ЕГЭ – 2016

Сложные вопросы алгоритмизиции и программирования

Петрова Ирина Александровна,

МБУ «Лицей №6»

Задания ЕГЭ из раздела «АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ», ежегодно вызывающие затруднения

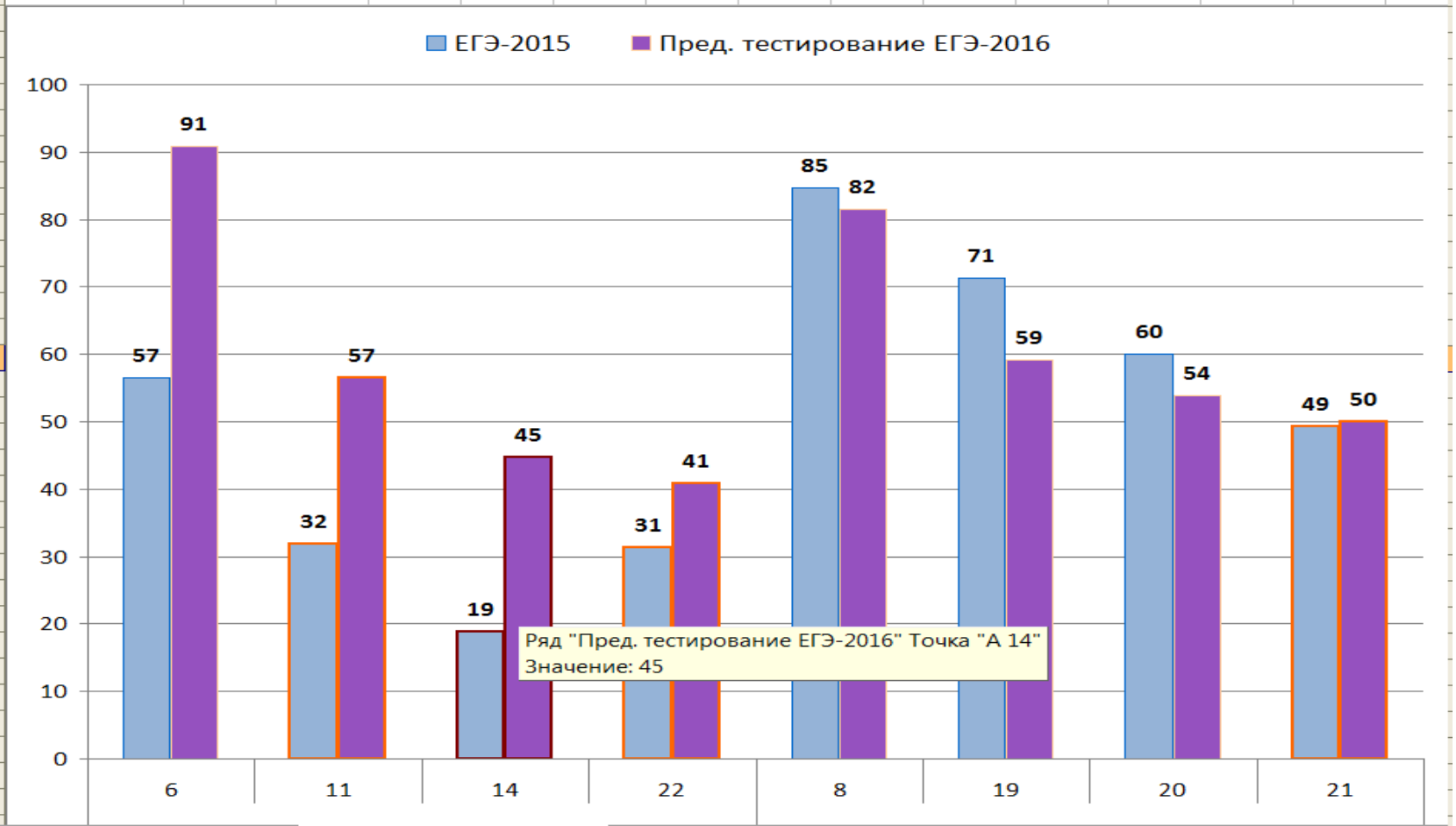
2015	6	8	11	14	19	20	21	22
	A5	B5	B6	A13	A12	B8	B14	B13
ЕГЭ-2013	85	77	67	76	76	54	36	47
ЕГЭ-2014	63	80	12	79	36	22	26	39
ЕГЭ-2015	57%	85%	32%	19%	71%	60%	49%	31%

Задания ЕГЭ из раздела «АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ», вызывающие затруднения

- **№ 11** (рекурсия в алгоритмах);
- **№ 14** (анализ результата исполнения алгоритма для конкретного исполнителя);
- **№ 20** (анализ алгоритма, содержащего подпрограммы, циклы и ветвления);
- **№ 21** (функции в алгоритмах);
- **№ 22** (применение метода динамического программирования для анализа результатов исполнения алгоритма).

