соответствии с данным алгоритмом все игрушки для девочек были распределены, количество сформированных подарков в списке соответственно равно количеству девочек - всё верно.

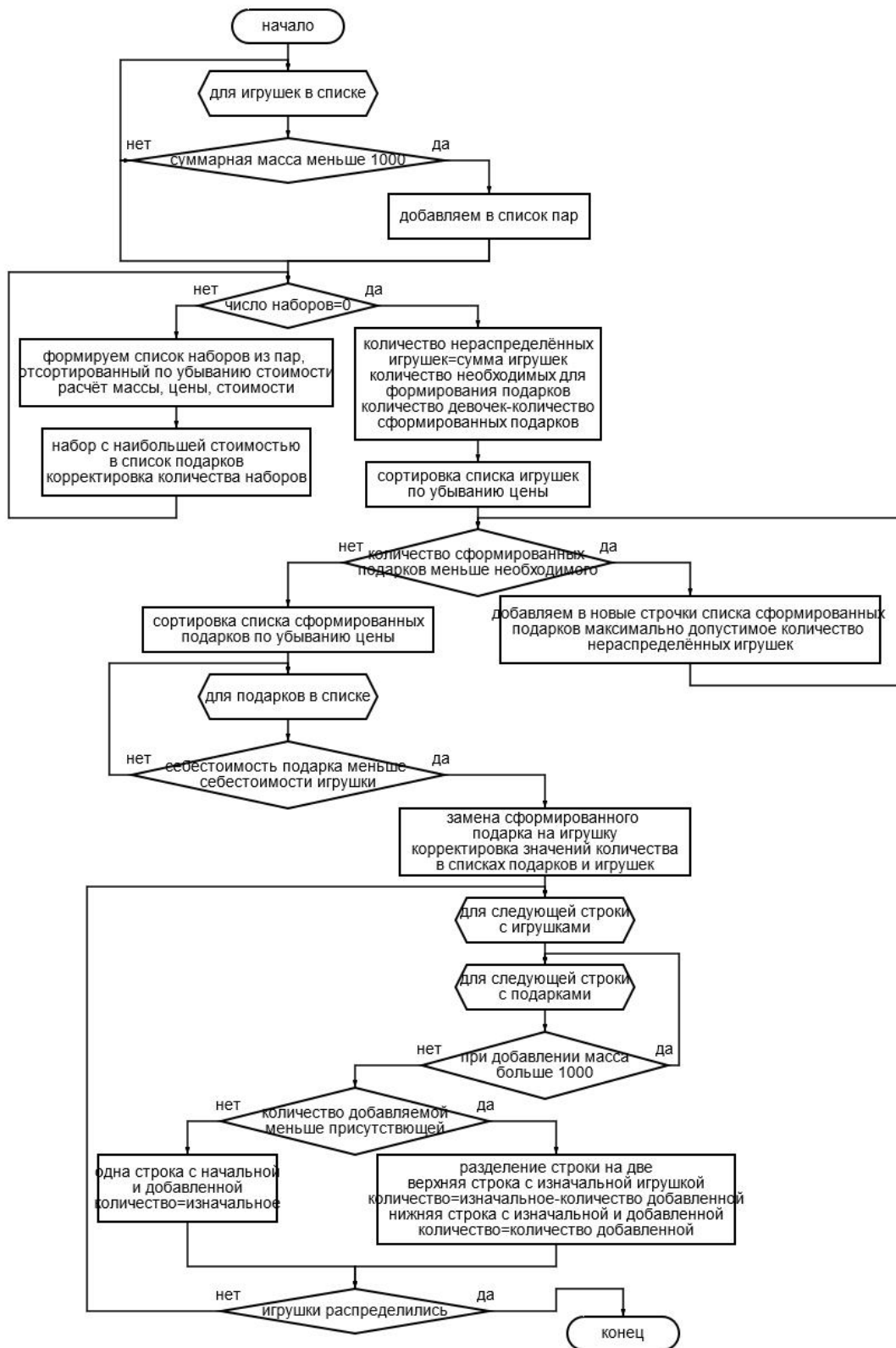


Рисунок 2 - Алгоритм распределения для девочек

Далее остаются универсальные игрушки, которые могут быть как у мальчиков, так и у девочек, поэтому для распределения объединяем списки

сформированных подарков для мальчиков и для девочек в один и сортируем его по увеличению цены подарка.

