МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРОСВЕЩЕНИЯ»
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРОСВЕЩЕНИЯ)

Кафедра вычислительной математики и информационных технологий

УТВЕРЖДЕН
на заседании кафедры
Протокол от «_19_»_марта_____2025 г., №_10_
Зав. кафедрой _____ /Шевчук М.В./

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине (модулю)
Методика решения олимпиадных задач по информатике

Направление подготовки (специальности) 44.03.05 Педагогическое образование
(с двумя профилями подготовки)

Профиль (программа подготовки, специализация)
Математика и информатика

Москва
2025

Содержание

1. Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы

Код и наименование компетенции	Этапы формирования
ПК-8 - Способен организовывать образовательный процесс с использованием современных образовательных технологий, в том числе дистанционных.	1.Работа на учебных занятиях 2.Самостоятельная работа

2. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Оцениваемые компетенции	**Уровень сформированности**	**Этап формирования**	**Описание показателей**	**Критерии оценивания**	**Шкала оценивания**
ПК-8	Пороговый	1. Работа на учебных занятиях 2. Самостоятельная работа	Знать: - основные виды олимпиад по информатике для школьников; - основные понятия теоретической информатики, теории чисел, теории графов и программирования; Уметь: - использовать знания по информатике и применять технологии, в том числе дистанционные, для решения олимпиадных задач;	конспект, лабораторные работы	Шкала оценивания конспекта Шкала оценивания лабораторных работ
	Продвинутый	1. Работа на учебных занятиях 2. Самостоятельная работа	Знать: - основные виды олимпиад по информатике для школьников; - основные понятия теоретической информатики, теории чисел, теории графов и программирования; Уметь: - применять технологии для решения олимпиадных задач;	конспект, лабораторные работы	Шкала оценивания конспекта Шкала оценивания лабораторных работ

Оцениваемые компетенции	Уровень сформированности	Этап формирования	Описание показателей	Критерии оценивания	Шкала оценивания
			Владеть: - методическими приемами и современными образовательными технологиями для решения олимпиадных задач.		

Описание шкал оценивания

Шкала оценивания лабораторных работ

Критерий оценивания	Баллы
Аккуратность и полнота выполнения всех пунктов задания	0-6
Понимание логики выполнения задания и значения полученных результатов	0-4
Максимальное количество баллов	10

Шкала оценивания конспекта

Критерии оценивания	Баллы
Текст конспекта логически выстроен и точно изложен, ясен весь ход рассуждения	0-2
Даны ответы на все поставленные вопросы, изложены научным языком, с применением терминологии	0-3
Максимальное количество баллов	5

3. Контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

Текущий контроль

ПК-8. Способен организовывать образовательный процесс с использованием современных образовательных технологий, в том числе дистанционных.

Знать: основные виды олимпиад по информатике для школьников; основные понятия теоретической информатики, теории чисел, теории графов и программирования;

Задания, необходимые для оценивания сформированности ПК-8 на пороговом и продвинутом уровнях

Перечень вопросов для конспекта

1. История олимпиадного движения. Олимпиады: сущность понятия, виды и функции.
2. Нормативно-правовая база организации олимпиад по информатике.
3. Технические вопросы проведения олимпиад по информатике.
4. Методические рекомендации по подготовке к участию в олимпиадах по информатике.
5. Всероссийская олимпиада школьников по информатике: основные правила и требования к проведению.
6. Применение информационно-коммуникационных технологий в современном процессе подготовки и организации олимпиад по информатике.
7. Интренет-ресурсы с коллекциями олимпиадных задач по информатике.
8. Сайты интернет-олимпиад, конкурсов и олимпиад для школьников по информатике.
9. Основные построения теории рекуррентных соотношений и способы их решения.
10. Формальные логические доказательства и логическое рассуждение при моделировании алгоритмов.
11. Основные алгоритмы теории чисел. Числовые алгоритмы. Алгоритмы на строках.
12. Основные понятия и вычислительные формулы комбинаторики. Классические комбинаторные алгоритмы.
13. Основные понятия и факты из теории графов. Типы графов. Операции над графами.
14. Рекурсия.
15. Алгоритмы теории игр. Простейшие игры.
16. Динамическое программирование.

Перечень практических работ

Тема 1. Олимпиады по информатике

Задание 1.1 Выполнение практического задания «Обзор олимпиад по информатике для школьников».

Задание 1.2 Выполнение практического задания «Технологии решения олимпиадных задач».

Задание 1.3 Выполнение практического задания «Формы подготовки учащихся к олимпиаде по информатике».

Тема 2. Математические основы информатики

Задание 2.1 Выполнение практического задания «Построение теории рекуррентных соотношений и способы их решения».

Задание 2.2 Выполнение практического задания «Формальные логические доказательства и логическое рассуждение при моделировании алгоритмов».

Тема 3. Основы теории чисел. основы комбинаторики. Основы теории графов

Задание 3.1 Выполнение практического задания «Перестановки, размещения и сочетания множества».

Задание 3.2 Выполнение практического задания «Теория графов".

Тема 4. Алгоритмы и их свойства.

Задание 4.1 Выполнение практического задания «Рекурсия».

Задание 4.2 Выполнение практического задания «Числовые алгоритмы».

Задание 4.3 Выполнение практического задания «Алгоритмы на строках».

Задание 4.4 Выполнение практического задания «Алгоритмы на графах».

Задание 4.5 Выполнение практического задания «Алгоритмы теории игр».

Тема 5. Программирование.

Задание 5.1 Выполнение практического задания «Классические задачи программирования».

Задание 5.2 Выполнение практического задания «Классические задачи динамического программирования».

Задания для отчета по лабораторным практическим работам

Лабораторная практическая работа 1

Нормативно-правовая база организации олимпиад по информатике. Технические вопросы проведения олимпиад по информатике

Цель работы. Знакомство с нормативно-правовыми нормами и техническими вопросами организации олимпиад по информатике, а также методическими особенностями подготовки школьников к участию в олимпиадах по информатике.

Задания.

1. Провести анализ нормативно-правовых норм организации олимпиад по информатике (выполнить конспект).

2. Сделать обзор олимпиад по информатике для школьников, на основе интернет-ресурсов (тип олимпиады, статус, уровень олимпиады, сроки проведения, требования к знаниям, умениям и владениям школьников по информатике)

Интернет-ресурсы:

- Образовательный цент «Взлёт» – Режим доступа: https://olympmo.ru/
- Олимпиады для школьников - Режим доступа: https://olimpiada.ru/
- Сайт «Олимпиадные задачи по программированию» – Режим доступа: http://algolist.manual.ru/olimp;
- Московская олимпиада школьников по информатике – Режим доступа: https://mos-inf.olimpiada.ru/ ;
- Олимпиады по информатике. Санкт-Петербург, Россия – Режим доступа: http://neerc.ifmo.ru/school ;

3. Описать типы и виды олимпиадных задач, рассмотрев основные технологии их решения на примере Всероссийской олимпиады школьников по информатике (подготовить конспект и презентацию).

4. Рассмотреть формы подготовки обучающихся к олимпиаде по информатике; выделить методические особенности организации подготовки школьников к участию в олимпиадах по информатике.

Цель работы. Проектирование элективного курса подготовки школьников к участию в олимпиаде по информатике для 7-9 класса.

Задание.

1. Создать методические документы, моделирующие педагогическую систему обучения элективному курсу (название курса, цели его преподавания, тематическое планирование курса с указанием названия занятий, их типов, методов обучения (включая методы

персонализированного обучения) источников информации, межпредметных и внутрипредметных связей).

2. Создать презентацию курса.

3. Подготовить подробный конспект одного из занятий, предусмотренных тематическим планом и дидактические материалы к нему в электронной форме, используя необходимые информационные технологии, в том числе дистанционные.

Лабораторная практическая работа 2
Математические основы информатики

Цель работы. Рассмотреть основы вычислений и построения теории рекуррентных соотношений, выделить способы их решения; формальные логические доказательства и логическое рассуждение при моделировании алгоритмов в олимпиадных задачах по информатике.

Задания.

1. Построение теории рекуррентных соотношений и способы их решения.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задача 1. Вычислить значение суммы $S = 1/1! + 1/2! + ... + 1/k!$

Решение: возможно, каждый раз вычислять очередное p!, затем 1/p!, и полученное слагаемое добавлять к сумме, но обратим внимание на следующее очевидное равенство: $1/(p+1)! = (1/p!)/(p+1)$, и программа вычисления запишется следующим образом

```
S:=1; p:=1; {S = p = 1/1!}
for i:=2 to k do
begin
p:=p/i;
S:=S+p;
end;
```

Задача 2. Написать программу определения количества шестизначных «счастливых» билетов, у которых сумма первых 3 десятичных цифр равна сумме 3 последних десятичных цифр.

Решение:

1) Выполним перебор всех возможных комбинаций шести цифр и подсчитаем число "счастливых" билетов.

```
Count:=0; {количество "счастливых" билетов}
```

```
for a1:=0 to 9 do
for a2:=0 to 9 do
for a3:=0 to 9 do
for a4:=0 to 9 do
for a5:=0 to 9 do
for a6:=0 to 9 do
if a1+a2+a3=a4+a5+a6
then Count:=Count+1;
```

Условие if во вложенных циклах будет проверяться 10^6 раз, поэтому будем говорить, что сложность этого алгоритма 10^6.

2) Обратим внимание на то, что в "счастливом" билете последняя цифра a6 однозначно определяется первыми пятью:

a6=(a1+a2+a3)-(a4+a5).

Если 0<=a6<=9, то билет "счастливый", иначе - нет. Таким образом, мы можем убрать шестой вложенный цикл:

```
Count:=0;
for a1:=0 to 9 do
for a2:=0 to 9 do
for a3:=0 to 9 do
for a4:=0 to 9 do
for a5:=0 to 9 do
begin
a6:=(a1+a2+a3)-(a4+a5);
if (a6>=0) and (a6<=9)
then Count:=Count+1;
end;
```

Сложность алгоритма 10^5.

3) Если комбинаций a1 a2 a3 первых трех цифр с суммой T=a1+a2+a3 насчитывается C[T], то всего "счастливых" билетов с суммой половины T=a1+a2+a3=a4+a5+a6 будет C[T]*C[T]. Всех возможных сумм T-28 (от 0=0+0+0 до 27=9+9+9). Подсчитаем C[i], i=0, ..., 28, затем найдем интересующее нас количество "счастливых" билетов

$C[0]^2 + C[1]^2 + ... + C[27]^2$.

Заметим, что "счастливых" билетов с суммой T столько же, сколько и с суммой 27-T. Действительно, если билет a1 a2 a3 a4 a5 a6 с суммой T - "счастливый", то таковым же является и билет (999999 - a1 a2 a3 a4 a5 a6) с суммой 27-T. Поэтому число билетов можно вычислять и по формуле

$2*(C[0]^2+ ... +C[13]^2)$, т.е. рассматривать только суммы T от 0 до 13.

```
var C:array[0..13] of longint;
Count:=0;
for T:=0 to 13 do C[T]:=0;
for a1:=0 to 9 do {перебираем все}
for a2:=0 to 9 do {возможные a1 a2 a3}
```

```
for a3:=0 to 9 do
begin
T:=a1+a2+a3;
C[T]:=C[T]+1 {нашли еще один билет}
end; {с суммой T}
for T:=0 to 13 do {считаем число билетов} Count:=Count+C[T]*C[T];
Count:=Count*2; {удваиваем сумму}
```

Сложность этого алгоритма 10^3.

4) В пункте 3 мы перебирали комбинации цифр и искали количество комбинаций с суммами C[T]. Сейчас мы пойдем от суммы T, и по ней будем определять, какое количество комбинаций a1 a2 a3 ее имеет.