Задания ЕГЭ из раздела «АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ», вызывающие затруднения

	А				П							
	6	11	14	22	8	19	20	21	24	25	26	27
ЕГЭ-2015	57	32	19	31	85	71	60	49	66	49	69	28
Пред. тестирование ЕГЭ-2016	91	57	45	41	82	59	54	50				



ЕГЭ – 2016: результаты предварительного тестирования

	A	B	C	D	E	F	G	H	I	J	K	L
1	**B11**		**B14**		**B19**		**B20**		**B21**		**B22**	
2	**Ответ**	**кол-во человек**	**Ответ**	**кол-во человек**	**Ответ**	**кол-во человек**	**Ответ**	**кол-во человек**	**Ответ**	**кол-во человек**	**Ответ**	**кол-во человек**
3	3	43	8	34	1	45	154	41	21	38	6	31
4	2	13	28288	18	4	9	66	3	3	3	7	8
5	4	2	22	5	2	5	220	3	5	3	8	8
6	8	2	2	4	3	4	22	2	1	2	5	7
7	10	2	2288	4	6	2	152	2	65	2	9	3
8	1	1	28	1	7	2	176	2	2	1	13	3
9	6	1	82	1	0	1	130	1	9	1	1	2
10	7	1	88	1	5	1	155	1	11	1	2	2
11	13	1	228	1	9	1	160	1	13	1	4	2
12	208	1	288	1	16	1	164	1	26	1	10	2
13	нет ответа	9	822	1	нет ответа	5	165	1	29	1	37	1
14			88228	1			167	1	50	1	нет ответа	7
15			нет ответа	4			168	1	52	1		
16							170	1	нет ответа	20		
17							188	1				
18							198	1				
19							300	1				
20							нет ответа	12				
21												
22												
23		76		76		76		76		76		76
24		57		45		59		54		50		41

№ 11. Рекурсия в алгоритмах

№11.1. Ниже на пяти языках программирования записана рекурсивная процедура (функция) F. Что выведет программа при вызове F(9)? В ответе запишите последовательность выведенных цифр слитно (без пробелов).

Бейсик

```
SUB F(n)
  print n,
  IF n >= 7 THEN
    F(n - 3)
    F(n - 1)
  END IF
END SUB
```

Python

```
def F(n):
    print(n, end='')
    if n >= 7:
        F(n - 3)
        F(n - 1)
```

Алгоритмический язык

```
алг F(цел n)
нач
  вывод n
  если n >= 7 то
    F(n - 3)
    F(n - 1)
  все
кон
```

Паскаль

```
procedure F(n: integer);
begin
  write(n);
  if n >= 7 then
  begin
    F(n – 3);
    F(n - 1)
  end
end;
```

Си

```
void F(int n) {
  printf("%d", n);
  if (n >= 7) {
    F(n - 3);
    F(n - 1);
```

№11.1. Решение:

Команда алгоритма	Результат исполнения	Вывод	Примечание
вывод n	вывод «9»	9	
если n >= 7 то	9 ≥7, истина		
F(n - 3)	вызов F(6)		1 уровень рекурсии
вывод n	вывод «6»	6	
если n >= 7 то	6 ≥7, ложь		возврат в F(9)
F(n - 1)	вызов F(8)		1 уровень рекурсии
вывод n	вывод «8»	8	
если n >= 7 то	8 ≥7, истина		
F(n - 3)	вызов F(5)		2 уровень рекурсии
вывод n	вывод «5»	5	
если n >= 7 то	5 ≥7, ложь		возврат в F(8)
F(n - 1)	вызов F(7)		2 уровень рекурсии
вывод n	вывод «7»	7	
если n >= 7 то	7 ≥7, истина		
F(n - 3)	вызов F(4)		3 уровень рекурсии
вывод n	вывод «4»	4	
если n >= 7 то	4 ≥7, ложь		возврат в F(7)
F(n - 1)	вызов F(6)		3 уровень рекурсии
вывод n	вывод «6»	6	
если n >= 7 то	6 ≥7, ложь		возврат в F(7)
кон			возврат в F(8)
кон			возврат в F(9)
кон			завершение алгоритма

Ответ: 9685746

№ 11.2.

