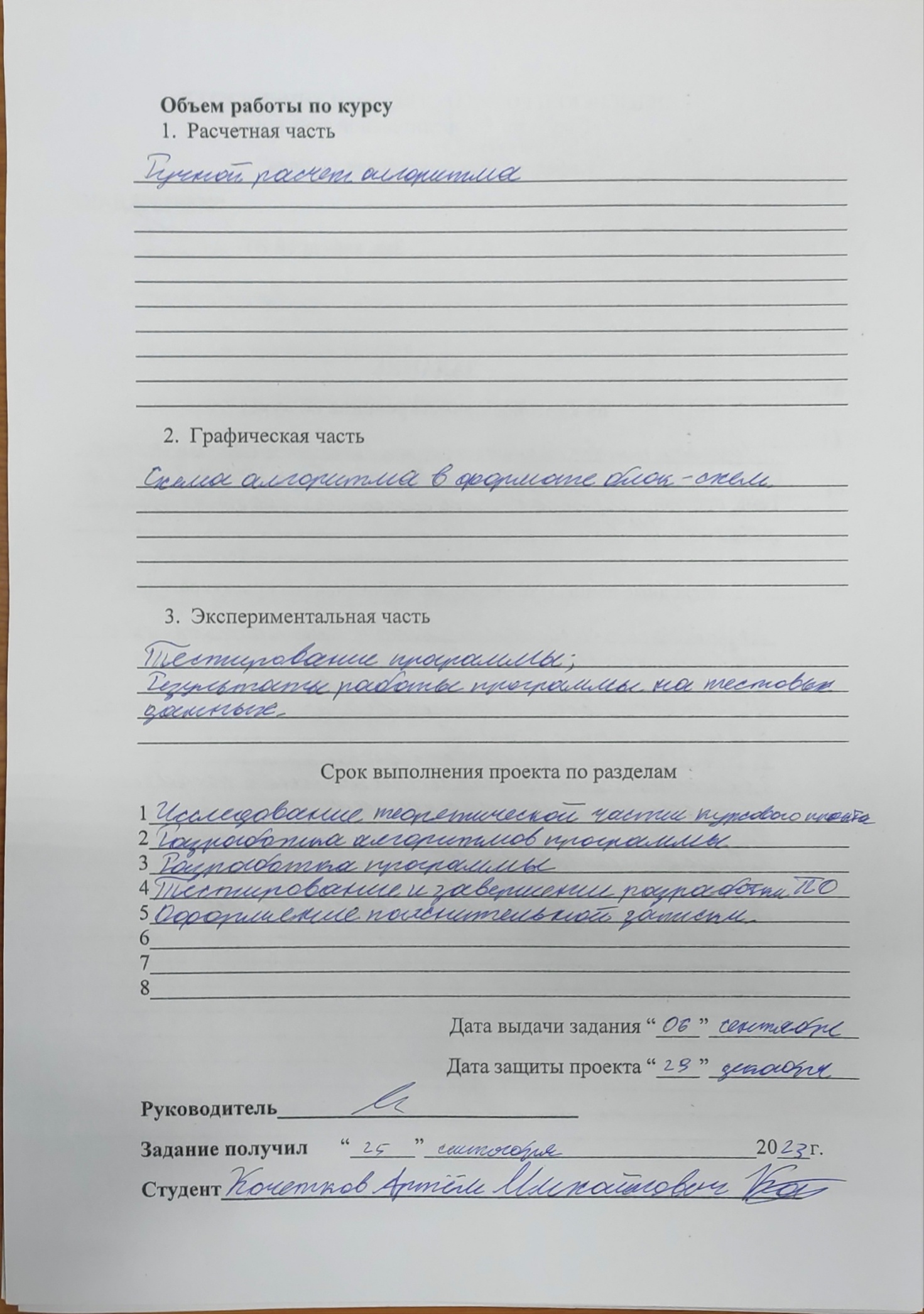