Итак T=a1+a2+a3.

Минимальное значение, которое может принимать a1, - это MAX{0,T-18}. Член T-18 появляется из следующих соображений: пусть a2=a3=9, тогда a1=T-18, но a1 не может быть меньше 0. Максимальное значение a1=MIN{9,T} (так как a2 и a3 неотрицательны, то a1<=T и одновременно a1<=9).

Для цифры a2 аналогично получаем, что она лежит в пределах от max{0,T-a1-9} до min{9,T-a1}.

Цифра a3 по T, a1 и a2 определяется однозначно.

Получаем, что комбинаций a1 a2 a3 с суммой T и с первой цифрой a1 столько же, сколько возможных цифр a2, а именно min{9,T-a1}-max{0,T-a1-9}+1.

Как и в пункте 3 мы можем рассматривать диапазон сумм T от 0 до 13.

```
Count:=0;
for T:=0 to 13 do
begin
CT:=0;
for a1:=max(0,T-18) to min(9,T) do CT:=CT+min(9,T-a1)-max(0,T-a1-9)+1;
Count:=Count+CT*CT
end;
Count:=Count*2;
```

Сложность этого алгоритма (т.е. количество выполнений операций присваивания внутри двух вложенных циклов) не превышает 140.

Задача 3:
Фишка может двигаться по полю длины N только вперед. Длина хода фишки не более K. Найти число различных путей, по которым фишка может пройти поле от начала до конца.
Пример. N=3, K=2
Возможные пути:
1,1,1
1,2
2,1
Ответ: 3.

2. Формальные логические доказательства и логическое рассуждение при моделировании алгоритмов.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задача № 1. Логическая функция F задается выражением ¬x V y V (¬z Λ w). На рисунке приведен фрагмент таблицы истинности функции F, содержащий все наборы аргументов, при которых функция F ложна. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z, w (таблица).

?	?	?	?	F
0	0	0	1	0
0	1	0	1	0
0	1	1	1	0

В ответе напишите буквы x, y, z, w в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Решение (I способ):

1. Запишем выражение в более понятной форме: $F = \overline{x} + y + \overline{z} * w$.
2. Функция F представляет собой логическое сложение, состоящие из трех операндов, где третий операнд представлен логическим умножением из двух операндов. Для того, чтобы функция $F = \overline{x} + y + \overline{z} * w$ была ложна (все операнды сложения должны быть ложны), необходимо, чтобы x всегда был равен 1 (при логической инверсии всегда будет 0), а y всегда был равен 0; поэтому x – это последний столбец в таблице, а y – первый (таблица):

y	?	?	x	F
0	0	0	1	0
0	1	0	1	0
0	1	1	1	0

3. Разберемся с двумя столбцами по середине; обратим внимание на вторую строчку таблицы, в которой одна из оставшихся переменных равна 1, а вторая – 0; так как функция равна 0, то $\overline{z} * w = 0$, откуда следует, что z = 1 и w = 0 (иначе произведение будет равно 1). 4. Ответ: yzwx.

Решение (II способ – инверсия выражения):

1. Запишем выражение в более понятной форме: $F = \overline{x} + y + \overline{z} * w$.
2. Попытаемся свести задачу к уже известной задаче; если при каком-то наборе аргументов функция F ложна, то ее обратная функция $\overline{F}$, истинна.

3. Построим для функции F обратную функцию $\overline{F}$, используя законы де Моргана: $\overline{F} = \overline{\overline{x}+\overline{y}+\overline{z}*\overline{w}} = x * \overline{y} * (z + \overline{w})$.

4. Тогда при тех же значениях аргументов функция $\overline{F}$ истинна (таблица)

?	?	?	?	$\overline{F}$
0	0	0	1	1
0	1	0	1	1
0	1	1	1	1

5. Для истинности функции $\overline{F} = x * \overline{y} * (z + \overline{w})$ необходимо, чтобы x всегда был равен 1, а y всегда был равен 0; поэтому x – это последний столбец в таблице, а y – первый (таблица):

y	?	?	x	$\overline{F}$
0	0	0	1	1
0	1	0	1	1
0	1	1	1	1

6. Разберемся с двумя столбцами по середине; обратим внимание на вторую строчку таблицы, в которой одна из оставшихся переменных равна 1, а вторая – 0; так как функция равна 1, то $z + \overline{w} = 1$, откуда следует, что z = 1 и w = 0 (иначе сумма будет равна 0).

7. Ответ: yzwx.

Задача № 2. Логическая функция F задается выражением $(x \rightarrow y) \vee (y \rightarrow z)$. Ниже приведен фрагмент таблицы истинности. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z (таблица)

?	?	?	F
1	0	1	1
0	0	1	0

Решение:

1. Выражение представляет собой логическое произведение импликаций. Для истинности этого выражения обе импликации должны быть истинны.

2. Рассмотрим верхнюю строчку таблицы, где функция принимает значение 1. Здесь одна из переменных равна 0, а две другие равны 1.

3. Нулю в 1 строке может быть равен только x, так как при y = 0 получаем $(1\rightarrow0)\wedge(0\rightarrow1) = 0 \wedge 1 = 0$, а при z = 0 имеем $(1\rightarrow1)\wedge(1\rightarrow0) = 1 \wedge 0 = 0$, то есть эти два варианта не подходят. Таким образом, 2 столбец – x.

4. Рассмотрим вторую строку, там должен получиться 0. Мы знаем, что 2 столбец – x, поэтому во второй строке x = 0, и $(0\rightarrow y)\wedge(y\rightarrow z) = 0$.

5. Первая импликация $0\rightarrow y = 1$ независимо от значения y. Поэтому для того, чтобы все выражение было равно 0, нужно обеспечить $y\rightarrow z = 0$.

6. Это условие сразу дает y = 1 и z = 0. Поэтому третий столбец – y, а первый – z.

7. Ответ: zxy

Лабораторная практическая работа 3

Основы теории чисел. Основы комбинаторики. Основы теории графов.

Цель работы. *Рассмотреть основы теории чисел, комбинаторики и теории графов, используемых в решении олимпиадных задач по информатике.*

Задания.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

1. Перестановки, размещения и сочетания множества.

Задачи для решения.

1. Элементами множества A являются натуральные числа. Известно, что выражение ¬(x ∈ {2, 4, 6, 8, 10, 12}) ∨ (¬(x ∈ {3, 6, 9, 12, 15}) → (x ∈ A)) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение произведения элементов множества A.

2. Элементами множества A являются натуральные числа. Известно, что выражение ¬(x ∈ {1, 2, 3, 4, 5, 6}) ∨ (¬(x ∈ {3, 6, 9, 12, 15}) → (x ∈ A)) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение суммы элементов множества A.

3. Элементами множества A являются натуральные числа. Известно, что выражение ((x ∈ {3, 5, 7, 11, 12, 15}) → (x ∈ {5, 6, 12, 15})) ∨ (x ∈ A) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение произведения элементов множества A.

4. Элементами множества A являются натуральные числа. Известно, что выражение ((x ∈ {1, 3, 5, 7, 9, 12}) → (x ∈ {3, 6, 9, 12})) ∨ (x ∈ A) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение суммы элементов множества A.

5. Элементами множества A являются натуральные числа. Известно, что выражение (x ∈ {2, 4, 8, 12, 15}) → ((x ∈ {3, 6, 8, 15}) ∨ (x ∈ A)) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение произведения элементов множества A.

6. Элементами множества A являются натуральные числа. Известно, что выражение ((x ∈ {3, 5, 7, 11, 12}) → ¬(x ∈ {5, 6, 12, 15})) ∨ (x ∈ A) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение суммы элементов множества A.

7. На числовой прямой даны два отрезка: P = [5, 10] и Q = [15, 18]. Выберите такой отрезок A, что формула ((x ∈ A) → (x ∈ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [3, 11] 2) [6, 10] 3) [8, 16] 4) [17, 23]

8. На числовой прямой даны два отрезка: P = [25, 30] и Q = [15, 20]. Выберите такой отрезок A, что формула ((x ∈ A) → (x ∈ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [10, 15] 2) [12, 30] 3) [20, 25] 4)[26, 28]

9. На числовой прямой даны два отрезка: P = [2, 20] и Q = [15, 30]. Выберите такой отрезок A, что формула ((x ∉ A) → (x ∉ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [0, 15] 2) [3, 20] 3) [10, 25] 4)[25, 40]

10. На числовой прямой даны два отрезка: P = [10, 25] и Q = [0, 12]. Выберите такой отрезок A, что формула ((x ∉ A) → (x ∉ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [10, 15] 2) [20, 35] 3) [5, 20] 4)[12, 40]

Лабораторная работа 4

Алгоритмы и их свойства.

Цель работы. Рассмотреть алгоритмы и их свойства и выделить способы их решения в олимпиадных задачах по информатике.

Задания.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задание 1 Выполнение практического задания «Рекурсия».

Задача 1. Вычислить n!

Решение:

```
program factorial;
var N: word;
function F (n:word): longint;
begin
 if n=0 then F:= else F:= n*F (n-1)
end;
begin
write (‘N=’);
readln (N);
writeln (‘N!=’, F (N))
end.
```

Задача 2. Дано целое число A. Вставить между некоторыми цифрами 1, 2, 3, 4, 5, 6, 7, 8, 9, записанными именно в таком порядке, знаки «+» и «-» так, чтобы значением получившегося выражения было число n. Например, если A=122, то подойдет выражение: 12+34-5-6+78+9.

Решение:

```
programzi;
var t: array [2..9] of string;
procedurezi_fra (s, r: longint; o:char; n:integer);
var newr: longint;
begin if o = ‘+’ then newr:= r+s else newr:= r-s;
if n <= 9 then begin
t[n]:= ‘+’; zi_fra (n, newr, ‘+’, n+1);
t[n]:= ‘-’; zi_fra (n, newr, ‘-’, n+1);
t[n]:= ‘ ’; zi_fra (s*10+n, r, o, n+1);
end
else// проверка равенства
if newr = A then begin for i:= 2 to 9 do write (i-1, t[i]);
```

```
writeln (‘9’);
end;
end;
begin readln (A);
zi_fra (1, 0, ‘+’, 2);
end.
```

Задание 2 Выполнение практического задания «Числовые алгоритмы».

Задача 1. Сообщение о том, что ваш друг живет на 5 этаже, несет 4 бита информации. Сколько этажей в доме?

Решение: $N=2^i$; $2^4=16$ этажей.

Ответ: 16.

2. Проверить, поместится ли на диске компьютера музыкальная композиция, которая длится m минут и n секунд, если свободное дисковое пространство 6 мегабайт, а для записи одной секунды звука необходимо 16 килобайт.

Решение: переведем мегабайты в килобайты, так как 1 мегабайт = 1024 килобайта, то 6 мегабайт = 6144 килобайт. Обозначим t – время, в секундах, v – объем файла, в килобайтах, тогда: t=60*m+n; v=16*t.

Программа на Паскале будет иметь вид:

```
programmuzi;
varm, n, t, v: integer;
begin
writeln (‘Введитемин’);
readln (m, n);
t:=60*m+n;
v:=16*t;
if v<=6144 then writeln (‘Композиция поместится’)
else writeln (‘Не хватает ’, v-6144, ‘ килобайт’);
end.
```

Задание 3 Выполнение практического задания «Алгоритмы на строках».

Задача 1. Счастливые билеты.

Создать программу определения числа билетов с 6-значными номерами, у которых сумма первых 3 десятичных цифр равна сумме 3 последних десятичных цифр.

Формат выходных данных.

Вывести количество счастливых билетов на экран.