11 Ниже на пяти языках программирования записаны две рекурсивные функции (процедуры): F и G.

Бейсик

```
DECLARE SUB F(n)
DECLARE SUB G(n)

SUB F(n)
  IF n > 0 THEN G(n - 2)
END SUB

SUB G(n)
  PRINT "*"
  IF n > 2 THEN F(n - 5)
END SUB
```

Python

```
def F(n):
    if n > 0:
        G(n - 2)

def G(n):
    print("*")
    if n > 2:
        F(n - 5)
```

Алгоритмический язык

```
алг F(цел n)
нач
  если n > 0 то
    G(n - 2)
  все
кон

алг G(цел n)
нач
  вывод "*"
  если n > 2 то
    F(n - 5)
  все
кон
```

Паскаль

```
procedure F(n: integer); forward;
procedure G(n: integer); forward;

procedure F(n: integer);
begin
  if n > 0 then
    G(n - 2);
end;

procedure G(n: integer);
begin
  writeln('*');
  if n > 2 then
    F(n - 5);
end;
```

Си

```
void F(int n);
void G(int n);

void F(int n){
  if (n > 0)
    G(n - 2);
}

void G(int n){
  printf("*");
  if (n > 2)
    F(n - 5);
}
```

Сколько символов «звёздочка» будет напечатано на экране при выполнении вызова F(15)?

№11.1. Решение:

Исполним алгоритм для указанного аргумента (15).

Команда алгоритма	Результат исполнения	Вывод	Примечание
`если n > 0 то`	15 >0, истина		выполняется F(15)
`G(n - 2)`	вызов G(13)		
`вывод "*"`	вывод «*»	*	выполняется G(13)
`если n > 2 то`	13 >2, истина		
`F(n - 5)`	вызов F(8)		
`если n > 0 то`	8 >0, истина		выполняется F(8)
`G(n - 2)`	вызов G(6)		
`вывод "*"`	вывод «*»	*	выполняется G(6)
`если n > 2 то`	6 >2, истина		
`F(n - 5)`	вызов F(1)		
`если n > 0 то`	1 >0, истина		выполняется F(1)
`G(n - 2)`	вызов G(-1)		
`вывод "*"`	вывод «*»	*	выполняется G(-1)
`если n > 2 то`	-1 >2, ложь		
`кон`			завершение G(-1)
`кон`			завершение F(1)
`кон`			завершение G(6)
`кон`			завершение F(8)
`кон`			завершение G(13)
`кон`			завершение F(15)

Таким образом, символ «звёздочка» будет напечатан 3 раза. Ответ: **3**

№ 11.3. Усложнение задания подобного 11.2. добавлением в процедуры еще одного оператор вывода:

Алгоритмический язык	Паскаль

алг F(цел n)
нач
  если n > 0 то
  G(n - 1)
  все
  вывод "+"
кон
 	
алг G(цел n)
нач
  вывод "*"
  если n > 1 то
  F(n - 2)
  все
кон
	procedure F(n: integer); forward;
	procedure G(n: integer); forward;
	
	procedure F(n: integer);
	begin
	  if n > 0 then
	  G(n - 1);
	  writeln('*');
	end;
	
	procedure G(n: integer);
	begin
	  writeln('*');
	  if n > 1 then
	  F(n - 2);
	end;

Сколько символов «*» будет напечатано при вызове F(11)?