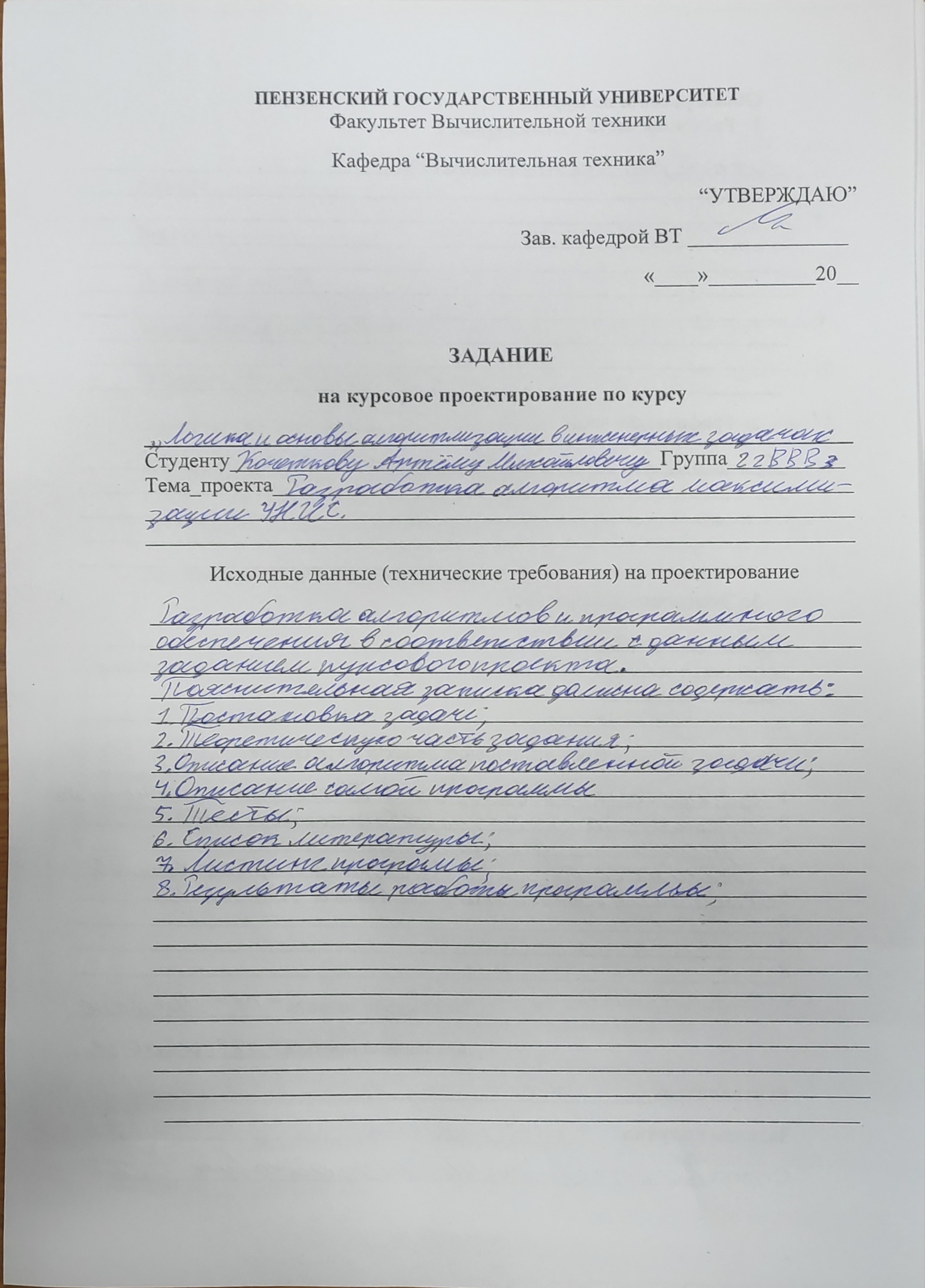