Разбор задачи

Просматриваем числа от 0 до 999999. Делим число на 2 части: первые 3 цифры и последние 3 цифры, находим сумму цифр каждой из частей, сравниваем результат.

Код программы:

```
program z1;
function summ(x: longint): byte;{возвращает сумму цифр числа}
var k, l: byte;
y: longint;
begin
y := x; l := 0;
while (y <> 0) do
begin
k := y mod 10;
```

```
y := y div 10;
l := l + k end;
summ := l;
end;
var w1, w2, i, j, count: longint;
n, m: byte;
begin
count := 0;
for j := 0 to 999999 do begin w1 := j div 1000;
w2 := j mod 1000;
if summ(w1) = summ(w2) then
begin
count := count + 1;
 writeln(j, ' --> ', count);
end;
 end;
 end.
```

Задание 4 Выполнение практического задания «Алгоритмы на графах».

Задача 1. Каждый элемент квадратной матрицы размеренности N*N равен нулю, либо единицы. Найти количество групп, образованных единицами. Единицы относятся к одной группе, если из одной из них можно перейти к другой «наступая» на единицы, расположенные в соседних клетках. Соседними являются клетки, граничащие по горизонтали или вертикали.

Входные данные: в первой строке файла input.txtзаписано натуральное число N не больше 100 – размер квадратной матрицы. В следующих N строках задаются элементы матрицы через пробел.

Выходные данные: output.txt выведите единственное число – количество групп.

Решение: необходимо обойти матрицу и вычислить количество групп. После того, как мы попадаем в группу, надо это зафиксировать, увеличив переменную – результат на единицу. Чтобы второй раз не посчитать одну и ту же группу, сразу после посещения необходимо его уничтожить, то есть присвоить всем клеткам значение ноль.

Тест задачи не слишком мал, поэтому необходимо написать процедуру уничтожения группы, назовем ее “del”. Чтобы во время процедуры не выйти за пределы массива, сделаем его размером N+2 * N+2, а не размером N*N. Это дает нам возможность окружить искомый массив размером N*N нулями.

```
programgrup;
const m =102;
var a:array [1..m, 1..m] of integer;
 i, j, n, rez: integer;
input, output: text;
procedure del (i, j: integer);
begin
 ifa[i, j] <> 1 then exit;
a [i, j]:= 0;
del (i+1, j);
del (i-1, j);
del (i, j+1);
del (i, j-1);
end;
```

```
begin 40 rez:= 0;
assing (input, ‘input.txt’);
reset (input);
assing (output, ‘output.txt’);
rewrite (output);
read (input, n); // заполняем массив нулям
 for i:= 1 to n+2 do
for j:= 1 to n+2 do
a [i, j]:=0; // считать матрицу из файла
for i:= 2 to n+1 do
for j:= 2 to n+1 do
read (input, a[i, j]); // обход матрицы в поиске групп
for i:= 2 to n+1 do
for j:= 2 to n+1 do
if a[i,j] = 1 then begin inc (rez);
count (i, j);
end;
write (output, rez);
close (input);
close (output);
end.
```

Задание 5 Выполнение практического задания «Алгоритмы теории игр».

Задача 1. Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу один или четыре камня либо увеличить количество камней в куче в пять раз. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16, 19 или 75 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней. Игра завершается в тот момент, когда количество камней в куче становится не менее 73.

Победителем считается игрок, сделавший последний ход, т.е. первым получивший кучу, в которой будет 73 или больше камней.

В начальный момент в куче было *S* камней; $1 \le S \le 72$.

Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника.

Укажите такое значение *S*, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Назовите минимальное значение *S*, при котором это возможно.

Решение: Из условия мы знаем, что Петя не может выиграть своим первым ходом, но Ваня, независимо от хода Пети, выигрывает. Петя парень не глупый и, естественно, сделает самый маленький ход – добавит в кучу один камень (*S+1)*, чтобы Ваня от выигрыша был как можно дальше. Далее Ваня делает свой первый ход и выигрывает. Ване необходимо максимально увеличить предыдущее значение камней в куче, следовательно, Ваня увеличивает количество камней в куче в пять раз (*S+1)*5*. Теперь просто решаем неравенство:

$(S+1)*5 \ge 73$

$5S \ge 70$

$S \ge 13,6$

$S = 14$

Ответ: 14**.**

Лабораторная работа 5
Программирование

Цель работы. Рассмотреть способы использования языков программирования для решения олимпиадных задач по информатике.

Задания.

Отчет представить в виде следующего алгоритма:
1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задание 1 Выполнение практического задания «Классические задачи программирования».
Пример задачи:
Сортировка методом “пузырьков”.
1. Функция для сортировки:

```
Public Function сортировка(массив() As Variant) As Boolean
Dim a As Boolean, t As Variant, x As Integer
For x = LBound(массив) To UBound(массив)
If IsNull(массив(X)) Then Exit Function
Next x
Do
a = False
For x = UBound(массив) To (LBound(массив) + 1) Step -1
If массив(x - 1) > массив(x) Then
t = массив(x - 1)
массив(x - 1) = массив(x)
массив(x) = t
a = True
End If
Next x
For x = (LBound(массив) + 1) To UBound(массив)
If массив(x - 1) > массив(x) Then
t = массив(x - 1)
массив(x - 1) = массив(x)
массив(x) = t
a = True
End If
Next x
Loop While a
сортировка = True
End Function
```

Задание 2 Выполнение практического задания «Классические задачи динамического программирования».

Задача 1. Вычислить значение суммы S=1/1!+1/2!+…+ 1/k!

Решение: можно, конечно, каждый раз вычислять очередное p!, затем 1/p!, и полученное слагаемое добавлять к сумме. Но обратим внимание на следующее очевидное равенство: 1/(p+1)! = (1/p!)/(p+1), И программа вычисления запишется следующим образом:

```
Program_sum;
var i, p: longint;
S:real;
begin
S:=1;
p:=1;
for i:= 2 to k do begin p:= p/I; S:=S+p;
end;
writeln('S=', S:8:2);
 end.
```

Задача 2. На числовой прямой школьник Ваня может идти вправо на одну или две единицы. Первоначально Ваня находится в точке с координатой 0. Определите количество маршрутов Вани, приводящих его в точку с координатой n.

Обозначим количество маршрутов Вани, ведущих в точку с координатой n, как F(n). Теперь вычислим функцию F(n). Заметим, что F(0)=1 (это вырожденный случай, существует ровно один маршрут из точки 0 в точку 0 – он не содержит ни одного прыжка), F(1)=1, F(2)=2. Как вычислить F(n)? В точку n школьник может попасть двумя способами – из точки n–2 при помощи прыжка длиной 2 и из точки n-1 прыжком длины 1. То есть число способов попасть в точку n равно сумме числа способов попасть в точку n-1 и n-2, что позволяет выписать рекуррентное соотношение: F(n) = F (n-2) + F (n-1) , верное для всех n> 2.

Решение на языке Pascal в виде рекурсивной функции:

```
function f (n: longint): longint;
begin
if n < 2 then
f:=1 else f:= f (n-1) + f (n-2);
end;
```

Вычисление решения этой функции, например, для n=20, оказывается, что функция работает крайне медленно. То есть одна из причин неэффективности рекурсивного решения – одно и то же значение функции вычисляется несколько раз, так как оно используется для вычисления нескольких других значений функции.

В данном случае, значения рекурсивной функции совпадает с числами Фибоначчи, так как вычисляются по тем же рекуррентным соотношениям. Для вычисления чисел Фибоначчи можно использовать цикл.

Решение:

```
var F: array [0..100] of longint;
i, n: longint;
begin
readln (n);
for i:= 1 to n do F [i]:=0;
 F[0]:=1; F[1]:=1;
 for i:= 2 to n do
F[i]:= F [i – 1] + F [i – 2];
```

```
writeln (F [n]);
end.
```

Динамическое программирование использует те же рекуррентные соотношения, что и рекурсивное решение, но в отличии от рекурсии в динамическом программировании значения вычисляются в цикле и сохраняются в списке.

Уметь: применять технологии для решения олимпиадных задач;

Задания, необходимые для оценивания сформированности ПК-8 на пороговом и продвинутом уровнях

Перечень вопросов для конспекта

1. История олимпиадного движения. Олимпиады: сущность понятия, виды и функции.
2. Нормативно-правовая база организации олимпиад по информатике.
3. Технические вопросы проведения олимпиад по информатике.
4. Методические рекомендации по подготовке к участию в олимпиадах по информатике.
5. Всероссийская олимпиада школьников по информатике: основные правила и требования к проведению.
6. Применение информационно-коммуникационных технологий в современном процессе подготовки и организации олимпиад по информатике.
7. Интренет-ресурсы с коллекциями олимпиадных задач по информатике.
8. Сайты интернет-олимпиад, конкурсов и олимпиад для школьников по информатике.
9. Основные построения теории рекуррентных соотношений и способы их решения.
10. Формальные логические доказательства и логическое рассуждение при моделировании алгоритмов.
11. Основные алгоритмы теории чисел. Числовые алгоритмы. Алгоритмы на строках.
12. Основные понятия и вычислительные формулы комбинаторики. Классические комбинаторные алгоритмы.
13. Основные понятия и факты из теории графов. Типы графов. Операции над графами.
14. Рекурсия.
15. Алгоритмы теории игр. Простейшие игры.
16. Динамическое программирование.

Перечень практических работ

Тема 1. Олимпиады по информатике

Задание 1.1 Выполнение практического задания «Обзор олимпиад по информатике для школьников».

Задание 1.2 Выполнение практического задания «Технологии решения олимпиадных задач».

Задание 1.3 Выполнение практического задания «Формы подготовки учащихся к олимпиаде по информатике».

Тема 2. Математические основы информатики

Задание 2.1 Выполнение практического задания «Построение теории рекуррентных соотношений и способы их решения».

Задание 2.2 Выполнение практического задания «Формальные логические доказательства и логическое рассуждение при моделировании алгоритмов».

Тема 3. Основы теории чисел. основы комбинаторики. Основы теории графов

Задание 3.1 Выполнение практического задания «Перестановки, размещения и сочетания множества».

Задание 3.2 Выполнение практического задания «Теория графов".
Тема 4. Алгоритмы и их свойства.
Задание 4.1 Выполнение практического задания «Рекурсия».
Задание 4.2 Выполнение практического задания «Числовые алгоритмы».
Задание 4.3 Выполнение практического задания «Алгоритмы на строках».
Задание 4.4 Выполнение практического задания «Алгоритмы на графах».
Задание 4.5 Выполнение практического задания «Алгоритмы теории игр».
Тема 5. Программирование.
Задание 5.1 Выполнение практического задания «Классические задачи программирования».
Задание 5.2 Выполнение практического задания «Классические задачи динамического программирования».

Задания для отчета по лабораторным практическим работам

Лабораторная практическая работа 1
Нормативно-правовая база организации олимпиад по информатике. Технические вопросы проведения олимпиад по информатике

Цель работы. Знакомство с нормативно-правовыми нормами и техническими вопросами организации олимпиад по информатике, а также методическими особенностями подготовки школьников к участию в олимпиадах по информатике.

Задания.

1. Провести анализ нормативно-правовых норм организации олимпиад по информатике (выполнить конспект).

