

Семинар #1: Основы. Домашнее задание.

Задача 1. Плюс 1:

Напишите программу, которая будет считывать число и печатать это число, увеличенное на 1. Известно, что число больше 0 и меньше 1000.

ВХОД	ВЫХОД
1	2
100	101

Задача 2. Условие:

Напишите программу, которая будет считывать число и проверять, является ли число чётным и принадлежащим следующему множеству $[0, 20] \cup (100, 200)$ и печатать **Yes** или **No**. Используйте один оператор **if**.

ВХОД	ВЫХОД	ВХОД	ВЫХОД
4	Yes	22	No
5	No	100	No
20	Yes	102	Yes

Задача 3. Число, квадрат и куб:

Напишите программу, которая будет печатать само число, его квадрат и его куб от 1 до **n**, разделённые стрелочкой. Число **n** считывается с помощью **scanf**. Например, при **n = 5**, программа должна напечатать следующее:

```
1 -> 1 -> 1
2 -> 4 -> 8
3 -> 9 -> 27
4 -> 16 -> 64
5 -> 25 -> 125
```

Для того чтобы все числа печатались выровнено, можно использовать спецификатор **%3i** за место **%i** в **printf**. В этом случае, если число имеет в записи меньше 3-х цифр, то **printf** напечатает необходимое число пробелов перед числом.

Задача 4. Последовательность:

Пример программы, которая считывает число **n**. Затем считывает **n** чисел и находит среди них максимум.

```
#include <stdio.h>
#include <limits.h>
int main()
{
    int n;
    scanf("%i", &n);
    int max = INT_MIN;
    for (int i = 0; i < n; ++i)
    {
        int a;
        scanf("%i", &a);
        if (a > max)
            max = a;
    }
    printf("Max = %i\n", max);
}
```

В этой программе используется константа `INT_MIN` из библиотеки `limits.h`. Эта константа равна минимальному возможному значению чисел типа `int`, то есть `INT_MIN = -2147483648`. В этой задаче нельзя использовать массивы. Измените программу выше так чтобы программа находила максимум и количество элементов, равных этому максимуму.

ВХОД	ВЫХОД
3 1 2 3	3 1
3 7 7 7	7 3
10 1 8 2 4 8 8 1 5 2 8	8 4

Задача 5. Числа-градины:

Пусть нам на вход поступает число `n`. Мы преобразуем это число следующим образом $n = f(n)$, где

$$f(n) = \begin{cases} 3n + 1, & \text{если } n - \text{нечётное} \\ n/2, & \text{если } n - \text{чётное} \end{cases}$$

Затем повторяем этот алгоритм до тех пор пока число не достигнет единицы. Получится некоторая последовательность. Например, если изначально `n = 7`, то последовательность будет выглядеть следующим образом:

7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Ваша задача заключается в том, чтобы напечатать эту последовательность, её длину и максимальный элемент этой последовательности по изначальному числу `n`.

ВХОД	ВЫХОД
3	3 10 5 16 8 4 2 1 Length = 8, Max = 16
256	256 128 64 32 16 8 4 2 1 Length = 9, Max = 256
7	7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1 Length = 17, Max = 52

Задача 6. Сумма:

На вход программе подаются два целых числа `n` и `m`. Нужно посчитать следующую сумму:

$$S_{n,m} = \sum_{i=1}^n \sum_{j=1}^m (-1)^{i+j} i \cdot j$$

Например, если `n = 3`, а `m = 4`, то сумма будет равна:

$$S_{3,4} = 1 - 2 + 3 - 4 - 2 + 4 - 6 + 8 + 3 - 6 + 9 - 12 = -4$$

ВХОД	ВЫХОД
1 1	1
2 2	1
3 3	4
3 4	-4
5 7	12
10 10	25
77 107	2106

Задача 7. Печать всех делимых:

На вход программе подаются числа `a`, `b`, `c`. Программа должна напечатать все числа, делящиеся на `c` на отрезке `[a, b]` через пробел. Решите эту задачу как можно более эффективно.

ВХОД	ВЫХОД
1 20 4	4 8 12 16 20
1 20 7	7 14
1 10000 9500	9500
1 1000000000 500000000	500000000 1000000000
1 1000000000 123456789	123456789 246913578 370370367 493827156 617283945 740740734 864197523 987654312

Задача 8. Повтор массива

На вход программе приходит количество элементов некоторой последовательности, а затем сама последовательность. Вам нужно напечатать эту последовательность 2 раза.

