# ЭЛЕКТИВНЫЙ КУРС ПО ТЕМЕ «АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ»: ПОДГОТОВКА К ЕГЭ

Ризванов Зимфир Зуфарович, учитель информатики
МБОУ «СОШ №143» г. Казань
*rizvanov.zemfir@mail.ru*
Фазлеева Эльмира Илдаровна, к.п.н., доцент
Казанский (Приволжский) федеральный университет
*elmira.fazleeva@mail.ru*

*Аннотация:* В работе приведен составленный авторами элективный курс по алгоритмизации и программированию, который содержит пояснительную записку, содержание, учебно-тематический план и методические рекомендации по изучению тем курса.

*Ключевые слова:* алгоритмизация, программирование, единый государственный экзамен, элективный курс.

*Abstract:* In this paper the authors refer compiled program elective course on algorithms and programming, which includes teaching and thematic plan and guidelines on the form of organization and building lessons.

*Keywords*: algorithmization, programming, unified state exam, elective course

## I. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Данный курс предлагается учащимся 11-х классов старшей школы, сдающих ЕГЭ по информатике.

**Цель** курса: углубление знаний и развитие навыков решения задач, расширение содержания темы «Алгоритмизация и программирование».

Достижение поставленной цели реализуется решением следующих **задач**:

1) изучение основных понятий, связанных с использованием основных алгоритмических конструкций;
2) повторение методов решения задач на исполнение и анализ отдельных алгоритмов, записанных в виде блок-схемы, на алгоритмическом языке или на языках программирования;
3) повторение методов решения задач на составление алгоритмов для конкретного исполнителя (задание с кратким ответом);
4) отработка навыка выполнения заданий, связанных с алгоритмизацией и программированием.

В структуре изучаемого курса выделяются следующие три раздела:

- «Элементы теории алгоритмов»;
- «Программирование»;
- «Тестирование (по вариантам)».

Содержание раздела «Элементы теории алгоритмов» включает такие темы: «Выполнение и анализ простых алгоритмов» или «Поиск алгоритма минимальной длины для исполнителя», «Рекурсивные алгоритмы», «Выполнение алгоритмов для исполнителя», «Работа с массивами и матрицами в языке программирования», «Анализ программы, содержащей подпрограммы, циклы и ветвления», «Динамическое программирование», «Обработка массива», «Дерево игры. Поиск выигрышной стратегии».

В основу раздела «Программирование» вошли такие темы, как «Анализ программы», «Анализ программы с подпрограммами», «Исправление ошибок в простой программе с условными операторами», «Обработка данных, вводимых в виде символьных строк».

Последний раздел посвящен тестированию учащихся по вариантам. Важным моментом данной работы является анализ полученных результатов.

**Требования к уровню подготовки обучающихся:**

В результате изучения данного элективного курса обучающиеся должны:

**знать:**

- понятие алгоритма, его свойств, способов записи;
- основные алгоритмические конструкции;

**уметь:**

- использовать стандартные алгоритмические конструкции при программировании;
- формально исполнять алгоритмы, записанные на естественных и алгоритмических языках, в том числе на языках программирования;
- анализировать обстановку исполнителя алгоритма;
- анализировать результат исполнения алгоритма;
- анализировать текст программы, с точки зрения соответствия записанного алгоритма, поставленной задаче и изменять его в соответствии с заданием;
- реализовывать сложный алгоритм с использованием современных систем программирования.

Курс рассчитан на 36 часов практических занятий и проводится в течение учебного года по 1 часу в неделю. Формы организации занятий – практикумы по решению задач. Результатом изучения является освоение учащимися содержания курса: овладение умениями и навыками решения задач, связанных с алгоритмизацией и программированием.

Каждое занятие тематических блоков может быть построено по следующему алгоритму:

1. Повторение основных методов решения заданий по теме.
2. Совместное решение заданий ЕГЭ.
3. Самостоятельная работа обучающихся по решению задач.

Курс завершается тестированием в форме ЕГЭ.