Начинаем распределять универсальные игрушки:

- Сначала сортируем список универсальных игрушек по убыванию цены.
- Затем начинаем в цикле брать каждую строку с игрушкой из списка игрушек и проверять, можно ли добавить эти игрушки в подарок и не нарушить условие с массой подарка ≤ 1000 грамм. Если можно, то добавляем новую игрушку в подарок, корректируя массу и стоимость, при этом если количество добавляемых игрушек меньше, чем количество игрушек в подарке, то строка в списке с подарками делится на две: в верхней остаётся одна изначальная игрушка в количестве, равном разнице её изначального количества и количества добавляемых игрушек, а в нижней строчке подарка содержатся уже изначальная и добавленная игрушки со скорректированными стоимостью и весом и в количестве, равном числу добавленных игрушек.

В соответствии с данным алгоритмом все универсальные игрушки были распределены, количество сформированных подарков в списке соответственно равно общему количеству детей - всё верно.

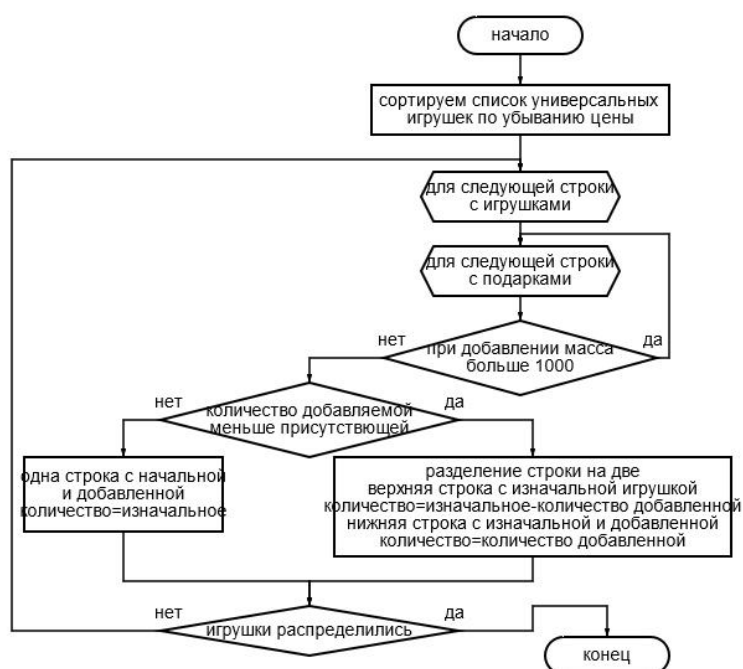


Рисунок 2 - Алгоритм распределения универсальных

Алгоритм программы выполняет полное распределение всех игрушек по подаркам и стремится достичь максимальной стоимости подарков. Универсальность данного алгоритма под вопросом, поскольку алгоритм для распределения подарков для девочек был разработан в течение отладки и получения промежуточных результатов, однако для текущего набора данных он полностью выполняет свою функцию.

Описание программы

Для решения задачи потребуются 3 двумерных массива, в которых будет содержаться информация о доступных игрушках, их весе, цене и оставшемся количестве. Необходимо составить 13263223 подарков для мальчиков и 12556143 подарков для девочек (по количеству человек) - суммарно 25819366 подарков. Эти данные также записываем, так как они понадобятся во время подсчётов и проверок.

Сначала выполняется алгоритм по распределению игрушек для мальчиков, после чего массив, являющийся результатом его работы и содержащий сформированные подарки сохраняется. Далее выполняется алгоритм по распределению игрушек для девочек, массив с результатами которого также сохраняется.

Далее массивы с результатами объединяются в один, к которому применяется алгоритм по распределению универсальных игрушек.