****



**Содержание**

Реферат ............................................................................................................... 6

Введение............................................................................................................. 7

Постановка задачи............................................................................................. 8

Теоретическая часть задания ........................................................................... 9

Описание алгоритма программы..................................................................... 10

Описание программы....................................................................................... 13

Тестирование .................................................................................................... 15

Заключение ....................................................................................................... 17

Приложение A. Листинг программы ............................................................. 18

**Реферат**

Отчет 17стр, 6 рисунков.

РАЗРАБОТКА АЛГОРИТМА МАКСИМИЗАЦИИ УНИС

Цель исследования – разработка программы, способной создавать выходной файл на основе набора входных данных изаранеезаданныхограничений и правил наиболее эффективным образом.

В работе рассмотрены правила подсчета индекса УНИС. Установлено, что с помощью данного алгоритма можно получить корректные выходные данные с оптимальным итоговым индексом.

**Введение**

Разработка алгоритма была реализована в ограниченные сроки из-за чего необходимо было соблюсти баланс между наибольшим финальным значением УНИС и нормальной скоростью выполнения алгоритма.

В качестве среды разработки мною была выбрана среда разработкиMicrosoftVisualStudio, язык программирования - C++.

Целью данной курсовой работы является разработка программы на языкеC++, который является широко используемым. Именно с его помощью в данной работе реализуется алгоритм.

**Постановка задачи**

Требуется разработать программу, которая анализирует два входных csvфайла и составляет выходной файл (пары idребенка,id подарка). Значение УНИС, которое считается по выходному файлу должно быть как можно больше.

Исходный файлkids\_wish.csv содержат в себе184000 последовательных idдетей с последующими idподарков в количестве 100 штук на каждого ребенка; файл ded\_moroz\_wish.csvсодержит 1000 последовательных idподарков с последующими 184 idдетей для каждого подарка. Наиболее приоритетные idрасполагаются левее (имеют наименьший номер столбца).

Данные представлены в формате .csv. Алгоритм должен правильно составлять выходной файл, учитывая заданные ограничения и, по возможности, время выполнения алгоритма должно находится в разумных пределах.

Устройство ввода – клавиатура, мышь.

**1 Теоретическая часть**

Файлkids\_wish.csvсодержит список желаемых подарков детей, его можно представить как граф, в котором IDребенка (значение в первом столбце для каждой строчки) – есть вершина, которая должна быть соединена с другой вершиной являющейся ID подарка. Вес ребра между двумя вершинами будет определятся номером столбца IDподарка (среди 100 других подарков) относительно строчки с ID ребенка. Аналогично можно представить файл ded\_moroz\_wish.csv. Тогда задача сводится к созданию графа, где для каждой вершины с IDребенка будет смежно одно ребро, причем сумма веса этих ребер должна быть максимальной.

УНИС (Усредненный Нормализованный Индекс Счастья) рассчитывается как УНИСД (Усредненный Нормализованный Индекс Счастья Детей) + УНИСДМ (Усредненный Нормализованный Индекс Счастья Деда Мороза).  
УНИСД есть сумма всех значений индекса счастья детей, деленное на количество детей. Счастье ребенка рассчитывается по формуле:“СчастьеРебенка” = 2\*(100 - №столбцаПодаркаИзСписка). Если подарок отсутствует в списке ребенка, то “СчастьеРебенка” принимает значение = -1.

УНИСДМ– сумма значения “СчастьеДедаМороза” для каждого подарка, деленная на количество подарков. “СчастьеДедаМороза” рассчитывается по формуле: “СчастьеДедаМороза” = 2\*(184 - №столбцаРебенкаИзСписка). Если ребенок отсутствует в списке, то “СчастьеДедаМороза” =-1.

Дополнительные условия заключались в наличии близнецов: дети, с id = [0;5519] должны получить попарно одинаковые подарки. Также количество каждого вида подарка ограничено 250 штуками.