## II. СОДЕРЖАНИЕ КУРСА

(здесь в скобках после темы указаны номера заданий в ЕГЭ по информатике)

***Раздел 1. Элементы теории алгоритмов***

*1.1. Выполнение и анализ простых алгоритмов (6-1 (А5)) или Поиск алгоритма минимальной длины для исполнителя (6-2 (В1)).*

Проверка закономерностей методом рассуждений. Представление о системах счисления. Формальное исполнение алгоритма, записанного на естественном языке.

Поиск алгоритма минимальной длины для исполнителя.

*1.2. Рекурсивные алгоритмы (11 (В6)).*

Основные понятия о рекурсивных алгоритмах.

*1.3. Выполнение алгоритмов для исполнителя (14 (А13)).*

Правила выполнения линейных, разветвляющихся и циклических алгоритмов. Основные операции с символьными строками (определение длины, выделение подстроки, удаление и вставка символов, «сцепка» двух строк в одну).

*1.4. Работа с массивами и матрицами на языке программирования (19 (А12)).*

Работа с массивами (заполнение, считывание, поиск, сортировка, массовые операции).

*1.5. Анализ программы, содержащей подпрограммы, циклы и ветвления (20 (В8)).*

Анализ алгоритма, содержащего вспомогательные алгоритмы, цикл и ветвление.

*1.6. Динамическое программирование (22 (В13)).*

Знания о динамическом программировании.

*1.7. Обработка массива (25 (С2)).*

Умение писать короткие (10-15 строк) простые программы (например, обработки массива) на языке программирования или записывать алгоритмы на естественном языке.

*1.8. Дерево игры. Поиск выигрышной стратегии (26 (С3)).*

Умение строить дерево игры по заданному алгоритму и обосновывать выигрышную стратегию.

***Раздел 2. Программирование***

*2.1. Анализ программы (8 (В5)).*

Знания основных конструкций языка программирования.

*2.2. Анализ программы с подпрограммами (21 (В14)).*

Анализ программы, использующий процедуры и функции.

*2.3. Исправление ошибок в простой программе с условными операторами(24 (С1)).*

Умение прочитывать фрагменты программы на языке программирования и исправлять допущенные ошибки.

*2.4. Обработка данных, вводимых в виде символьных строк или последовательности чисел (27 (С4)).*

Умение создавать собственные программы (30-50 строк) для решения задач средней сложности.

***Раздел 3. «Тестирование (по вариантам)»***

*3.1. Алгоритмизация и программирование.*

Выполнение тренировочных заданий. Проведение пробного ЕГЭ с последующим разбором результатов.

## III. УЧЕБНО-ТЕМАТИЧЕСКИЙ ПЛАН

| Наименование разделов и тем | Кол-во часов | Формы организации занятий |
|---|---|---|
| **Раздел 1. «Элементы теории алгоритмов»** | | |
| 1.1. Выполнение и анализ простых алгоритмов (6-1 (A5)) или Поиск алгоритма минимальной длины для исполнителя (6-2 (B1)) | 1 | **Индивидуальная работа** |
| 1.2. Рекурсивные алгоритмы (11 (B6)) | 1 | **Индивидуальная работа** |
| 1.3. Выполнение алгоритмов для исполнителя (14 (A13)) | 2 | **Работа в парах** |
| 1.4. Работа с массивами и матрицами на языке программирования (19 (A12)) | 2 | **Фронтальная работа** |
| 1.5. Анализ программы, содержащей подпрограммы, циклы и ветвления (20 (B8)) | 2 | **Фронтальная работа** |
| 1.6. Динамическое программирование (22 (B13)) | 2 | **Фронтальная работа** |
| 1.7. Обработка массива (25 (C2)) | 3 | **Работа в парах** |
| 1.8. Дерево игры. Поиск выигрышной стратегии (26 (C3)) | 6 | **Работа в парах** |
| **Раздел 2. «Программирование»** | | |
| 2.1. Анализ программы (8 (B5)) | 1 | **Индивидуальная работа** |
| 2.2. Анализ программы с подпрограммами (21 (B14)) | 2 | **Фронтальная работа** |
| 2.3. Исправление ошибок в простой программе с условными операторами (24 (C1)) | 4 | **Работа в парах** |
| 2.4. Обработка данных, вводимых в виде символьных строк или последовательности чисел (27 (C4)) | 8 | **Работа в парах** |
| **Раздел 3. «Тестирование (по вариантам)»** | | |
| 3.1. Алгоритмизация и программирование. | 2 | **Индивидуальная работа** |
| **ВСЕГО:** | **36** | |