2. Сделать обзор олимпиад по информатике для школьников, на основе интернет-ресурсов (тип олимпиады, статус, уровень олимпиады, сроки проведения, требования к знаниям, умениям и владениям школьников по информатике)

Интернет-ресурсы:

- Образовательный цент «Взлёт» – Режим доступа: https://olympmo.ru/
- Олимпиады для школьников - Режим доступа: https://olimpiada.ru/
- Сайт «Олимпиадные задачи по программированию» – Режим доступа: http://algolist.manual.ru/olimp;
- Московская олимпиада школьников по информатике – Режим доступа: https://mos-inf.olimpiada.ru/ ;
- Олимпиады по информатике. Санкт-Петербург, Россия – Режим доступа: http://neerc.ifmo.ru/school ;

3. Описать типы и виды олимпиадных задач, рассмотрев основные технологии их решения на примере Всероссийской олимпиады школьников по информатике (подготовить конспект и презентацию).

4. Рассмотреть формы подготовки обучающихся к олимпиаде по информатике; выделить методические особенности организации подготовки школьников к участию в олимпиадах по информатике.

Цель работы. Проектирование элективного курса подготовки школьников к участию в олимпиаде по информатике для 7-9 класса.

Задание.

1. Создать методические документы, моделирующие педагогическую систему обучения элективному курсу (название курса, цели его преподавания, тематическое планирование курса с указанием названия занятий, их типов, методов обучения (включая методы персонализированного обучения) источников информации, межпредметных и внутрипредметных связей).

2. Создать презентацию курса.

3. Подготовить подробный конспект одного из занятий, предусмотренных тематическим планом и дидактические материалы к нему в электронной форме, используя необходимые информационные технологии, в том числе дистанционные.

Лабораторная практическая работа 2
Математические основы информатики

Цель работы. Рассмотреть основы вычислений и построения теории рекуррентных соотношений, выделить способы их решения; формальные логические доказательства и логическое рассуждение при моделировании алгоритмов в олимпиадных задачах по информатике.

Задания.

1. Построение теории рекуррентных соотношений и способы их решения.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задача 1. Вычислить значение суммы S = 1/1! + 1/2! + ... + 1/k!

Решение: возможно, каждый раз вычислять очередное p!, затем 1/p!, и полученное слагаемое добавлять к сумме, но обратим внимание на следующее очевидное равенство: 1/(p+1)! = (1/p!)/(p+1), и программа вычисления запишется следующим образом

```
S:=1; p:=1; {S = p = 1/1!}
for i:=2 to k do
begin
```

```
    p:=p/i;
    S:=S+p;
    end;
```

Задача 2. Написать программу определения количества шестизначных «счастливых» билетов, у которых сумма первых 3 десятичных цифр равна сумме 3 последних десятичных цифр.

Решение:

1) Выполним перебор всех возможных комбинаций шести цифр и подсчитаем число "счастливых" билетов.

```
Count:=0; {количество "счастливых" билетов}
for a1:=0 to 9 do
for a2:=0 to 9 do
for a3:=0 to 9 do
for a4:=0 to 9 do
for a5:=0 to 9 do
for a6:=0 to 9 do
if a1+a2+a3=a4+a5+a6
then Count:=Count+1;
```

Условие if во вложенных циклах будет проверяться 10^6 раз, поэтому будем говорить, что сложность этого алгоритма 10^6.

2) Обратим внимание на то, что в "счастливом" билете последняя цифра a6 однозначно определяется первыми пятью:

a6=(a1+a2+a3)-(a4+a5).

Если 0<=a6<=9, то билет "счастливый", иначе - нет. Таким образом, мы можем убрать шестой вложенный цикл:

```
Count:=0;
for a1:=0 to 9 do
for a2:=0 to 9 do
for a3:=0 to 9 do
for a4:=0 to 9 do
for a5:=0 to 9 do
begin
a6:=(a1+a2+a3)-(a4+a5);
if (a6>=0) and (a6<=9)
then Count:=Count+1;
end;
```

Сложность алгоритма 10^5.

3) Если комбинаций a1 a2 a3 первых трех цифр с суммой T=a1+a2+a3 насчитывается C[T], то всего "счастливых" билетов с суммой половины T=a1+a2+a3=a4+a5+a6 будет C[T]*C[T]. Всех возможных сумм T-28 (от 0=0+0+0 до 27=9+9+9). Подсчитаем C[i], i=0, ..., 28, затем найдем интересующее нас количество "счастливых" билетов

$C[0]^2 + C[1]^2 + ... + C[27]^2$.

Заметим, что "счастливых" билетов с суммой T столько же, сколько и с суммой 27-T. Действительно, если билет a1 a2 a3 a4 a5 a6 с суммой T - "счастливый", то таковым же является и билет (999999 - a1 a2 a3 a4 a5 a6) с суммой 27-T. Поэтому число билетов можно вычислять и по формуле

$2*(C[0]^2+ ... +C[13]^2)$, т.е. рассматривать только суммы T от 0 до 13.

```
var C:array[0..13] of longint;
Count:=0;
for T:=0 to 13 do C[T]:=0;
for a1:=0 to 9 do {перебираем все}
for a2:=0 to 9 do {возможные a1 a2 a3}
for a3:=0 to 9 do
begin
T:=a1+a2+a3;
C[T]:=C[T]+1 {нашли еще один билет}
end; {с суммой T}
for T:=0 to 13 do {считаем число билетов} Count:=Count+C[T]*C[T];
Count:=Count*2; {удваиваем сумму}
```

Сложность этого алгоритма 10^3.

4) В пункте 3 мы перебирали комбинации цифр и искали количество комбинаций с суммами C[T]. Сейчас мы пойдем от суммы T, и по ней будем определять, какое количество комбинаций a1 a2 a3 ее имеет.

Итак T=a1+a2+a3.

Минимальное значение, которое может принимать a1, - это MAX{0,T-18}. Член T-18 появляется из следующих соображений: пусть a2=a3=9, тогда a1=T-18, но a1 не может быть меньше 0. Максимальное значение a1=MIN{9,T} (так как a2 и a3 неотрицательны, то a1<=T и одновременно a1<=9).

Для цифры a2 аналогично получаем, что она лежит в пределах от max{0,T-a1-9} до min{9,T-a1}.

Цифра a3 по T, a1 и a2 определяется однозначно.

Получаем, что комбинаций a1 a2 a3 с суммой T и с первой цифрой a1 столько же, сколько возможных цифр a2, а именно min{9,T-a1}-max{0,T-a1-9}+1.

Как и в пункте 3 мы можем рассматривать диапазон сумм T от 0 до 13.

```
Count:=0;
for T:=0 to 13 do
begin
CT:=0;
for a1:=max(0,T-18) to min(9,T) do CT:=CT+min(9,T-a1)-max(0,T-a1-9)+1;
Count:=Count+CT*CT
end;
Count:=Count*2;
```

Сложность этого алгоритма (т.е. количество выполнений операций присваивания внутри двух вложенных циклов) не превышает 140.

Задача 3:
Фишка может двигаться по полю длины N только вперед. Длина хода фишки не более K. Найти число различных путей, по которым фишка может пройти поле от начала до конца.
Пример. N=3, K=2
Возможные пути:
1,1,1
1,2
2,1
Ответ: 3.

2. Формальные логические доказательства и логическое рассуждение при моделировании алгоритмов.

Отчет представить в виде следующего алгоритма:
1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задача № 1. Логическая функция F задается выражением ¬x V y V (¬z Λ w). На рисунке приведен фрагмент таблицы истинности функции F, содержащий все наборы аргументов, при которых функция F ложна. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z, w (таблица).

?	?	?	?	F
0	0	0	1	0
0	1	0	1	0
0	1	1	1	0

В ответе напишите буквы x, y, z, w в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Решение (I способ):
1. Запишем выражение в более понятной форме: $F = \overline{x} + y + \overline{z} * w$.
2. Функция F представляет собой логическое сложение, состоящие из трех операндов, где третий операнд представлен логическим умножением из двух операндов. Для того, чтобы функция $F = \overline{x} + y + \overline{z} * w$ была ложна (все операнды сложения должны быть ложны), необходимо, чтобы x всегда был равен 1 (при логической инверсии всегда будет 0), а y всегда был равен 0; поэтому x – это последний столбец в таблице, а y – первый (таблица):

y	?	?	x	F
0	0	0	1	0
0	1	0	1	0
0	1	1	1	0

3. Разберемся с двумя столбцами по середине; обратим внимание на вторую строчку таблицы, в которой одна из оставшихся переменных равна 1, а вторая – 0; так как функция равна 0, то $\overline{z} * w = 0$, откуда следует, что z = 1 и w = 0 (иначе произведение будет равно 1). 4. Ответ: yzwx.

Решение (II способ – инверсия выражения):

1. Запишем выражение в более понятной форме: $F = \overline{x} + y + \overline{z} * w$.

2. Попытаемся свести задачу к уже известной задаче; если при каком-то наборе аргументов функция F ложна, то ее обратная функция $\overline{F}$, истинна.

3. Построим для функции F обратную функцию $\overline{F}$, используя законы де Моргана: $\overline{F} = \overline{\overline{x}+\overline{y}+\overline{z}*\overline{w}} = x * \overline{y} * (z + \overline{w})$.

4. Тогда при тех же значениях аргументов функция $\overline{F}$ истинна (таблица)

?	?	?	?	$\overline{F}$
0	0	0	1	1
0	1	0	1	1
0	1	1	1	1

5. Для истинности функции $\overline{F} = x * \overline{y} * (z + \overline{w})$ необходимо, чтобы x всегда был равен 1, а y всегда был равен 0; поэтому x – это последний столбец в таблице, а y – первый (таблица):

y	?	?	x	$\overline{F}$
0	0	0	1	1
0	1	0	1	1
0	1	1	1	1

6. Разберемся с двумя столбцами по середине; обратим внимание на вторую строчку таблицы, в которой одна из оставшихся переменных равна 1, а вторая – 0; так как функция равна 1, то $z + \overline{w} = 1$, откуда следует, что z = 1 и w = 0 (иначе сумма будет равна 0).

7. Ответ: yzwx.

Задача № 2. Логическая функция F задается выражением $(x \rightarrow y) \vee (y \rightarrow z)$. Ниже приведен фрагмент таблицы истинности. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z (таблица)

?	?	?	F
1	0	1	1
0	0	1	0

Решение:

1. Выражение представляет собой логическое произведение импликаций. Для истинности этого выражения обе импликации должны быть истинны.

2. Рассмотрим верхнюю строчку таблицы, где функция принимает значение 1. Здесь одна из переменных равна 0, а две другие равны 1.

3. Нулю в 1 строке может быть равен только x, так как при y = 0 получаем (1→0)Λ(0→1) = 0 Λ 1 = 0, а при z = 0 имеем (1→1)Λ(1→0) = 1 Λ 0 = 0, то есть эти два варианта не подходят. Таким образом, 2 столбец – x.

4. Рассмотрим вторую строку, там должен получиться 0. Мы знаем, что 2 столбец – x, поэтому во второй строке x = 0, и (0→y)Λ(y→z) = 0.

5. Первая импликация 0→y =1 независимо от значения y. Поэтому для того, чтобы все выражение было равно 0, нужно обеспечить y→z = 0.

6. Это условие сразу дает y = 1 и z = 0. Поэтому третий столбец – y, а первый – z.

7. Ответ: zxy

Лабораторная практическая работа 3

Основы теории чисел. Основы комбинаторики. Основы теории графов.

Цель работы. *Рассмотреть основы теории чисел, комбинаторики и теории графов, используемых в решении олимпиадных задач по информатике.*

Задания.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

1. Перестановки, размещения и сочетания множества.

Задачи для решения.

1. Элементами множества A являются натуральные числа. Известно, что выражение ¬(x ∈ {2, 4, 6, 8, 10, 12}) ∨ (¬(x ∈ {3, 6, 9, 12, 15}) → (x ∈ A)) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение произведения элементов множества A.

