Министерство образования и науки Российской Федерации

Севастопольский Государственный Университет

Кафедра ИС

Отчет по учебной практике

Выполнил: ст. гр. ИТб-11

Куркчи А.Э.

Проверил:

Сметанина Т. И.

Севастополь

2015

1.Введение

Практика была пройдена в Севастопольском государственном университете. Во время практики были выполнены олимпиадные задачи по таким темам: комбинаторика, битовые операции, алгоритм Евклида, частичная сумма, бинарное возведение в степень, динамическое программирование, поиск в глубину.

2.ВАРИАНТЫ ЗАДАНИЙ

1. 903 Первая или последняя?
2. 926 Формула Герона
3. 927 Количество игрушек
4. 1325 Васькины дорожки
5. 1326 В хоккей играют настоящие
6. 7336 Пирожки
7. 7337 Скидки
8. 388 Превращение
9. 488 Обход матрицы: обход «змейкой»
10. 2385 Количество перестановок
11. 2806 Числа
12. 1286 Школьный буфет
13. 390 Анаграммы
14. 1289 Ланч
15. 5716 Карточные домики
16. 124 Количество числа N
17. 5320 Дополнительный код-1
18. 5321 Дополнительный код-2
19. 1612 Изменение на единицу
20. 1355 Количество N-значных чисел, содержащих цифру 7
21. 1364 Магический квадрат
22. 137 НОД
23. 6941 Сумма НОД
24. 2214 Функция 9
25. 1033 Торт от Толи
26. 136 Отрезок
27. 1148 НОД НОК
28. 1244 НОД и НОК
29. 2327 Сортировка подсчетом
30. 480 Возведение в степень-2
31. 1121 A^B mod C
32. 682 Сумма на матрице
33. 684 Сумма на отрезке
34. 2860 Сумма чисел на промежутке
35. 4018 Черепашки
36. 595 Новый Лабиринт Амбера
37. 4020 Культ-орки на лестнице
38. 982 Связность
39. 4000 Обход в глубину
40. 4004 Есть ли цикл?
41. 3165 Двухцветность
42. 977 Дерево

3. Текст программы

3.1 Задачи с пояснением

7 задание

В супермаркете электроники, если верить телерекламе, существует система скидок: с двух купленных товаров полностью оплачивается только стоимость дорогого товара, а другой предоставляется бесплатно. Какой суммы достаточно, чтобы оплатить покупку трех товаров, если известна цена каждого.

Алгоритм состоит в том, чтобы найти минимальную и максимальную стоимость товара и затем вывести их сумму. Это и будет решением задачи. Для поиска максимальной и минимальной стоимости мы сначала записали все цены в массив, а дальше библиотечной функцией sort отсортировали массив цен по возрастанию.

#include <iostream>

#include <algorithm>

**using** **namespace** std**;**

int main**()** **{**

int a**[**3**];**

cin **>>** a**[**0**]** **>>** a**[**1**]** **>>** a**[**2**];**

sort**(**a**,**a**+**3**);**

cout **<<** **(**a**[**0**]** **+** a**[**2**])** **<<** endl**;**

**return** 0**;**

**}**

Тестовый пример:

https://pp.vk.me/c625426/v625426900/3d5aa/WwUPCdeBQr8.jpg

10 задание

По заданному натуральному числу n вычислить количество разных перестановок чисел от 1 до n.

Алгоритм заключается в том, чтобы последовательно перемножить цифры от 1 до заданного n. для этого мы воспользовались циклом.

#include <iostream>

**using** **namespace** std**;**

int main**()** **{**

int n**;**

cin **>>** n**;**

int res **=** 1**;**

**for(**int i**=**2**;**i**<=**n**;**i**++)** **{**

res **\*=** i**;**

**}**

cout **<<** res **<<** endl**;**

**}**

Тестовый пример:

https://pp.vk.me/c625426/v625426900/3d5b1/_Jt7Wkto8fQ.jpg

15 задание

Славик мечтает стать артистом оригинального жанра, поэтому постоянно занимается манипуляциями с разнообразными предметами. В данный момент он всё своё свободное время занят построеним карточных домиков. Ему уже даже удалось построить в домашних условиях 4-х этажный домик, на что у него ушло 26 карт. А на уроке технологии ему удалось тайком от учителя продемонстрировать своё умение одноклассникам и построить домик в 3 этажа, на что было потрачено всего 15 карт. "Художества" будущего артиста запечатлены на фотографиях из его мобильного телефона ниже.

Для подсчета количества затраченных карт мы воспользовались циклом, в котором суммировали карты, затраченные на постройку уровней от первого до n.

#include <cstdlib>

#include <cstdio>

#include <cstring>

#include <math.h>

**using** **namespace** std**;**

int main**(**int argc**,** char**\*\*** argv**)** **{**

int n**;**

scanf**(**"%d"**,&**n**);**

long long answer **=** 0**;**

**for(**int i**=**1**;**i**<=**n**;**i**++)** **{**

answer **+=** 2**\***i**+**i**-**1**;**

**}**

printf**(**"%lld\n"**,**answer**);**

**return** 0**;**

**}**

Тестовый пример:

https://pp.vk.me/c625426/v625426900/3d5b8/-fRrGudjVMc.jpg https://pp.vk.me/c625426/v625426900/3d5bf/bHyNK4hsXMg.jpg

19 задание

Дано число n. Используя только битовые операции, заменить первую справа единицу в двоичной записи на ноль.

Алгоритм состоит в том, чтобы сдвигать число n с помощью битовой операции "сдвиг вправо" пока не встретим единицу. Это мы делаем с помощью цикла, где дополнительно запоминаем количество сдвигов k до первой единицы. Далее единицу меняем на ноль и сдвигаем число на k влево.

#include <stdio.h>

#include <math.h>

#include <iostream>

**using** **namespace** std**;**

int main**()** **{**

int n**;**

cin **>>** n**;**

int k**=**0**,**f**=**1**;**

**while(**f**)** **{**

**if(**n**&**1**)** **{**

n **^=** 1**;**

f **=** 0**;**

n **<<=** k**;**

**}** **else** **{**

n **>>=** 1**;**

k**++;**

**}**

**}**

cout **<<** n **<<** endl**;**

**return** 0**;**

**}**

Тестовый пример:

https://pp.vk.me/c625426/v625426900/3d5c6/JWxxol-Z2Ak.jpg https://pp.vk.me/c625426/v625426900/3d5cd/i2HqQXtA5_w.jpg

34 задание

Найти сумму целых чисел на промежутке от a до b.