В результате итоговый массив содержит в себе количество каждого подарка, его стоимость, вес и состав(игрушки в нём). В соответствии с количеством каждого подарка, они построчно с нумерацией выводятся в файл.



Рисунок 2 - Алгоритм программы

Тестирование

Среда разработки Python 3.8.2 Shell предоставляет большинство средств, необходимых при разработке и отладке программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом и выводом данных, алгоритмом программы, взаимодействием циклов и массивов.

Ниже продемонстрирован результат тестирования программы.

```
===== исходные данные ===== RESTART: C:\Users\Данила\Downloads\newyearchallenge-antropov.py =====
-----
Мальчиков - 13263223
Девочек - 12556143
Надо всего подарков - 25819366

Игрушек для мальчиков - 23539524
Игрушек для девочек - 22643418
Игрушек для всех - 24502092
В среднем игрушек - 2.665687453363495
-----

объединенный список подарков отсортированный по уменьшению стоимости
[1689522, 2300, 1000, 'Игрушечный домик', 'Детские украшения']
[1238, 1900, 1000, 'Игрушечный домик', 'Детская косметика']
[68507, 1500, 800, 'Игрушечный домик']
[449196, 1400, 850, 'Самолет', 'Легковой автомобиль', 'Пожарная машина']
[1768341, 1200, 350, 'Вертолет на радиоуправлении']
[1262830, 1100, 900, 'Самолет', 'Легковой автомобиль', 'Грузовик']
[1460838, 1050, 950, 'Набор "Кухня"', 'Книга', 'Набор "Выращиваем кристаллы"', 'Слайд']
[380409, 1000, 900, 'Набор "Кухня"', 'Книга', 'Набор "Выращиваем кристаллы"']
[945376, 1000, 700, 'Конструктор "Радиолубитель"']
[236881, 1000, 900, 'Кукла', 'Детская косметика', 'Мягкая игрушка']
[2589632, 950, 950, 'Самолет', 'Грузовик', 'Книга', 'Пластелин', 'Поп-ит']
[131107, 950, 950, 'Самолет', 'Грузовик', 'Книга', 'Фломастеры', 'Поп-ит']
[236881, 950, 950, 'Набор "Кухня"', 'Кукла']
[117385, 900, 800, 'Набор "Кухня"', 'Набор "Выращиваем кристаллы"', 'Фломастеры']
[2231608, 850, 950, 'Кукла', 'Мягкая игрушка', 'Книга', 'Поп-ит']
[1796038, 850, 950, 'Кукла', 'Мягкая игрушка', 'Книга', 'Слайд']
[2227871, 800, 500, 'Самолет', 'Легковой автомобиль', 'Фломастеры']
[441729, 750, 600, 'Набор "Кухня"', 'Фломастеры']
[236881, 750, 600, 'Набор "Кухня"', 'Фломастеры']
[1903639, 700, 400, 'Самолет', 'Легковой автомобиль']
[1985231, 700, 450, 'Пожарная машина']
[3658226, 700, 650, 'Кукла', 'Детская косметика']

25819366
25819366

>>>
```

Рисунок 4 - Результат работы программы

Ручной расчёт задачи

Произвести ручной расчёт задачи в данном случае не представляется возможным в течение обозримого времени, так как подсчёт вручную займёт очень много времени и человеческих ресурсов. Предполагаемое время подсчёта для данного случая с помощью Microsoft Office Excel сравнимо с временем, затраченным на разработку алгоритма и написание программы. В случае увеличения видов игрушек время подсчёта возрастает кратно.

Заключение

Таким образом, в процессе создания данного курсового проекта разработана программа, выполняющая обработку большого количества данных в соответствии с заданными условиями и выводящая результаты в файл.

При данной курсовой работы были получены навыки разработки программ и освоены приёмы работы с большим количеством данных. Углублены навыки в программировании на языке Python.

Недостатками разработанной программы являются довольно длительное время выполнения и выходной файл большого размера, однако программа также выводит сжатую версию в консоль.

Программа имеет небольшой, но достаточный для выполнения поставленной задачи функционал, и, при необходимости, может быть отредактирована и реорганизована таким образом, чтобы её можно было применить к решению другой похожей задачи.