## IV. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ ТЕМ КУРСА

### 1. Элементы теории алгоритмов

**6-1 (базовый уровень, время – 4 мин)**

**Тема**: Выполнение и анализ простых алгоритмов.

**Что нужно знать**:

- нужно лишь умение логически рассуждать (эту задачу можно давать даже детям начальной школы для развития логического мышления);
- в некоторых задачах нужно иметь представление о системах счисления (могут использоваться цифры восьмеричной и шестнадцатеричной систем счисления).

**6-2 (базовый уровень, время – 4 мин)**

**Тема**: Поиск алгоритма минимальной длины для исполнителя.

**Что нужно знать**:

- каких-либо особых знаний из курса информатики не требуется, просто его нужно уметь организовать оптимальным образом;
- *исполнитель* – это человек, группа людей, животное, машина или другой объект, который может понимать и выполнять некоторые команды.

**11 (базовый уровень, время – 5 мин)**

**Тема**: Рекурсивные алгоритмы.

**Что нужно знать**:

- *рекурсия* – это приём, позволяющий свести исходную задачу к одной или нескольким более простым задачам того же типа;
- чтобы определить рекурсию, нужно задать:
  - условие остановки рекурсии (базовый случай или несколько базовых случаев);
  - рекуррентную формулу;
- любую рекурсивную процедуру можно запрограммировать с помощью цикла;
- рекурсия позволяет заменить цикл и в некоторых сложных задачах делает решение более понятным, хотя часто менее эффективным;
- существуют языки программирования, в которых рекурсия используется как один из основных приемов обработки данных (Lisp, Haskell).

**14 (повышенный уровень, время – 6 мин)**

**Тема**: Выполнение алгоритмов для исполнителя.

**Что нужно знать**:

- правила выполнения линейных, разветвляющихся и циклических алгоритмов;
- основные операции с символьными строками (определение длины, выделение подстроки, удаление и вставка символов, «сцепка» двух строк в одну);
- *исполнитель* – это человек, группа людей, животное, машина или другой объект, который может понимать и выполнять некоторые команды;
- в школьном алгоритмическом языке **нц** обозначает «начало цикла», а **кц** – «конец цикла»; все команды между **нц** и **кц** – это *тело цикла*, они выполняются несколько раз;
- запись **нц для i от 1 до n** обозначает начало цикла, в котором переменная **i** (она называется *переменной цикла*) принимает последовательно все значения от **1** до **n** с шагом 1.

**19 (повышенный уровень, время – 5 мин)**

**Тема**: Работа с массивами и матрицами на языке программирования.

**Что нужно знать**:

- работу цикла **for** (цикла с переменной);
- *массив* – это набор однотипных элементов, имеющих общее имя и расположенных в памяти рядом;
- для обращения к элементу массива используют квадратные скобки, запись **A[i]** обозначает элемент массива **A** с номером (индексом) **i;**
- матрица (двухмерный массив) – это прямоугольная таблица однотипных элементов;
- если матрица имеет имя A, то обращение **A[i,k]** обозначает элемент, расположенный на пересечении строки **i** и столбца **k;**
- элементы, у которых номера строки и столбца совпадают, расположены на главной диагонали:

| | | | |
|---|---|---|---|
| A[1,1] | | | |
| | A[2,2] | | |
| | | A[3,3] | |
| | | | A[4,4] |

- выше главной диагонали расположены элементы, у которых номер строки **меньше** номера столбца:

| | | | |
|---|---|---|---|
| | A[1,2] | A[1,3] | A[1,4] |

| | | | |
|---|---|---|---|
| | | A[2,3] | A[2,4] |
| | | | A[3,4] |
| | | | |

- ниже главной диагонали расположены элементы, у которых номер строки **больше** номера столбца:

| | | | |
|---|---|---|---|
| | | | |
| A[2,1] | | | |
| A[3,1] | A[3,2] | | |
| A[4,1] | A[4,2] | A[4,3] | |

**20 (повышенный уровень, время – 5 мин)**

**Тема**: Анализ программы, содержащей подпрограммы, циклы и ветвления.

**Что нужно знать**:

- операции целочисленного деления (**div**) и взятия остатка (**mod**);
- как работают операторы присваивания, циклы и условные операторы в языке программирования.