2. Элементами множества A являются натуральные числа. Известно, что выражение ¬(x ∈ {1, 2, 3, 4, 5, 6}) ∨ (¬(x ∈ {3, 6, 9, 12, 15}) → (x ∈ A)) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение суммы элементов множества A.

3. Элементами множества A являются натуральные числа. Известно, что выражение ((x ∈ {3, 5, 7, 11, 12, 15}) → (x ∈ {5, 6, 12, 15})) ∨ (x ∈ A) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение произведения элементов множества A.

4. Элементами множества A являются натуральные числа. Известно, что выражение ((x ∈ {1, 3, 5, 7, 9, 12}) → (x ∈ {3, 6, 9, 12})) ∨ (x ∈ A) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение суммы элементов множества A.

5. Элементами множества A являются натуральные числа. Известно, что выражение (x ∈ {2, 4, 8, 12, 15}) → ((x ∈ {3, 6, 8, 15}) ∨ (x ∈ A)) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение произведения элементов множества A.

6. Элементами множества A являются натуральные числа. Известно, что выражение ((x ∈ {3, 5, 7, 11, 12}) → ¬(x ∈ {5, 6, 12, 15})) ∨ (x ∈ A) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение суммы элементов множества A.

7. На числовой прямой даны два отрезка: P = [5, 10] и Q = [15, 18]. Выберите такой отрезок A, что формула ((x ∈ A) → (x ∈ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [3, 11] 2) [6, 10] 3) [8, 16] 4) [17, 23]

8. На числовой прямой даны два отрезка: P = [25, 30] и Q = [15, 20]. Выберите такой отрезок A, что формула ((x ∈ A) → (x ∈ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [10, 15] 2) [12, 30] 3) [20, 25] 4)[26, 28]

9. На числовой прямой даны два отрезка: P = [2, 20] и Q = [15, 30]. Выберите такой отрезок A, что формула ((x ∉ A) → (x ∉ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [0, 15] 2) [3, 20] 3) [10, 25] 4)[25, 40]

10. На числовой прямой даны два отрезка: P = [10, 25] и Q = [0, 12]. Выберите такой отрезок A, что формула ((x ∉ A) → (x ∉ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [10, 15] 2) [20, 35] 3) [5, 20] 4)[12, 40]

Лабораторная работа 4

Алгоритмы и их свойства.

Цель работы. Рассмотреть алгоритмы и их свойства и выделить способы их решения в олимпиадных задачах по информатике.

Задания.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задание 1 Выполнение практического задания «Рекурсия».

Задача 1. Вычислить n!

Решение:

```
program factorial;
var N: word;
function F (n:word): longint;
begin
 if n=0 then F:= else F:= n*F (n-1)
end;
begin
write (‘N=’);
readln (N);
writeln (‘N!=’, F (N))
end.
```

Задача 2. Дано целое число A. Вставить между некоторыми цифрами 1, 2, 3, 4, 5, 6, 7, 8, 9, записанными именно в таком порядке, знаки «+» и «-» так, чтобы значением получившегося выражения было число n. Например, если A=122, то подойдет выражение: 12+34-5-6+78+9.

Решение:

```
programzi;
var t: array [2..9] of string;
procedurezi_fra (s, r: longint; o:char; n:integer);
var newr: longint;
begin if o = '+' then newr:= r+s else newr:= r-s;
if n <= 9 then begin
t[n]:= '+'; zi_fra (n, newr, '+', n+1);
t[n]:= '-'; zi_fra (n, newr, '-', n+1);
t[n]:= ' '; zi_fra (s*10+n, r, o, n+1);
end
else// проверка равенства
if newr = A then begin for i:= 2 to 9 do write (i-1, t[i]);
writeln ('9');
end;
end;
begin readln (A);
zi_fra (1, 0, '+', 2);
end.
```

Задание 2 Выполнение практического задания «Числовые алгоритмы».

Задача 1. Сообщение о том, что ваш друг живет на 5 этаже, несет 4 бита информации. Сколько этажей в доме?

Решение: $N=2^i$; $2^4=16$ этажей.

Ответ: 16.

2. Проверить, поместится ли на диске компьютера музыкальная композиция, которая длится m минут и n секунд, если свободное дисковое пространство 6 мегабайт, а для записи одной секунды звука необходимо 16 килобайт.

Решение: переведем мегабайты в килобайты, так как 1 мегабайт = 1024 килобайта, то 6 мегабайт = 6144 килобайт. Обозначим t – время, в секундах, v – объем файла, в килобайтах, тогда: t=60*m+n; v=16*t.

Программа на Паскале будет иметь вид:

```
programmuzi;
varm, n, t, v: integer;
begin
writeln ('Введитемин');
readln (m, n);
t:=60*m+n;
v:=16*t;
if v<=6144 then writeln ('Композиция поместится')
else writeln ('Не хватает ', v-6144, ' килобайт');
end.
```

Задание 3 Выполнение практического задания «Алгоритмы на строках».

Задача 1. Счастливые билеты.

Создать программу определения числа билетов с 6-значными номерами, у которых сумма первых 3 десятичных цифр равна сумме 3 последних десятичных цифр.

Формат выходных данных.

Вывести количество счастливых билетов на экран.

Разбор задачи

Просматриваем числа от 0 до 999999. Делим число на 2 части: первые 3 цифры и последние 3 цифры, находим сумму цифр каждой из частей, сравниваем результат.

Код программы:

```
program z1;
function summ(x: longint): byte;{возвращает сумму цифр числа}
var k, l: byte;
y: longint;
begin
y := x; l := 0;
while (y <> 0) do
begin
k := y mod 10;
y := y div 10;
l := l + k end;
summ := l;
end;
var w1, w2, i, j, count: longint;
n, m: byte;
begin
count := 0;
for j := 0 to 999999 do begin w1 := j div 1000;
w2 := j mod 1000;
if summ(w1) = summ(w2) then
begin
count := count + 1;
 writeln(j, ' --> ', count);
end;
 end;
 end.
```

Задание 4 Выполнение практического задания «Алгоритмы на графах».

Задача 1. Каждый элемент квадратной матрицы размеренности N*N равен нулю, либо единицы. Найти количество групп, образованных единицами. Единицы относятся к одной группе, если из одной из них можно перейти к другой «наступая» на единицы, расположенные в соседних клетках. Соседними являются клетки, граничащие по горизонтали или вертикали.

Входные данные: в первой строке файла input.txtзаписано натуральное число N не больше 100 – размер квадратной матрицы. В следующих N строках задаются элементы матрицы через пробел.

Выходные данные: output.txt выведите единственное число – количество групп.

Решение: необходимо обойти матрицу и вычислить количество групп. После того, как мы попадаем в группу, надо это зафиксировать, увеличив переменную – результат на единицу. Чтобы второй раз не посчитать одну и ту же группу, сразу после посещения необходимо его уничтожить, то есть присвоить всем клеткам значение ноль.

Тест задачи не слишком мал, поэтому необходимо написать процедуру уничтожения группы, назовем ее “del”. Чтобы во время процедуры не выйти за пределы массива, сделаем его размером N+2 * N+2, а не размером N*N. Это дает нам возможность окружить искомый массив размером N*N нулями.

```
programgrup;
const m =102;
var a:array [1..m, 1..m] of integer;
 i, j, n, rez: integer;
input, output: text;
procedure del (i, j: integer);
begin
 ifa[i, j] <> 1 then exit;
a [i, j]:= 0;
del (i+1, j);
del (i-1, j);
del (i, j+1);
del (i, j-1);
end;
begin 40 rez:= 0;
assing (input, ‘input.txt’);
reset (input);
assing (output, ‘output.txt’);
rewrite (output);
read (input, n); // заполняем массив нулям
 for i:= 1 to n+2 do
for j:= 1 to n+2 do
a [i, j]:=0; // считать матрицу из файла
for i:= 2 to n+1 do
for j:= 2 to n+1 do
read (input, a[i, j]); // обход матрицы в поиске групп
for i:= 2 to n+1 do
for j:= 2 to n+1 do
if a[i,j] = 1 then begin inc (rez);
count (i, j);
end;
write (output, rez);
close (input);
close (output);
end.
```

Задание 5 Выполнение практического задания «Алгоритмы теории игр».

Задача 1. Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу один или четыре камня либо увеличить количество камней в куче в пять раз. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16, 19 или 75 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней. Игра завершается в тот момент, когда количество камней в куче становится не менее 73.

Победителем считается игрок, сделавший последний ход, т.е. первым получивший кучу, в которой будет 73 или больше камней.

В начальный момент в куче было *S* камней; $1 \le S \le 72$.

Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника.

Укажите такое значение *S*, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Назовите минимальное значение *S*, при котором это возможно.

Решение: Из условия мы знаем, что Петя не может выиграть своим первым ходом, но Ваня, независимо от хода Пети, выигрывает. Петя парень не глупый и, естественно, сделает самый маленький ход – добавит в кучу один камень (*S+1)*, чтобы Ваня от выигрыша был как можно дальше. Далее Ваня делает свой первый ход и выигрывает. Ване необходимо максимально увеличить предыдущее значение камней в куче, следовательно, Ваня увеличивает количество камней в куче в пять раз (*S+1)*5*. Теперь просто решаем неравенство:

$(S+1)*5 \ge 73$

$5S \ge 70$

$S \ge 13{,}6$

$S = 14$

Ответ: 14**.**

Лабораторная работа 5

Программирование

Цель работы. Рассмотреть способы использования языков программирования для решения олимпиадных задач по информатике.

Задания.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задание 1 Выполнение практического задания «Классические задачи программирования».

Пример задачи:

Сортировка методом “пузырьков”.

1. Функция для сортировки:

```
Public Function сортировка(массив() As Variant) As Boolean
Dim a As Boolean, t As Variant, x As Integer
For x = LBound(массив) To UBound(массив)
If IsNull(массив(X)) Then Exit Function
Next x
Do
a = False
For x = UBound(массив) To (LBound(массив) + 1) Step -1
If массив(x - 1) > массив(x) Then
t = массив(x - 1)
```

```
массив(x - 1) = массив(x)
массив(x) = t
a = True
End If
Next x
For x = (LBound(массив) + 1) To UBound(массив)
If массив(x - 1) > массив(x) Then
t = массив(x - 1)
массив(x - 1) = массив(x)
массив(x) = t
a = True
End If
Next x
Loop While a
сортировка = True
End Function
```

Задание 2 Выполнение практического задания «Классические задачи динамического программирования».

Задача 1. Вычислить значение суммы S=1/1!+1/2!+…+ 1/k!

Решение: можно, конечно, каждый раз вычислять очередное p!, затем 1/p!, и полученное слагаемое добавлять к сумме. Но обратим внимание на следующее очевидное равенство: 1/(p+1)! = (1/p!)/(p+1), И программа вычисления запишется следующим образом:

```
Program_sum;
var i, p: longint;
S:real;
begin
S:=1;
p:=1;
for i:= 2 to k do begin p:= p/I; S:=S+p;
end;
writeln('S=', S:8:2);
 end.
```

Задача 2. На числовой прямой школьник Ваня может идти вправо на одну или две единицы. Первоначально Ваня находится в точке с координатой 0. Определите количество маршрутов Вани, приводящих его в точку с координатой n.

Обозначим количество маршрутов Вани, ведущих в точку с координатой n, как F(n). Теперь вычислим функцию F(n). Заметим, что F(0)=1 (это вырожденный случай, существует ровно один маршрут из точки 0 в точку 0 – он не содержит ни одного прыжка), F(1)=1, F(2)=2. Как вычислить F(n)? В точку n школьник может попасть двумя способами – из точки n–2 при помощи прыжка длиной 2 и из точки n-1 прыжком длины 1. То есть число способов попасть в точку n равно сумме числа способов попасть в точку n-1 и n-2, что позволяет выписать рекуррентное соотношение: F(n) = F (n-2) + F (n-1) , верное для всех n> 2.