Приложение А. Листинг программы

```
def podarok():
    sp = []
    for i in range(len(tois)):
        for j in range(i+1, len(tois)):
            if (tois[i][1] + tois[j][1]) <= 1000:
                sp.append([min(tois[i][3], tois[j][3]), tois[i][2]+tois[j][2],
tois[i][1]+tois[j][1], tois[i][0], tois[j][0], i, j,
min(tois[i][3], tois[j][3])*(tois[i][1]+tois[j][1])])
    sp = sorted(sp, key=lambda st: (st[0], st[1]), reverse=True)
    if sp[0][0] > 0:
        make_present.append([sp[0][0], sp[0][1], sp[0][2], sp[0][3], sp[0][4]])
        tois[sp[0][5]][3] = tois[sp[0][5]][3] - sp[0][0]
        tois[sp[0][6]][3] = tois[sp[0][6]][3] - sp[0][0]
        if sum([row[0] for row in sp]) > 0:
            return True
    else:
        return False

b = 13263223
g = 12556143
b_and_g = b + g
```

```

b_tois = [['Легковой автомобиль', 300, 500, 5843536], ['Грузовик', 500, 400, 3983569],
['Пожарная машина', 450, 700, 2434427], ['Самолет', 100, 200, 8564275], ['Вертолет на
радиоуправлении', 350, 1200, 1768341], ['Конструктор "Радиолюбитель"', 700, 1000, 945376]]
b_tois_cору = [['Легковой автомобиль', 300, 500, 5843536], ['Грузовик', 500, 400, 3983569],
['Пожарная машина', 450, 700, 2434427], ['Самолет', 100, 200, 8564275], ['Вертолет на
радиоуправлении', 350, 1200, 1768341], ['Конструктор "Радиолюбитель"', 700, 1000, 945376]]
g_tois = [['Кукла', 450, 300, 8159634], ['Игрушечный домик', 800, 1500, 1759267], ['Мягкая
игрушка', 250, 300, 4264527], ['Набор "Кухня"', 500, 650, 2874123], ['Детские украшения',
200, 800, 1689522], ['Детская косметика', 200, 400, 3896345]]
g_tois_cору = [['Кукла', 450, 300, 8159634], ['Игрушечный домик', 800, 1500, 1759267],
['Мягкая игрушка', 250, 300, 4264527], ['Набор "Кухня"', 500, 650, 2874123], ['Детские
украшения', 200, 800, 1689522], ['Детская косметика', 200, 400, 3896345]]
o_tois = [['Поп ит', 50, 50, 4952347], ['Пластин', 100, 100, 2589632], ['Слайм', 50, 50,
3256876], ['Набор "Выращиваем кристаллы"', 200, 150, 1958632], ['Книга', 200, 200, 8589632],
['Фломастеры', 100, 100, 3154973]]
o_tois_cору = [['Поп ит', 50, 50, 4952347], ['Пластин', 100, 100, 2589632], ['Слайм', 50,
50, 3256876], ['Набор "Выращиваем кристаллы"', 200, 150, 1958632], ['Книга', 200, 200,
8589632], ['Фломастеры', 100, 100, 3154973]]
print('----- исходные данные -----')
print('Мальчиков          - ', b)
print('Девочек           - ', g)
print('Надо всего подарков - ', b_and_g)
print('')
print('Игрушек для мальчиков - ', sum([row[3] for row in b_tois]))
print('Игрушек для девочек  - ', sum([row[3] for row in g_tois]))
print('Игрушек для всех     - ', sum([row[3] for row in o_tois]))
print('В среднем игрушек    - ', (sum([row[3] for row in b_tois])+sum([row[3] for row in
g_tois])+sum([row[3] for row in o_tois]))/b_and_g)
print(33*'-')
p = input('')

make_present = []

tois = b_tois.copy()
tois = sorted(tois, key=lambda kolvo: kolvo[3], reverse=True)
make_present.append([tois[0][3], tois[0][2], tois[0][1], tois[0][0]])
tois[0][3] = 0

konec = False
for i in range(1, len(tois)):
    if konec:
        break
    if (b - sum([row[0] for row in make_present])) >= (sum([row[3] for row in tois]) -
tois[i][3]): # если при распределении очередной группы игрушек их станет меньше чем осталось
собрать подарков
        for j in range(len(make_present)):