**22 (повышенный уровень, время –7 мин)**

**Тема**: Динамическое программирование.

**Что нужно знать**:

- *динамическое программирование* – это способ решения сложных задач путем сведения их к более простым задачам того же типа;
- с помощью динамического программирования решаются задачи, которые требуют полного перебора вариантов:
  - «подсчитайте количество вариантов…»;
  - «как оптимально распределить…»;
  - «найдите оптимальный маршрут…»;
- динамическое программирование позволяет ускорить выполнение программы за счет использования дополнительной памяти; полный перебор не требуется, поскольку запоминаются решения всех задач с меньшими значениями параметров.

**25 (C2) (высокий уровень, время – 30 мин)**

**Тема**: Обработка массива (написать программу из 10-15 строк на языке программирования или алгоритм на естественном языке).

**Что нужно знать**:

- *массив* – это набор однотипных элементов, имеющих общее имя и расположенных в памяти рядом;
- для обращения к элементу массива используют квадратные скобки, запись **A[i]** обозначает элемент массива **A** с номером (индексом) **i;**
- для обработки всех элементов массива используется цикл вида

```
for i:=1 to N do begin
  { что-то делаем с элементом A[i] }
end;
```

  переменная **i** обозначает номер текущего элемента массива, она меняется от 1 до N с шагом 1, то есть мы «проходим» последовательно все элементы;
- *матрица* (двухмерный массив) – это прямоугольная таблица однотипных элементов;
- если матрица имеет имя **A**, то обращение **A[i,k]** обозначает элемент, расположенный на пересечении строки **i** и столбца **k:**

- каждая *строка матрицы* – это обычный (одномерный, линейный) массив; для того, чтобы обработать строку **i** в матрице из **M** столбцов, нужно использовать цикл, в котором меняется номер столбца **k**:

```
for k:=1 to M do begin
  { что-то делаем с элементом A[i,k] }
end;
```

- каждый *столбец матрицы* – это обычный (одномерный, линейный) массив; для того, чтобы обработать столбец **k** в матрице из **N** строк, нужно использовать цикл, в котором изменяется номер строки **i**:

```
for i:=1 to N do begin
  { что-то делаем с элементом A[i,k] }
end;
```

- условие задачи записано на нескольких языках (алгоритмический язык, Паскаль, Бейсик и Си); в принципе, решение можно писать и на любом другом языке, в том числе на естественном языке или в виде блок-схемы; но нужно помнить, что

Если вы пишите решение на языке, в котором есть встроенные функции для обработки массивов (списков), например, на Python, использовать эти функции НЕЛЬЗЯ; в первую очередь, это касается функций (методов) **min, max, sort.**

**26 (С3) (высокий уровень, время – 30 мин)**

**Тема**: Дерево игры. Поиск выигрышной стратегии.

**Что нужно знать**:

- в простых играх можно найти выигрышную стратегию, просто перебрав все возможные варианты ходов соперников;
- все позиции в простых играх делятся на выигрышные и проигрышные;
- **выигрышная позиция** – это такая позиция, в которой игрок, делающий первый ход, может гарантированно выиграть при любой игре соперника, если не сделает ошибку; при этом говорят, что у него есть выигрышная стратегия – алгоритм выбора очередного хода, позволяющий ему выиграть;
- если игрок начинает играть в **проигрышной** позиции, он обязательно проиграет, если ошибку не сделает его соперник; в этом случае говорят, что у него нет выигрышной стратегии; таким образом, общая стратегия игры состоит в том, чтобы своим ходом создать проигрышную позицию для соперника;
- выигрышные и проигрышные позиции можно охарактеризовать так:
  - позиция, из которой все возможные ходы ведут в выигрышные позиции – **проигрышная**;
  - позиция, из которой хотя бы один из возможных ходов ведет в проигрышную позицию – **выигрышная**, при этом стратегия игрока состоит в том, чтобы перевести игру в эту проигрышную (для соперника) позицию.