Решение на языке Pascal в виде рекурсивной функции:

```
function f (n: longint): longint;
begin
if n < 2 then
f:=1 else f:= f (n-1) + f (n-2);
```

```
end;
```

Вычисление решения этой функции, например, для n=20, оказывается, что функция работает крайне медленно. То есть одна из причин неэффективности рекурсивного решения – одно и то же значение функции вычисляется несколько раз, так как оно используется для вычисления нескольких других значений функции.

В данном случае, значения рекурсивной функции совпадает с числами Фибоначчи, так как вычисляются по тем же рекуррентным соотношениям. Для вычисления чисел Фибоначчи можно использовать цикл.

Решение:

```
var F: array [0..100] of longint;
i, n: longint;
begin
readln (n);
for i:= 1 to n do F [i]:=0;
 F[0]:=1; F[1]:=1;
 for i:= 2 to n do
F[i]:= F [i – 1] + F [i – 2];
writeln (F [n]);
end.
```

Динамическое программирование использует те же рекуррентные соотношения, что и рекурсивное решение, но в отличии от рекурсии в динамическом программировании значения вычисляются в цикле и сохраняются в списке.

Владеть: методическими приемами и современными образовательными технологиями для решения олимпиадных задач.

Задания, необходимые для оценивания сформированности ПК-8 на продвинутом уровне

Перечень вопросов для конспекта

1. История олимпиадного движения. Олимпиады: сущность понятия, виды и функции.
2. Нормативно-правовая база организации олимпиад по информатике.
3. Технические вопросы проведения олимпиад по информатике.
4. Методические рекомендации по подготовке к участию в олимпиадах по информатике.
5. Всероссийская олимпиада школьников по информатике: основные правила и требования к проведению.
6. Применение информационно-коммуникационных технологий в современном процессе подготовки и организации олимпиад по информатике.
7. Интренет-ресурсы с коллекциями олимпиадных задач по информатике.
8. Сайты интернет-олимпиад, конкурсов и олимпиад для школьников по информатике.
9. Основные построения теории рекуррентных соотношений и способы их решения.
10. Формальные логические доказательства и логическое рассуждение при моделировании алгоритмов.
11. Основные алгоритмы теории чисел. Числовые алгоритмы. Алгоритмы на строках.
12. Основные понятия и вычислительные формулы комбинаторики. Классические комбинаторные алгоритмы.

13. Основные понятия и факты из теории графов. Типы графов. Операции над графами.
14. Рекурсия.
15. Алгоритмы теории игр. Простейшие игры.
16. Динамическое программирование.

Перечень практических работ

Тема 1. Олимпиады по информатике

Задание 1.1 Выполнение практического задания «Обзор олимпиад по информатике для школьников».

Задание 1.2 Выполнение практического задания «Технологии решения олимпиадных задач».

Задание 1.3 Выполнение практического задания «Формы подготовки учащихся к олимпиаде по информатике».

Тема 2. Математические основы информатики

Задание 2.1 Выполнение практического задания «Построение теории рекуррентных соотношений и способы их решения».

Задание 2.2 Выполнение практического задания «Формальные логические доказательства и логическое рассуждение при моделировании алгоритмов».

Тема 3. Основы теории чисел. основы комбинаторики. Основы теории графов

Задание 3.1 Выполнение практического задания «Перестановки, размещения и сочетания множества».

Задание 3.2 Выполнение практического задания «Теория графов".

Тема 4. Алгоритмы и их свойства.

Задание 4.1 Выполнение практического задания «Рекурсия».

Задание 4.2 Выполнение практического задания «Числовые алгоритмы».

Задание 4.3 Выполнение практического задания «Алгоритмы на строках».

Задание 4.4 Выполнение практического задания «Алгоритмы на графах».

Задание 4.5 Выполнение практического задания «Алгоритмы теории игр».

Тема 5. Программирование.

Задание 5.1 Выполнение практического задания «Классические задачи программирования».

Задание 5.2 Выполнение практического задания «Классические задачи динамического программирования».

Задания для отчета по лабораторным практическим работам

Лабораторная практическая работа 1

Нормативно-правовая база организации олимпиад по информатике. Технические вопросы проведения олимпиад по информатике

Цель работы. Знакомство с нормативно-правовыми нормами и техническими вопросами организации олимпиад по информатике, а также методическими особенностями подготовки школьников к участию в олимпиадах по информатике.

Задания.

1. Провести анализ нормативно-правовых норм организации олимпиад по информатике (выполнить конспект).

2. Сделать обзор олимпиад по информатике для школьников, на основе интернет-ресурсов (тип олимпиады, статус, уровень олимпиады, сроки проведения, требования к знаниям, умениям и владениям школьников по информатике)

Интернет-ресурсы:

- Образовательный цент «Взлёт» – Режим доступа: https://olympmo.ru/
- Олимпиады для школьников - Режим доступа: https://olimpiada.ru/
- Сайт «Олимпиадные задачи по программированию» – Режим доступа: http://algolist.manual.ru/olimp;
- Московская олимпиада школьников по информатике – Режим доступа: https://mos-inf.olimpiada.ru/ ;
- Олимпиады по информатике. Санкт-Петербург, Россия – Режим доступа: http://neerc.ifmo.ru/school ;

3. Описать типы и виды олимпиадных задач, рассмотрев основные технологии их решения на примере Всероссийской олимпиады школьников по информатике (подготовить конспект и презентацию).

4. Рассмотреть формы подготовки обучающихся к олимпиаде по информатике; выделить методические особенности организации подготовки школьников к участию в олимпиадах по информатике.

Цель работы. Проектирование элективного курса подготовки школьников к участию в олимпиаде по информатике для 7-9 класса.

Задание.

1. Создать методические документы, моделирующие педагогическую систему обучения элективному курсу (название курса, цели его преподавания, тематическое планирование курса с указанием названия занятий, их типов, методов обучения (включая методы персонализированного обучения) источников информации, межпредметных и внутрипредметных связей).

2. Создать презентацию курса.

3. Подготовить подробный конспект одного из занятий, предусмотренных тематическим планом и дидактические материалы к нему в электронной форме, используя необходимые информационные технологии, в том числе дистанционные.

Лабораторная практическая работа 2
Математические основы информатики

Цель работы. Рассмотреть основы вычислений и построения теории рекуррентных соотношений, выделить способы их решения; формальные логические доказательства и логическое рассуждение при моделировании алгоритмов в олимпиадных задачах по информатике.

Задания.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

1. Построение теории рекуррентных соотношений и способы их решения.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задача 1. Вычислить значение суммы $S = 1/1! + 1/2! + ... + 1/k!$

Решение: возможно, каждый раз вычислять очередное p!, затем 1/p!, и полученное слагаемое добавлять к сумме, но обратим внимание на следующее очевидное равенство: $1/(p+1)! = (1/p!)/(p+1)$, и программа вычисления запишется следующим образом

```
S:=1; p:=1; {S = p = 1/1!}
for i:=2 to k do
begin
p:=p/i;
S:=S+p;
end;
```

Задача 2. Написать программу определения количества шестизначных «счастливых» билетов, у которых сумма первых 3 десятичных цифр равна сумме 3 последних десятичных цифр.

Решение:

1) Выполним перебор всех возможных комбинаций шести цифр и подсчитаем число "счастливых" билетов.

```
Count:=0; {количество "счастливых" билетов}
for a1:=0 to 9 do
for a2:=0 to 9 do
for a3:=0 to 9 do
for a4:=0 to 9 do
for a5:=0 to 9 do
for a6:=0 to 9 do
if a1+a2+a3=a4+a5+a6
then Count:=Count+1;
```

Условие if во вложенных циклах будет проверяться 10^6 раз, поэтому будем говорить, что сложность этого алгоритма 10^6.

2) Обратим внимание на то, что в "счастливом" билете последняя цифра a6 однозначно определяется первыми пятью:

a6=(a1+a2+a3)-(a4+a5).

Если 0<=a6<=9, то билет "счастливый", иначе - нет. Таким образом, мы можем убрать шестой вложенный цикл:

```
Count:=0;
for a1:=0 to 9 do
for a2:=0 to 9 do
for a3:=0 to 9 do
for a4:=0 to 9 do
for a5:=0 to 9 do
begin
a6:=(a1+a2+a3)-(a4+a5);
if (a6>=0) and (a6<=9)
then Count:=Count+1;
end;
```

Сложность алгоритма 10^5.

3) Если комбинаций a1 a2 a3 первых трех цифр с суммой T=a1+a2+a3 насчитывается C[T], то всего "счастливых" билетов с суммой половины T=a1+a2+a3=a4+a5+a6 будет C[T]*C[T]. Всех возможных сумм T-28 (от 0=0+0+0 до 27=9+9+9). Подсчитаем C[i], i=0, ..., 28, затем найдем интересующее нас количество "счастливых" билетов

$C[0]^2 + C[1]^2 + ... + C[27]^2$.

Заметим, что "счастливых" билетов с суммой T столько же, сколько и с суммой 27-T. Действительно, если билет a1 a2 a3 a4 a5 a6 с суммой T - "счастливый", то таковым же является и билет (999999 - a1 a2 a3 a4 a5 a6) с суммой 27-T. Поэтому число билетов можно вычислять и по формуле

$2*(C[0]^2+ ... +C[13]^2)$, т.е. рассматривать только суммы T от 0 до 13.

```
var C:array[0..13] of longint;
Count:=0;
for T:=0 to 13 do C[T]:=0;
for a1:=0 to 9 do {перебираем все}
for a2:=0 to 9 do {возможные a1 a2 a3}
for a3:=0 to 9 do
begin
T:=a1+a2+a3;
C[T]:=C[T]+1 {нашли еще один билет}
end; {с суммой T}
for T:=0 to 13 do {считаем число билетов} Count:=Count+C[T]*C[T];
Count:=Count*2; {удваиваем сумму}
```

Сложность этого алгоритма 10^3.

4) В пункте 3 мы перебирали комбинации цифр и искали количество комбинаций с суммами C[T]. Сейчас мы пойдем от суммы T, и по ней будем определять, какое количество комбинаций a1 a2 a3 ее имеет.

Итак T=a1+a2+a3.

Минимальное значение, которое может принимать a1, - это MAX{0,T-18}. Член T-18 появляется из следующих соображений: пусть a2=a3=9, тогда a1=T-18, но a1 не может быть меньше 0. Максимальное значение a1=MIN{9,T} (так как a2 и a3 неотрицательны, то a1<=T и одновременно a1<=9).

Для цифры a2 аналогично получаем, что она лежит в пределах от max{0,T-a1-9} до min{9,T-a1}.

Цифра a3 по T, a1 и a2 определяется однозначно.

Получаем, что комбинаций a1 a2 a3 с суммой T и с первой цифрой a1 столько же, сколько возможных цифр a2, а именно min{9,T-a1}-max{0,T-a1-9}+1.

Как и в пункте 3 мы можем рассматривать диапазон сумм T от 0 до 13.

```
Count:=0;
for T:=0 to 13 do
begin
CT:=0;
for a1:=max(0,T-18) to min(9,T) do CT:=CT+min(9,T-a1)-max(0,T-a1-9)+1;
Count:=Count+CT*CT
end;
Count:=Count*2;
```

Сложность этого алгоритма (т.е. количество выполнений операций присваивания внутри двух вложенных циклов) не превышает 140.