```

```

        if (tois[i][3] > 0) and ((tois[i][1] + make_present[j][2]) < 1000) and
(tois[i][0] not in make_present[j]): # будем добавлять игрушку в подарок если общий вес
нового подарка не превышает и этой игрушки в подарке еще нет
            podar = sum([row[0] for row in make_present]) + sum([row[3] for row in
tois]) - b

            if make_present[j][0] < podar:
                make_present.insert(j+1, [make_present[j][0], make_present[j][1] +
tois[i][2], make_present[j][2] + tois[i][1]]) # добавляем новый подарок без названия игрушек
                for el in (3, len(make_present[j])-1): # добавляем в этот подарок
названия игрушек которые были
                    make_present[j+1].append(make_present[j][el])
                make_present[j+1].append(tois[i][0]) # добавляем в этот подарок
название игрушки которую включили в подарок
                make_present.pop(j) # удаляем тот подарок который разделяли
                tois[i][3] = tois[i][3] - make_present[j][0] # уменьшаем кол-во
нераспределенных игрушек
            else:
                make_present.insert(j+1, [podar, make_present[j][1] + tois[i][2],
make_present[j][2] + tois[i][1]]) # добавляем новый подарок без названия игрушек
                for el in (3, len(make_present[j])-1): # добавляем в этот подарок
названия игрушек которые были
                    make_present[j+1].append(make_present[j][el])
                make_present[j+1].append(tois[i][0]) # добавляем в этот подарок
название игрушки которую включили в подарок
                make_present[j][0] = make_present[j][0] - podar # изменяем кол-во
игрушек в исходном подарке
                tois[i][3] = tois[i][3] - podar # изменяем количество нераспределенных
игрушек

            for el in range(len(tois)):
                if (tois[el][3] > 0):
                    make_present.append([tois[el][3], tois[el][2], tois[el][1],
tois[el][0]]) # добавляем новый подарок
                    tois[el][3] = 0 # уменьшаем количество нераспределенных игрушек
                konec = True
                break
        else:
            for j in range(len(make_present)):
                if (tois[i][3] > 0) and ((tois[i][1] + make_present[j][2]) < 1000) and
(tois[i][0] not in make_present[j]): # кол-во игрушек в строке > 0, общий вес не превышает 1
кг, добавляемой игрушки нет в этом подарке
                    if make_present[j][0] < tois[i][3]:
                        make_present[j][1] = make_present[j][1] + tois[i][2] # плюсуем
стоимость

                        make_present[j][2] = make_present[j][2] + tois[i][1] # плюсуем вес
                        make_present[j].append(tois[i][0]) # добавляем название игрушки
которую включили в подарок