Решением этой задачи является сумма n членов арифметической прогрессии, где первый элемент a, а последний b и количество элементов между ними это модуль их разности, увеличенный на единицу

#include <stdio.h>

#include <math.h>

#include <iostream>

#define ABS(A) (((A)<0) ? -(A) : (A))

**typedef** long long LL**;**

**using** **namespace** std**;**

int main**()** **{**

LL a**,**b**;**

scanf**(**"%lld%lld"**,&**a**,&**b**)**

printf**(**"%lld\n"**,(**a**+**b**)\*(**ABS**(**b**-**a**)+**1**)/**2**);**

**return** 0**;**

**}**

Тестовый пример:

https://pp.vk.me/c625426/v625426900/3d5d4/zuyXeS0Q4n8.jpg

3.2 Задачи без пояснения

1 задание

Задано трицифровое число. Определить, какая цифра в нем есть большей – первая или последняя.

#include <cstdlib>

#include <cstdio>

#include <iostream>

#include <algorithm>

#include <math.h>

**using** **namespace** std**;**

int main**(**int argc**,** char**\*\*** argv**)** **{**

int n**;**

cin **>>** n**;**

**if(**n**/**100 **>** n**%**10**)** **{**

cout **<<** n**/**100**;**

**}** **else** **if(**n**/**100 **<** n**%**10**)** **{**

cout **<<** n**%**10**;**

**}** **else** **{**

cout **<<** "="**;**

**}**

cout **<<** endl**;**

**return** 0**;**

**}**

2 задание

Задано стороны a, b, c, d и диагональ f выпуклого четырехугольника. Определить площадь четырехугольника, используя вспомогательную функцию вычисления площади треугольника по формуле Герона.

#include <cstdlib>

#include <cstdio>

#include <iostream>

#include <algorithm>

#include <math.h>

**using** **namespace** std**;**

int main**(**int argc**,** char**\*\*** argv**)** **{**

double a**,**b**,**c**,**d**,**f**;**

cin **>>** a **>>** b **>>** c **>>** d **>>** f**;**

double p1 **=** **(**a**+**b**+**f**)/**2.0**;**

double p2 **=** **(**c**+**d**+**f**)/**2.0**;**

double S1 **=** sqrt**(**p1**\*(**p1**-**a**)\*(**p1**-**b**)\*(**p1**-**f**));**

double S2 **=** sqrt**(**p2**\*(**p2**-**c**)\*(**p2**-**d**)\*(**p2**-**f**));**

printf**(**"%.4f\n"**,**S1**+**S2**);**

**return** 0**;**

**}**

3 задание

Задано количество видов игрушек в магазине, количество игрушек каждого вида и стоимость игрушки каждого вида. Определить количество игрушек, стоимость которых меньше 50 грн.

#include <cstdlib>

#include <cstdio>

#include <iostream>

#include <algorithm>

#include <math.h>

**using** **namespace** std**;**

int main**(**int argc**,** char**\*\*** argv**)** **{**

int n**;**

cin **>>** n**;**

int result **=** 0**,**tmp1**;**

double tmp2**;**

**for(**int i**=**0**;**i**<**n**;**i**++)** **{**

cin **>>** tmp1 **>>** tmp2**;**

**if(**tmp2 **<** 50.0**)** **{**

result **+=** tmp1**;**

**}**

**}**

cout **<<** result **<<** endl**;**

**return** 0**;**

**}**

4 задание

Кот Василий узнал, что у соседа Димы, проживающего от него через какое-то количество заборов завелись мыши. Так как в своём хозяйстве всех мышей он уже давно выловил, кот отправляется на охоту за мышами к соседу, пролезая через дыры в ограде. На каждом участке Василий, как любой воспитанный кот, перемещается по уже проложенным там тропинкам.

В деревне Старые Васюки, где проживает Василий, всего одна улица и та протянулась вдоль реки, поэтому домики расположены только по одну сторону улицы. Известно, что между любыми соседними участками в заборе ровно одна дыра. Сколькими способами Василий может попасть на участок Димы, если известно, что Дима проживает на участке под номером k, а сам Василий проживает на участке под номером m?

#include <iostream>

**using** **namespace** std**;**

int main**()** **{**

long long n**,**m**,**k**,**tmp**,**res**=**1**;**

cin **>>** n **>>** m **>>** k**;**

**if(**m **>** k**)** **{**

tmp **=** m**;**

m **=** k**;**

k **=** tmp**;**

**}**

**for(**int i**=**1**;**i**<=**n**;**i**++)** **{**

cin **>>** tmp**;**

**if(**i **>=** m **&&** i **<=** k**)** **{**

res **\*=** tmp**;**

**}**

**}**

cout **<<** res **<<** endl**;**

**return** 0**;**

**}**

5 задание

Лесные жители решили провести хоккейный турнир между N командами. Сколькими способами могут быть распределены комплекты золотых, серебряных и бронзовых медалей, если одно призовое место может занять только одна команда?

#include <iostream>

#include <algorithm>

**using** **namespace** std**;**

int main**()** **{**

long long n**;**

cin **>>** n**;**

**switch(**n**)** **{**

**case** 0**:**

cout **<<** 0**;**

**break;**

**case** 1**:**

cout **<<** 1**;**

**break;**

**case** 2**:**

cout **<<** 2**;**

**break;**

**default:**

cout **<<** **(**n**\*(**n**-**1**)\*(**n**-**2**));**

**break;**

**}**

cout **<<** endl**;**

**}**

6 задание

Пирожок в школьной столовой стоит а гривен и b копеек. Найдите сколько гривен и копеек заплатит Петя за n пирожков.

#include <iostream>

**using** **namespace** std**;**

int main**()** **{**

int a**,**b**,**n**;**

cin **>>** a **>>** b **>>** n**;**

a **\*=** n**;**

b **\*=** n**;**

a **+=** b**/**100**;**

b **%=** 100**;**

cout **<<** a **<<** " " **<<** b **<<** endl**;**

**return** 0**;**

**}**

8 задание

Возьмем какое-нибудь натуральное число N. Будем изменять его следующим образом: если число четное, то разделим его на 2, если нечетное, прибавим 1. После нескольких таких изменений мы всегда получаем число 1. Например, из числа 11 получается число 12, затем 6, 3, 4, 2 и, наконец, 1. Таким образом, для получения 1 из 11 нужно проделать 6 изменений.

Напишите программу, которая считывает натуральное число и выводит количество изменений данного числа до получения 1.