**2 Описание алгоритма программы**

Принцип работы алгоритма заключается в последовательном считывании каждой отдельной строки из файла kids\_wish.csv. Из этой строки берется 100 значений IDподарков, каждый из них проверяется на ряд условий.Выбранный IDподарка проверяется по массиву GiftCounter[gift], дабы удостовериться, что подарок не был использован 250 раз. Первый из 100 подарков, который удовлетворит этому условию (если таковой находится в строке), сохраняется в переменную best\_gift, так как он имеет наибольший приоритет (имеет наименьший номер столбца по сравнению с последующими) и, как следствие даст наибольший показатель “СчастьеРебенка”. Оставшиеся подарки в строке, которые проходят проверку по массиву GiftCounter[gift], сравниваются со строкой из файла ded\_moroz\_wish.csv. Номер строки равен ID текущего подарка. Если среди первых 100 значений из этой строки происходит совпадение с IDтекущего ребенка, то значение best\_gift меняется на IDподарка, для которого произошло совпадение.

После проверки всех 100 подарков для данной строки в выходной файл для данного IDребенка записывается значение IDподарка из переменной best\_gift. Если ни один из 100 подарков не подошел по условиям, то выбирается первый IDподарка из диапазона [0;1000], который удовлетворяет условию проверки через массив GiftCounter[gift] (Т.е. любой подарок, который возможно записать в файл, не нарушив условия задачи).

Для детей с ID< 5520, т.е. для близнецов, происходит аналогичный выбор IDподарка, но без проверки с файлом ded\_moroz\_wish.csv. После нахождения лучшего подарка из 100 подарков для ребенка с ID = n, находится лучший подарок для ребенка с ID = n+1. Номера столбцов этих подарков сохраняются в переменные, они сравниваются и по итогу для детей с IDnи n+1 присваивается подарок, который имеет меньший номер столбца, т.е имеет больший приоритет.

Выбранный idподарка записывается в файл вместе с id ребенка, далее считывается следующая строчка входного файла kids\_wish.csv и алгоритм повторяется до EOF.

Ниже представлен псевдокод функции TempIn(), которая отвечает за создание конечного выходного файла.

**TempIn(string line, unsigned short int\* giftCounter, vector<int>child\_wish\_list)**

1. Открывается 3 рабочих файла для записи/ чтения. Вызывается функция ded\_copy(), которая копирует первые 100 столбцов всех строчек файла ded\_moroz\_wish.csv, после этого работа с ним прекращается.
2. Пока в файле kids\_wish.csv имеется строчка с значениями (не EOF), выполнять:

3. Считывается idребенка и вся строчка преобразуется в вектор для дальнейшего использования.

4. Если idребенка < 5520, то.

5. Записать строчку для id+1 в другой вектор.

6. Цикл, пока cycling == true.

7. Цикл, проходящий через все 100 подарков ребенка

8. Если подарок был использован < 250раз

9. Если idребенка < 5520, то.

10. Выбирается лучший подарок из 100 подарков для

текущего ChildId и ChildId2.

11. Иначе (еслиidребенка > 5520 )

12. Каждый подарок проверяется на наличие в children[][],

сравнивается УНИС и если он больше, чем ранее

выбранная опция для подарка, то выбирается новый.

(Если иной опции не было найдено, присваивается   
 подарок, полученный на входе в цикл на шаге 8.)

13. Выбранный подарок gift записывается в выходной файл

giftCounter[gift]++;

cycling = false;

14. Если подарок был использован >= 250 раз

15. Если все 100 подарков в векторе были просмотрены и

текущий был использован >250 раз, то выбирается

первый свободныйподарок (random\_gift)

16. Если все 100 подарков в векторе были просмотрены и

текущий был использован <250 раз, то присваивается

именноон.

giftCounter[gift]++;

cycling = false;

17.Если текущий подарок не последний в списке из 100

подарков для этого ребенка, то проверяется следующий

подарок.

TakenIndex++;

18. Очищаетсявекторchild\_wish\_list.

**3 Описание программы**

Для написания данной программы использован языкC++. Данный язык был выбран из-за приспособленности работы нашей команды с C-образными языками и его скорости в сравнении с интерпретируемыми языками.