ВХОД	ВЫХОД
4	10 20 30 40 10 20 30 40
10 20 30 40	
1	123 123
123	

Задача 9. Операция над массивом

Ниже приведён пример программы, которая считывает массив, затем удаляет первый элемент в этом массиве и печатает массив на экран:

```
#include <stdio.h>
int main()
{
    int a[1000];
    int n;
    scanf("%i", &n);
    for (int i = 0; i < n; ++i)
        scanf("%i", &a[i]);

    for (int i = 0; i < n - 1; ++i)
        a[i] = a[i + 1];
    n--;

    for (int i = 0; i < n; ++i)
        printf("%i ", a[i]);

    printf("\n");
}
```

Напишите программу, которая будет считывать массив и удалять все отрицательные элементы из массива, а затем печатать этот массив на экран. Постарайтесь написать как можно более эффективный код. Например, каждый элемент нужно переместить максимум 1 раз. Вам нужно изменить сам массив, а не просто напечатать отдельные его элементы.

ВХОД	ВЫХОД
6	0 2 5
0 -1 2 -3 -4 5	
2	9
9 -5	

Задача 10. Сумма столбцов

На вход поступают размеры матрицы n и m и элементы матрицы. Нужно найти сумму элементов в каждом столбце. Для этой задачи не нужно использовать двумерный массив, достаточно будет одномерного.

ВХОД	ВЫХОД
3 4	14 15 16 12
1 2 3 6	
6 5 4 2	
7 8 9 4	

Задача 11. Сортировка по сумме цифр

На вход подаётся последовательность чисел. Отсортируйте эти числа по возрастанию суммы цифр.

ВХОД	ВЫХОД
6	401 15 516 932 96 673
673 96 15 401 932 516	

В файле `numbers.txt` хранится 10000 чисел. Используйте перенаправление потока, чтобы отсортировать эти числа по сумме цифр. Сохраните результат в файле `sorted.txt`. Нужно написать в терминал следующее:

```
a.exe < numbers.txt > sorted.txt
```

Задача 12. Умножение матриц

На вход поступает число `n` и две квадратных матрицы размера `nхn`. Нужно перемножить эти матрицы и напечатать результат. Формула перемножения матриц:

$$C_{ij} = \sum_{k=0}^{n-1} A_{ik} \cdot B_{kj}$$

ВХОД	ВЫХОД
3	21 30 130
7 7 2	-6 21 82
1 8 3	48 12 -1
2 1 6	
5 2 9	
-4 2 11	
7 1 -5	

ВХОД	ВЫХОД
3	55 60 70
5 2 9	-4 -1 64
-4 2 11	40 52 -13
7 1 -5	
7 7 2	
1 8 3	
2 1 6	

В файле `matAB.txt` сохранены две матрицы 10х10. Используйте перенаправление потока, чтобы считать эти матрицы из файла `matAB.txt`, перемножьте их и сохраните в новый файл `matC.txt`. Нужно написать в терминал следующее:

```
a.exe < matAB.txt > matC.txt
```

В результате должна получиться такая матрица:

$$\begin{pmatrix} 259 & -15 & 237 & 257 & 231 & 67 & 237 & -64 & 152 & 363 \\ 555 & 233 & 539 & 188 & 356 & 325 & 423 & -47 & 123 & 387 \\ 497 & 512 & 572 & 95 & 619 & 155 & 414 & 207 & 203 & 217 \\ 455 & 280 & 675 & 354 & 664 & 346 & 483 & 177 & 168 & 404 \\ 264 & 182 & 272 & 290 & 474 & -33 & 234 & 99 & 379 & 156 \\ 272 & 180 & 469 & 286 & 326 & 282 & 325 & 215 & 195 & 231 \\ 421 & 363 & 475 & 506 & 359 & 481 & 468 & 101 & 325 & 328 \\ 384 & 218 & 567 & 395 & 475 & 488 & 361 & 168 & 291 & 298 \\ 387 & 297 & 480 & 170 & 318 & 423 & 483 & 10 & -17 & 406 \\ 193 & 241 & 486 & 38 & 403 & 146 & 286 & 326 & 212 & 172 \end{pmatrix}$$

Необязательные задачи (не входят в ДЗ, никак не учитываются)

Задача 1. Три числа:

На вход программе подаются три числа: `a`, `b` и `c`. Нужно проверить следующие условия:

1. Если числа `a`, `b` и `c` являются последовательными, то нужно напечатать `Consecutive`.
2. Если последовательность `a`, `b`, `c` является возрастающей, то нужно напечатать `Increasing`.
3. Если последовательность `a`, `b`, `c` является убывающей, то нужно напечатать `Decreasing`.
4. Если все три числа равны, то нужно напечатать `Equal`.
5. В ином случае нужно напечатать `None`.