#include <iostream>

#include <algorithm>

**using** **namespace** std**;**

int main**(){**

int n**;**

cin **>>** n**;**

int cnt **=** 0**;**

**while(**n **>** 1**)** **{**

cnt**++;**

**if(**n**%**2**)** **{**

n**++;**

**}** **else** **{**

n **/=** 2**;**

**}**

**}**

cout **<<** cnt **<<** endl**;**

**return** 0**;**

**}**

9 задание

По данному числу n требуется заполнить квадратную матрицу размером nxn целыми числами от 1 до n2 следующим образом:

- в левом верхнем углу находится число 1

- далее числа располагаются "змейкой", т.е. по возрастанию слева направо в нечётных строках, и справа налево - в чётных

#include <iostream>

#include <algorithm>

**using** **namespace** std**;**

int main**(){**

int n**;**

cin **>>** n**;**

**for(**int i**=**0**;**i**<**n**;**i**++)** **{**

**for(**int j**=**0**;**j**<**n**;**j**++)** **{**

**if(**i**%**2**)** **{**

cout **<<** i**\***n**+**n**-**j**;**

**}** **else** **{**

cout **<<** i**\***n**+**j**+**1**;**

**}**

**if(**j **==** n**-**1**)** **{**

cout **<<** endl**;**

**}** **else** **{**

cout **<<** " "**;**

**}**

**}**

**}**

**return** 0**;**

**}**

11 задание

Напишите программу, которая находит количество натуральных чисел, не превышающих N и не делящихся ни на одно из чисел 2, 3, 5.

#include <iostream>

**using** **namespace** std**;**

int main**(){**

long long n**;**

cin **>>** n**;**

int a**[]={**0**,**1**,**1**,**1**,**1**,**1**,**1**,**2**,**2**,**2**,**2**,**3**,**3**,**4**,**4**,**4**,**4**,**5**,**5**,**6**,**6**,**6**,**6**,**7**,**7**,**7**,**7**,**7**,**7**,**8**,**8**};**

cout **<< (**n**/**30**)\***8**+**a**[**n**%**30**] <<** endl**;**

**return** 0**;**

**}**

12 задание

В школьном буфете к окончанию уроков осталось несколько пирожных: A ванильных, B шоколадных и C фруктовых. Дима собирается купить пирожные перед закрытием буфета. Сколько пирожных может выбрать Дима?

#include <iostream>

**using** **namespace** std**;**

int main**()** **{**

int a**,**b**,**c**;**

cin **>>** a **>>** b **>>** c**;**

cout **<<** a**+**b**+**c **<<** endl**;**

**}**

13 задание

Анаграммой слова называется любая перестановка всех букв слова. Например, из слова SOLO можно получить 12 анаграмм: SOLO, LOSO, OSLO, OLSO, OSOL, OLOS, SLOO, LSOO, OOLS, OOSL, LOOS, SOOL.

Напишите программу, которая выводит количество различных анаграмм, которые могут получиться из этого слова.

#include <iostream>

#include <algorithm>

**typedef** long long ll**;**

**using** **namespace** std**;**

int main**()** **{**

int freq**[**30**]** **=** **{**0**};**

char str**[**15**];**

cin **>>** str**;**

int len **=** strlen**(**str**);**

**for(**int i**=**0**;**i**<**len**;**i**++)** **{**

freq**[**str**[**i**]-**'A'**]++;**

**}**

ll res **=** 1**;**

**for(**int i**=**len**;**i**>**0**;**i**--)** **{**

res **\*=** i**;**

**}**

**for(**int i**=**'A'**;**i**<=**'Z'**;**i**++)** **{**

**for(**int j**=**2**;**j**<=**freq**[**i**-**'A'**];**j**++)** **{**

res **/=** j**;**

**}**

**}**

cout **<<** res **<<** endl**;**

**return** 0**;**

**}**

14 задание

Влад хочет взять с собой для ланча пару фруктов. У него есть A бананов, B яблок и C груш. Сколькими способами он может выбрать 2 фрукта разного вида из имеющихся у него?

#include <iostream>

**typedef** long long ll**;**

**using** **namespace** std**;**

int main**()** **{**

ll a**,**b**,**c**;**

cin **>>** a **>>** b **>>** c**;**

cout **<<** a**\***b**+**b**\***c**+**a**\***c **<<** endl**;**

**}**

16 задание

По заданному натуральному числу N необходимо вычислить количество натуральных чисел, которые являются делителями N! (факториала числа N).

#include <iostream>

#include <cmath>

const int ch**[**14**]={**2**,**3**,**5**,**7**,**11**,**13**,**17**,**19**,**23**,**29**,**31**,**37**,**41**,**43**};**

**using** **namespace** std**;**

int main**(){**

int ex**[**14**]={**0**};**

signed long long int k**=**1**;**

int j**,**a**,**i**,**n**;**

cin **>>** n**;**

**for(**int q**=**0**;** q**<**14**;** q**++)**

ex**[**q**]=**0**;**

**for(**i**=**1**;** i**<=**n**;** i**++){**

j**=**0**;**

a**=**i**;**

**while(**a**!=**1**){**

**while(**a**%**ch**[**j**]==**0**){**

a**/=**ch**[**j**];**

**++**ex**[**j**];**

**}**

**++**j**;**

**}**

**}**

**for(**int i**=**0**;** i**<**14**;** i**++)**

k**\*=**ex**[**i**]+**1**;**

cout **<<** k **<<** endl**;**

**return** 0**;**

**}**

Далее для задач использовался следующий заголовок:

#include <stdio.h>

#include <math.h>

#include <algorithm>

#include <string.h>

#include <iostream>

#include <iomanip>

#include <vector>

#define SZ(X) ((int)(X).size())

#define ALL(X) (X).begin(), (X).end()

#define REP(I, N) for (int I = 0; I < (N); ++I)

#define REPR(I, N) for (int I = (N)-1; I >= 0; --I)

#define REPP(I, A, B) for (int I = (A); I < (B); ++I)

#define REPPR(I, A, B) for (int I = (A)-1; I >= (B); --I)

#define RLL(X) scanf("%lld", &(X))

#define RI(X) scanf("%d", &(X))

#define RII(X, Y) scanf("%d%d", &(X), &(Y))

#define RIII(X, Y, Z) scanf("%d%d%d", &(X), &(Y), &(Z))

#define DRLL(X) LL (X); scanf("%lld", &(X))

#define DRI(X) int (X); scanf("%d", &X)

#define DRII(X, Y) int (X), (Y); scanf("%d%d", &(X), &(Y))

#define DRIII(X, Y, Z) int (X), (Y), (Z); scanf("%d%d%d", &(X), &(Y), &(Z))

#define RS(X) scanf("%s", (X))

#define MP make\_pair

#define PB push\_back

#define MS0(X) memset((X), 0, sizeof((X)))

#define MS1(X) memset((X), -1, sizeof((X)))

#define LEN(X) strlen(X)

#define PII pair<int,int>

#define VPII vector<PII>

#define PLL pair<long long,long long>

#define F first

#define S second

#define INF (int) 1e9

#define ABS(A) (((A)<0) ? -(A) : (A))

**typedef** long long LL**;**

17 задание

Напишите программу, которая по данным числам A и n записывает представление числа A в n-разрядном двоичном дополнительном коде.

Первая строка входных данных содержит число A, вторая строка - число n, при этом 2 ≤ n ≤ 16, -2n-1 ≤ A ≤ 2n-1-1.

Программа должна вывести строку из n символов, содержащих запись числа A в n-разрядном двоичном дополнительном коде, первый символ - старший знаковый разряд.