Данная программа является единичным файлом .cppсодержащим функции для обработки данных, их записи и считывания в угоду производительности и простоте написания/ редактирования кода.

После запуска исполняемого файла алгоритм начинает работу по анализу всех значений из файлов формата .csv

Чтобы иметь представление о правильности работы алгоритма, непосредственно во время его работы, была написана функция, выводящая прогресс, в текущий момент времени в процентом виде.

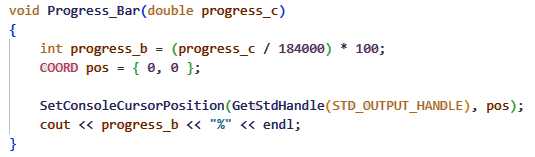
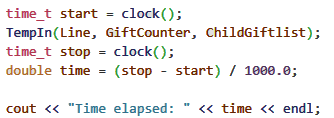


Рисунок 1 – Функция Progress\_Bar

Также после успешного завершения программы в консоль выводится затраченное время в секундах.

  
Рисунок 2 – Реализация подсчета времени работы

В начале работы программы содержимое файла ded\_moroz\_wish.csv записывается в локальную переменную в виде вектора векторов. Тестирование показало, что обращение к локальной переменной уменьшает время работы программы значительно в сравнении с постоянным обращением к файлу и поиску значений в нем.

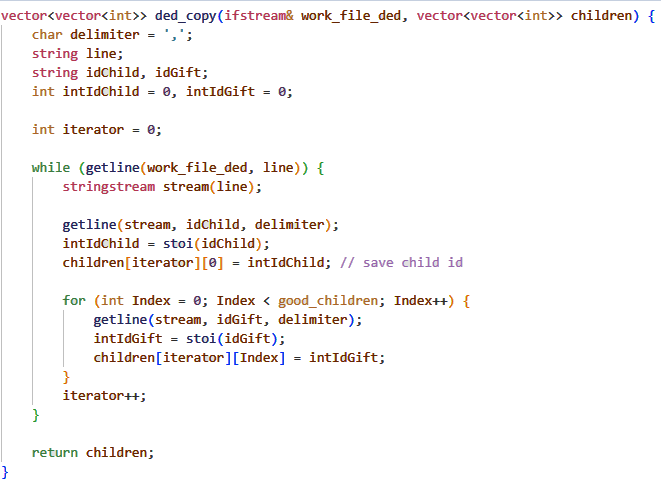


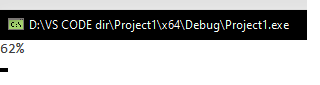
Рисунок 3 – Реализация функции по копировании данных из файла в двумерный массив

**4 Тестирование**

Тестирование программы во время нашей разработки было разделено на два этапа: 1) откладка самого кода, дабы достичь его работоспособность; 2) Проверка правильности выполнения алгоритма, путем анализа выходного файла.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения.

Для проверки выходного файла, использовался предоставленный в репозитории код python check\_submission.py.

  
Рисунок 4– Процесс работы программы

Время работы программы составляет от ~10 минут (ПК №1) до 5,5 минут (ПК №2) на машине с более мощным железом. Корректность работы определяется по изменения progressbar’а в консоли.

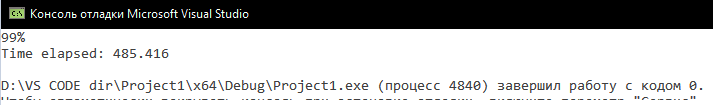
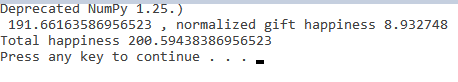


Рисунок 5 – Результат работы программы

Проверка через скрипт на Python, дает положительный результат и выводит УНИС в 200 единиц

  
Рисунок 6 – Результат расчета УНИС

В результате тестирования было выявлено, что программа успешно генерирует выходной файл согласно всем ограничениям и требованиям ТЗ.

**Заключение**

Таким образом, в процессе создания данного проекта разработанапрограмма, реализующая анализ двух табличных файлов, их сопоставление и генерацию выходного файла по заранее заданным правилам и ограничениям.