ВХОД	ВЫХОД
1 2 3	Consecutive Increasing
1 2 4	Increasing
1 1 2	None
1 2 1	None
1 5 9	Increasing
1 0 -1	Consecutive Decreasing
1 5 4	None
7 7 7	Equal
20 15 5	Decreasing

Задача 2. Последовательность++:

Пример программы, которая считывает число `n`. Затем считывает `n` чисел и находит среди них максимум.

```
#include <stdio.h>
#include <limits.h>
int main()
{
    int n;
    scanf("%i", &n);
    int max = INT_MIN;
    for (int i = 0; i < n; ++i)
    {
        int a;
        scanf("%i", &a);
        if (a > max)
            max = a;
    }
    printf("Max = %i\n", max);
}
```

В этой программе используется константа `INT_MIN` из библиотеки `limits.h`. Эта константа равна минимальному возможному значению чисел типа `int`, то есть `INT_MIN = -2147483648`. В этой задаче нельзя использовать массивы.

Подзадачи:

Измените программу выше так чтобы:

1. Программа находила минимум, а не максимум. Может понадобиться константа `INT_MAX = 2147483647`.
2. Программа находила минимальное чётное число и максимальное нечётное. Если чётных или нечётных чисел нет, то программа должна печатать `None` за место числа.

ВХОД	ВЫХОД
3 4 5 6	4 5
3 7 7 7	None 7
10 1 8 2 4 8 8 1 5 2 8	2 5
4 10 8 6 8	6 None

3. Программа печатала **Increasing** если последовательность чисел строго возрастает, **Decreasing**, если последовательность чисел строго убывает и **Equal**, если все члены последовательности равны. В любом ином случае программа должна печатать **None**.

ВХОД	ВЫХОД
3 1 2 3	Increasing
3 7 7 7	Equal
5 20 15 10 7 5	Decreasing
4 1 1 4 5	None

Задача 3. Числа-градины II:

На вход поступает 2 числа **a** и **b**. Нужно найти такое число **n** ($a \leq n \leq b$), для которого последовательность чисел-градин будет самой длинной. Нужно напечатать число **n**, а также длину последовательности, которая начинается с **n**.

ВХОД	ВЫХОД
1 5	3 8
1 8	7 17
1 10	9 20
10 15	14 18
1 100	97 119
1 500	327 144
400 500	487 142
1 1000	871 179
1 10000	6171 261
1 100000	77031 351

Задача 4. Пифагоровы тройки:

На вход приходит целое число **n**. Нужно напечатать все возможные пифагоровы тройки **a**, **b** и **c**, такие что $a \leq n$, $b \leq n$ и $c \leq n$. Пифагорова тройка – это тройка натуральных чисел, для которых верно:

$$a^2 + b^2 = c^2$$

Пифагоровы тройки, получаемые из некоторой пифагоровой тройки путём обмена местами чисел **a** и **b** считаются дублирующими. Пифагоровы тройки, получаемые из некоторой пифагоровой тройки путём умножения всех чисел на некоторое натуральное число, также считаются дублирующими. Печатать дублирующие тройки не нужно.

Подсказка: Просто переберите все возможные значения **a**, **b** и **c**.

ВХОД	ВЫХОД
15	3 4 5 5 12 13
50	3 4 5 5 12 13 8 15 17 7 24 25 20 21 29 12 35 37 9 40 41

Задача 5. Операции над массивом++

Во всех подзадачах этой задачи вам нужно изменить массив **a** и, возможно, размер **n** между считыванием массива и его печатью. Каждая программа должна иметь такой вид:

```
#include <stdio.h>
int main()
{
    int a[1000];
    int n;
    scanf("%i", &n);
    for (int i = 0; i < n; ++i)
        scanf("%i", &a[i]);

    // ||||| Ваш код между считыванием и печатью массива |||||

    // ||||| Ваш код между считыванием и печатью массива |||||
    for (int i = 0; i < n; ++i)
        printf("%i ", a[i]);

    printf("\n");
}
```

Внутри вашего кода нужно считать дополнительные данные и изменить массив и переменную **n**.

1. **Удвоение массива:** Нужно увеличить массив **a** в 2 раза, заполнив новую часть копией массива **a**. Предполагается, что количество места в массиве (1000) больше чем $2n$, то есть места хватит. Не забудьте изменить переменную **n**.