**using** **namespace** std**;**

int main**()** **{**

int a**,**n**;**

cin **>>** a **>>** n**;**

int f **=** 0**;**

**if(**a **<** 0**)** **{**

cout **<<** 1**;**

a **\*=** **-**1**;**

a**--;**

f **=** 1**;**

**}** **else** **{**

cout **<<** 0**;**

**}**

int m **=** 1 **<<** n**-**2**;**

**while(**m**)** **{**

cout **<<** **((**a**&**m**)** **?** 1**^**f **:** 0**^**f**);**

m **>>=** 1**;**

**}**

cout **<<** endl**;**

**return** 0**;**

**}**

18 задание

Дана запись некоторого числа в двоичном дополнительном коде. Выведите десятичную запись этого числа.

**using** **namespace** std**;**

int main**()** **{**

char a**[**17**];**

cin **>>** a**;**

int f **=** 0**;**

**if(**a**[**0**]** **==** '1'**)** **{**

f **=** 1**;**

**}**

int n **=** 0**;**

REPP**(**i**,**1**,**strlen**(**a**))** **{**

n **\*=** 2**;**

n **+=** **(**a**[**i**]-**'0'**)^**f**;**

**}**

cout **<<** **(**f **?** **-(**n**+**f**)** **:** n**)** **<<** endl**;**

**return** 0**;**

**}**

20 задание

Найти K - количество N-значных натуральных чисел, которые содержат в своей записи хотя бы одну цифру 7.

**using** **namespace** std**;**

int main**(){**

long long n**;**

long long a**;**

cin **>>** n**;**

a **=** 9**\***pow**(**10**,(**n**-**1**))-**8**\***pow**(**9**,(**n**-**1**));**

**if(**n **>** 2**)** **{**

a**++;**

**}**

cout **<<** a **<<** endl**;**

**return** 0**;**

**}**

21 задание

Магический квадрат — это квадратная таблица, заполненная n2 чисел таким образом, что сумма чисел в каждой строке, каждом столбике и на обеих диагоналях одинакова. Сумма чисел в каждой строке, столбике и по диагоналям, называется магической постоянной. Проверьте, или заданный квадрат является магическим.

**using** **namespace** std**;**

int main**()** **{**

DRI**(**n**);**

int a**[**101**][**101**];**

MS0**(**a**);**

int d1**=**0**,**d2**=**0**;**

REPP**(**i**,**1**,**n**+**1**)** **{**

REPP**(**j**,**1**,**n**+**1**)** **{**

RI**(**a**[**i**][**j**]);**

a**[**i**][**0**]** **+=** a**[**i**][**j**];**

a**[**0**][**j**]** **+=** a**[**i**][**j**];**

**if(**i **==** j**)** **{**

d1 **+=** a**[**i**][**j**];**

**}**

**if(**i **==** n**-**j**+**1**)** **{**

d2 **+=** a**[**i**][**j**];**

**}**

**}**

**}**

**if(**d1 **==** d2**)** **{**

REPP**(**i**,**1**,**n**+**1**)** **{**

**if(**a**[**i**][**0**]** **!=** d1 **||** a**[**0**][**i**]** **!=** d1**)** **{**

d1 **=** **-**1**;**

**break;**

**}**

**}**

**}** **else** **{**

d1 **=** **-**1**;**

**}**

cout **<<** d1 **<<** endl**;**

**return** 0**;**

**}**

22 задание

Найти НОД (наибольший общий делитель) n чисел.

**using** **namespace** std**;**

LL gcd**(**LL a**,**LL b**)** **{**

**while(**b**)** **{**

a **%=** b**;**

swap**(**a**,**b**);**

**}**

**return** a**;**

**}**

LL lcm**(**LL a**,**LL b**)** **{**

**return** a**/**gcd**(**a**,**b**)\***b**;**

**}**

int main**()** **{**

DRI**(**n**);**

LL b **=** 0**;**

REP**(**i**,**n**)** **{**

DRLL**(**a**);**

b **=** gcd**(**a**,**b**);**

**}**

printf**(**"%lld\n"**,**b**);**

**return** 0**;**

**}**

23 задание

Для заданных n целых положительных чисел Вы должны найти сумму НОД (наибольших общих делителей) всех возможных пар этих чисел.

**using** **namespace** std**;**

LL gcd**(**LL a**,**LL b**)** **{**

**while(**b**)** **{**

a **%=** b**;**

swap**(**a**,**b**);**

**}**

**return** a**;**

**}**

LL lcm**(**LL a**,**LL b**)** **{**

**return** a**/**gcd**(**a**,**b**)\***b**;**

**}**

int main**()** **{**

int a**[**150**];**

DRI**(**n**);**

REP**(**t**,**n**)** **{**

MS0**(**a**);**

DRI**(**m**);**

REP**(**i**,**m**)** **{**

RI**(**a**[**i**]);**

**}**

LL sum **=** 0**;**

REP**(**i**,**m**-**1**)** **{**

REPP**(**j**,**i**+**1**,**m**)** **{**

sum **+=** gcd**(**a**[**i**],**a**[**j**]);**

**}**

**}**

printf**(**"%lld\n"**,**sum**);**

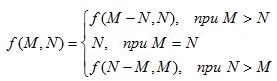
**}**

**return** 0**;**

**}**

24 задание

Дана функция, аргументы которой - произвольные натуральные числа



Составить алгоритм (написать программу), вычисляющий значение функции без помощи рекурсии.

**using** **namespace** std**;**

LL gcd**(**LL a**,**LL b**)** **{**

**while(**b**)** **{**

a **%=** b**;**

swap**(**a**,**b**);**

**}**

**return** a**;**

**}**

LL lcm**(**LL a**,**LL b**)** **{**

**return** a**/**gcd**(**a**,**b**)\***b**;**

**}**

int main**()** **{**

DRLL**(**a**);**

DRLL**(**b**);**

printf**(**"%lld\n"**,**gcd**(**a**,**b**));**

**return** 0**;**

**}**

25 задание

Толя на день рождения собирается угостить друзей тортом. Известно, что на дне рождения может быть либо n, либо m человек, включая самого именинника. На какое минимальное количество частей ему нужно разрезать торт (не обязательно всех равных), чтобы при любом из указанных количестве собравшихся, все съели торт поровну?

**using** **namespace** std**;**

int main**()** **{**

DRLL**(**a**);**

DRLL**(**b**);**

LL cnt **=** 0**;**

**while((**a **!=** 0**)** **&&** **(**b **!=** 0**))** **{**

cnt **+=** b**-**b**%**a**;**

b **=** b**%**a**;**

swap**(**a**,**b**);**

**}**

cout **<<** cnt **<<** endl**;**

**return** 0**;**

**}**

26 задание

Задан отрезок, концы которого имеют целочисленные координаты. Подсчитайте количество точек отрезка, имеющих целочисленные координаты.