## 2. Программирование

**8 (базовый уровень, время – 3 мин)**

**Тема**: Анализ программы.

**Что нужно знать**:

- основные конструкции языка программирования:
  - объявление переменных;
  - оператор присваивания;
  - оператор вывода;
  - циклы;
- уметь выполнять ручную прокрутку программы;
- уметь выделять переменную цикла, от изменения которой зависит количество шагов цикла;
- уметь определять количество шагов цикла;
- уметь определять переменную, которая выводится на экран;

- формулу для вычисления $n$-ого элемента арифметической прогрессии:

$$a_n = a_1 + d(n-1)$$

- формулу для вычисления суммы первых $n$ членов арифметической прогрессии:

$$S_n = \sum_{i=1}^{n} a_i = a_1 + a_2 + \ldots + a_n = \frac{a_1 + a_n}{2} \cdot n = \frac{2a_1 + d(n-1)}{2} \cdot n$$

где $a_i$ – $i$-ый элемент последовательности, $d$ – шаг (разность) последовательности.

**21 (повышенный уровень, время – 6 мин)**

**Тема**: Анализ программы с подпрограммами.

**Что нужно знать**:

- *функция* – это вспомогательный алгоритм, который возвращает некоторое значение – результат;
- в Паскале функция располагается выше основной программы и оформляется следующим образом (вместо многоточия могут быть любые операторы):

```
function F(x: integer):integer;
begin
  ...
  F:= <результат функции>
end;
```

- в заголовке функции записывают имя функции, в скобках – список параметров, далее через двоеточие – тип возвращаемого значения; в приведенном примере функция **F** принимает один целый параметр, к которому внутри функции нужно обращаться по имени **x**, и возвращает целое число;
- результат функции записывается в специальную переменную, имя которой совпадает с именем функции; объявлять эту переменную не нужно;
- если параметров несколько, для каждого из них указывают тип:

```
function F(x: integer; y: integer):integer;
```

- если несколько соседних параметров имеют одинаковый тип, можно их объединить в список:

```
function F(x, y: integer):integer;
```

- следующая программа ищет наименьшее значение функции **F(x)** на интервале **[a,b]**, просматривая значения от **a** до **b** с шагом 1:

```
M:=a; R:=F(a);
for t:=a to b do
  if F(t) < R then begin
    R:=F(t); M:=t;
  end;
```

- цикл для поиска наибольшего значения выглядит точно так же, только знак < нужно заменить на знак >;
- если функция представляет собой квадратный трехчлен вида $F(x) = ax^2 + bx + c$, то абсцисса, соответствующая точке минимума, вычисляется по формуле

$$x_{\min} = \frac{-b}{2a}.$$

Этот результат можно получить (вывести, если забыли), например, так:

  - в критической точке (точке минимума, точке максимума или точке перегиба) производная функции обращается в 0;
  - находим производную $F'(x) = 2ax + b$;
  - приравниваем ее к нулю: $2ax + b = 0 \quad \Rightarrow \quad x = -\frac{b}{2a}$;

- если квадратный трехчлен задан в виде $F(x) = a(x-p)(x-q)$, то абсцисса, соответствующая точке минимума, вычисляется по формуле

$$x_{\min} = \frac{p+q}{2}.$$

**24 (С1) (повышенный уровень, время – 30 мин)**

**Тема**: Исправление ошибок в простой программе с условными операторами.

**Что нужно знать**:

- правила построения программы на Паскале, Бэйсике или Си;
- правила работы с переменными (объявление, ввод, вывод, оператор присваивания);
- *ветвление* – это выбор одного из двух возможных вариантов действий в зависимости от того, выполняется ли некоторое условие;
- *условный оператор* **if–else** служит для организации ветвления в программе на языке Паскаль;
- после **else** не надо **(нельзя!)** ставить какое-то условие, эта часть выполняется тогда, когда условие после **if** неверно (частая ошибка – после **else** пытаются написать условие, обратное тому, которое стоит после соответствующего ему **if**);
- в Паскале перед **else** не ставится точка с запятой, поскольку это ключевое слово обозначает не начало нового оператора, а вторую часть условного оператора **if–else.**

**27(С4) (высокий уровень, время – 55 мин)**

**Тема**: Обработка данных, вводимых в виде символьных строк (написать программу средней сложности из 30-50 строк) или последовательности чисел.