ВХОД	ВЫХОД
4	0 1 2 3 0 1 2 3
0 1 2 3	
3	6 4 3 6 4 3
6 4 3	

2. **Вставка:** На вход подаётся массив, новый элемент массива и индекс – положение в массиве, после которого нужно вставить элемент. Чтобы освободить место в массиве нужно передвинуть часть элементов вправо. Предполагается, что количество места в массиве (1000) больше чем **n**, то есть места на 1 элемент хватит. Будьте осторожны, не перепишите элементы массива при их перемещении. Не забудьте изменить переменную **n**.

ВХОД	ВЫХОД
6	0 1 2 9 3 4 5
0 1 2 3 4 5	
9 2	
2	1 5 4
1 5	
4 1	

3. **Удаление:** На вход подаётся массив и индекс элемента, который нужно удалить. При этом понадобится передвинуть часть элементов влево.

ВХОД	ВЫХОД
6	0 1 2 4 5
0 1 2 3 4 5	
3	
2	5
1 5	
0	

4. **Удаление подмассива:** На вход подаётся массив и подмассив(2 индекса). Нужно удалить этот подмассив из массива. Постарайтесь написать как можно более эффективный код. Например, каждый элемент нужно переместить только 1 раз.

ВХОД	ВЫХОД
6 0 1 2 3 4 5 1 4	0 4 5
9 9 8 7 6 5 4 3 2 1 0 7	2 1

5. **Разделение на чётные/нечётные:** Переставьте элементы массива **a** так, чтобы сначала в нём шли нечётные элементы, а потом чётные. Причём порядок следования внутри чётной или нечётной части не важен. Эту задачу можно решить с использованием дополнительных массивов, а можно и без них.

ВХОД	ВЫХОД
7 0 1 2 3 4 5 6	1 3 5 0 4 2 6
9 9 8 7 6 5 4 3 2 1	9 7 5 3 1 8 2 4 6
2 2 1	1 2

6. **Раздвоение:** Увеличьте массив в 2 раза, раздвоив каждый элемент. Постарайтесь написать более оптимальный код без использования дополнительного массива.

ВХОД	ВЫХОД
6 0 1 2 3 4 5	0 0 1 1 2 2 3 3 4 4 5 5
1 1	1 1

7. **Циклический сдвиг:** На вход подаётся массив и целое положительное число **k** нужно циклически сдвинуть массив на **k** элементов вправо.

ВХОД	ВЫХОД
6 0 1 2 3 4 5 2	4 5 0 1 2 3
6 0 1 2 3 4 5 5	1 2 3 4 5 0

Подсказка: Новое положение **i**-го элемента в массиве будет задаваться формулой $(i + k) \% n$. Эту задачу проще всего решить с использованием дополнительного массива, но можно и без него.

Задача 6. Сортировка столбцов

На вход поступают размеры матрица **n** и **m** и элементы матрицы. Нужно отсортировать элементы в каждом столбце.

ВХОД	ВЫХОД
5 3	1 1 1
8 1 9	2 2 3
2 5 1	4 2 7
7 5 7	7 5 7
4 2 3	8 5 9
1 2 7	

ВХОД	ВЫХОД
2 6	5 2 1 3 1 2
6 2 8 3 2 4	6 4 8 5 2 4
5 4 1 5 1 2	

Задача 7. Нижняя граница

Пусть дан массив и некоторое число x . Нижняя граница – это индекс первого элемента, который больше или равен x . Напишите эффективную программу, которая ищет нижнюю границу на отсортированном массиве. Если такого элемента нет, то нужно вернуть n .

ВХОД	ВЫХОД
7 1 1 1 2 2 5 6 2	3
7 0 1 1 2 6 6 9 3	4

ВХОД	ВЫХОД
5 1 2 3 4 5 5	4
5 1 1 1 1 1 1	0

ВХОД	ВЫХОД
3 2 2 6 1	0
3 2 2 6 9	3

Задача 8. Поиск пика

Пусть дан массив. Известно, что у этого массива и первые $k \geq 0$ элементов строго возрастают, а остальные – строго убывают. Напишите эффективную программу, которая будет искать индекс пика (максимального элемента) в этом массиве. Используйте бинарный поиск, чтобы сделать алгоритм поиска пика более эффективным.

ВХОД	ВЫХОД
7 1 2 3 4 3 2 1	3
7 1 9 8 6 4 3 1	1

ВХОД	ВЫХОД
5 1 2 3 4 5	4
5 1 2 3 9 1	3

ВХОД	ВЫХОД
3 1 2 1	1
1 5	0