**using** **namespace** std**;**

LL gcd**(**LL a**,**LL b**)** **{**

**while(**b**)** **{**

a **%=** b**;**

swap**(**a**,**b**);**

**}**

**return** a**;**

**}**

LL lcm**(**LL a**,**LL b**)** **{**

**return** a**/**gcd**(**a**,**b**)\***b**;**

**}**

int main**()** **{**

DRLL**(**x1**);**

DRLL**(**y1**);**

DRLL**(**x2**);**

DRLL**(**y2**);**

x1 **=** ABS**(**x1**-**x2**);**

y1 **=** ABS**(**y1**-**y2**);**

cout **<<** gcd**(**x1**,**y1**)+**1 **<<** endl**;**

**return** 0**;**

**}**

27 задание

НОД двух положительных целых чисел является наибольшее целое число, на которое делятся оба целые числа без остатка. НОК двух положительных целых чисел является наименьшее положительное целое число, которое делится на оба числа. В этой задаче вам будет задано два положительных целых числа, где НОД - это первое число, а НОК - это второе число.

**using** **namespace** std**;**

LL gcd**(**LL a**,**LL b**)** **{**

**while(**b**)** **{**

a **%=** b**;**

swap**(**a**,**b**);**

**}**

**return** a**;**

**}**

LL lcm**(**LL a**,**LL b**)** **{**

**return** a**/**gcd**(**a**,**b**)\***b**;**

**}**

int main**()** **{**

DRI**(**n**);**

REP**(**i**,**n**)** **{**

DRLL**(**a**);**

DRLL**(**b**);**

**if(**b**%**a **==** 0**)** **{**

cout **<<** a **<<** ' ' **<<** b **<<** endl**;**

**}** **else** **{**

cout **<<** **-**1 **<<** endl**;**

**}**

**}**

**return** 0**;**

**}**

28 задание

Задано два натуральных числа A и B. Найти количество таких пар чисел (P, Q), что для них A является НОД(P, Q), а B - НОК(P, Q).

**using** **namespace** std**;**

LL gcd**(**LL a**,**LL b**)** **{**

**while(**b**)** **{**

a **%=** b**;**

swap**(**a**,**b**);**

**}**

**return** a**;**

**}**

LL lcm**(**LL a**,**LL b**)** **{**

**return** a**/**gcd**(**a**,**b**)\***b**;**

**}**

int main**()** **{**

DRLL**(**a**);**

DRLL**(**b**);**

LL n**,**cnt **=** 0**;**

**for(**LL i**=**a**;**i**<=**b**;**i**+=**a**)** **{**

n **=** **(**b**/**i**)\***a**;**

**if((**gcd**(**i**,**n**)** **==** a**)** **&&** **(**lcm**(**i**,**n**)** **==** b**))** **{**

cnt**++;**

**}**

**}**

cout **<<** cnt **<<** endl**;**

**return** 0**;**

**}**

29 задание

Воспользуйтесь магической силой сортировки подсчётом и отсортируйте n чисел из диапазона [0; 100000].

**using** **namespace** std**;**

LL a**[**SIZE**];**

int main**()** **{**

MS0**(**a**);**

DRI**(**n**);**

REP**(**i**,**n**)** **{**

DRI**(**tmp**);**

a**[**tmp**]++;**

**}**

n**--;**

REP**(**i**,**SIZE**)** **{**

REP**(**j**,**a**[**i**])** **{**

printf**(**n **?** "%d " **:** "%d"**,**i**);**

n**--;**

**}**

**}**

printf**(**"\n"**);**

**return** 0**;**

**}**

30 задание

Для заданных A, B и M вычислить AB mod M.

long long binpow**(**long long a**,** long long n**,** int m**)** **{**

**if(**n **==** 0**)** **return** 1**;**

**if(**n **==** 1**)** **return** a**;**

long long result **=** n**&**1 **?** a **:** 1**;**

long long tmp **=** binpow**(**a**,**n**/**2**,**m**);**

result **\*=** **(**tmp**\***tmp**)%**m**;**

**return** result**%**m**;**

**}**

int main**(){**

long long a**,**b**;**

int m**;**

cin **>>** a **>>** b **>>** m**;**

cout **<<** binpow**(**a**%**m**,**b**,**m**)** **<<** endl**;**

**return** 0**;**

**}**

31 задание

Зная a, b, c, Вам необходимо вычислить значение ab mod c (1 ≤ a, b, c < 2^63).

**using** **namespace** std**;**

ULL mult**(**ULL base**,** ULL exp**,** ULL mod**)** **{**

ULL ret **=** 0**;**

**while(**exp**)** **{**

**if(**exp**%**2**)** **{**

ret **+=** base**;**

**}**

base **+=** base**;**

exp **/=** 2**;**

**if(**mod**)** **{**

ret **%=** mod**;**

base **%=** mod**;**

**}**

**}**

**return** ret**;**

**}**

ULL power**(**ULL base**,** ULL exp**,** ULL mod**)** **{**

ULL ret **=** 1**;**

**while(**exp**)** **{**

**if(**exp**%**2**)** **{**

ret **=** mult**(**ret**,**base**,**mod**);**

**}**

base **=** mult**(**base**,**base**,**mod**);**

exp **/=** 2**;**

**if(**mod**)** **{**

ret **%=** mod**;**

base **%=** mod**;**

**}**

**}**

**return** ret**;**

**}**

int main**()** **{**

ULL a**,**b**,**c**;**

**while(**RLL**(**a**)** **!=** EOF**)** **{**

RLL**(**b**);**

RLL**(**c**);**

printf**(**"%lld\n"**,**power**(**a**%**c**,**b**,**c**));**

**}**

**return** 0**;**

**}**

32 задание

Задан набор чисел a1 ... an. Для заданных запросов l и r найдите

prb682

**using** **namespace** std**;**

LL a**[**SIZE**];**

int main**()** **{**

MS0**(**a**);**

DRI**(**n**);**

REPP**(**i**,**1**,**n**+**1**)** **{**

RLL**(**a**[**i**]);**

a**[**i**]** **+=** a**[**i**-**1**];**

**}**

DRI**(**m**);**

REP**(**i**,**m**)** **{**

DRII**(**l**,**r**);**

printf**(**"%lld\n"**,(**a**[**r**]-**a**[**l**-**1**]));**

**}**

**return** 0**;**

**}**

33 задание

Задана матрица чисел ai,j, где 1 ≤ i ≤ n, 1 ≤ j ≤ m. Для заданных lx, ly, rx, ry найдите

prb684

**using** **namespace** std**;**

LL a**[**SIZE**][**SIZE**];**

int main**()** **{**

MS0**(**a**);**

DRII**(**n**,**m**);**

REPP**(**i**,**1**,**n**+**1**)** **{**

REPP**(**j**,**1**,**m**+**1**)** **{**

RLL**(**a**[**i**][**j**]);**

a**[**i**][**j**]** **+=** a**[**i**-**1**][**j**]** **+** a**[**i**][**j**-**1**]** **-** a**[**i**-**1**][**j**-**1**];**

**}**

**}**

DRI**(**q**);**

REP**(**t**,**q**)** **{**

DRII**(**lx**,**ly**);**

DRII**(**rx**,**ry**);**

LL sum **=** a**[**rx**][**ry**]-**a**[**lx**-**1**][**ry**]-**a**[**rx**][**ly**-**1**]+**a**[**lx**-**1**][**ly**-**1**];**

printf**(**"%lld\n"**,**sum**);**

**}**

**return** 0**;**

**}**

35 задание

В левом верхнем углу прямоугольной таблицы размером N×M находится черепашка. На каждой клетке таблицы разлито некоторое количество кислоты. Черепашка может перемещаться вправо или вниз, при этом маршрут черепашки заканчивается в правом нижнем углу таблицы.