№11.3. Решение:

Команда алгоритма	Результат исполнения	Вывод	Примечание
`если n > 0 то`	11 >0, истина		выполняется F(11)
`G(n - 1)`	вызов G(10)		
`вывод "*"`	вывод «*»	*	выполняется G(10)
`если n > 1 то`	10 >1, истина		
`F(n - 2)`	вызов F(8)		
`если n > 0 то`	8 >0, истина		выполняется F(8)
`G(n - 1)`	вызов G(7)		
`вывод "*"`	вывод «*»	*	выполняется G(7)
`если n > 1 то`	7 >1, истина		
`F(n - 2)`	вызов F(5)		
`если n > 0 то`	5 >0, истина		выполняется F(5)
`G(n - 1)`	вызов G(4)		
`вывод "*"`	вывод «*»	*	выполняется G(4)
`если n > 1 то`	4 >1, истина		
`F(n - 2)`	вызов F(2)		
`если n > 0 то`	2 >0, истина		выполняется F(2)
`G(n - 1)`	вызов G(1)		

№11.3. Решение (продолжение):

Команда алгоритма	Результат исполнения	Вывод	Примечание
вывод "*"	вывод «*»	*	выполняется G(1)
если n > 1 то	1 >1, ложь		
кон			завершение G(1)
вывод "*"	вывод «*»	*	возврат в F(2)
кон			завершение F(2)
кон			завершение G(4)
вывод "*"	вывод «*»	*	возврат в F(5)
кон			завершение F(5)
кон			завершение G(7)
вывод "*"	вывод «*»	*	возврат в F(8)
кон			завершение F(8)
кон			завершение G(10)
вывод "*"	вывод «*»	*	возврат в F(11)
кон			завершение F(11)

Таким образом, символ «звёздочка» будет напечатан 8 раз.
Ответ: 8

№ 14.

Это задание предполагает применение знаний в новой для экзаменующихся ситуации (практически это означает, что в задании может быть описан неизвестный экзаменуемым исполнитель), поэтому выпускникам надо уметь разбираться в системе команд исполнителя и исполнять алгоритм для него.

14

Исполнитель Редактор получает на вход строку цифр и преобразовывает её. Редактор может выполнять две команды, в обеих командах *v* и *w* обозначают цепочки цифр.

А) **заменить** (*v*, *w*).

Эта команда заменяет в строке первое слева вхождение цепочки *v* на цепочку *w*. Например, выполнение команды

заменить (111, 27)

преобразует строку 05111150 в строку 0527150.

Если в строке нет вхождений цепочки *v*, то выполнение команды **заменить** (*v*, *w*) не меняет эту строку.

Б) **нашлось** (*v*).

Эта команда проверяет, встречается ли цепочка *v* в строке исполнителя Редактор. Если она встречается, то команда возвращает логическое значение «истина», в противном случае возвращает значение «ложь». Строка исполнителя при этом не изменяется.

Цикл

ПОКА *условие*

последовательность команд

КОНЕЦ ПОКА

выполняется, пока условие истинно.

В конструкции

ЕСЛИ *условие*

ТО *команда1*

ИНАЧЕ *команда2*

КОНЕЦ ЕСЛИ

выполняется *команда1* (если условие истинно) или *команда2* (если условие ложно).

№14.1.

Какая строка получится в результате применения приведённой ниже программы к строке, состоящей из 41 идущих подряд цифр 8? В ответе запишите полученную строку.

```
НАЧАЛО
ПОКА нашлось (222) ИЛИ нашлось (888)
        ЕСЛИ нашлось (222)
            ТО заменить (222, 8)
            ИНАЧЕ заменить (888, 2)
        КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

№14.1. Решение:

Для правильного решения данной задачи ученики должны были применить данный алгоритм для строки меньшего размера. Например, рассмотреть строку из 10 идущих подряд цифр 8.

8888888888

Заметить, что данный алгоритм удаляет 8 восьмерок:

~~888~~ ~~888~~ ~~888~~ 8

~~2~~ ~~2~~ ~~2~~ 8

8 8

Таким образом, от 10 первоначальных восьмерок осталось 2, т.е. данный алгоритм действительно удаляет 8 восьмерок.

Если такой «круговой» алгоритм к строке из 41 идущих подряд цифр 8 применить 5 раз (больше нельзя, это ясно из таблицы умножения), в строке останется 41 – 5*8 = 41 – 40 = 1 восьмерка.

Ответ: **8**.

№14.2. Условие предыдущкй задачи применить к строке из 41 идущих подряд цифр 2

Решение: нужно обратить внимание учащихся на то, что алгоритм сначала заменит ВСЕ «222» на «8». Затем нужно подсчитать, сколько раз «222» заменятся на «8». Вычислим целую часть от деления 41 на 3. Это 13. То есть в последовательности станет 13 восьмерок + «22» (целочисленный остаток от деления 41 на 3):

8888888888822

Найдем целочисленный остаток от деления 13 на 8. Это будет 1, то есть теперь к полученной строке 1 раз применим алгоритм, описанный выше:

~~888~~ ~~888~~ ~~888~~ 888822

~~2~~ ~~2~~ ~~2~~ 888822

8 888822

Таким образом, останется «8888822». В этой строке есть «888», которые заменим на «2». Получится строка «28822».

Ответ: **28822**.

№ 20. Анализ алгоритма, содержащего подпрограммы, циклы и ветвления

20 Ниже на пяти языках программирования записан алгоритм. Получив на вход число x, этот алгоритм печатает число M. Известно, что $x > 150$. Укажите **наименьшее** такое (т.е. большее 150) число x, при вводе которого алгоритм печатает число 22.

Алгоритмический язык