При выполнении данной курсовой работы были получены навыки

разработки программ и освоены приемы оптимизации кода и работы с большими файлами.

Углублены знания языка программирования C++.

**Приложение А.**

**Листинг программы.**

**Файлconcur.cpp**

//Выполнено командой Кочетков А.М., Бормотов А.А. 22ВВВ3

#include <iostream>

#include <sstream>

#include <fstream>

#include <vector>

#include <windows.h>

#define kids "kids\_wish.csv"

#define ded "ded\_moroz\_wish.csv"

#define result "result.csv"

#define stockList\_size 1000

#define wishList\_size 100

#define dedList\_size 184

#define maxGiftCount 250

#define good\_children 110

using namespace std;

void Progress\_Bar(double progress\_c)

{

int progress\_b = (progress\_c / 184000) \* 100;

COORD pos = { 0, 0 };

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), pos);

cout<<progress\_b<< "%" <<endl;

}

vector<vector<int>>ded\_copy(ifstream&work\_file\_ded, vector<vector<int>> children) {

char delimiter = ',';

string line;

string idChild, idGift;

int intIdChild = 0, intIdGift = 0;

int iterator = 0;

while (getline(work\_file\_ded, line)) {

stringstream stream(line);

getline(stream, idChild, delimiter);

intIdChild = stoi(idChild);

children[iterator][0] = intIdChild; // save child id

for (int Index = 0; Index <good\_children; Index++) {

getline(stream, idGift, delimiter);

intIdGift = stoi(idGift);

children[iterator][Index] = intIdGift;

}

iterator++;

}

return children;

}

void TempIn(string line, unsigned short int\* giftCounter, vector<int>child\_wish\_list)

{

double progress\_counter = 0;

ifstreamwork\_file\_kids(kids);

ofstreamwork\_file\_result(result);

char limiter = ',';

char delimiter = ',';

int intIdGift = 0;

int intIdChild = 0;

//mine

int present\_option = 0;

int takenIndex\_2 = 0;

int intIdChild\_2 = 0;

vector<int> child\_wish\_list\_2;

int gift\_pos\_1 = 0, gift\_pos\_2 = 0;

int temp\_gift = 0;

//mine for ded

ifstreamwork\_file\_ded(ded);

work\_file\_result<< "ChildId" << limiter << "GiftId" <<endl;

vector<vector<int>>children(1000, vector<int>(good\_children));

children = ded\_copy(work\_file\_ded, children);

work\_file\_ded.close();

//going through every str

while (getline(work\_file\_kids, line))

{

stringstream stream(line);

string idGift;

string idChild;

bool cycling = true;

//index to iterate through gifts that are given to child

int takenIndex = 0;

int takenGiftCounter = 0;

//taking only child id from the str

getline(stream, idChild, delimiter);

intIdChild = stoi(idChild);

for (int Index = 0; Index <wishList\_size; Index++)

{

//storing all desired gifts id's into array

getline(stream, idGift, delimiter);

intIdGift = stoi(idGift);

//intIdChild = stoi(idChild);

child\_wish\_list.push\_back(intIdGift);

}

///

string idChild\_2;

if (intIdChild< 5520) {

//take the next line (second sibling)

getline(work\_file\_kids, line);

stringstream stream\_2(line); //convert it

getline(stream\_2, idChild\_2, delimiter); // we have to do it to cut off the id from the string of 100 present id's

intIdChild\_2 = stoi(idChild\_2);

for (int Index = 0; Index <wishList\_size; Index++) {

//storing all desired gifts id's into array

getline(stream\_2, idGift, delimiter);

intIdGift = stoi(idGift);

child\_wish\_list\_2.push\_back(intIdGift);

}

}

///

int random\_gift = 0;

//giving gifts to children

while (cycling)

{

bool best\_taken = false;

int best\_gift = -1;

for (int Index = 0; Index <wishList\_size; Index++)