```

```

        tois[i][3] = tois[i][3] - make_present[j][0] # уменьшаем количество
нераспределенных игрушек
    else:
        p = make_present[j].copy()
        p[0] = tois[i][3] # изменяем количество
        p[1] = p[1] + tois[i][2] # плюсуем стоимость
        p[2] = p[2] + tois[i][1] # плюсуем вес
        p.append(tois[i][0]) # добавляем название игрушки которую включили в
подарок

        make_present.insert(j+1, p) # добавляем первый подарок
        make_present[j][0] = make_present[j][0] - tois[i][3]
        tois[i][3] = 0 # уменьшаем количество нераспределенных игрушек

bp = make_present.copy()
make_present.clear()
tois = g_tois.copy()

cicle = True
while cicle:
    cicle = podarok()

while (g-sum([row[0] for row in make_present]) > 0):
    tois = sorted(tois, key=lambda cena: cena[2], reverse=True)
    for i in range (len(tois)):
        if tois[i][3] > 0:
            if tois[i][3] < (g-sum([row[0] for row in make_present])):
                make_present.append([tois[i][3], tois[i][2], tois[i][1], tois[i][0]])
                tois[i][3] = 0
            else:
                tois[i][3] = tois[i][3] - (g-sum([row[0] for row in make_present]))
                make_present.append([g-sum([row[0] for row in make_present]), tois[i][2],
tois[i][1], tois[i][0]])
        if (sum([row[3] for row in tois]) > 0):
            make_present = sorted(make_present, key=lambda st: st[1], reverse=True)
            for i in range(len(tois)):
                if tois[i][3] > 0:
                    for j in range(len(make_present)):
                        if (tois[i][2] >= make_present[j][1] and (len(make_present[j]) > 4):
                            make_present.append([tois[i][3], tois[i][2], tois[i][1], tois[i][0]])
                            n = tois[i][3]
                            make_present[j][0] = make_present[j][0] - tois[i][3]
                            tois[i][3] = tois[i][3] - tois[i][3]
                            for i in range(len(tois)):
                                if tois[i][0] in make_present[j][3:]:
                                    tois[i][3] = n
                            make_present = sorted(make_present, key=lambda st: st[1], reverse=True)

```

```

        for i in range(len(tois)):
            if tois[i][3] > 0:
                for j in range(len(make_present)):
                    if (make_present[j][0] >= tois[i][3]) and
((make_present[j][2] + tois[i][1]) < 1000) and (tois[i][0] not in make_present[j]):
                        make_present.insert(j+1, make_present[j].copy())
                        make_present[j][0] = make_present[j][0] - tois[i][3]
                        make_present[j+1][0] = tois[i][3]
                        make_present[j+1][1] = make_present[j+1][1] +
tois[i][2]
                        make_present[j+1][2] = make_present[j+1][2] +
tois[i][1]
                        make_present[j+1].append(tois[i][0])
                        tois[i][3] = tois[i][3] - tois[i][3]
                        break

                make_present = sorted(make_present, key=lambda st: st[1], reverse=True) #сортируем по
убыванию количества игрушек
                gp = make_present.copy()

                op = bp + gp
                op = sorted(op, key=lambda st: st[1]) #сортируем по увеличению стоимости

                o_tois = sorted(o_tois, key=lambda st: st[2], reverse=True) #сортируем по убыванию
СТОИМОСТИ

                while sum([row[3] for row in o_tois]) > 0:
                    for i in range(len(o_tois)):
                        for j in range(len(op)):
                            if ((op[j][2] + o_tois[i][1]) < 1000) and (o_tois[i][0] not in op[j]) and
(o_tois[i][3] > 0):
                                if op[j][0] < o_tois[i][3]:
                                    op[j][1] = op[j][1] + o_tois[i][2]
                                    op[j][2] = op[j][2] + o_tois[i][1]
                                    op[j].append(o_tois[i][0])
                                    o_tois[i][3] = o_tois[i][3] - op[j][0]
                                else:
                                    op.insert(j+1, op[j].copy())
                                    op[j].append(o_tois[i][0])
                                    op[j][0] = o_tois[i][3]
                                    op[j][1] = op[j][1] + o_tois[i][2]
                                    op[j][2] = op[j][2] + o_tois[i][1]
                                    op[j+1][0] = op[j+1][0] - o_tois[i][3]
                                    o_tois[i][3] = o_tois[i][3] - o_tois[i][3]
                            if o_tois[i][3] == 0:
                                break

```

```

op = sorted(op, key=lambda st: st[1], reverse=True) #сортируем по уменьшению стоимости
print('объединенный список подарков отсортированный по уменьшению стоимости')
print(*op, sep='\n')
p = input('')
print(sum([row[0] for row in op]))
print(b_and_g)

```

```

with open("test.txt", "w") as f:
    f.write('Номер, Стоимость, Вес')
    f.write('\n')
    n = 1
    for j in range (len(op)):
        for k in range(op[j][0]):
            out = str(n)+' , '+str(op[j][1:]).lstrip('[').rstrip(']').replace('"', '')
            f.write(out)
            f.write('\n')
            n = n+1

```