```
алг
нач
  цел x, L, M
  ввод x
  L := x
  M := 55
  если mod(L,2) = 0
    то
      M := 44
  все
  нц пока L <> M
    если L > M
      то
        L := L - M
      иначе
        M := M - L
    все
  кц
  вывод M
кон
```

Паскаль

```
var x, L, M: integer;
begin
  readln(x);
  L := x;
  M := 55;
  if L mod 2 = 0 then
    M := 44;
  while L <> M do
    if L > M then
      L := L - M
    else
      M := M - L;
  writeln(M);
end.
```

№ 20. Анализ алгоритма, содержащего подпрограммы, циклы и ветвления

Сложность этой задачи состоит в том, чтобы разобраться в алгоритме.

- Сначала вводится число x,
которое помещается в переменную L.
Переменной M присваивается значение 55.

```
readln(x);
L := x;
M := 55;
```

- Если L четное,
то переменной M присваивается значение 44.

```
if L mod 2 = 0 then
  M := 44;
```

- Затем с переменными L и M выполняется цикл:
его суть сводится к тому, что большее из двух чисел
L и M каждый раз заменяется на их разность до тех пор,
пока они не станут равны.

```
while L <> M do
  if L > M then
    L := L - M
  else
    M := M - L;
```

- Делаем вывод, что это классический **Алгоритм Евклида**, который служит для **вычисления наибольшего общего делителя (НОД) двух чисел**; этот делитель в результате оказывается и в переменной L, и в переменной M.
- Смотрим, что выводится на экран: значение переменной M=22 (наибольший общий делитель исходных чисел, НОД(x,55) или НОД(x,44)).

Наименьшее число, которое больше 150 и делится на 22, равно 154.

Ответ: **154**.

Алгоритм Евклида (2 вариант)

- Для того, чтобы выпускники узнали **Алгоритм Евклида**, они обязательно должны решать подобные задачи, в том числе и с таким вариантом этого алгоритма:

```
while b > 0 do
    begin
        r:= a mod b;
        a:= b;
        b:= r;
    end;
```

- Его суть сводится к тому, что большее из двух чисел, a и b, каждый раз заменяется на остаток от деления большего на меньшее до тех пор, пока этот остаток не станет равен нулю.

№20.2. (№53 ЕГЭ-2016 http://kpolyakov.spb.ru).

Получив на вход число x, этот алгоритм печатает два числа K и R. Укажите наименьшее из таких чисел x, при вводе которых алгоритм печатает сначала 4, а потом 3.

```
var x, i, K, R, y: integer;
begin
  readln(x);
  K := 0; R := 9;
  y := x mod 10;
  while x > 0 do begin
    K := K + 1;
    if R > x mod 10 then R := x mod 10;
    x := x div 10
  end;
  R := y - R;
  writeln(K); writeln(R)
end.
```

№20.2. (продолжение)

Выполним трассировку для какого-то простого числа, например, для числа 253:

оператор	условие	x	K	R	y
readln(x);		253			
K:=0; R:=9;			0	9	
y := x mod 10;					3
while x > 0 do…	253 > 0? да				
K:=K+1;			1		
if R> x mod 10 then…	9 > 3 ? да				
R:=x mod 10;				3	
x:=x div 10;		25			
while x > 0 do…	25 > 0? да				
K:=K+1;			2		
if R> x mod 10 then…	3 > 5 ? нет				
x:=x div 10;		2			
while x > 0 do…	2 > 0? да				
K:=K+1;			3		
if R> x mod 10 then…	3 > 2? да				
R:=x mod 10;				2	
x:=x div 10;		0			
while x > 0 do…	0 > 0? нет				
R := y - R;				1	
writeln(K); write(R);			3	1	

№20.2. (продолжение)

- Можно догадаться, что в результате работы программы в переменной K окажется число цифр числа, а в переменной R – разность между последней и наименьшей цифрами.
- Нужно запомнить, что для целого числа x остаток от деления на 10 (x mod 10) – это последняя цифра в десятичной записи числа, а целочисленное деление (x div 10) отсекает последнюю цифру, то есть из 123 получается 12.
- Итак, по условию задачи фактически требуется найти наименьшее четырехзначное число, в котором разность между последней и наименьшей цифрами = 3. Значит последняя цифра должна быть >=3. Наименьшая цифра в числе – 0, но она не должна стоять на первом месте.
- Очевидно, что это 1003.

Ответ: **1003.**

- Кроме того, выпускникам нужно уметь «узнавать» алгоритмы перевода десятичных чисел в **другие системы счисления**. Например, после выполнения следующего фрагмента программы:

```
while x > 0 do
    begin
        L:=L+1;
        M:= M*(x mod 8);
        x:= x div 8;
    end;
```

- Переменная **L** будет равна количеству цифр числа **x**, записанного в **восьмеричной системе счисления**, а в **M** будет записано произведение всех цифр восьмеричной записи числа **x**.

№ 21. Функции в алгоритмах

21 Напишите в ответе наименьшее значение входной переменной k, при котором программа выдаёт тот же ответ, что и при входном значении $k = 25$. Для Вашего удобства программа приведена на пяти языках программирования.

Алгоритмический язык

```
алг
нач
  цел i, k
  ввод k
  i := 1
  нц пока f(i) < g(k)
    i := i + 1
  кц
  вывод i
кон

алг цел f(цел n)
нач
  знач := n * n * n
кон

алг цел g(цел n)
нач
  знач := 3*n + 2
кон
```

Паскаль

```
var
  k, i : longint;

function f(n: longint): longint;
begin
  f := n * n * n;
end;

function g(n: longint): longint;
begin
  g := 3*n + 2;
end;

begin
  readln(k);
  i := 1;
  while f(i) < g(k) do
    i := i+1;
  writeln(i)
end.
```

Решение:

- В программе используются две функции: f, которая возводит значение n в куб, и g, которая умножает значение n на 3 и прибавляет к полученному результату 2. В основной программе послу ввода k переменной i присваивается 1 и начинает работу цикл с предусловием. После чего выводится значение переменной i. Нужно найти наименьшее значение входной переменной k, при котором программа выдаёт тот же ответ, что и при входном значении k = 25.
- Найдем, какой программа выдаёт ответ при входном значении k = 25. Сначала запишем условие цикла через переменные k и i:

$$i^3 < 3*k + 2$$

- Цикл в программе будет выполняться, пока выполняется это условие и с каждым проходом цикла переменная i увеличивается на 1. Значение k не изменяется, поэтому перепишем условие цикла:

$$i^3 < 3*25 + 2 \quad \text{или}$$

$$i^3 < 77$$

Решение (продолжение):

1 проход цикла (i =1): **$1^3 = 1 < 77$ (да)**

2 проход цикла (i =2): **$2^3 = 8 < 77$ (да)**

3 проход цикла (i =3): **$3^3 = 27 < 77$ (да)**

4 проход цикла (i =4): **$4^3 = 64 < 77$ (да)**

5 проход цикла (i =5): **$5^3 = 125 < 77$ (нет)**

- Таким образом, при k = 25 цикл остановится на значении i = 5, которое программа и выведет на экран.
- Последнее выполнившееся условие цикла: 64 < 77.
- Перепишем его через k: **64 < 3*k + 2** и решим полученное неравенство:

  64 < 3*k + 2

  62 < 3*k

  20,7 < k

- Очевидно, наименьшее целое k = 21, при котором программа выдаёт тот же ответ, что и при входном значении k = 25.

Ответ: **21**.

Усложним задание (из материалов сайта К.Ю. Полякова http://kpolyakov.spb.ru)

№70. Напишите в ответе число, равное количеству различных значений входной переменной k, при которых приведённая ниже программа выводит тот же ответ, что и при входном значении k = 17. Значение k = 17 также включается в подсчёт различных значений k.