Задача 3:
Фишка может двигаться по полю длины N только вперед. Длина хода фишки не более K. Найти число различных путей, по которым фишка может пройти поле от начала до конца.
Пример. N=3, K=2
Возможные пути:
1,1,1
1,2
2,1
Ответ: 3.

2. Формальные логические доказательства и логическое рассуждение при моделировании алгоритмов.

Отчет представить в виде следующего алгоритма:
1. Проведите анализ задачи. Какими методами ее можно решить.
2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.

5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задача № 1. Логическая функция F задается выражением ¬x V y V (¬z Λ w). На рисунке приведен фрагмент таблицы истинности функции F, содержащий все наборы аргументов, при которых функция F ложна. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z, w (таблица).

?	?	?	?	F
0	0	0	1	0
0	1	0	1	0
0	1	1	1	0

В ответе напишите буквы x, y, z, w в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Решение (I способ):

1. Запишем выражение в более понятной форме: $F = \overline{x} + y + \overline{z} * w$.

2. Функция F представляет собой логическое сложение, состоящие из трех операндов, где третий операнд представлен логическим умножением из двух операндов. Для того, чтобы функция $F = \overline{x} + y + \overline{z} * w$ была ложна (все операнды сложения должны быть ложны), необходимо, чтобы x всегда был равен 1 (при логической инверсии всегда будет 0), а y всегда был равен 0; поэтому x – это последний столбец в таблице, а y – первый (таблица):

y	?	?	x	F
0	0	0	1	0
0	1	0	1	0
0	1	1	1	0

3. Разберемся с двумя столбцами по середине; обратим внимание на вторую строчку таблицы, в которой одна из оставшихся переменных равна 1, а вторая – 0; так как функция равна 0, то $\overline{z} * w = 0$, откуда следует, что z = 1 и w = 0 (иначе произведение будет равно 1). 4. Ответ: yzwx.

Решение (II способ – инверсия выражения):

1. Запишем выражение в более понятной форме: $F = \overline{x} + y + \overline{z} * w$.

2. Попытаемся свести задачу к уже известной задаче; если при каком-то наборе аргументов функция F ложна, то ее обратная функция $\overline{F}$, истинна.

3. Построим для функции F обратную функцию $\overline{F}$, используя законы де Моргана: $\overline{F} = \overline{\overline{x}+\overline{y}+\overline{z}*\overline{w}} = x * \overline{y} * (z + \overline{w})$.

4. Тогда при тех же значениях аргументов функция $\overline{F}$ истинна (таблица)

?	?	?	?	$\overline{F}$
0	0	0	1	1
0	1	0	1	1
0	1	1	1	1

5. Для истинности функции $\overline{F} = x * \overline{y} * (z + \overline{w})$ необходимо, чтобы x всегда был равен 1, а y всегда был равен 0; поэтому x – это последний столбец в таблице, а y – первый (таблица):

y	?	?	x	$\overline{F}$
0	0	0	1	1
0	1	0	1	1
0	1	1	1	1

6. Разберемся с двумя столбцами по середине; обратим внимание на вторую строчку таблицы, в которой одна из оставшихся переменных равна 1, а вторая – 0; так как функция равна 1, то $z + \overline{w} = 1$, откуда следует, что z = 1 и w = 0 (иначе сумма будет равна 0).

7. Ответ: yzwx.

Задача № 2. Логическая функция F задается выражением $(x \rightarrow y) \vee (y \rightarrow z)$. Ниже приведен фрагмент таблицы истинности. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z (таблица)

?	?	?	F
1	0	1	1
0	0	1	0

Решение:

1. Выражение представляет собой логическое произведение импликаций. Для истинности этого выражения обе импликации должны быть истинны.

2. Рассмотрим верхнюю строчку таблицы, где функция принимает значение 1. Здесь одна из переменных равна 0, а две другие равны 1.

3. Нулю в 1 строке может быть равен только x, так как при y = 0 получаем $(1 \rightarrow 0) \wedge (0 \rightarrow 1) = 0 \wedge 1 = 0$, а при z = 0 имеем $(1 \rightarrow 1) \wedge (1 \rightarrow 0) = 1 \wedge 0 = 0$, то есть эти два варианта не подходят. Таким образом, 2 столбец – x.

4. Рассмотрим вторую строку, там должен получиться 0. Мы знаем, что 2 столбец – x, поэтому во второй строке x = 0, и $(0 \rightarrow y) \wedge (y \rightarrow z) = 0$.

5. Первая импликация $0 \rightarrow y = 1$ независимо от значения y. Поэтому для того, чтобы все выражение было равно 0, нужно обеспечить $y \rightarrow z = 0$.

6. Это условие сразу дает y = 1 и z = 0. Поэтому третий столбец – y, а первый – z.

7. Ответ: zxy

Лабораторная практическая работа 3

Основы теории чисел. Основы комбинаторики. Основы теории графов.

Цель работы. *Рассмотреть основы теории чисел, комбинаторики и теории графов, используемых в решении олимпиадных задач по информатике.*

Задания.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.

2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

1. Перестановки, размещения и сочетания множества.

Задачи для решения.

1. Элементами множества A являются натуральные числа. Известно, что выражение ¬(x ∈ {2, 4, 6, 8, 10, 12}) ∨ (¬(x ∈ {3, 6, 9, 12, 15}) → (x ∈ A)) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение произведения элементов множества A.

2. Элементами множества A являются натуральные числа. Известно, что выражение ¬(x ∈ {1, 2, 3, 4, 5, 6}) ∨ (¬(x ∈ {3, 6, 9, 12, 15}) → (x ∈ A)) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение суммы элементов множества A.

3. Элементами множества A являются натуральные числа. Известно, что выражение ((x ∈ {3, 5, 7, 11, 12, 15}) → (x ∈ {5, 6, 12, 15})) ∨ (x ∈ A) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение произведения элементов множества A.

4. Элементами множества A являются натуральные числа. Известно, что выражение ((x ∈ {1, 3, 5, 7, 9, 12}) → (x ∈ {3, 6, 9, 12})) ∨ (x ∈ A) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение суммы элементов множества A.

5. Элементами множества A являются натуральные числа. Известно, что выражение (x ∈ {2, 4, 8, 12, 15}) → ((x ∈ {3, 6, 8, 15}) ∨ (x ∈ A)) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение произведения элементов множества A.

6. Элементами множества A являются натуральные числа. Известно, что выражение ((x ∈ {3, 5, 7, 11, 12}) → ¬(x ∈ {5, 6, 12, 15})) ∨ (x ∈ A) истинно (т. е. принимает значение 1) при любом значении переменной x. Определите наименьшее возможное значение суммы элементов множества A.

7. На числовой прямой даны два отрезка: P = [5, 10] и Q = [15, 18]. Выберите такой отрезок A, что формула ((x ∈ A) → (x ∈ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [3, 11] 2) [6, 10] 3) [8, 16] 4) [17, 23]

8. На числовой прямой даны два отрезка: P = [25, 30] и Q = [15, 20]. Выберите такой отрезок A, что формула ((x ∈ A) → (x ∈ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [10, 15] 2) [12, 30] 3) [20, 25] 4)[26, 28]

9. На числовой прямой даны два отрезка: P = [2, 20] и Q = [15, 30]. Выберите такой отрезок A, что формула ((x ∉ A) → (x ∉ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [0, 15] 2) [3, 20] 3) [10, 25] 4)[25, 40]

10. На числовой прямой даны два отрезка: P = [10, 25] и Q = [0, 12]. Выберите такой отрезок A, что формула ((x ∉ A) → (x ∉ P)) ∨ (x ∈ Q) тождественно истинна, то есть принимает значение 1 при любом значении переменной x: 1) [10, 15] 2) [20, 35] 3) [5, 20] 4)[12, 40]

Лабораторная работа 4

Алгоритмы и их свойства.

Цель работы. Рассмотреть алгоритмы и их свойства и выделить способы их решения в олимпиадных задачах по информатике.

Задания.

Отчет представить в виде следующего алгоритма:
1. Проведите анализ задачи. Какими методами ее можно решить.

2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задание 1 Выполнение практического задания «Рекурсия».

Задача 1. Вычислить n!

Решение:

```
program factorial;
var N: word;
function F (n:word): longint;
begin
 if n=0 then F:= else F:= n*F (n-1)
end;
begin
write (‘N=’);
readln (N);
writeln (‘N!=’, F (N))
end.
```

Задача 2. Дано целое число A. Вставить между некоторыми цифрами 1, 2, 3, 4, 5, 6, 7, 8, 9, записанными именно в таком порядке, знаки «+» и «-» так, чтобы значением получившегося выражения было число n. Например, если A=122, то подойдет выражение: 12+34-5-6+78+9.

Решение:

```
programzi;
var t: array [2..9] of string;
procedurezi_fra (s, r: longint; o:char; n:integer);
var newr: longint;
begin if o = ‘+’ then newr:= r+s else newr:= r-s;
if n <= 9 then begin
t[n]:= ‘+’; zi_fra (n, newr, ‘+’, n+1);
t[n]:= ‘-’; zi_fra (n, newr, ‘-’, n+1);
t[n]:= ‘ ’; zi_fra (s*10+n, r, o, n+1);
end
else// проверка равенства
if newr = A then begin for i:= 2 to 9 do write (i-1, t[i]);
writeln (‘9’);
end;
end;
begin readln (A);
zi_fra (1, 0, ‘+’, 2);
end.
```

Задание 2 Выполнение практического задания «Числовые алгоритмы».

Задача 1. Сообщение о том, что ваш друг живет на 5 этаже, несет 4 бита информации. Сколько этажей в доме?

Решение: $N=2^i$; $2^4=16$ этажей.

Ответ: 16.

2. Проверить, поместится ли на диске компьютера музыкальная композиция, которая длится m минут и n секунд, если свободное дисковое пространство 6 мегабайт, а для записи одной секунды звука необходимо 16 килобайт.

Решение: переведем мегабайты в килобайты, так как 1 мегабайт = 1024 килобайта, то 6 мегабайт = 6144 килобайт. Обозначим t – время, в секундах, v – объем файла, в килобайтах, тогда: t=60*m+n; v=16*t.

Программа на Паскале будет иметь вид:

```
programmuzi;
varm, n, t, v: integer;
begin
writeln (‘Введитемин’);
readln (m, n);
t:=60*m+n;
v:=16*t;
if v<=6144 then writeln (‘Композиция поместится’)
else writeln (‘Не хватает ’, v-6144, ‘ килобайт’);
end.
```

Задание 3 Выполнение практического задания «Алгоритмы на строках».

Задача 1. Счастливые билеты.

Создать программу определения числа билетов с 6-значными номерами, у которых сумма первых 3 десятичных цифр равна сумме 3 последних десятичных цифр.

Формат выходных данных.

Вывести количество счастливых билетов на экран.

Разбор задачи

Просматриваем числа от 0 до 999999. Делим число на 2 части: первые 3 цифры и последние 3 цифры, находим сумму цифр каждой из частей, сравниваем результат.

Код программы:

```
program z1;
function summ(x: longint): byte;{возвращает сумму цифр числа}
var k, l: byte;
y: longint;
begin
y := x; l := 0;
while (y <> 0) do
begin
k := y mod 10;
y := y div 10;
l := l + k end;
summ := l;
end;
var w1, w2, i, j, count: longint;
n, m: byte;
begin
count := 0;
for j := 0 to 999999 do begin w1 := j div 1000;
w2 := j mod 1000;
```

```
if summ(w1) = summ(w2) then
begin
count := count + 1;
 writeln(j, ' --> ', count);
end;
 end;
 end.
```

Задание 4 Выполнение практического задания «Алгоритмы на графах».