**Что нужно знать**:

- *символьная строка* – это цепочка символов, которая может обрабатываться как единое целое;
- для обращения к символу с номером **i** строки **s** используется запись **s[i],** это говорит о том, что строка – особый вариант массива, в котором хранятся символы;
- знак сложения при работе с символьными строками означает сцепку, объединение двух строк в одну (добавление второй строки в конец первой);
- для работы со строками в наиболее распространенных Паскаль-средах (*Turbo Pascal, Borland Pascal, PascalABC,* среда *АЛГО*) используют стандартные функции (здесь **s** – это переменная типа **string**, символьная строка; **n** и **r** – целые переменные)

| | |
|---|---|
| **n := Length(s);** | записать длину строки **s** в целую переменную **n** |
| **s1 := Copy(s, 2, 5);** | записать в символьную строку **s1** подстроку строки **s**, которая начинается с символа с номером 2 и состоит из 5 символов (важно – не со 2-го по 5-ый символ!) |
| **n := Pos('Вася', s);** | записать в целую переменную **n** номер символа, с которого в строке **s** начинается подстрока 'Вася' (если ее нет, в переменную **n** записывается 0); так же можно искать отдельные символы (важно: сначала указываем, **что** ищем, а потом – **где**) |
| **n := StrToInt(s);** | преобразовать строку **s** в целое число и записать результат в переменную **n** (*PascalABC, Delphi*) |

и процедуры

| | |
|---|---|
| **Delete(s, 2, 5);** | удалить из строки **s** 5 символов, начиная со второго |
| **Insert('Вася', s, 3);** | вставить в строку **s** фрагмент 'Вася', начиная с третьего символа (между 2-м и 3-м) |
| **Val(s, n, r);** | преобразовать строку **s** в целое число и записать результат в переменную **n**; если при этом произошла ошибка, в переменной **r** будет номер ошибочного символа, если все нормально – ноль |

- структура (в Паскале она называется «запись», *record*) – это сложный тип данных, который может включать в себя несколько элементов – полей; поля могут иметь различный тип;
- записи в Паскале объявляются с помощью ключевого слова **record**; в простейшем случае можно выделить память под одну запись так:

```
var x: record
    name: string;
    code: integer;
   end;
```

эта запись состоит из двух полей: символьной строки **name** и целого числа **code;**

- для обращения к полям записи используют точку, например **x.name** означает «поле **name** записи **x**»

**3. Тестирование (некоторые задания из пробного варианта ЕГЭ по информатике)**

**1 (6-1).** Автомат получает на вход четырехзначное число. По этому числу строится новое число по следующим правилам.

1. Складываются первая и вторая, а также третья и четвертая цифры исходного числа.
2. Полученные два числа записываются друг за другом в порядке убывания (без разделителей).

Пример. Исходное число: 3165. Суммы 3 + 1 = 4; 6 + 5 = 11. Результат: 114.

Укажите наименьшее число, в результате обработки которого, автомат выдаст число 1311.

Ответ: _____.

**2 (11).** Ниже на различных языках программирования записан рекурсивный алгоритм F.

| **Бейсик** | **Алгоритмический язык** |
|---|---|
| SUB F(n)<br>  PRINT n<br>  IF n < 5 THEN<br>  F(n+1)<br>  F(n+3)<br>  END IF<br>END SUB | алг F (цел n)<br>нач<br>  вывод n, нс<br>  если n < 5 то<br>  F(n+1)<br>  F(n+3)<br>  все<br>кон |
| **Паскаль** | **Си** |
| procedure F(n: integer);<br>begin<br>  writeln (n);<br>  if n < 5 then<br>  begin<br>  F(n+1);<br>  F(n+3)<br>  end;<br>end. | void F (int n)<br>{<br>  printf («%d\n», n);<br>  if (n < 5) {<br>  F (n+1);<br>  F (n+3);<br>  }<br>} |

Чему равна сумма всех чисел, напечатанных на экране при выполнении вызова F(1)?