Каждый миллилитр кислоты приносит черепашке некоторое количество урона. Найдите наименьшее возможное значение урона, которое получит черепашка после прогулки по таблице.

**using** **namespace** std**;**

LL board**[**2**][**1001**];**

int n**,**m**;**

int main**(**int argc**,** char**\*\*** argv**)** **{**

scanf**(**"%d%d"**,&**n**,&**m**);**

memset**(**board**,**0**,sizeof(**board**));**

int line **=** 1**;**

**for(**int i**=**1**;**i**<=**n**;**i**++)** **{**

**for(**int j**=**1**;**j**<=**m**;**j**++)** **{**

scanf**(**"%d"**,&**board**[**line**][**j**]);**

**if(**i **==** 1**)** **{**

board**[**line**][**j**]** **+=** board**[**line**][**j**-**1**];**

**}** **else** **if(**j **==** 1**)** **{**

board**[**line**][**j**]** **+=** board**[**1**-**line**][**j**];**

**}** **else** **{**

board**[**line**][**j**]** **+=** min**(**board**[**1**-**line**][**j**],**board**[**line**][**j**-**1**]);**

**}**

**}**

line **=** 1**-**line**;**

**}**

printf**(**"%d\n"**,**board**[**n**%**2**][**m**]);**

**return** 0**;**

**}**

36 задание

Как-то Корвину – принцу Амбера, по каким-то важным делам срочно понадобилось попасть в самую далекую тень, которую он только знал. Как всем известно, самый быстрый способ путешествия для принцев Амбера – это Лабиринт Амбера. Но у Корвина были настолько важные дела, что он не хотел тратить время на спуск в подземелье (именно там находится Амберский Лабиринт). Поэтому он решил воспользоваться Новым Лабиринтом, который нарисовал Дворкин. Но этот Лабиринт не так прост, как кажется…

Новый Лабиринт имеет вид последовательных ячеек, идущих друг за другом, пронумерованных от 1 до N. Из ячейки под номером i можно попасть в ячейки под номерами i+2 (если i+2 ≤ N) и i+3 (если i+3 ≤ N). На каждой ячейке лежит какое-то количество золотых монет ki. Для того чтобы пройти лабиринт нужно, начиная ходить из-за границ лабиринта (с нулевой ячейки) продвигаться по выше описанным правилам, при этом подбирая все монетки на ячейках, на которых вы делаете промежуточные остановки. Конечная цель путешествия – попасть на ячейку с номером N. Дальнейшее путешествие (в любое место Вселенной) возможно лишь тогда, когда достигнув ячейки с номером N, вы соберете максимально количество монеток. Напишите программу, которая поможет Корвину узнать, какое максимальное количество монеток можно собрать, проходя Новый Лабиринт Амбера.

**using** **namespace** std**;**

int a**[**SIZE**];**

int b**[**SIZE**];**

int main**()** **{**

MS0**(**a**);**

MS0**(**b**);**

DRI**(**n**);**

REPP**(**i**,**1**,**n**+**1**)** **{**

RI**(**a**[**i**]);**

**}**

REP**(**i**,**n**)** **{**

b**[**i**+**2**]** **=** max**(**b**[**i**+**2**],**b**[**i**]+**a**[**i**+**2**]);**

b**[**i**+**3**]** **=** max**(**b**[**i**+**3**],**b**[**i**]+**a**[**i**+**3**]);**

**}**

printf**(**"%d\n"**,**b**[**n**]);**

**return** 0**;**

**}**

37 задание

В Летней Кинематографической Школе пришло время обеда и эльф Коля поспешил в столовую. Однако для того, чтобы попасть в столовую, Коле нужно подняться по длинной лестнице, а на каждой её ступеньке в это время суток стоит по культорку. Каждый культорк разрешает Коле пройти по своей ступеньке только после того, как Коля запишется на мероприятие, которое этот кульорк организует. При этом никакие два культорка не проводят одно и то же мероприятие, и все мероприятия проходят в разное время.

Коля - честный эльф, и если уж он запишется на какую игру или конкурс, то потом обязательно придёт поучаствовать. Однако Коля хочет тратить как можно меньше времени на развлечения, ведь иначе ему некогда будет дорешивать кинематографические задачки. К счастью, Коле не надо наступать на каждую ступеньку, он может перепрыгнуть через несколько. Коля хочет узнать, какое минимальное количество времени ему придётся распланировать за один проход по лестнице до столовой.

**using** **namespace** std**;**

int N**,**K**,**result**=**0**;**

int l**[**20000**];**

int ans**[**20000**];**

int f**(**int n**)** **{**

**if(**n **<** 0**)** **return** 0**;**

**if(**ans**[**n**]** **==** **-**1**)** **{**

int tmp**=**f**(**n**-**1**);**

**for(**int i**=**1**;**i**<=**K**;**i**++)** **{**

**if(**f**(**n**-**i**-**1**)<**tmp**)**

tmp **=** f**(**n**-**i**-**1**);**

**}**

ans**[**n**]** **=** tmp**+**l**[**n**];**

**}**

**return** ans**[**n**];**

**}**

int main**(**int argc**,** char**\*\*** argv**)** **{**

cin **>>** N **>>** K**;**

memset**(**ans**,-**1**,sizeof(**ans**));**

**for(**int i**=**1**;**i**<=**N**;**i**++)** **{**

cin **>>** l**[**i**];**

**}**

f**(**N**);**

cout **<<** ans**[**N**]** **<<** endl**;**

**return** 0**;**

**}**

38 задание

В этой задаче требуется проверить, что неориентированный граф является связным, то есть что из любой вершины можно по рёбрам этого графа попасть в любую другую.