```
var k, i : longint;
function f(n: longint) : longint;
begin
  f := n*n*n + 5*n*n;
end;
begin
  readln(k);
  i := 1;
  while f(i) < k do
    i:= i+1;
  if f(i)-k <= k-2*f(i-1) then
       writeln(i)
  else writeln(i-1);
end.
```

Решение:

- сначала заметим, что функция **f** возвращает сумму куба и квадрата умноженного на 5 переданного ей числа
- после ввода **k** работает цикл, который увеличивает **i** до тех пор, пока значение **f (i)** не станет больше или равно **k** – тогда нарушится условие **f (i) < k** и цикл завершится
- построим таблицу значений функции **f (i)**:

i	f (i)	Цикл завершается для ...
1	6	k=1, . . . , 6
2	28	k=7, . . . , 28

- таким образом, при **k = 17** цикл завершится при **i = 2**
- главная «новинка» – в условном операторе

  f (i) – k < = k – 2 * f (i – 1) then writeln (i) else writeln (i – 1);
- например, при k = 17 и i = 2 условие

  f (i) – k < = k – 2 * f (i – 1)

  28 – 17 < = 17 – 2 * 6

  11 < = 5

неверно, из-за этого выводится на экран не **i**, а **i – 1**, то есть 1

- итак, нам нужно найти, сколько значений **k** дадут на выходе **1**
- посмотрим внимательно на условие в условном операторе, преобразуем его к виду

  f (i) + 2 * f (i – 1) < = 2 * k
- тогда

  при (f (i) + 2 * f (i – 1)) / 2 < = k выводится на экран i,

  при (f (i) + 2 * f (i – 1)) / 2 > k выводится на экран i – 1.

i	(f(i)+ 2*f(i-1))/2	Выводится i-1 для …	Выводится i для …
1	3	k=1,2	k=3,...,6
2	20	k=7,...,19	k=20,...,28

- таким образом, в этой задаче нам подходят числа в диапазоне [3;19], всего их 19-3+1 = 17

Ответ: **17**.

№ 22. Динамическое программирование

22 Исполнитель Тренер преобразует число на экране.
У исполнителя есть две команды, которым присвоены номера:
1. Прибавить 1
2. Умножить на 2
Первая команда увеличивает число на экране на 1, вторая умножает его на 2.
Программа для исполнителя Тренер – это последовательность команд.
Сколько существует программ, для которых при исходном числе 7 результатом является число 29 и при этом траектория вычислений содержит число 9 и не содержит числа 15?
Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. Например, для программы **121** при исходном числе 7 траектория будет состоять из чисел 8, 16, 17.

Для пояснения решение этой задачи воспользуемся разбором решений подобного задания на сайте К.Ю. Полякова http://kpolyakov.spb.ru.

Решение:

1) у нас в задании две особые точки – числа 9 (через которое должна проходить траектория) и 15 (а сюда она попасть НЕ должна)
2) сначала составляем рекуррентную формулу, по которой будем вычислять количество K_N обозначить количество разных программ для получения числа N из начального числа 1:
3) число N могло быть получено одной из двух операций:
   – увеличением на 1 числа N-1;
   – умножением на 2 числа N/2 (только для N, которые делятся на 2);

   $K_N = K_{N-1}$ для нечётных чисел

   $K_N = K_{N-1} + K_{N/2}$ для чётных чисел
4) для начального числа 7 количество программ равно 1: существует только одна пустая программа, не содержащая ни одной команды; $K_7 = 1$.

Решение (продолжение):

5) составляем таблицу до первой особой точки – числа 9:

N	7	8	9
K_N	1	1	1

6) поскольку число 9 должно обязательно войти в траекторию, начинаем составлять вторую часть таблицы (до второй контрольной точки, 15) с этого числа заново, считая, что все ячейки для меньших чисел – нулевые

N	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
K_N	1	1	1	1	1	1	0	0	0	1	1	2	2	3	3	4	4	5	5	6	6

7) поскольку траектория не может проходить через 15, для $N = 15$ принимаем $K_N = 0$ (в таблице эта ячейка выделена красным цветом)

8) дальше заполняем оставшиеся ячейки второй части таблицы обычным образом.

Ответ: **6**.

№ 22.2. Исполнитель Калькулятор преобразует целое число, записанное на экране. У исполнителя две команды, каждой команде присвоен номер:

1. **Прибавь 1**
2. **Прибавь 2**