**3 (14).** Исполнитель Чертежник перемещается на координатной плоскости, оставляя след в виде линии. Чертежник может выполнить команду ***сместиться на (a, b)***, где *a, b* – целые числа. Эта команда перемещает Чертежника из точки с координатами *(x, y)* в точку с координатами *(x + a, y + b).*

*Например, если Чертежник находится в точке с координатами (4, 2), то команда* ***сместится на (2, -3)*** *переместит Чертежника в точку (6, -1).*

Цикл

ПОВТОРИ число РАЗ

последовательность команда

КОНЕЦ ПОВТОРИ

означает, что *последовательность команд* будет выполнена указанное *число* раз (число должно быть натуральным).

Чертежнику был дан для исполнения следующий алгоритм (буквами *n, a, b* обозначены неизвестные числа, при этом *n* > 1):

НАЧАЛО
сместиться на *(-3, -3)*
ПОВТОРИ *n* РАЗ
**сместиться на (*a, b*)**
**сместиться на (27, 12)**
КОНЕЦ ПОВТОРИ
**сместится на (-22, -7)**
КОНЕЦ

Укажите наименьшее возможное значение числа *n,* для которого найдутся такие значения чисел *a* и *b*, что после выполнения программы Чертежник возвратиться в исходную точку.

**4 (19).** В программе используется одномерный целочисленный массив A с индексами от 0 до 9. Значения элементов равны 6; 9; 7; 2; 1; 5; 0; 3; 4; 8 соответственно, т.е. A[0] = 6; A[1] = 9 и т.д.

Определите значение переменной c после выполнения следующего фрагмента программы, записанного ниже на различных языках программирования.

| **Бейсик** | **Алгоритмический язык** |
|---|---|
| c = 0<br>FOR i = 1 TO 9<br>  IF A(i - 1) < A(i) THEN<br>  c = c + 1<br>  t = A(i)<br>  A(i) = A(i - 1)<br>  A(i - 1) = t<br>  END IF<br>NEXT i | c := 0<br>нц для i от 1 до 9<br>  если A[i – 1] < A[i] то<br>  c := c + 1<br>  t := A[i]<br>  A[i] := A[i – 1]<br>  A[i – 1] := t<br>  все<br>кц |
| **Паскаль** | **Си** |
| c := 0;<br>for i :=1 to 9 do<br>  if A[i – 1] < A[i] then<br>  begin<br>  c := c + 1;<br>  t := A[i];<br>  A[i] := A[i – 1];<br>  A[i – 1] := t<br>  end. | c := 0;<br>for (i = 1; i =< 9; i++)<br>  if (A[i – 1] < A[i]) {<br>  c++;<br>  t := A[i];<br>  A[i] = A[i – 1];<br>  A[i – 1] = t<br>  } |

**5 (22).** Исполнитель Май4 преобразует число, записанное на экране. У исполнителя три команды, которым присвоены номера:

**1. Прибавь 1**
**2. Прибавь 2**
**3. Прибавь 4**

Первая из них увеличивает число на экране на 1, вторая увеличивает это число на 2, а третья – на 4. Программа для исполнителя Май4 – это последовательность команд.

Сколько есть программ, которые **число 21** преобразуют в **число 30**?

**Список литературы**


1. Информатика: ЕГЭ за 30 дней: экспресс-репетитор / О.Б. Богомолова. – Москва: АСТ, Астрель, 2014. – 446 с. – (Единый государственный экзамен).
2. Информатика: Полный справочник для подготовки к ЕГЭ / О.Б. Богомолова. – Москва: АСТ: Астрель, 2014. – 415 с.
3. Преподавание, наука и жизнь // http://kpolyakov.spb.ru/school/ege.htm