bool vertex\_visited**[**SIZE**];** // Graphs data

vector**<**int**>** vertex\_list**[**SIZE**];** // Graphs data

queue**<**int**>** vertex\_queue**;** // Graphs data

int vertex\_count**,**edge\_count**;** // Graphs data

int start\_vertex**,**finish\_vertex**,**distance\_limit**;** // Task data

void bfs**()** **{**

vertex\_visited**[**1**]** **=** **true;**

vertex\_queue**.**push**(**1**);**

int vertex\_target**;**

**while(!**vertex\_queue**.**empty**())** **{**

int vertex **=** vertex\_queue**.**front**();**

vertex\_queue**.**pop**();**

**for(**int i**=**0**;**i**<**vertex\_list**[**vertex**].**size**();**i**++)** **{**

vertex\_target **=** vertex\_list**[**vertex**][**i**];**

**if(!**vertex\_visited**[**vertex\_target**])** **{**

vertex\_visited**[**vertex\_target**]** **=** **true;**

vertex\_queue**.**push**(**vertex\_target**);**

**}**

**}**

**}**

**}**

int main**(**int argc**,** char**\*\*** argv**)** **{**

memset**(**vertex\_visited**,**0**,sizeof(**vertex\_visited**));**

cin **>>** vertex\_count **>>** edge\_count**;**

int edge\_start**,**edge\_finish**;**

**for(**int i**=**0**;**i**<**edge\_count**;**i**++)** **{**

cin **>>** edge\_start **>>** edge\_finish**;**

vertex\_list**[**edge\_start**].**push\_back**(**edge\_finish**);**

vertex\_list**[**edge\_finish**].**push\_back**(**edge\_start**);**

**}**

bfs**();**

bool result **=** **true;**

**for(**int i**=**1**;**i**<=**vertex\_count**;**i**++)** **{**

**if(!**vertex\_visited**[**i**])** **{**

result **=** **false;**

**break;**

**}**

**}**

cout **<<** **(**result **?** "YES" **:** "NO"**)** **<<** endl**;**

**return** 0**;**

**}**

39 задание

Дан неориентированный невзвешенный граф, в котором выделена вершина. Вам необходимо найти количество вершин, лежащих с ней в одной компоненте связности (включая саму вершину).

bool vertex\_visited**[**SIZE**];** // Graphs data

vector**<**int**>** vertex\_list**[**SIZE**];** // Graphs data

queue**<**int**>** vertex\_queue**;** // Graphs data

int vertex\_count**,**edge\_count**;** // Graphs data

int start\_vertex**,**finish\_vertex**,**distance\_limit**;** // Task data

void bfs**()** **{**

vertex\_visited**[**start\_vertex**]** **=** **true;**

vertex\_queue**.**push**(**start\_vertex**);**

int vertex\_target**;**

**while(!**vertex\_queue**.**empty**())** **{**

int vertex **=** vertex\_queue**.**front**();**

vertex\_queue**.**pop**();**

**for(**int i**=**0**;**i**<**vertex\_list**[**vertex**].**size**();**i**++)** **{**

vertex\_target **=** vertex\_list**[**vertex**][**i**];**

**if(!**vertex\_visited**[**vertex\_target**])** **{**

vertex\_visited**[**vertex\_target**]** **=** **true;**

vertex\_queue**.**push**(**vertex\_target**);**

**}**

**}**

**}**

**}**

int main**(**int argc**,** char**\*\*** argv**)** **{**

memset**(**vertex\_visited**,**0**,sizeof(**vertex\_visited**));**

cin **>>** vertex\_count **>>** start\_vertex**;**

int temp**;**

**for(**int i**=**1**;**i**<=**vertex\_count**;**i**++)** **{**

**for(**int j**=**1**;**j**<=**vertex\_count**;**j**++)** **{**

cin **>>** temp**;**

**if(**temp **==** 1**)** **{**

vertex\_list**[**i**].**push\_back**(**j**);**

**}**

**}**

**}**

bfs**();**

int result **=** 0**;**

**for(**int i**=**1**;**i**<=**vertex\_count**;**i**++)** **{**

**if(**vertex\_visited**[**i**])** **{**

result**++;**

**}**

**}**

cout **<<** result **<<** endl**;**

**return** 0**;**

**}**

40 задание

Дан ориентированный граф. Требуется определить, есть ли в нём цикл.

int vertex\_visited**[**SIZE**];** // Graphs data

vector**<**int**>** vertex\_list**[**SIZE**];** // Graphs data

queue**<**int**>** vertex\_queue**;** // Graphs data

int vertex\_count**,**edge\_count**;** // Graphs data

int start\_vertex**,**finish\_vertex**;** // Task data

bool result **=** **false;**

void dfs**(**int vertex**)** **{**

vertex\_visited**[**vertex**]** **=** 1**;**

**for(**int i**=**0**;**i**<**vertex\_list**[**vertex**].**size**();**i**++)** **{**

**if(**vertex\_visited**[**vertex\_list**[**vertex**][**i**]]** **==** 0**)** **{**

dfs**(**vertex\_list**[**vertex**][**i**]);**

**}** **else** **if(**vertex\_visited**[**vertex\_list**[**vertex**][**i**]]** **==** 1**)** **{**

result **=** **true;**

**}**

**if(**result**)** **{**

**break;**

**}**

**}**

vertex\_visited**[**vertex**]** **=** 2**;**

**}**

int main**(**int argc**,** char**\*\*** argv**)** **{**

memset**(**vertex\_visited**,**0**,sizeof(**vertex\_visited**));**

cin **>>** vertex\_count**;**

int temp**;**

**for(**int i**=**1**;**i**<=**vertex\_count**;**i**++)** **{**

**for(**int j**=**1**;**j**<=**vertex\_count**;**j**++)** **{**

cin **>>** temp**;**

**if(**temp **==** 1**)** **{**

vertex\_list**[**i**].**push\_back**(**j**);**

**}**

**}**

**}**

**for(**int i**=**1**;**i**<=**vertex\_count**;**i**++)** **{**

**if(**vertex\_visited**[**i**]** **==** 0**)** **{**

dfs**(**i**);**

**}**

**if(**result**)** **{**

**break;**

**}**

**}**

cout **<<** **(**result **?** 1 **:** 0**)** **<<** endl**;**

**return** 0**;**

**}**

41 задание

Необходимо выяснить, является ли заданный связный граф двухцветным. То есть можно ли его вершинам назначить цвета (имеется всего два цвета) таким образом, чтобы никакие две смежные вершины не имели одинаковый цвет. Для упрощения задачи можно предположить, что:

- граф не имеет петель.

- граф неориентированный. То есть если вершина a связана с вершиной b, то и b связана с a.

- граф связный. То есть всегда существует как минимум один путь из любой вершины в любою другую.