Задача 1. Каждый элемент квадратной матрицы размеренности N*N равен нулю, либо единицы. Найти количество групп, образованных единицами. Единицы относятся к одной группе, если из одной из них можно перейти к другой «наступая» на единицы, расположенные в соседних клетках. Соседними являются клетки, граничащие по горизонтали или вертикали.

Входные данные: в первой строке файла input.txtзаписано натуральное число N не больше 100 – размер квадратной матрицы. В следующих N строках задаются элементы матрицы через пробел.

Выходные данные: output.txt выведите единственное число – количество групп.

Решение: необходимо обойти матрицу и вычислить количество групп. После того, как мы попадаем в группу, надо это зафиксировать, увеличив переменную – результат на единицу. Чтобы второй раз не посчитать одну и ту же группу, сразу после посещения необходимо его уничтожить, то есть присвоить всем клеткам значение ноль.

Тест задачи не слишком мал, поэтому необходимо написать процедуру уничтожения группы, назовем ее “del”. Чтобы во время процедуры не выйти за пределы массива, сделаем его размером N+2 * N+2, а не размером N*N. Это дает нам возможность окружить искомый массив размером N*N нулями.

```
programgrup;
const m =102;
var a:array [1..m, 1..m] of integer;
 i, j, n, rez: integer;
input, output: text;
procedure del (i, j: integer);
begin
 ifa[i, j] <> 1 then exit;
a [i, j]:= 0;
del (i+1, j);
del (i-1, j);
del (i, j+1);
del (i, j-1);
end;
begin 40 rez:= 0;
assing (input, ‘input.txt’);
reset (input);
assing (output, ‘output.txt’);
rewrite (output);
read (input, n); // заполняем массив нулям
 for i:= 1 to n+2 do
for j:= 1 to n+2 do
a [i, j]:=0; // считать матрицу из файла
for i:= 2 to n+1 do
```

```
for j:= 2 to n+1 do
read (input, a[i, j]); // обход матрицы в поиске групп
for i:= 2 to n+1 do
for j:= 2 to n+1 do
if a[i,j] = 1 then begin inc (rez);
count (i, j);
end;
write (output, rez);
close (input);
close (output);
end.
```

Задание 5 Выполнение практического задания «Алгоритмы теории игр».

Задача 1. Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу один или четыре камня либо увеличить количество камней в куче в пять раз. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16, 19 или 75 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней. Игра завершается в тот момент, когда количество камней в куче становится не менее 73.

Победителем считается игрок, сделавший последний ход, т.е. первым получивший кучу, в которой будет 73 или больше камней.

В начальный момент в куче было *S* камней; $1 \le S \le 72$.

Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника.

Укажите такое значение *S*, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Назовите минимальное значение *S*, при котором это возможно.

Решение: Из условия мы знаем, что Петя не может выиграть своим первым ходом, но Ваня, независимо от хода Пети, выигрывает. Петя парень не глупый и, естественно, сделает самый маленький ход – добавит в кучу один камень (*S+1)*, чтобы Ваня от выигрыша был как можно дальше. Далее Ваня делает свой первый ход и выигрывает. Ване необходимо максимально увеличить предыдущее значение камней в куче, следовательно, Ваня увеличивает количество камней в куче в пять раз (*S+1)*5*. Теперь просто решаем неравенство:

$(S+1)*5 \ge 73$

$5S \ge 70$

$S \ge 13,6$

$S = 14$

Ответ: 14**.**

Лабораторная работа 5

Программирование

Цель работы. Рассмотреть способы использования языков программирования для решения олимпиадных задач по информатике.

Задания.

Отчет представить в виде следующего алгоритма:

1. Проведите анализ задачи. Какими методами ее можно решить.

2. Выберите оптимальный метод решения, объясните свой выбор.
3. Составьте и расскажите алгоритм решения задачи, какие используются переменные, типы данных, функции.
4. Напишите программу на языке программирования.
5. Расскажите о типичных ошибках, возникающих при написании и выполнении подобной программы.
6. Приведите примеры задач данного типа.

Задание 1 Выполнение практического задания «Классические задачи программирования».

Пример задачи:

Сортировка методом “пузырьков”.

1. Функция для сортировки:

```
Public Function сортировка(массив() As Variant) As Boolean
Dim a As Boolean, t As Variant, x As Integer
For x = LBound(массив) To UBound(массив)
If IsNull(массив(X)) Then Exit Function
Next x
Do
a = False
For x = UBound(массив) To (LBound(массив) + 1) Step -1
If массив(x - 1) > массив(x) Then
t = массив(x - 1)
массив(x - 1) = массив(x)
массив(x) = t
a = True
End If
Next x
For x = (LBound(массив) + 1) To UBound(массив)
If массив(x - 1) > массив(x) Then
t = массив(x - 1)
массив(x - 1) = массив(x)
массив(x) = t
a = True
End If
Next x
Loop While a
сортировка = True
End Function
```

Задание 2 Выполнение практического задания «Классические задачи динамического программирования».

Задача 1. Вычислить значение суммы S=1/1!+1/2!+…+ 1/k!

Решение: можно, конечно, каждый раз вычислять очередное p!, затем 1/p!, и полученное слагаемое добавлять к сумме. Но обратим внимание на следующее очевидное равенство: 1/(p+1)! = (1/p!)/(p+1), И программа вычисления запишется следующим образом:

```
Program_sum;
var i, p: longint;
S:real;
begin
S:=1;
p:=1;
for i:= 2 to k do begin p:= p/I; S:=S+p;
```

```
end;
writeln('S=', S:8:2);
 end.
```

Задача 2. На числовой прямой школьник Ваня может идти вправо на одну или две единицы. Первоначально Ваня находится в точке с координатой 0. Определите количество маршрутов Вани, приводящих его в точку с координатой n.

Обозначим количество маршрутов Вани, ведущих в точку с координатой n, как F(n). Теперь вычислим функцию F(n). Заметим, что F(0)=1 (это вырожденный случай, существует ровно один маршрут из точки 0 в точку 0 – он не содержит ни одного прыжка), F(1)=1, F(2)=2. Как вычислить F(n)? В точку n школьник может попасть двумя способами – из точки n–2 при помощи прыжка длиной 2 и из точки n-1 прыжком длины 1. То есть число способов попасть в точку n равно сумме числа способов попасть в точку n-1 и n-2, что позволяет выписать рекуррентное соотношение: F(n) = F (n-2) + F (n-1) , верное для всех n> 2.

Решение на языке Pascal в виде рекурсивной функции:

```
function f (n: longint): longint;
begin
if n < 2 then
f:=1 else f:= f (n-1) + f (n-2);
end;
```

Вычисление решения этой функции, например, для n=20, оказывается, что функция работает крайне медленно. То есть одна из причин неэффективности рекурсивного решения – одно и то же значение функции вычисляется несколько раз, так как оно используется для вычисления нескольких других значений функции.

В данном случае, значения рекурсивной функции совпадает с числами Фибоначчи, так как вычисляются по тем же рекуррентным соотношениям. Для вычисления чисел Фибоначчи можно использовать цикл.

Решение:

```
var F: array [0..100] of longint;
i, n: longint;
begin
readln (n);
for i:= 1 to n do F [i]:=0;
 F[0]:=1; F[1]:=1;
 for i:= 2 to n do
F[i]:= F [i – 1] + F [i – 2];
writeln (F [n]);
end.
```

Динамическое программирование использует те же рекуррентные соотношения, что и рекурсивное решение, но в отличии от рекурсии в динамическом программировании значения вычисляются в цикле и сохраняются в списке.

Промежуточная аттестация

ПК-8 - Способен организовывать образовательный процесс с использованием современных образовательных технологий, в том числе дистанционных.
Знать:

- основные виды олимпиад по информатике для школьников;
- основные понятия теоретической информатики, теории чисел, теории графов и программирования

Уметь:
- применять технологии для решения олимпиадных задач;

Владеть:
- методическими приемами и современными образовательными технологиями для решения олимпиадных задач.

Перечень вопросов для зачета

1. Нормативно-правовая база организации олимпиад по информатике.
2. Обзор олимпиад по информатике.
3. Техническое сопровождение олимпиад по информатике.
4. Системы автоматизированного проведения турниров.
5. Особенности подготовки задач для олимпиад по информатике.
6. Методические особенности организации подготовки школьников к участию в олимпиадах по информатике.
7. Типовые алгоритмы решения задач по теме ««Рекурсия».
8. Типовые алгоритмы решения задач по теме «Числовые алгоритмы».
9. Типовые алгоритмы решения задач по теме «Алгоритмы на строках».
10. Типовые алгоритмы решения задач по теме «Алгоритмы теории игр».
11. Типовые алгоритмы решения задач по теме «Рекурсивные алгоритмы».
12. Построение и решение рекуррентных соотношений.
13. Теоретико-числовые алгоритмы.
14. Целочисленная арифметика. Алгоритмы для работы с большими числами.
15. Сортировка. Вычислительная сложность основных алгоритмов сортировки.
16. Основные комбинаторные алгоритмы. Перебор.
17. Структуры данных.
18. Поиск. Строки и последовательности.
19. Графы и маршруты.

4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Общее количество баллов по дисциплине – 100 баллов.

Максимальное количество баллов, которое можно набрать в течение семестра за выполнение лабораторных работ, тестирование и самостоятельную работу (написание конспектов) – 70 баллов.

За выполнение лабораторных работ обучающийся может набрать максимально 30 баллов.

За тестирование обучающийся может набрать максимально 20 баллов.

За написание конспектов 20 баллов.

Зачет с оценкой

К зачету допускаются студенты, выполнившие все лабораторные работы, задания для самостоятельной работы, тесты и набравшие не менее 40 баллов.

На зачете с оценкой студент получает 1 вопрос и 1 задачу. Готовит ответ 30-40 минут, отвечает преподавателю подготовленные теоретический вопрос и решение задачи на компьютере. Для задачи студент должен дать методический анализ. Студент должен быть готов ответить на дополнительные вопросы.

Шкала оценивания зачета с оценкой

Критерии оценивания	**Баллы**
Ставится, если студент обнаруживает всестороннее, систематическое и глубокое знание программного материала по дисциплине; обстоятельно анализирует структурную взаимосвязь рассматриваемых тем и разделов дисциплины; усвоил основную и знаком с дополнительной литературой, рекомендованной программой, а также усвоил взаимосвязь основных понятий дисциплины в их значении для приобретаемой профессии; проявил творческие способности в понимании, изложении и использовании учебного материала.	26-30
Ставится, если студент, обнаруживает полное знание программного материала, успешно выполняет предусмотренные в программе задания; усвоил основную литературу, рекомендованную в программе; показал систематический характер знаний по дисциплине и способен к их самостоятельному пополнению и обновлению в ходе дальнейшей образовательной деятельности.	21-25
Ставится, если студент обнаруживает знание основного программного материала в объеме, необходимом для дальнейшего обучения и профессиональной деятельности; справляется с выполнением заданий, предусмотренных программой; знаком с основной литературой, рекомендованной программой; допускает погрешности непринципиального характера в ответе на зачете с оценкой.	16-20
Ставится в том случае, если студент обнаруживает пробелы в знаниях основного программного материала, допускает принципиальные ошибки в выполнении предусмотренных программой заданий.	0-15

Итоговая шкала оценивания результатов освоения дисциплины.

Итоговая оценка по дисциплине выставляется по приведенной ниже шкале. При выставлении итоговой оценки преподавателем учитывается работа обучающегося в течение освоения дисциплины, а также оценка по промежуточной аттестации.

Количество баллов	**Оценка по традиционной шкале**
81-100	Отлично
61-80	Хорошо
41-60	Удовлетворительно
0-40	Неудовлетворительно