Первая команда увеличивает число на экране на 1, вторая увеличивает – на 2. Сколько существует программ, которые число 3 преобразуют в число 18 и в которых предпоследняя команда 2?

Решение:

- итак, мы знаем предпоследнюю команду – 2, при этом последняя команда может быть любая – 1 или 2
- выходит, что нужно получить количество всех программ вида «…21» и «…22», где « **…**» обозначает любую траекторию;
- если программа заканчивается на «21», то до ее выполнения у нас было число (18 – 1) – 2 = 15;
  поэтому нужно найти число программ для преобразования 3 в 15
- для начального числа 3 количество программ равно 1: существует только одна пустая программа, не содержащая ни одной команды;.
- теперь построим рекуррентную формулу

№ 22.2. (продолжение)

- составляем таблицу:

  для «…21»: (18 – 1) – 2 = 15

N	3	4	5	6	7	8	9	10	11	12	13	14	15
K_N	1	1	2	3	5	8	13	21	34	55	89	144	233

- теперь рассматриваем случай, когда программа заканчивается на «22», это значит, что до ее выполнения у нас было число (18 – 2) – 2 = 14; поэтому нужно найти число программ для преобразования 3 в 18:

  для «…22»: (18 – 2) – 2 = 14

N	3	4	5	6	7	8	9	10	11	12	13	14
K_N	1	1	2	3	5	8	13	21	34	55	89	144

- ответ к задаче – сумма двух значений, выделенных жёлтым маркером, поскольку мы рассмотрели все варианты программ, в которых предпоследняя команда – 1
- 233 + 144 = 377

Ответ: **377**

№ 22.3. Исполнитель A12S преобразует целое число, записанное на экране. У исполнителя три команды, каждой команде присвоен номер:

1. **Прибавь 1**
2. **Прибавь 2**
3. **Прибавь предыдущее**

Первая команда увеличивает число на экране на 1, вторая увеличивает это число на 3, третья прибавляет к числу на экране число, меньшее на 1 (к числу 3 прибавляется 2, к числу 11 прибавляется 10 и т. д.). Программа для исполнителя A12S – это последовательность команд. Сколько существует программ, которые число 3 преобразуют в число 10?

№ 22.3. Решение:

- для начального числа 3 количество программ равно 1: существует только одна пустая программа, не содержащая ни одной команды;

  теперь рассмотрим общий случай, чтобы построить рекуррентную формулу

  Число N могло быть получено одной из трёх операций сложения:

- увеличением на 1 числа N-1;
- увеличением на 2 числа N-2;
- из некоторого числа X увеличением на X-1 (предыдующее число), так что $N = X + (X - 1)$

  откуда $X = (N + 1) / 2$; так могут быть получены только нечетные числа;

  поэтому

- $K_N = K_{N-1} + K_{N-2}$ для чётных чисел
- $K_N = K_{N-1} + K_{N-2} + K_{(N+1)/2}$ для нечётных чисел

N	3	4	5	6	7	8	9	10
K_N	1	1	3	4	8	12	23	35

Ответ: **35**

В классах, где информатика изучается на базовом уровне (т.е. отводится минимальное количество часов на изучение информатики и ИКТ) следует использовать дополнительное время (факультативы, спецкурсы), а также организовывать дистанционную поддержку для подготовки к ЕГЭ. В арсенале учителя должны быть средства и методы, позволяющие обеспечить дифференцированный подход к учащимся, предоставить для учащихся со слабой подготовкой по информатике и ИКТ, возможность более длительной отработки умений в ходе решения простых задач, а для более подготовленных – достаточно быстрый переход к решению задач повышенного уровня. В этом большую помощь могут оказать практикумы, включающие наборы задач по разным темам, допускающие самопроверку. В качестве ресурсов, которые полезно использовать учителям при подготовке к ЕГЭ по информатике и ИКТ, можно рекомендовать и учащимся для самостоятельной подготовки:

- **http://kpolyakov.spb.ru/**
- **http://ege.yandex.ru/**
- **http://reshuege.ru/**
- **http://ege-go.ru/**

Литература:

- Учебно-методические письма, размещенные на сайте ФИПИ http://www.fipi.ru.
- В.Р. Лещинер, М.А. Ройтберг. Методические рекомендации для учителей, подготовленные на основе анализа типичных ошибок участников ЕГЭ 2015 года по информатике и ИКТ [Текст]: –М: ФИПИ, 2015г.
- Материалы сайта К.Ю. Полякова «Преподавание, наука и жизнь» kpolyakov.spb.ru.