#include <iostream>

#include <vector>

#include <math.h>

#include <stdlib.h>

**using** **namespace** std**;**

vector**<**int**>** a**[**1005**];**

int used**[**1005**];**

int n**;**

int flag **=** 0**;**

int main**()** **{**

int l**,**x**,**y**;**

**while(**1**)** **{**

flag **=** 0**;**

scanf**(**"%d"**,&**n**);**

**if(**n **==** 0**)** **{**

**break;**

**}**

scanf**(**"%d"**,&**l**);**

memset**(**a**,**0**,sizeof(**a**));**

memset**(**used**,**0**,sizeof(**used**));**

**for(**int i**=**0**;**i**<**l**;**i**++)** **{**

scanf**(**"%d%d"**,&**x**,&**y**);**

a**[**x**].**push\_back**(**y**);**

a**[**y**].**push\_back**(**x**);**

**}**

dfs**(**1**,**1**);**

printf**(**flag **?** "NOT BICOLOURABLE.\n" **:** "BICOLOURABLE.\n"**);**

**}**

**return** 0**;**

**}**

42 задание

Неориентированный граф без петель и кратных ребер задан матрицей смежности. Определить, является ли этот граф деревом.

#include <iostream>

#include <vector>

#include <math.h>

#include <stdlib.h>

**using** **namespace** std**;**

int a**[**105**][**105**];**

int used**[**105**];**

int n**;**

int flag **=** 0**;**

int dfs**(**int v**,**int c**,**int prev**)** **{**

**if(**used**[**v**]** **<=** c **&&** used**[**v**]** **!=** 0**)** **{**

flag **=** 1**;**

**return** 0**;**

**}**

used**[**v**]** **=** c**;**

**for(**int i**=**1**;**i**<=**n**;**i**++)** **{**

**if((**a**[**v**][**i**]** **==** 1**)** **&&** **(**i **!=** prev**))** **{**

dfs**(**i**,**c**+**1**,**v**);**

**}**

**}**

**return** 0**;**

**}**

int main**(**int argc**,**char **\*\***argv**)** **{**

memset**(**a**,**0**,sizeof(**a**));**

memset**(**used**,**0**,sizeof(**used**));**

cin **>>** n**;**

**for** **(**int i**=**1**;**i**<=**n**;**i**++)** **{**

**for(** int j**=**1**;**j**<=**n**;** j**++)** **{**

cin **>>** a**[**i**][**j**];**

**}**

**}**

int cnt **=** 0**;**

**for(**int j**=**1**;**j**<=**n**;**j**++)** **{**

**if(**used**[**j**]** **==** 0**)** **{**

cnt**++;**

dfs**(**j**,**1**,**0**);**

**}**

**}**

cout **<<** **((**flag **||** **(**cnt **>** 1**))** **?** "NO" **:** "YES"**)** **<<** endl**;**

**return** 0**;**

**}**

5.Блок схемы

Блок схема программы по задаче 7 из пункта 3.1

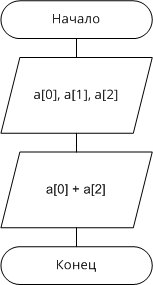


Рисунок 5.1 – Алгоритм работы программы 7

Блок схема программы по задаче 10 из пункта 3.1

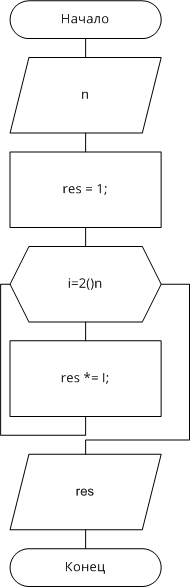


Рисунок 5.2 – Алгоритм работы программы 10

Блок схема программы по задаче 15 из пункта 3.1

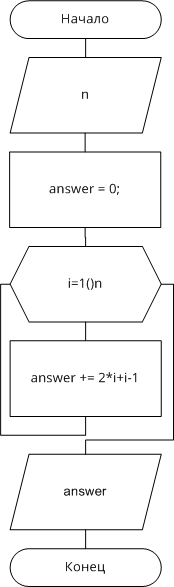


Рисунок 5.3 – Алгоритм работы программы 15

Блок схема программы по задаче 19 из пункта 3.1

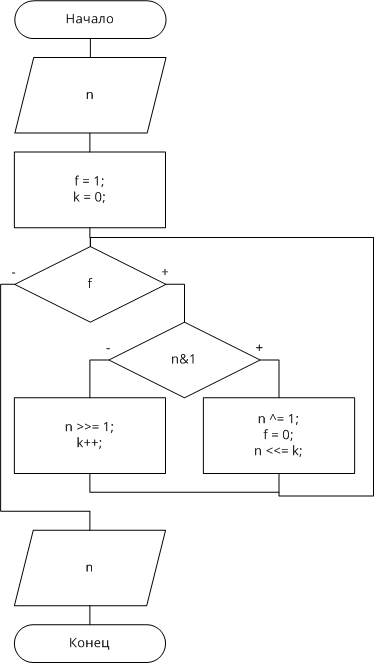


Рисунок 5.4 – Алгоритм работы программы 19

Блок схема программы по задаче 34 из пункта 3.1

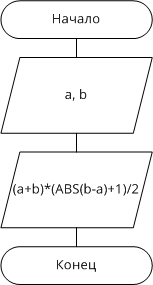


Рисунок 5.5 – Алгоритм работы программы 34

Вывод

Во время практики учебный процесс проходил в два этапа: теоретический и практический. При этом практический этап проходил в особой форме − соревновательной. Это повышало интерес к поиску оптимальных алгоритмов решения задач и их реализации.

По окончании прохождения учебной практики были изучены методы решения олимпиадных задач различного уровня сложности. Были рассмотрены более детально такие темы, как алгоритм Евклида, бинарное возведение в степень, динамическое программирование, поиск в глубину.

Было выяснено, что алгоритм Евклида для нахождения НОД и НОК можно применять в различных задачах, где на первый взгляд решение задачи не предполагает его использования.

Бинарное возведение в степень значительно ускоряло работу программ, что позволяет не превышать лимит времени, который отведен на ее работу.

В основе метода динамического программирования лежит принцип последовательной оптимизации: решение исходной задачи оптимизации большой размерности заменяется решением последовательности задач оптимизации малой размерности. Некоторые процессы (операции) расчленяются на шаги естественно, но существуют такие операции, которые приходится делить на этапы искусственно. Динамическое программирование дает возможность решать задачи, которые раньше не исследовались из-за отсутствия соответствующего математического аппарата. Также в ряде случаев этот метод сокращает объем при поиске оптимальных решений.

Поиск в глубину является рекурсивным алгоритм обхода вершин графа. Он используется для решения задач, таких как проверка на связность графа, поиска количества вершин, лежащих с ней в одной компоненте связности, определение цикличности ориентированного графа и других.

В отчете по учебной практике представлены все решенные задачи, а также отдельно вынесены некоторые задачи с пояснением алгоритма решения. Для них представлены тестовые примеры.

Учебная практика позволила расширить знания в области программирования на языке С++ и полученные знания будут применяться в дальнейшем процессе обучения.