{

int gift = child\_wish\_list[takenIndex];

bool suitable = 0;

gift\_pos\_1 = Index;

//if gift is in child's wishlist,

//then check if there is this type of gift in stock

//if there is, give it to child

//if the gift is not in wishlist,

//but the gifts that are in it are out of stock,

//give child this gift

if (giftCounter[gift] <maxGiftCount)

{

if (intIdChild< 5520) {

for (int Index\_2 = 0; Index\_2 <wishList\_size; Index\_2++) {

gift\_pos\_2 = Index\_2;

present\_option = child\_wish\_list\_2[takenIndex\_2];

if (giftCounter[present\_option] <maxGiftCount) {

//fuck

if (gift\_pos\_1 <= gift\_pos\_2) break;

else {

gift = present\_option;

break;

}

}

if (giftCounter[present\_option] >= maxGiftCount) {

break;

}

takenIndex\_2++;

}

}

else {

if (!best\_taken) {

best\_gift = gift; // saving the first (left most) gift suitable for this kid

best\_taken = true;

}

for (int j = 0; j <good\_children; j++) {

if (children[gift][j] == intIdChild) {

best\_gift = gift; // change the gift to better option (the gift IN the kid list AND IN ded list)

suitable = true;

break;

}

}

if (suitable == false &&takenIndex<wishList\_size - 1) {

takenIndex++;

continue;

}

takenIndex++;

gift = best\_gift;

}

work\_file\_result<<intIdChild<< limiter << gift <<endl;

if (intIdChild< 5520) {

work\_file\_result<< intIdChild\_2 << limiter << gift <<endl;

giftCounter[gift]++;

progress\_counter++;

}

//counting how much of certain type of gift is left

giftCounter[gift]++;

progress\_counter++;

cycling = false;

break;

}

if (giftCounter[gift] >= maxGiftCount)

{

//optimisation?

if (takenIndex>= wishList\_size - 1 &&giftCounter[random\_gift] >= maxGiftCount) {

while (giftCounter[random\_gift] >= maxGiftCount) {

random\_gift++;

}

}

//

if (takenIndex>= wishList\_size - 1 &&giftCounter[random\_gift] <maxGiftCount)

{

if (intIdChild< 5520) {

gift\_pos\_1 = 0;

for (int Index\_2 = 0; Index\_2 <wishList\_size; Index\_2++) {

gift\_pos\_2 = Index\_2;

present\_option = child\_wish\_list\_2[takenIndex\_2];

if (giftCounter[present\_option] <maxGiftCount) {

//fuck

if (gift\_pos\_1 <= gift\_pos\_2) break;

else {

//random\_gift = present\_option;

temp\_gift = present\_option;

break;

}

}

if (giftCounter[present\_option] >= maxGiftCount) {

break;

}

takenIndex\_2++;

}

}

if (intIdChild>= 5520) temp\_gift = random\_gift; // put this in else{}

if (best\_taken)

{

temp\_gift = best\_gift;

}

work\_file\_result<<intIdChild<< limiter <<temp\_gift<<endl;

if (intIdChild< 5520) {

work\_file\_result<< intIdChild\_2 << limiter <<temp\_gift<<endl;

giftCounter[temp\_gift]++;

progress\_counter++;

}

//counting how much of certain type of gift is left

giftCounter[temp\_gift]++;

progress\_counter++;

cycling = false;

break;

}

//if there isn't, check other gift

if (takenIndex<wishList\_size - 1)

{

takenIndex++;

}

}

//silly progress bar :3

Progress\_Bar(progress\_counter);

}

}

//clear previous child gifts id's data

child\_wish\_list.clear();

}

work\_file\_kids.close();

}

int main()

{

unsigned short int GiftCounter[stockList\_size] = {};

vector<int>ChildGiftlist;

string Line;

time\_t start = clock();

TempIn(Line, GiftCounter, ChildGiftlist);

time\_t stop = clock();

double time = (stop - start) / 1000.0;

cout<< "Time elapsed: " << time <<endl;

}